Proceedings of the 21st Workshop

From Objects to Agents

Bologna, Italy
September 14–16, 2020

CEUR Workshop Proceedings Vol. 2706
http://ceur-ws.org/Vol-2706/

Edited by

Roberta Calegari
Giovanni Ciatto
Enrico Denti
Andrea Omicini
Giovanni Sartor

Table of Contents

Preface

The Workshop "From Objects to Agents" (WOA) is the reference event for Italian researchers active in the agents and multi-agent systems research domain. Since its very first edition in 2000, located in Parma, WOA was conceived as a meeting occasion for researchers and practitioners from MAS-AI*IA (the AI*IA working group on MAS) and TABOO (the association for advanced technologies based on concepts from object-orientation). Since then, WOA was held on a yearly basis in many different Italian locations, from north to south (islands included), gaining a conspicuous success and succeeding in gathering researchers and practitioners from various research fields, thanks to its format.

Despite stemming from an Italian initiative, WOA is an international workshop where presenters and participants exchange opinions and discuss on-going works in a friendly yet rigorous setting. Furthermore, since 2004, WOA includes a one-day mini-school, where experienced scientists and professionals can introduce younger researchers as well as Ph.D. and undergraduate students to hot topics in the fields of AI, MAS, and programming languages.

The 21st edition of the workshop was held on September 14–16, 2020 in Bologna, as an on-line virtual meeting. During the three days, more than 18 speakers joined the workshop, as well as many more attendants. In particular, this edition was structured in two mini-school sessions, one keynote speech, and in six technical sessions—each one discussing three papers. The six technical sessions hosted the presentation of 18 papers, 17 of which are collected in this virtual volume published by CEUR in the AI*IA Series.

The topics discussed in the papers cover some of the hottest topics laying under the umbrella of "MAS for human-centred intelligent systems", as requested by the call for papers. The choice of this theme was deliberate. In fact, it is widely recognised that nowadays intelligent systems have to be human-centred, with the human(s) in the loop acting synergistically within the system. Accordingly, human-centred AI focuses on the design, development, and deployment of intelligent systems that co-operate with humans in real-time in a deep and meaningful way. There, the AI system is expected to continuously improve itself by learning from humans while creating an effective interactive experience. In such a scenario, sub-symbolic techniques play a major role to provide sophisticated features that would be hard for developers to implement otherwise. However, symbolic approaches are getting more and more attention as those that could make AI amenable to human understanding and interpretation, once suitably integrated with sub-symbolic approaches. In this context, MAS are the core of the design of intelligent systems, as they represent the *glue* making symbolic and sub-symbolic components fruitfully interoperable.

As far as the mini-school is concerned, two sessions were organised, hosting talks from experts in the fields of Logic Programming and MAS. In particular, in the first session, Fabrizio Riguzzi illustrated the role of Probabilistic Logic Programming (PLP). PLP provides a powerful combination that has already achieved successful applications in a variety of fields. The talk discussed PLP languages under the distribution semantics – one of the most impactful semantics in this area – and introduced the types of reasoning that can be performed with these languages: inference, weight learning and structure learning. In the second session, Viviana Mascardi provided an enlightening perspective on the current status of logic-based technologies for

MAS, based on her review "Logic-based Technologies for Multi-agent Systems: A Systematic Literature Review". The talk emphasised the core role of MAS in the design of intelligent systems since their very beginning and their long-term connection with logic-based technologies, thus opening new ways to engineer explainable intelligent systems.

The "Fabio Bellifemine" keynote speech was given by Alessandro Ricci discussing the topic of programming multi-agent systems—"Reflections after a decade building and using JaCaMo". There, JaCaMo is a platform that allows to program multi-agent systems integrating agent, environment and organisation as first-class design and programming dimensions, and exploiting Jason, CArtAgO and Moise as concrete technologies. Within the talk, Alessandro Ricci shared his experience, reflections, and thoughts about the future of agent-oriented programming.

The 17 papers collected in this issue were organised, presented, and discussed into six thematic sessions. The final versions here includes also include the outcomes of some of the several interesting discussions that followed the presentations at the workshop. The authors' contributions cover quite relevant research areas that include *(i)* simulation, *(ii)* organisations, norms, and argumentation, *(iii)* features of MAS as robustness, trust, and explainability, *(iv)* healthcare applications, *(v)* agents and actors for data science and *(vi)* tools and application for MAS.

Finally, the Organising Scientific Committee gratefully thanks all those who have contributed, with their work and their enthusiasm, to the success of this edition of WOA: the members of the WOA Board; the members of the Program Committee; the Department of Informatics – Engineering and Information Sciences (DISI) of the University of Bologna; the Alma Mater Research Institute for Human-Centered Artificial Intelligence of the University of Bologna; the local organisers; the speakers of the workshop sessions; the mini-school lecturers; the sponsors; and all collaborators who participated in the organisation. More generally, we would like to thank the lively, creative, and sometimes even volcanic community that has been regularly meeting for 21 years at the workshop.

Bologna, Italy
October 15, 2020

Roberta Calegari
Giovanni Ciatto
Enrico Denti
Andrea Omicini
Giovanni Sartor

Session 1

Simulation in MAS

An Agent-based Simulator for Urban Air Mobility Scenarios

Maria Nadia Postorino[a], Francesco A. Sarné[b] and Giuseppe M. L. Sarné[c]

[a]Department DICAM, Alma Mater Studiorum, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
[b]Politecnico of Milan, P.za Leonardo da Vinci, 32 20133 Milano, Italy
[c]Department DICEAM, University Mediterranea of Reggio Calabria, Loc. Feo di Vito, 89122 Reggio Calabria, Italy

Abstract

In the next years, flying cars are expected to become a real opportunity to realize Urban Air Mobility (UAM) systems. Most of the appeal is given by the opportunity of avoiding congestion, gaining time and reducing environmental impacts with respect to conventional mobility. However, UAM implementation is not trivial as it has several implications in manifold areas like safety, security, traffic control, legal issues and urban design among the others. To investigate on the impacts of UAM, a dedicated agent-based framework has been designed. The results of some preliminary tests carried out to verify the capabilities of this simulator are presented.

Keywords

Flying cars, Simulator, Software agents, Transportation network, Urban Air Mobility

1. Introduction

In the wake of Icarus, thanks to recent technological advancements, Personal Aerial Vehicles (PAV) and Passengers Unmanned Aerial Vehicles (PUAV) moving on both land and aerial modalities, also known as “flying cars”, make it real the opportunity to realize an *Urban Air Mobility* (UAM) for point-to-point connections. To this aim, a growing number of flying cars is being developed or tested all over the world also by commercial companies like Uber [1], which is planning to start with aerial services [2] when technical, urban, legal and economic criticisms will be solved. Indeed, until now UAM requirements have not been considered neither in urban planning policies (e.g., landing and take-off spaces for transition from ground to aerial mode and vice versa) nor from laws and regulations point of view (e.g., safety, security and privacy issues due to flights over or close to buildings have not been considered yet).

Consequences on urban transportation contexts and economic convenience of UAM scenarios are not fully understood and, therefore, there is the need to investigate about them. To this aim, the state of a transportation network [3], where conventional vehicles coexist with flying cars, has to be simulated. In particular, interactions and decision processes not taken into account in

WOA 2020: Workshop “From Objects to Agents”, September 14–16, 2020, Bologna, Italy
marianadia.postorino@unibo.it (M. N. Postorino); francescoalessandro.sarne@polimi.it (F. A. Sarné); sarne@unirc.it (G. M. L. Sarné)
https://www.unibo.it/sitoweb/marianadia.postorino/ (M. N. Postorino); https://www.unirc.it/scheda_persona.php?id=50234 (G. M. L. Sarné)
0000-0002-6609-4818 (M. N. Postorino); 0000-0003-3753-6020 (G. M. L. Sarné)

usual traffic simulations (e.g., interactions of flying cars with other vehicles and obstacles or criteria adopted to choose of moving in aerial or ground modality) have to be considered for evaluating their effects in UAM scenarios.

Intelligent software agent technology (from here on only agent) has been extensively applied to simulate and manage different aspects, at different level of detail, of a wide variety of transportation systems [4, 5, 6, 7]. In transportation systems, agents can play different roles (e.g., travelers, vehicles, signals, etc.). Many studies have explored the opportunity of taking advantage from the autonomous, adaptive, learning, pro-active and social abilities of agents [8] as well as their capabilities to work in large, centralized or distributed contexts also in presence of uncertainty or dynamic behaviors [9, 10].

Such agent features well fit with the need to simulate autonomous vehicles, their motion on transportation networks and their choice processes. Therefore, the agent technology has been adopted also to implement an UAM simulator by associating an agent with each moving flying or ground vehicle that, similarly to Connected Automated Vehicles (CAVs), has been assumed to be fully automated. By using this UAM simulator, we want to investigate on the potential advantages, in terms of travel times, deriving by the possible, future realization of UAM scenarios but without to simulate other aspects which depend from laws and regulations that at the moment are not defined. To this aim, some test transportation networks of different size have been considered, in order to have comparable scenarios. In fact, it is expected that UAM scenarios in existing urban contexts of different size, to which transportation networks refer, will be affected by the urban features, such as location of spaces for landing and take-off, urban structure, building height and specific vertical obstacles among the others, which will result in specific requirements for each tested real network. Then, to avoid specific-feature effects and provide appropriate comparisons, in this study modular test transportation networks have been used, which are based on the aggregation of suitable, unitary modules and refer to the same urban features. The preliminary campaign of experiments carried out on these modular transportation networks of different size has allowed to calibrate the agent-based simulator, including agents' behaviors.

To compare UAM scenarios, the index called "Travel Time Advantage" (TTA) has been introduced, which is the ratio between travel times computed when both ground and flying mobility are allowed on the examined transportation network and travel times computed when only ground mobility is admitted.

The paper is organized as follows. In Section 2 some of the main characteristics of flying cars and some scenarios are presented. In Section 3 the agent-based UAM model is described and in Section 4 the UAM agent-based simulator is presented and discussed. Section 5 some related work are described and, finally, in Section 6 some conclusions are drawn.

2. Urban Air Mobility (UAM)

In this Section the main features characterizing (prototype) flying cars and UAM scenarios will be shortly introduced.

The main flying car characteristics can be summarized in:

1. Vehicle architecture. Shape and size of vehicles must be compatible with both flying

(e.g., aerodynamic) and land (e.g., road lanes width, take-off, landing and parking spaces) constraints [11]. Vehicles mainly differ for take-off and landing (TOL) operations, which can be Conventional (CTOL) or Vertical (VTOL). In urban contexts, VTOL vehicles are expected to be preferred to CTOL ones for the smaller TOL spaces required and the higher maneuverability [12].

2. Operability. Different aspects can influence the vehicle operability [13, 14, 15, 16], which is usually defined in terms of:
   a) Range - the maximum flight distance, measured on the ground, traveled for the maximum fuel/charge capacity;
   b) Endurance - the maximum flight time with respect to the maximum fuel/charge capacity;
   c) Speed - with respect to both “on-the-road” and “in-flight” modalities.

3. Vertical position and main flight rules. The vertical position of flying objects in low level space may be identified by the following vertical distances, namely:
   a) Height - measured from the Above Ground Level (AGL);
   b) Altitude - measured from the Mean Sea Level (MSL);

   Flying conditions [17] currently operating are:
   a) Visual Flight Rules (VFR), for Visual Meteorological Conditions (VMC), permitted until $3000\ ft$ from the ground or sea level;
   b) Instrument Flight Rules (IFR), applying to Instrument Meteorological Conditions (IMC) [18].

   Flying car VFR conditions are expected to be realized at a Very Low Level (VLL) airspace, i.e. $0 - 500\ ft$ AGL, where cabin pressure plant does not need.

4. Automation level. Flying-cars can have different automation/autonomy and flight assistance degrees, depending on the on-board driving systems [19, 20] and communication features, e.g. FANET [21]. Note that autonomous vehicles are expected to be driverless and fully automated (e.g., they monitor the environment around them to adapt their positions/behaviors).

The main expected UAM scenarios are:

I) Point-to-point services between origin/destination prefixed points (i.e., from/to relevant places to/from suitable collecting areas), by certified transportation operators with authorized flight plans;

II) Long/medium-distance trips, with flying mode for the longer legs and ground mode within cities, and with take-off and landing areas on external or dedicated transition roads completely separated from ground mode operations;

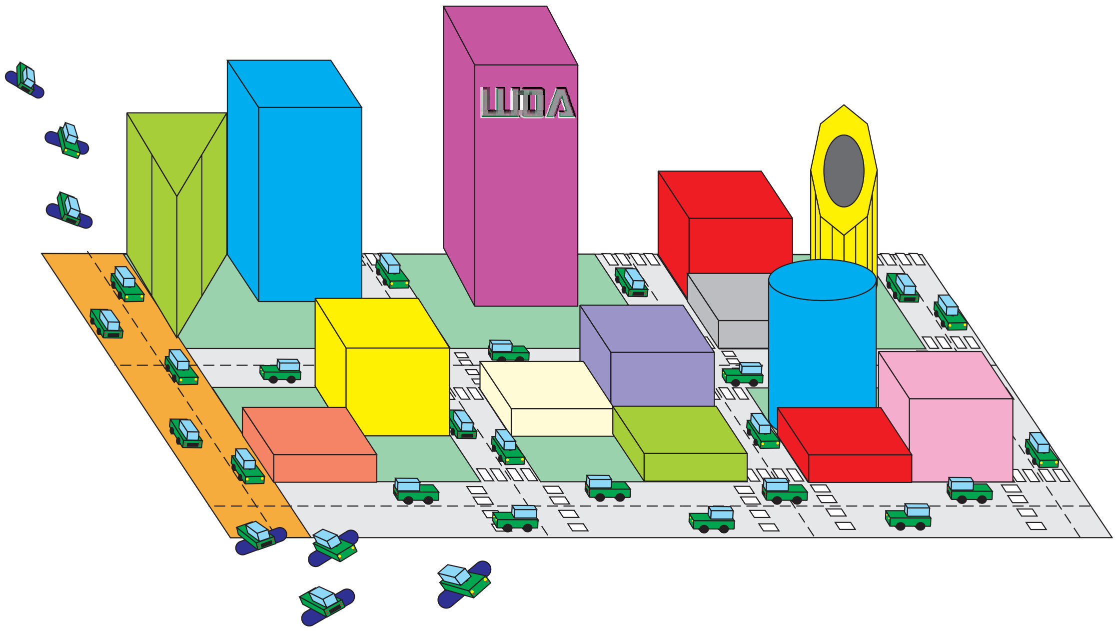


Figure 1: Case C: Short/medium-distance trips within cities.

III) Short/medium-distance trips (Figure 1), where flying cars can move both between city pairs and almost everywhere within cities, although TOL operations happen only at dedicated areas linked to roads for only ground mode.

The agent-based simulator has been designed for the latter one, which includes the main features of the other two cases.

3. The Agent-based UAM Model

In this Section we describe the within-city trip scenarios (case III in Section 2) for which the agent-based UAM simulator (see Section 4) has been designed. Simulation results have been evaluated based on the travel times required to move between origin/destination pairs. Moreover, agents simulate flying cars assumed to be electric and autonomous (coherently with expectations for the still-in-progress CAVs models) to *i*) keep separations in the three dimensions, right trajectories and altitude (according to meteorological conditions) and *ii*) exchange data to be processed on-board to avoid collisions or wake turbulence effects.

The interactions among *i*) flying cars, *ii*) flying and ground cars and *iii*) flying cars and ground obstacles (e.g., buildings, cables) have been considered. Security issues have not been explicitly simulated, while safety aspects have been considered in terms of suitable distances kept from each type of obstacle, including other moving objects. Interactions among flying and ground cars within the city have been allowed only at pre-fixed "transition areas" (TAs) (*i*) placed where the urban structure is sufficiently dispersed, at no less than d_{min} from each other transition area and (*ii*) sufficient to provide safe conditions for entering/leaving the ground transportation network also along the TAs [22]. Moreover, flights have been allowed only along prefixed, safe routes. Other issues concerning rules and prescriptions for security reasons are behind the focus of this research.

To obtain realistic simulations, we assumed that: (*i*) flying cars keep safe distances with ground obstacles and other flying cars; (*ii*) TOL transition areas are suitably connected to the ground transportation network; (*iii*) flying mode can be chosen to move between TAs by maintaining a height suitably greater than the highest building or ground obstacle.

In detail, the agent framework has been specified as follows:

1. Vehicles are homogeneous for characteristics and equipment and each vehicle i is associated with an agent A_i.

2. For each couple (A_i, A_j) of agents:

   a) In ground mode, each A_i follows A_j at a minimum distance $d_{ij} = t \cdot v_{gi} + s_{ib}$, where: $t = 1\ sec$ is the time to start braking; v_{gi} is the ground speed of A_i; s_{ib} is the braking space at a constant deceleration.

   b) In flying mode, the vertical position of A_i, flying over A_j, is $h_i = d_0 + n \cdot s_z$, where: d_0 is the minimum height to overfly urban areas; n is the number of agents under A_i on the z axis; s_z is the minimum vertical separation between agents. Note that more vehicles can use the same horizontal route but at different heights.

   c) In flying mode, $\forall A_i$ that follows A_j on the same horizontal route, their minimum gap s_x is constant.

   d) Transition from ground/flying to flying/ground mode happens at dedicated TAs based on a booked and confirmed time slot authorization; the time slot depends on the estimated arrival/leaving time at the TA depending on ground and flying traffic conditions.

3. For a given origin/destination (O/D) pair [23] the following conditions hold:

   a) The flying leg of a trip follows the Euclidean route. If the Euclidean distance of a trip is greater than d_{min} it will take place by combining ground and aerial links, otherwise it will be only on ground mode.

   b) For each O/D trip, the minimum travel time path is computed as $t(f_g) = l_g/v_g + l_f/v_f$. The ground speed v_g is empirically computed for urban roads as $v_g = c_1 - c_2 \cdot f_g$, where f_g is the traffic volume (i.e., the number of agents) on the ground link at a given time, $c_1 = 37.5$ and $c_2 = 8.5 \cdot 10^{-6}$ (for v_g measured in Km/h) are coefficients empirically computed for averaged road features (e.g., width, slope, etc.), l_g, l_f and v_f are respectively the length of the ground link, the length of the aerial link and the speed on the aerial link. Note that, congestion effects have been assumed to be caused only by ground traffic flows [24].

   c) Agents are autonomous in their choices, although coordinated by a central Agency to/from which they send/receive information about their position, those of other agents and obstacles in their neighboring and about the status of the transportation network.

Combined ground and aerial trips will start only after agents receive information by the Agency, in order to avoid congestion effects at the transition areas

d) All the agents adopt the same TOL procedures.

To compare UAM scenarios at increasing network size, the index "*Travel Time Advantage*" (TTA) is computed as the ratio between the total "flying-ground" travel time and the total ground travel time for "only ground" mode to move between an O/D pair over all the agents and O/D pairs, has been considered:

$$TTA = \frac{\sum_i T^{G+F}_{O/D,\ i}}{\sum_i T^{G}_{O/D,\ i}} \tag{1}$$

where, for each A_i, (*i*) $T^{G+F}_{O/D,\ i}$ is its travel time in ground+flying mode and (*ii*) $T^{G}_{O/D,\ i}$ is its travel time in only-ground mode.

4. The Agent-based UAM Simulator

This Section describes the agent-based simulator designed to implement the UAM model presented in Section 3 and the preliminary results obtained. This simulator has been written in C++ by expanding the one developed for simulating the ground mobility, exploited in [25, 26], and it is not equipped with a graphical user interface.

More in detail, each agent represents a vehicle and it is an object implemented by a specific class. All the vehicles (i.e., agents) can move on both ground and flying modalities and are assumed to be provided with homogeneous features (a reasonable assumption because it is expected that they will have standardized features and equipment).

Agents can autonomously decide in which modality to move among O/D pairs on the basis of the information that they mutually exchange with the other agents and with the Agency, that acts as a Traffic Controller. Information are exchanged by messages having a simplified JADE-like structure and implemented as objects of a dedicate class. In particular, each message stores information about (*i*) sender, (*ii*) receiver, (*iii*) type of the content (e.g., Information, Route, Action) and (*iv*) content (e.g., O/D pair, route, ground/flight modality flag, coordinates on the three axes, speed, action required).

As specified in Section 1, the aim of this simulator is to investigate UAM advantages in terms of travel times, with respect to transportation networks of different sizes. Because of real urban transportation networks have evolved without considering UAM features (e.g., TOL spaces for flying-cars, which should be based on standards for commercial flying cars that are still undefined), the simulation results could not be completely comparable among them if the simulator is applied to real contexts of different size. As introduced in Section 1, network size plays a role in the assessment of potential benefits coming from UAM contexts. Therefore, without loss of generality, we exploited artificial, modular transportation networks based on a specific transportation network module object. In this way, all the transportation networks are intended to share the same urban design, such as building heights and position, road features, and with particular attention to the location of TAs for flying-cars transition from/to flying mode to/from ground mode.

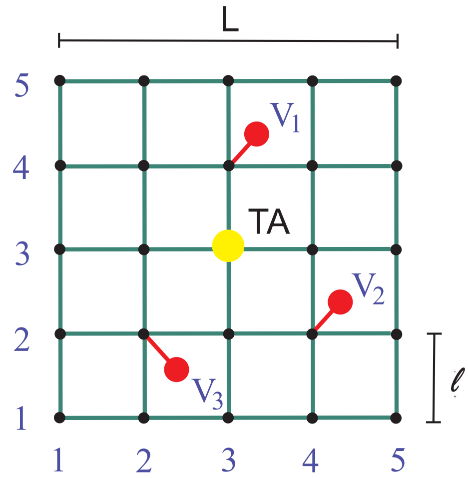


Figure 2: Baseline transportation network module (green lines represents real connections, red line represent virtual connections with trip origin/destination red nodes V, TA is the transition area of the module).

In particular, the basic transportation network module consists of a square mesh grid (see Figure 2) of dimensions $L \times L$ formed by 5×5 nodes and two-way road links of equal capacity and length $l = L/4$. TOL procedures at the transition area TA (represented by the yellow circle in Figure 2) are maintained distinct for ground traffic flows entering to/exiting from it. More in detail, each combined (i.e., ground + flying) trip: *i*) starts from an origin (o) node V_o; *ii*) reaches the transition area TA_o in ground mode; *iii*) takes-off from the transition area TA_o and lands at the destination (d) transition area TA_d in flying mode; *iv*) reaches in ground mode the destination node V_d where the trip ends.

The UAM simulator has been applied on two test transportation networks formed by 2×2 and 3×3 modules and by setting the length of each module to L=1600 m. Moreover, the O/D trip demand has been simulated by adopting an average value of 250 vehicles/h. For each O/D pair, the demand for time intervals of 5 minutes has been generated by using a variation coefficient set to 0.4. The minimum flight height has been set to 50 m[1], by assuming a maximum building height of 30 m. Based on the aerial link length and height, the cruise flying speed varies in the range $80 \div 120$ Km/h. In principle, departure times at a transition area depend on *i*) the expected ground travel time to reach the transition area from an origin node and *ii*) the queue at the transition area. To avoid or minimize waiting times at the transition area (i.e., queues for departures and arrivals), the Agency will inform each agent (i.e., vehicle) about the estimated times:

i) to reach, in ground mode, the transition area from an origin node by considering the current number of agents on the path;

ii) to fly between two transition areas by considering take-off and landing procedures, cruise

[1]Note that the adoption of a lower minimum flight height requires the assumption of additional conditions and hypothesis on the vehicle equipment, the air traffic control and the urban design.

Table 1
TTA results for the tested networks of 3 x 3 and 4 x 4 modules

Scenario	3 x 3	4 x 4
S_0 (O/D baseline)	0.38	0.35
S_1 (10% O/D increase)	0.45	0.39
S_2 (20% O/D increase)	0.66	0.57
S_3 (30% O/D increase)	0.73	0.65

speed and the time spent until a free slot is available, which depends on the current number of agent on that route.

The structure of the transportation test network is coherent with conventional city organization where only few areas could be available for transition processes, mainly for safety reasons and urban obstacles. Moreover, we assumed that the aerial network is considered virtually not congested because on the same route there is the opportunity of using more lanes, separated from each other by $5\ m$ in height (see Section 3). Note that, for short trips the travel time of only-ground paths could be less than the one of combined ground + flying paths.

Agent moves on the transportation links according to a minimum travel time path criterion [27]. The link travel times are continuously updated by considering the number of agents that are on the links (see Section 3, point b). At transition areas, the maximum acceleration and speed in ground modality have been set respectively to $2.5\ m/sec^2$ and $100\ km/h$ [28].

The reference (i.e., baseline) transportation UAM scenario is S_0, with baseline O/D trip demand level and only-ground mode. For the other scenarios, the O/D trip demand has been increased by 10%, 20% and 30% with respect to S_0. For the S_0 scenario, the value of TTA is 1, while enabling also the flying modality the obtained results are shown in Table 1. As it can be seen, the higher the level of demand, the more the link traffic flows increase that, in turn, causes travel times increase according to a congested network approach[2]. Finally, given that not all individual trip origins and destinations can be reached in a ground mode, and not all the trips are suitable for flying legs, flying and ground modes have to co-exist. However, when ground traffic increases then travel times generally increase and the times to reach transition areas to travel in aerial mode could not be more convenient than using only ground links.

5. Related Work

Decision processes underlying planning and management activities require the knowledge of the state of a system under different conditions and constraints, which can be obtained by using simulation tools to test hypotheses and architectures [3]. To this aim, the agent technology is widely adopted for its advantages, particularly the opportunity of providing agents with different degrees of intelligence, autonomy, learning, adaptive, time-persistent and pro-active capabilities [4, 6]. In the transportation field, agent-based simulations are mainly carried out at

[2]Agent's path choices change according to link travel times, which in turn depend on agents on the link, thus producing a traffic flow distribution on the network [29, 30].

a microscopic level [31], but there exist also a significant amount of both macroscopic (usually less competitive in terms of design and use of computational/storage resources) and mesoscopic (combining micro and macro aspects) agent-based tools for simulations [32, 33].

Agents have been exploited to study almost all the different aspects involved in usual transportation systems like, among the others, network management [34], transit [35], car-sharing and car-pooling [36, 37], vehicle emissions [38, 39], pedestrian mobility [40], flight recommender [41]. However, given the overwhelming body of researches presented in the literature and the impossibility to provide the interested readers with a comprehensive summary, they could refer to the many existing survey as [4, 42, 43, 44]

In the latter years, an increasing number of research dealt with different aspects involved in UAM and, also in this case, agent-based simulation have been widely exploited to study the opportunities offered by this new promising type of mobility [45]. For instance, high-dense traffic UAM scenarios have been considered in [46, 47] by adopting several scheduling horizons, in [48] airspace integration approaches have been investigated on air vehicle separation issues and in [49] autonomous vehicles, driven by an algorithm with collision avoidance capability, have been simulated on three free-flight scenarios. Other studies have simulated an UAM service on the Sioux Falls area to evaluate several parameter sets and contexts in [50] or by analyzing three case studies to identify possible constraints for UAM services on the basis of mission types or environments in [51].

Finally, communications play an important role for automated/autonomous vehicles and software agents are frequently adopted to simulate communication architecture, routing protocols and the coverage range of the ground infrastructure in complex urban environments. For ground and flying vehicles, Vehicular and Flying Ad hoc Networks (i.e., VANET [52] and FANET [53]) have been respectively proposed to improve the safety of vehicles and prevent collision accidents. In particular, [54] highlights as usual Air Traffic Control (ATC) systems, in presence of high UAM traffic levels and complex urban environments, might fail in monitoring and supporting the vehicle safety and this requires that vehicles should be provided with high levels of autonomous driving.

6. Conclusions

This paper presented an agent-based simulator designed to simulate UAM by considering vehicle interactions (when they are in ground and aerial modality), transition processes, security and air traffic control issues. It allows to evaluate the benefits deriving from UAM, with the desired level of detail, on simulated transportation networks, by means of the value of TTA measure.

Forthcoming researches will test this simulator on different UAM contexts represented by transportation networks of different size and with different demand levels also to evaluate the potential advantage given by UAM with respect to the demand level, flight distance and location of transition nodes. However, note that current regulations do not admit private flights over the city at low altitudes, except some specific, authorized cases and, therefore, before UAM becomes a reality the whole regulatory framework should be changed/adapted to meet some specific requirements.

Finally, further advancements will include the simulation of aerial congestion phenomenon, the optimization of taking-off and landing processes under specific conditions and the effects due to the location of transition nodes.

Acknowledgments

This study has been supported by the Network and Complex Systems (**NeCS**) Laboratory at the University Mediterranea of Reggio Calabria, Department of Civil, Energy and Materials Engineering (DICEAM).

References

[1] Uber, https://www.uber.com, 2020.
[2] Uber, https://www.uber.com/elevate.pdf, 2020.
[3] J. Barceló, et al., Fundamentals of traffic simulation, volume 145, Springer, 2010.
[4] B. Chen, H. H. Cheng, A review of the applications of agent technology in traffic and transportation systems, Intelligent Transportation Systems, IEEE Transaction on 11 (2010) 485–497.
[5] L. Weng, F. Menczer, Computational analysis of collective behaviors via agent-based modeling, in: Handbook of Human Computation, Springer, 2013, pp. 761–767.
[6] M. N. Postorino, G. M. L. Sarné, Agents meet traffic simulation, control and management: A review of selected recent contributions, in: Proceedings of the 17th Workshop "from Objects to Agents", WOA 2016, volume 1664 of *CEUR Work. Proceedings*, 2016, pp. 112–117.
[7] S. Abar, G. K. Theodoropoulos, P. Lemarinier, G. M. P. O'Hare, Agent based modelling and simulation tools: A review of the state-of-art software, Computer Science Review 24 (2017) 13–33.
[8] G. S. Bhamra, A. K. Verma, R. B. Patel, Intelligent software agent technology: an overview, International Journal of Computer Applications 89 (2014).
[9] D. Ye, M. Zhang, A. V. Vasilakos, A survey of self-organization mechanisms in multiagent systems, IEEE Transaction on Systems, Man, and Cybernetics: Systems 47 (2016) 441–461.
[10] N. Naciri, M. Tkiouat, Multi-agent systems: theory and applications survey, International Journal of Intelligent Systems Technologies and Applications 14 (2015) 145–167.
[11] K. Rajashekara, Q. Wang, K. Matsuse, Flying cars: Challenges and propulsion strategies, IEEE Electrification Magazine 4 (2016) 46–57.
[12] B. Saeed, G. B. Gratton, An evaluation of the historical issues associated with achieving non-helicopter v/stol capability and the search for the flying car, The Aeronautical Journal 114 (2010) 91–102.
[13] C. Lan, J. Roskam, Airplane Aerodynamics and Performances, DARcorporation , US, Kansas, 1997.
[14] J. Seddon, S. Newman, Basic helicopter aerodynamics, American Institute of Aeronautics and Astronautics, 2001.
[15] D. McLean, Understanding aerodynamics: arguing from the real physics, John Wiley & Sons, 2012.

[16] S. Corda, Introduction to aerospace engineering with a flight test perspective, John Wiley & Sons, 2017.
[17] ICAO, Rules of the Air, Annex 2 to the Convention on International Civil Aviation, 10th Edition, July 2005, ICAO, 2005.
[18] M. Consiglio, S. Conway, C. Adams, H. Syed, Sats hvo procedures for priority landings and mixed vfr/ifr operations, in: 24th Digital Avionics Systems Conference, volume 2, IEEE, 2005, pp. 8–pp.
[19] W. Payre, J. Cestac, P. Delhomme, Intention to use a fully automated car: Attitudes and a priori acceptability, Transportation research part F: traffic psychology and behaviour 27 (2014) 252–263.
[20] E. Dogan, M.-C. Rahal, R. Deborne, P. Delhomme, A. Kemeny, J. Perrin, Transition of control in a partially automated vehicle: effects of anticipation and non-driving-related task involvement, Transportation research part F: traffic psychology and behaviour 46 (2017) 205–215.
[21] O. S. Oubbati, A. Lakas, F. Zhou, G. Mesut, M. B. Yagoubi, A survey on position-based routing protocols for flying ad hoc networks (fanets), Vehicular Communications 10 (2017) 29–56.
[22] M. N. Postorino, V. Barrile, F. Cotroneo, Surface movement ground control by means of a gps–gis system, Journal of Air Transport Management 12 (2006) 375–381.
[23] M. N. Postorino, G. Musolino, P. Velonà, Evaluation of o/d trip matrices by traffic counts in transit systems, in: Schedule-Based Dynamic Transit Modeling: Theory and applications, Springer, 2004, pp. 197–216.
[24] M. N. Postorino, G. M. L. Sarné, Reinventing mobility paradigms: Flying car scenarios and challenges for urban mobility, Sustainability 12 (2020) 3581.
[25] P. De Meo, F. Messina, M. N. Postorino, D. Rosaci, G. M. L. Sarné, A reputation framework to share resources into iot-based environments, in: 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), IEEE, 2017, pp. 513–518.
[26] M. N. Postorino, G. M. L. Sarné, A preliminary study for an agent blockchain-based framework supporting dynamic car-pooling, in: Proc. of 20th Workshop "from Objects to Agents"(WOA 2019), volume 2404, CEUR Workshop Proceedings, 2019, pp. 65–70.
[27] H. Ortega-Arranz, D. R. Llanos, A. Gonzalez-Escribano, The shortest-path problem: Analysis and comparison of methods, Synthesis Lectures on Theoretical Computer Science 1 (2014) 1–87.
[28] Mechanical engineering site, http://www.mechanicalengineeringsite.com/want-a-flying-car-here-it-is-the-pal-v-flying-car/#Technical_details, 2020.
[29] E. Cascetta, Transportation systems analysis: models and applications, volume 29, Springer Science & Business Media, 2009.
[30] A. Krylatov, V. Zakharov, T. Tuovinen, Optimization models and methods for equilibrium traffic assignment, Springer.
[31] J. Du, H. A. Rakha, A. Elbery, M. Klenk, Microscopic simulation and calibration of a large-scale metropolitan network: Issues and proposed solutions, Technical Report, 2018.
[32] T. Jeerangsuwan, A. Kandil, Agent-based model architecture for mesoscopic traffic simulations, Computing in Civil and Building Engineering (2014) 1246–1253.
[33] I. Caldas, J. Moreira, J. Rebelo, R. J. F. Rossetti, Exploring visualization metaphors in

macroscopic traffic simulation, in: 2018 IEEE International Smart Cities Conference (ISC2), IEEE, 2018, pp. 1–6.
[34] H. Hamidi, A. Kamankesh, An approach to intelligent traffic management system using a multi-agent system, International Journal of Intelligent Transportation Systems Research 16 (2018) 112–124.
[35] J. Liu, X. Zhou, Capacitated transit service network design with boundedly rational agents, Transportation Research Part B: Methodological 93 (2016) 225–250.
[36] E. Picasso, M. N. Postorino, G. M. L. Sarné, A study to promote car-sharing by adopting a reputation system in a multi-agent context., in: Proceedings of the 18th Workshop "from Objects to Agents", WOA 2017, volume 1867 of *CEUR Workshop Proceedings*, 2017, pp. 13–18.
[37] I. Hussain, L. Knapen, A. Yasar, T. Bellemans, D. Janssens, G. Wets, Negotiation and coordination in carpooling: An agent-based simulation model (2016).
[38] S. F. Smith, G. J. Barlow, X. F. Xie, Z. B. Rubinstein, Surtrac: scalable urban traffic control (2013).
[39] C. Hofer, G. Jager, M. Fullsack, Large scale simulation of co2 emissions caused by urban car traffic: an agent-based network approach, Journal of Cleaner Production 183 (2018) 1–10.
[40] S. Liu, S. Lo, J. Ma, W. Wang, An agent-based microscopic pedestrian flow simulation model for pedestrian traffic problems, Intelligent Transportation Systems, IEEE Transaction on 15 (2014) 992–1001.
[41] M. N. Postorino, G. M. L. Sarné, A neural network hybrid recommender system, in: Proceedings of the 2011 Conference on Neural Nets WIRN10, 2011, pp. 180–187.
[42] M. N. Postorino, G. M. L. Sarné, An agent-based sensor grid to monitor urban traffic, in: Proceedings of the 15th Workshop "from Objects to Agents", WOA 2014, volume 1260 of *CEUR Workshop Proceedings*, 2014, pp. 1–6.
[43] A. L. C. Bazzan, F. Klugl, A review on agent-based technology for traffic and transportation, The Knowledge Engineering Review 29 (2014) 375–403.
[44] G. O. Kagho, M. Balac, K. W. Axhausen, Agent-based models in transport planning: Current state, issues, and expectations, Procedia Computer Science 170 (2020) 726–732.
[45] Y. Mualla, W. Bai, S. Galland, C. Nicolle, Comparison of agent-based simulation frameworks for unmanned aerial transportation applications, Procedia computer science 130 (2018) 791–796.
[46] C. Bosson, T. A. Lauderdale, Simulation evaluations of an autonomous urban air mobility network management and separation service, in: 2018 Aviation Technology, Integration, and Operations Conference, 2018, p. 3365.
[47] G. Zhu, P. Wei, Pre-departure planning for urban air mobility flights with dynamic airspace reservation, in: AIAA Aviation 2019 Forum, 2019, p. 3519.
[48] D. P. Thipphavong, R. Apaza, B. Barmore, V. Battiste, B. Burian, Q. Dao, M. Feary, S. Go, K. H. Goodrich, J. Homola, et al., Urban air mobility airspace integration concepts and considerations, in: 2018 Aviation Technology, Integration, and Operations Conference, 2018, p. 3676.
[49] X. Yang, P. Wei, Autonomous on-demand free flight operations in urban air mobility using monte carlo tree search, in: International Conference on Research in Air Transportation

(ICRAT), Barcelona, Spain, 2018.
[50] R. Rothfeld, M. Balac, K. O. Ploetner, C. Antoniou, Initial analysis of urban air mobility's transport performance in sioux falls, in: 2018 Aviation Technology, Integration, and Operations Conference, 2018, p. 2886.
[51] P. D. Vascik, R. J. Hansman, N. S. Dunn, Analysis of urban air mobility operational constraints, J. of Air Transportation 26 (2018) 133–146.
[52] A. Rasheed, S. Gillani, S. Ajmal, A. Qayyum, Vehicular ad hoc network (vanet): A survey, challenges, and applications, in: Vehicular Ad-Hoc Networks for Smart Cities, Springer, 2017, pp. 39–51.
[53] A. Bujari, C. E. Palazzi, D. Ronzani, Fanet application scenarios and mobility models, in: Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications, 2017, pp. 43–46.
[54] W. B. Cotton, Adaptive airborne separation to enable uam autonomy in mixed airspace (2020).

Applying inferential processes to partner selection in large agents communities

Pasquale De Meo[a], Rino Falcone[b] and Alessandro Sapienza[b]

[a]*Department of Ancient and Modern Civilizations (University of Messina), Messina, 98122, Italy*
[b]*Institute of cognitive sciences and technologies (ISTC-CNR), Rome, 00185, Italy*

Abstract

The current literature clearly highlighted the need to define a fast and efficient tool for trust assessment, even in lack of direct information, as much as possessing mechanisms allowing a matching between a selected task and a reliable agent able to carry it out. Direct experience plays a big part, yet it requires a long time to offer a stable and accurate performance and this characteristics may represents a strong drawback especially within huge agents' communities. We support the idea that category-based evaluations and inferential processes represent a useful resource for trust assessment. Within this work, we exploit simulations to investigate how efficient this inferential strategy is, with respect to direct experience, focusing on when and to what extent the first prevails on the latter. Our results suggest that in some situations categories represent a valuable asset, providing even better results.

Keywords

trust, inference, multi-agent systems

1. Introduction and Background

A great deal of effort has been made to assess trust within agents' societies [1][2][3][4]. The great majority of the approaches makes extensive use of direct experience as the main source of information, considering recommendation/reputation and inferential processes just later, as a secondary mechanism to refine trust assessment. Unfortunately, direct experience might not always represent the best solution to assess trustworthiness. For instance, using direct experience within huge networks may become unfeasible, because of their very nature: it is hard and extremely costly to possess enough experience to judge a sufficient number of agents. It is therefore fundamental to find an effective approach for trust assessment even in lack of direct experience [5]. In particular, we argue that category-based evaluations and inferential processes may represent an effective solution. Let us then introduce the formal framework within which we will work.

We consider a population of agents $\mathcal{A} = \{a_1, \ldots, a_n\}$ which can collaborate by reciprocally delegating the execution of some tasks.

We suppose that any agent $a_i \in \mathcal{A}$ needs to achieve a goal g, which identifies a state of the environment (in short, the *world*) which the agent a_i plans to achieve. The agent a_i can reach

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
pdemeo@unime.it (P. D. Meo); rino.falcone@istc.cnr.it (R. Falcone); alessandro.sapienza@istc.cnr.it (A. Sapienza)
0000-0001-7421-216X (P. D. Meo); 0000-0002-0300-458X (R. Falcone); 0000-0001-9360-779X (A. Sapienza)

the goal g by executing a *task* τ, which affects the world. The most interesting case occurs if the agent a_i want/must delegate the execution of τ.

At an abstract level, each agent possesses some skills and resources, defined here as *features* which determine its ability in carrying out the tasks it has to face. Nevertheless, not all the features associated with an agent are crucial for the execution of τ and the majority of these will not even be necessary. If I were to ask someone to cook for me, it would be interesting to know how fast she/he is, how good is her/his culinary imagination or if she/he knows how to cook specific dishes; however, knowing that she/he loves reading astrophysics books would not be of any help.

It is therefore a fundamental precondition that an agent a_i identifies which features are necessary to carry out τ. Then, a_i needs a mental representation of any other a_j, which comprises, at least, the subset of the features which are relevant to execute τ. It is also important to underline that just the possession of these features is not enough, it is also very relevant a_j's willingness (following its motivations) to actually realize τ. Of course, two different tasks, say τ_1 and τ_2, require different features to be efficiently addressed.

Thanks to its mental model, a_i is able to estimate the likelihood $\phi(i, j)$ that a_j will positively bring to completion that specific agreed task, for each agent $a_j \in \mathcal{A}$. The function $\phi(i, j)$ measures the *degree of trust* [6] that a_i (hereafter, the *trustor*) puts in a_j (hereafter the *trustee*), i.e., quantifies to what extent a_i is confident that a_j is capable of successfully executing τ.

It is crucial to point out that the assessment of trust is not only *task-dependent* but also *context-dependent*, because external causes may amplify or downsize the trust between the trustor and trustee. For instance, assume that a_i wants to get to the airport one hour before the departure of her/his flight and suppose that a_i is confident that a_j is able to safely drive and she/he is knowledgeable of obstacles to traffic flow (e.g., limited access roads), and thus, a_i puts a high degree of trust in a_j. However, unforeseen circumstances (e.g., a_j stucks in a traffic jam) may prevent a_i from being at the airport at the scheduled time: such an event negatively influence the trust from a_i to a_j, even if, of course, the liability of a_j is limited.

The procedure to select the agent to which the task τ has to be delegated is thus entirely driven from the calculation of the function $\phi(i, j)$: the trustor should select the agent $a_j^\star$ for which $\phi(i, j)$ achieves its largest value, i.e., $j^\star = \arg\max_j \phi(i, j)$. Such a protocol is, unfortunately, infeasible in real-life applications: in fact, a_i is capable of estimating the trust of those agents – in short E_i – with which it interacted in the past and of which it knows features. In real-life applications, we expect that the size of $\mathcal{A}$ is much larger than that of E_i and, thus, the search of a successful partner is likely to end up in a failure.

An elegant solution to the problem of selecting partners in large agent communities is described in [7, 8] and it relies on the concept of *agent category* or, in short, *category*.

Broadly speaking, a category is a subset of agents in $\mathcal{A}$ such that each category member possesses homogeneous features. Their unique nature makes categories very interesting and particularly useful. Since the members of a category possess similar features, even their performance concerning the same task will be similar. For sure, we have to consider a certain degree of uncertainty, due to the specific peculiarities of the individuals.

The specific categories to take into consideration change with the context and with the task of interest. For instance, suppose that $\mathcal{A}$ correspond to a community of people working in food service with different roles; chefs, waiters, and sommeliers are possible examples of categories in this context.

Because of the existence of categories, the set of agents that the trustor can evaluate significantly expands in size and it consists of the following type of agents:

1. The aforementioned set E_i, which consists of the agents with which a_i has had a direct experience.
2. The set C_i of agents, such that each agent $a_j \in C_i$ belongs to at least one of the categories $CS = \{C_1, C_2, \ldots, C_p\}$; here we suppose that a_i has had a direct experience with at least one agent in each of the categories in CS.
3. The set of agents R_i with which a_i had no direct experience but which have been recommended to a_i by other agents in $\mathcal{A}$ (for instance, on the basis of their reputation).
4. The set of agents RC_i, such that each agent in RC_i belongs to a category which contain at least one agent in R_i.

Advantages arising from the introduction of categories have been extensively studied in past literature [9, 8, 7]: the trustor, in fact, could be able to estimate the performance of any other agent a_j, even if it has never met this agent (and, as observed in [7] without even suspecting its existence), through an *inferential mechanism*.

As the authors of [9] say, it is possible to take advantage of categories just if a few conditions are met. First of all, $\mathcal{A}$ must be partitioned into the categories $\mathcal{C} = \{C_1, C_2, \ldots C_m\}$, classifying the agents according to their features. We assume that this classification is given and accepted by all the agents in $\mathcal{A}$. It must be possible to clearly and unequivocally link a_j to a category c_l. Finally, we must somehow identify the average performance of the category C_l with respect to the task τ: we will discuss in detail in Section 3 a procedure to estimate the performance – called true quality – $\theta_l(\tau)$ of the category C_l for task τ.

When all three of these conditions are met, then the category C_l's evaluation can be used for the agent a_j, concerning the task τ since, by definition of category, all agents in C_l will share the same features of a_j and, thus, if the other agents in C_l are able to successfully execute the task τ (or not), we can reasonably assume that even a_j can do it (or not).

Of course, only some of the categories $C_1, \ldots, C_m$ possess the qualities to successfully execute the task τ while others do not. As a consequence, the first step to perform is to *match* the task τ with a set of categories capable of executing τ.

At a basic level, such a matching could be implemented through a a function $\psi(C_l, \tau)$ which takes a category C_l and a task τ and returns `True` if agents in C_l are capable of executing τ, `False` vice versa. The computation of the function ψ requires an analytical and explicit specification of: *(a)* the chain of actions to perform to execute τ and *(b)* for each action mentioned in *(a)*, the features an agent should possess to perform such an action.

The protocol above easily generalizes to the case in which the trustor has a limited experience (or, in the worst case it has no previous experience): in this case, in fact, the trustor a_i could leverage the sets of agents R_i and RC_i.

2. Related Work

The growing need to deal with bigger and bigger agents' networks makes it difficult to find reliable partners to delegate tasks. It becomes clear that, in such situations, direct experience

[10] is not enough to allow us facing this problem. Going beyond this dimension becomes essential, on the light of the knowledge we already have, identifying models and methodologies able to evaluate our interlocutors and possibly to select adequate partners for the collaborative goals we want to pursue.

Several authors proposed trust propagation as a solution to this topic. Trust propagation [11][12] starts from the assumption that if a_i trusts a_j and a_j trusts a_k, then it is reasonable to assume that a_i can trust a_k to some extent. Exploiting this and other assumptions, this technique allows propagating a trust value from an agent to another one, without requiring a direct interaction. The confusion in the reasoning process here is due to the consideration of a generic trust value for an individual, leaving aside the reason why we trust it: the task we want to delegate to it.

Many articles have discussed the use of categories/stereotypes in trust evaluations [13][14]. This is a very useful instrument, allowing to generalize an individual's evaluation, concerning a specific task to other agents owning similar characteristics. It represents a useful proxy for individuating knowledge about specific trustees [15], elicited in particular in all those circumstances precluding the development of person-based trust [16]. Here the intuition is that, given a specific task τ, the performance of the agent we are evaluating are related to the values of the features it needs to carry out the task itself. Along these lines, it is natural to assume that other individual owning similar values, i.e. belonging to the same category, have the same potential to solve τ.

Pursuant to these considerations, our contribution within this work concerns the investigation of how efficient this inferential strategy is, with respect to direct experience, focusing on when and to what extent the first prevails on the latter.

3. Inferring the quality of categories

In this section we illustrate our procedure to estimate the performance (in short called the *true quality*) $\theta_l(\tau)$ of agents in the category C_l to successfully execute a particular task τ.

Because of the assumptions of our model (illustrated in Section 1), agents belonging to the same category share the same features and, thus, their performances in executing τ are roughly similar; this implies that if an agent $a_j \in C_l$ is able (resp., not able) to execute τ, then we expect that any other agent $a_q \in C_l$ is also able (resp., not able) to execute τ.

In the following, we suppose that agents in C_l are able to execute τ, i.e. in compliance with notation introduced in Section 1, we assume that $\psi(C_l, \tau) = \texttt{True}$. In contrast, if $\psi(C_l, \tau) = \texttt{False}$, it does not make sense to estimate $\theta_l(\tau)$.

The next step of our protocol consists of selecting one of the agents, i.e., the *trustee*, in C_l to which delegate τ; to this purpose, we could select, uniformly at random, one of the agents in C_l, as illustrated in [7]. However, agents are individual entities and, thus, slight differences in their features exist. Because of these differences, an agent (say a_j) may have better (resp., worse) performance than another agent (say, a_q) in executing τ.

In the light of the reasoning above, the true quality $\theta_l(\tau)$ quantifies the *expected performance of an arbitrary trustee in C_l in the execution of τ.*

We assume that $\theta_l(\tau)$ ranges in $(-\infty, +\infty)$: positive (resp., negative) values of $\theta_l(\tau)$ are an indicator of good (resp., bad) performances.

The first step to compute $\theta_l(\tau)$ consists of modelling the performance $f_j(\tau)$ of an arbitrary agent $a_j \in C_l$ in executing τ. To capture uncertainty in the performance of a_j, we represent $f_j(\tau)$ as a Gaussian random variable with mean μ_j and variance σ_j^2.

The assumption that all of the agents in the same category should reach the same performance implies that $\mu_j = \theta_l(\tau)$ for each agent $a_j \in C_l$.

The variance σ_j^2 controls the amount of variability in the performances of the agent a_j: large (resp., small) values in σ_j^2 generate significant (resp., irrelevant) deviations from $\theta_l(\tau)$. In this paper we considered two options for σ_j^2, namely:

1. *Fixed Variance Model*: we suppose that $\sigma_j^2 = \sigma^2$ for each $a_j \in C_l$.
2. *Random Variance Model*: we suppose that σ_j^2 is a uniform random variable in the interval $[\alpha, \beta]$.

Based on these premises, the procedure to estimate $\theta_l(\tau)$ is *iterative* and, at the k-th iteration it works as follows:

a) We select, uniformly at random, an agent, say a_j from C_l
b) We sample the performance $\hat{f}_j(k) \sim f_j(\tau)$ of a_j

Steps a) and b) are repeated N times, being N the number of agents we need to sample before making a decision. In addition, in Step a), agents are sampled with replacement, i.e., an agent could be selected more than once. The algorithm outputs the average value of sampled performances, i.e.:

$$\hat{\theta}_l(\tau) = \frac{1}{N}\sum_{k=1}^{N} \hat{f}_j(k) \tag{1}$$

Our algorithm actually converges to the true value $\theta_l(\tau)$ as stated in the following theorem:

Let N be the number of agents queried by our algorithm and let $\hat{\theta}_l(\tau)$ be the estimation of the true quality $\theta_l(\tau)$ the algorithm returns after N rounds. We have that in both the fixed variance and random variance models $\hat{\theta}_l(\tau)$ converges to $\theta_l(\tau)$ at a rate of convergence of $\frac{1}{\sqrt{N}}$.

Let us first analyze the individual agent performances $f_j(\tau)$ and we are interested in computing the mean and variance of $f_j(\tau)$. If we opt for the Fixed Variance Model, then $f_j(\tau)$ is a Gaussian random variable with mean $\theta_l(\tau)$ and the variance is equal to a constant value σ^2. In contrast, if we are in the Random Variance Model, then the estimation of the mean and the variance of $f_j(\tau)$ can be obtained by law of total mean and the law of total variance [17], which state that for two arbitrary random variables X and Y, the following identities hold true:

$$\mathrm{E}(X) = \mathrm{E}(\mathrm{E}(X \mid Y)) \tag{2}$$

$$\mathrm{Var}(Y) = \mathrm{E}(\mathrm{Var}(Y \mid X)) + \mathrm{Var}(\mathrm{E}(Y \mid X)) \tag{3}$$

We apply Equations 2 and 3 to $X = f_j(\tau)$ and $Y = \sigma$; if we condition on $\sigma = \overline{\sigma}$, then $f_j(\tau)$ is a Gaussian random variable with mean equal to $\theta_l(\tau)$ and variance equal to $\overline{\sigma}$ and therefore:

$$\mathrm{E}(f_j(\tau)) = \mathrm{E}(\mathrm{E}(f_j(\tau) \mid \sigma = \overline{\sigma})) = \mathrm{E}(\theta_l(\tau)) = \theta_l(\tau)$$

In addition,

$$\mathrm{E}(\mathrm{Var}(f_j(\tau) \mid \sigma = \overline{\sigma})) = \mathrm{E}(\sigma) = \frac{\alpha + \beta}{2}$$

and

$$\mathrm{Var}(\mathrm{E}(f_j(\tau) \mid \sigma = \overline{\sigma})) = \mathrm{Var}(\mathrm{E}(\theta_l(\tau))) = \mathrm{Var}(\theta_l(\tau)) = 0$$

which jointly imply

$$\mathrm{Var}(f_j(\tau)) = \frac{\alpha + \beta}{2}$$

As a consequence, independently of the agent a_j, we have that the agent performances $f_j(\tau)$ have the same distribution which we denote as $f(\tau)$. Therefore, in both the Fixed Variance Model and Random Variance Model, the algorithm selects a random sample of agents $Z_1, Z_2, \ldots, Z_N$ of size N in which, for each k such that $1 \leq k \leq N$, Z_k is the average performance of the agent selected at the k-th iteration and it is distributed as $f(\tau)$. The algorithm calculates:

$$S_N = \frac{Z_1 + Z_2 + \ldots + Z_n}{N} \tag{4}$$

Because of the Central Limit Theorem [17], the distribution of S_N gets closer and closer to a Gaussian distribution with mean $\theta_l(\tau)$ as $N \to +\infty$ with a rate of convergence in the order of $\frac{1}{\sqrt{N}}$ and this end the proof.

4. Experimental Analysis

We designed our experiments to answer two main research questions, namely:

$\mathbf{RQ_1}$ What are the benefits arising from the introduction of categories in the selection of a trustee against, for instance, a pure random search or a direct-experience based strategy?

$\mathbf{RQ_2}$ How quickly our algorithm to estimate $\theta_l(\tau)$ converges?

In what follows, we first describe a reference scenario in which our task consists of recruiting a chef from a database of applicants (see Section 4.1). Then, in Sections 4.2 and 4.3, we provide an answer to $\mathbf{RQ_1}$ and $\mathbf{RQ_2}$.

4.1. The reference scenario

We assume that features associated with our task are as follows: *(i) Culinary Education*, measured as the (overall) number of hours spent in training courses with qualified chef trainers, *(ii) Expertise*, i.e., the number of years of professional experience, *(iii) Language Skills*, defined as the number of foreign languages in which the applicant is proficient, *(iv) Culture*, measured on

Table 1
Some tasks associated with the recruitment of a professional chef and their requirements

Task	Culinary Education	Expertise	Language Skills	Culture	Creativity
τ_1	150	5	2	6	7
τ_2	200	4	2	6	6
τ_3	300	4	2	7	6

Table 2
Some features associated with categories C_1- C_5

Category ID.	Culinary Education	Expertise	Language Skills	Culture	Creativity
C_1	250	5	2	8	7
C_2	250	3	2	6	5
C_3	300	3	1	4	5
C_4	100	6	1	4	5
C_5	400	3	1	4	5

a scale from 0 (worse) to 10 (best) and which is understood as the ability of preparing different kind dishes (e.g. fish, meat, vegetarian and so on) in different styles (e.g. Indian, Thai or Italian) and *(v) Creativity*, measured on a scale from 0 (worse) to 10 (best). The list of features is, of course, non-exhaustive. We suppose that each feature is associated with a plausible range: for instance, in Table 1, we consider three potential tasks and the corresponding requirements.

In the following, due to space limitations, we concentrate only on the task τ_1 and we suppose that five categories exist, namely: *Professional Chefs* - C_1, who are trained to master culinary art. Members in C_1 are able to provide creative innovation in menu, preparation and presentation, *Vegan Chefs* - C_2, specialized in the preparation of plant-based dishes, *Pastry Chefs* - C_3, who are capable of creating chocolates and pastries, *Roast Chefs* - C_4, who have expertise in preparing roasted/braised meats and *Fish Chefs* - C_5, who are mainly specialized in the preparation of dish fishes. Each category consists of 100 agents and, thus, the overall number of agents involved in our experiments is 500. Features associated with categories C_1-C_5 are reported in Table 2.

In our scenario, *only agents in* C_1 *are able to fulfill* τ_1; agents in other categories are, for different reasons, unable to execute τ_1: for instance, the expertise of agents in categories C_2, C_3 and C_5 is not sufficient while agents forming categories C_3-C_5 correspond to applicants with a high level of specialization in the preparation of some specific kind of dishes (e.g., fish-based dishes) but they are not sufficiently skilled in the preparation of other type of foods and, thus, agents in these categories showcase an insufficient level of culture.

To simplify discussion we suppose that, through a proper normalization, the performance $f(\tau_1)$ (see Section 3) of an individual agent as well as the true quality $\theta_l(\tau_1)$ of a category C_l (for l = 1 . . . 5) range from 0 to 1. Here, the best performance of an agent can provide (resp., the highest true quality of a category) is 1.

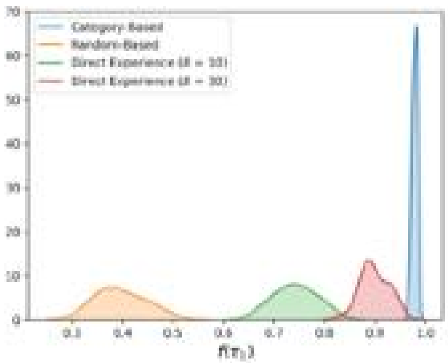


Figure 1: Feedback $f(\tau)$ provided by the trustee. Fixed Variance Model with $\sigma = 0.05$

4.2. A comparison of category-based search with random-based search and direct-experience search

In our first experiment, we compare three strategies to search for a trustee, namely: *a) Random-Based Search*: here, the trustor selects, uniformly at random, a trustee to execute τ_1. The trustor measures the performance $f(\tau_1)$ provided by the trustee in the execution of τ_1 . *b) Category-Based Search*: here, the trustor considers only agents in the most appropriate category (which in our reference scenario coincides with C_1); as suggested in [7], the trustor selects, uniformly at random, one of the agents in C_1 to act as trustee. Once again, the trustor measures the performance $f(\tau_1)$ provided by the trustee in the execution of τ_1. *(c) Direct-Experience Search*: we suppose that the trustor consults up to B agents in the community, being B a fixed integer called *budget*. The trustor records the performance of each consulted agent but it *does not memorize* its category (it may be that the trustor is unable to perceive/understand the trustee's category). At the end of this procedure, the trustor selects as trustee the agent providing the highest performance $f(\tau_1)$ among all consulted agents. The Direct-Experience Search strategy can be regarded as an evolution of the Random-Based Search strategy in which the trustor learns from its past interactions it uses its knowledge to spot the trustee. Here, the budget B regulates the duration of the learning activity the trustor pursues.

In our experimental setting, we considered two values of B, namely $B = 10$ and $B = 30$ and we discuss only results in the Fixed Variance Model with $\sigma = 0.05$ and $\sigma = 0.15$. We applied the Random-Based, the Category-Based and the Direct-Experience Search strategies 20 times; a sketch of the probability density function (pdf) of $f(\tau_1)$ for each strategy is shown in Figure 1 and 2.

As expected, Category-Based Search performs consistently better than the Random-Search one. In addition, the standard deviation of $f(\tau_1)$ in Category-Based Search is much smaller than that observed in the Random-Based strategy and such a behaviour depends on the different degree of matching of categories C_1-C_5 with the task τ_1: in other words, if the trustee is in C_1, the performances it provides are constantly very good; in contrast, in Random-Based Search strategy, the measured performance may significantly fluctuate on the basis of the category to

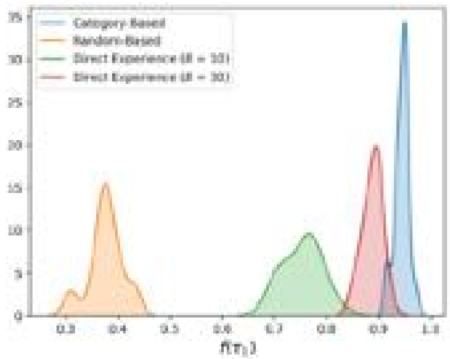


Figure 2: Feedback $f(\tau)$ provided by the trustee. Fixed Variance Model with $\sigma = 0.15$

which the trustee belongs to and this explains oscillations in $f(\tau_1)$.

The analysis of the Direct-Experience Search strategy offers many interesting insights which are valid for both $\sigma = 0.05$ and $\sigma = 0.15$. Firstly, notice that if $B = 10$, then the Direct-Experience Search strategy achieves significantly better performances than the Random-Based strategy, which indicates that an even short learning phase yields tangible benefits. If B increases, the trustor is able to see a larger number of agents before making its decision and, in particular, if B is sufficiently large, then the trustor might encounter the best performing agent $a^{\star}$ in the whole community. In this case, the Direct-Experience strategy would outperform the Category-Based Search strategy: in fact, in the Category-Based strategy, the trustor chooses, uniformly at random, one of the agents in C_1 which provides a performance worse than (or equal to) $a^{\star}$. In short, for large values of B, the Direct-Experience strategy achieves performances which are comparable and, in some cases, even better than those we would obtain in the Category-Based strategy, as shown in Figure 1 and 2. However, the budget B has the meaning of a *cost*, i.e., it is associated with the time the trustor has to wait before it chooses the trustee and, thus, in many practical scenarios, the trustor has to make its decisions as quick as possible.

It is also instructive to consider a further strategy, called *Mixed-Based Search*, which combines the Random-Based Search strategy with the Category-Based Search strategy.

In Mixed-Based Search, we assume the existence of a *warm-up* phase in which the trustor selects the trustee by means of the Random-Based Search strategy; unlike the Direct-Experience Search strategy, the trustor collects not only $f(\tau_1)$ but it also records the category of the trustee. In this way, the trustor is able to identify (after, hopefully, a small number of steps) the category with the highest true quality, i.e., C_1. From that point onward, the trustor switches to a Category-Based strategy and it selects only agents from C_1. From a practical standpoint, we suppose that a performance $f(\tau_1) \geq 0.6$ is classified as an indicator of good performance (in short, *positive signal*); as soon as the trustor has collected 2 positive signals, it makes a decision on the best performing category and it switches to the Category-Based Search strategy.

We are interested at estimating, through simulations, the length ℓ of the warm-up phase, i.e., the number of agents that the trustor has to contact before switching to the Category-Based search strategy. In Figure 3 and 4 we plot the pdf of ℓ.

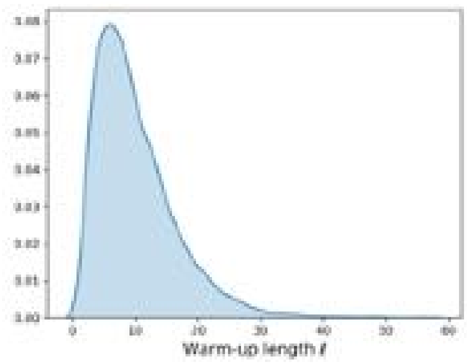


Figure 3: Probability Density Function of the Warm-up length (ℓ) in the Mixed-Search strategy. Fixed Variance Model with $\sigma = 0.05$

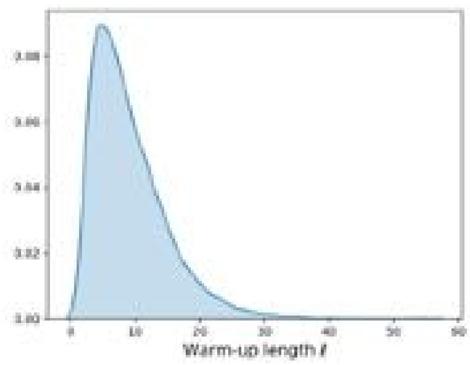


Figure 4: Probability Density Function of the Warm-up length (ℓ) in the Mixed-Search strategy. Fixed Variance Model with $\sigma = 0.15$

Here, the variance σ has a minor impact and we notice that, the pdf achieves its largest value at $\ell \simeq 10$, i.e., 10 iterations are generally sufficient to identify the best performing category.

4.3. The rate of convergence of our algorithm

We conclude our study by investigating how the Fixed Variance Model and the Random Variance model influence the rate at which our algorithm estimates the true quality $\theta_l(\tau)$ of a category.

To make exposition of experimental outcomes simple, we suppose that $\theta_l = 1$ (which models a scenario in which agents in C_1 showcase an exceptionally high ability in executing τ_1).

We considered the Fixed Variance Model with $\sigma \in \{0.05, 0.1, 0.15\}$ and the Random Variance Model in which σ is uniformly distributed in $[0.01, 0.3]$.

We investigated how $\hat{\theta}_l(\tau)$ varied as function of the number N of queried agents; obtained results are reported in Figure 5.

The main conclusions we can draw from our experiment are as follows:

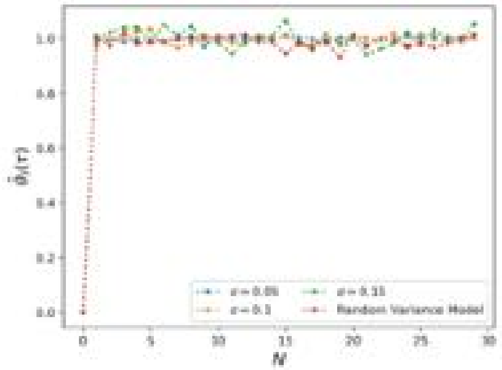


Figure 5: Variation of $\hat{\theta}_l(\tau)$ as function of N.

1. Individual agent variability (modelled through the parameter σ) greatly affects the rate at which $\hat{\theta}_l(\tau)$ converges to $\theta_l(\tau)$. Specifically, Figure 5 suggests that less than 5 iterations are enough to guarantee that $|\hat{\theta}_l(\tau) - \theta_l(\tau)| < 10^{-2}$ if $\sigma = 0.05$. In addition, as σ gets larger and larger, we highlight more and more fluctuations in $\hat{\theta}_l(\tau)$: as an example, if $\sigma = 0.15$ (green line), we highlight the largest fluctuation in $\hat{\theta}_l(\tau)$ and, at a visual inspection, at least $N = 30$ queries are needed to achieve a significant reduction in $|\hat{\theta}_l(\tau) - \theta_l(\tau)|$.
2. An interesting case occurs in the Random Variance Model: in some iterations of the algorithm, agents with a small variability are selected (i.e., we would sample agents with $\sigma \simeq 0.01$) while in other cases agent with a larger variability are selected (here $\sigma \simeq 0.3$). Overall, agents with small variability fully balance agents with high variability and, thus, the algorithm converges to $\theta_l(\tau)$ (red line) generally faster than the case $\sigma = 0.1$ (orange line) and $\sigma = 0.15$ (green line).

5. Conclusions

Although highly populated networks are a particularly useful environment for agents' collaboration, the very nature of these networks may represent a drawback for trust formation, given the lack of data for evaluating the huge number of possible partners. Many contributions in the literature [8, 18] showed that category-based evaluations and inferential processes represent a remarkable solution for trust assessment, since they allow agents to generalize from trust in individuals to trust in their category and vice versa, basing on their observable features. On that note, we cared about stressing the tight relationship between trust and the specific task, target of the trust itself. With the purpose of investigating the role of agents' categories, we considered a simulated scenario, testing in particular the performance of a category-based evaluation, with respect to a random-based search - which it is easily outperformed - and a direct-experience one, showing that, in case of little direct experience, categories grant a better result. Moreover, we proved that, if not available, it is possible to estimate the category's true quality $\theta_l(\tau)$ in a reasonably short amount of time. Future research will attempt to test these findings on a real data set.

References

[1] J. Yan, D. Wu, S. Sanyal, R. Wang, Trust-oriented partner selection in d2d cooperative communications, IEEE Access 5 (2017) 3444–3453.

[2] F. Messina, G. Pappalardo, C. Santoro, D. Rosaci, G. M. Sarné, A multi-agent protocol for service level agreement negotiation in cloud federations, International Journal of Grid and Utility Computing 7 (2016) 101–112.

[3] F. Messina, G. Pappalardo, D. Rosaci, G. M. Sarné, A trust-based, multi-agent architecture supporting inter-cloud vm migration in iaas federations, in: International Conference on Internet and Distributed Computing Systems, Springer, 2014, pp. 74–83.

[4] A. Sapienza, R. Falcone, Evaluating agents' trustworthiness within virtual societies in case of no direct experience, Cognitive Systems Research 64 (2020) 164–173.

[5] S. Abar, G. K. Theodoropoulos, P. Lemarinier, G. M. O'Hare, Agent based modelling and simulation tools: A review of the state-of-art software, Computer Science Review 24 (2017) 13–33.

[6] C. Castelfranchi, R. Falcone, Trust theory: A socio-cognitive and computational model, volume 18, John Wiley & Sons, 2010.

[7] R. Falcone, A. Sapienza, Selecting trustworthy partners by the means of untrustworthy recommenders in digitally empowered societies, in: Proc. of the International Conference on Practical Applications of Agents and Multi-Agent Systems, Springer, Avila, Spain, 2019, pp. 55–65.

[8] R. Falcone, A. Sapienza, C. Castelfranchi, The relevance of categories for trusting information sources, ACM Transactions on Internet Technology (TOIT) 15 (2015) 13.

[9] R. Falcone, M. Piunti, M. Venanzi, C. Castelfranchi, From manifesta to krypta: The relevance of categories for trusting others, ACM Transactions on Intelligent Systems and Technology (TIST) 4 (2013) 27.

[10] C.-W. Hang, Y. Wang, M. P. Singh, Operators for propagating trust and their evaluation in social networks, Technical Report, North Carolina State University. Dept. of Computer Science, 2008.

[11] R. Guha, R. Kumar, P. Raghavan, A. Tomkins, Propagation of trust and distrust, in: Proceedings of the 13th international conference on World Wide Web, 2004, pp. 403–412.

[12] M. Jamali, M. Ester, A matrix factorization technique with trust propagation for recommendation in social networks, in: Proceedings of the fourth ACM conference on Recommender systems, 2010, pp. 135–142.

[13] W. L. Teacy, M. Luck, A. Rogers, N. R. Jennings, An efficient and versatile approach to trust and reputation using hierarchical bayesian modelling, Artificial Intelligence 193 (2012) 149–185.

[14] H. Fang, J. Zhang, M. Sensoy, N. M. Thalmann, A generalized stereotypical trust model, in: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2012, pp. 698–705.

[15] R. M. Kramer, Collective trust within organizations: Conceptual foundations and empirical insights, Corporate Reputation Review 13 (2010) 82–97.
[16] B. D. Adams, R. D. Webb, Trust in small military teams, in: 7th international command and control technology symposium, 2002, pp. 1–20.
[17] D. Bertsekas, J. Tsitsiklis, Introduction to probability, Athena Scientific, 2008.
[18] R. Falcone, A. Sapienza, C. Castelfranchi, Trusting information sources through their categories, in: International Conference on Practical Applications of Agents and Multi-Agent Systems, Springer, 2015, pp. 80–92.

A preliminary experimentation for large scale epidemic forecasting simulations

Gianfranco Lombardo, Agostino Poggi

Dipartimento di Ingegneria e Architettura, Università di Parma - Parma, Italy

Abstract

Agent-based modeling and simulation are some powerful techniques that are widely used with success for analyzing complex and emergent phenomena in many research and application areas. Many different reasons are behind the success of such techniques, among which an important mention goes to the availability of a great variety of software tools, that ease the development of models, as well as the execution of simulations and the analysis of results. This paper presents an actor software library, called ActoDeS, for the development of concurrent and distributed systems, and shows how it can be a suitable mean for building flexible and scalable epidemic forecasting simulations. In particular, the paper presents the first results of the experimentation of ActoDeS for defining a COVID-19 epidemic diffusion model and for supporting the simulation in large populations.

Keywords

actor model, epidemic diffusion model, COVID-19, distributed simulation, HPC

1. Introduction

Agent-based modeling and simulation (ABMS) has been and is widely used with success for studying complex and emergent phenomena in many research and application areas, including agriculture, biomedical analysis, ecology, engineering, sociology, market analysis, artificial life, social studies, and others fields. Such studies are possible thanks to the availability of several tools and libraries that support the development of ABMS applications. Moreover, the availability of large-scale, dynamic, and heterogeneous networks of computational resources and the advent of multi-cores computers allow the development of high performance and scalable computationally intensive ABMS applications. As a matter of fact, such applications are able to manage very large and dynamic models, whose computational needs (in space and time) can be difficult to satisfy by a single machine.

The success and the diffusion of ABMS techniques is also due to the availability of software tools that ease the development of models, the execution of simulations and the analysis of results (see, for example, Mason [1], NetLogo [2], and Repast [3]). However, all the most known and used ABMS tools have been initially designed for the execution of simulations on a single machine and, only in a second step, they were extended for supporting distributed simulations (see, for example, D-Mason [4] and HLA_ACTOR_REPAST [5]). It is worth noting that such

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
gianfranco.lombardo@unipr.it (G. Lombardo); agostino.poggi@unipr.it (A. Poggi)
0000-0003-1808-4487 (G. Lombardo); 0000-0003-3528-0260 (A. Poggi)

extensions show a limitation in terms of reusability. In fact, the code of agents defined for a standalone execution must be modified in order to gain suitable implementations to be used in a distributed simulation.

In this paper we present an actor based software library, called ActoDeS, that provides the suitable features for simplifying the development of scalable and efficient agent based models and simulations. The next section introduces the actor model and discusses the advantages of using actors in ABMS applications. Section 3 introduces ActoDeS. Section 4 shows the advantages of using this software library for ABMS applications. Section 5 presents the COVID-19 diffusion model that is used in the simulations. Section 6 presents the results of the experimentation on the population of Bergamo province. Section 7 introduces the work necessary to extend the simulation to all the population of Lombardia region. Finally, section 8 concludes the paper by discussing its main features and the directions for future work.

2. ABMS with acotrs

Actors [6] are autonomous concurrent objects, which interact with other actors by exchanging asynchronous messages and which, in response of an incoming message, can perform some tasks, namely, sending messages to other actors, creating new actors, and designating a new behavior for processing the messages that the actor will receive.

Actors have the suitable features for defining agent models that can be used in ABMS applications and for modeling the computational agents found in MAS and DAI systems [7]. In fact, actors and computational agents share certain characteristics: i) both react to external stimuli (i.e., they are reactive), ii) both are self-contained, self-regulating, and self-directed, (i.e., they are autonomous and can be goal directed), and iii) both interact through asynchronous messages and such messages are the basis for their coordination and cooperation (i.e., they are social).

Moreover, the use of messages for exchanging state information decouples the code of agents. In fact, agents do not need to access directly to the code of the other agents (i.e., their methods) to get information about them, and so the modification of the code of a type of agent should cause smaller modifications in the code of the other types of agent. Finally, the use of actors simplifies the development of real computational agents in domains where, for example, they need to coordinate themselves or cooperate through direct interactions.

Different researchers propose the use of actors for agent based simulation. For example, Jang and Agha [8] proposed an actor-based software infrastructure, called adaptive actor architecture, to support the construction of large-scale agent-based simulations, by exploiting distributed computing techniques to efficiently distribute agents across a distributed network of computers. In particular, this software infrastructure uses some optimizing techniques in order to reduce the amount of exchanged data among nodes and to support dynamic agent distribution and search.

Cicirelli et al. [5] propose the use of actors for distributing Repast simulations. In particular, they defined a software infrastructure that allows: the migration of agents, a location transparent naming, and an efficient communication. This architecture allows the decomposition of a large system into sub-systems (theatres), each hosting a collection of application actors, that can be allocated on different computational nodes.

Finally, de Berardinis et al. [9] propose an actor based model which allows users to get an idea of the impact of a mobility initiative prior to deployment is a complex task for both urban planners and transport companies.

3. ActoDeS

ActoDeS is an actor-based software framework that has the goal of both simplifying the development of concurrent and distributed complex systems and guarantying an efficient execution of applications [10]. ActoDeS is implemented by using the Java language and is an evolution of CoDE [11] that simplifies the definition of actor behaviors and provides more scalable and performant implementations. Moreover, it takes advantages of some implementation solutions used in JADE [12],[13],[14] for the definition of some internal components. In particular, it has been used for the development of data analysis tools [15] and their use for the analysis of social networks data [16],[17],[18],[19].

ActoDeS has a layered architecture composed of an application and a runtime layer. The application layer provides the software components that an application developer needs to extend or directly use for implementing the specific actors of an application. The runtime layer provides the software components that implement the ActoDeS middleware infrastructures to support the development of standalone and distributed applications.

In ActoDeS an application is based on a set of interacting actors that perform tasks concurrently and interact with each other by exchanging asynchronous messages. Moreover, it can create new actors, update its local state, change its behavior and kill itself.

Depending on the complexity of the application and on the availability of computing and communication resources, one or more actor spaces can manage the actors of the application. An actor space acts as “container” for a set of actors and provides them the services necessary for their execution. An actor space contains a set of actors (application actors) that perform the specific tasks of the current application and two actors (runtime actors) that support the execution of the application actors. These two last actors are called executor and the service provider. The executor manages the concurrent execution of the actors of the actor space. The service provider enables the actors of an application to perform new kinds of action (e.g., to broadcast a message or to move from an actor space to another one).

Communication between actors is buffered: incoming messages are stored in a mailbox until the actor is ready to process them; moreover, an actor can set a timeout for waiting for a new message and then can execute some actions if the timeout fires. Each actor has a system-wide unique identifier called reference that allows it to be reached in a location transparent way independently of the location of the sender (i.e., their location can be the same or different). An actor can send messages only to the actors of which it knows the reference, that is, the actors it created and of which it received the references from other actors. After its creation, an actor can change several times its behavior until it kills itself. Each behavior has the main duty of processing a set of specific messages through a set of message handlers called cases. Therefore, if an unexpected message arrives, then the actor mailbox maintains it until a next behavior will be able to process it.

An actor can be viewed as a logical thread that implements an event loop [20],[21]. This event loop perpetually processes incoming messages. In fact, when an actor receives a message, then it looks for the suitable message handler for its processing and, if it exists, it processes the message. The execution of the message handler is also the means for changing its way of acting. In fact, the actor uses the return value of its message handlers for deciding to remain in the current behavior, to move to a new behavior or to kill itself. Moreover, an actor can set a timeout within receiving a new message and set a message handler for managing the firing of the timeout. This message handler is bound to the reception of the message notifying the firing of the timeout, and so the management of the timeout firing is automatically performed at the reception of such notification message.

ActoDeS supports the configuration of applications with different actor, scheduler ad service provider implementations. The type of the implementation of an actor is one of the factors that mainly influence the attributes of the execution of an application. In particular, actor implementations can be divided in two classes that allow to an actor either to have its own thread (from here named active actors) or to share a single thread with the other actors of the actor space (from here named passive actors). Moreover, the duties of a scheduler depend on the type of the actor implementation. Of course, a scheduler for passive actors is different from a scheduler for active actors, but for the same kind of actor can be useful to have different scheduler implementations. For example, it can allow the implementation of "cooperative" schedulers in which actors can cyclically perform tasks whose duties vary from the processing of the first message in the buffer to the processing of all the messages in it.

The most important decision that influences the quality of the execution of an application is the choice of the actor and scheduler implementations. In fact, the use of one or another couple of actor and scheduler causes large differences in the performance and in the scalability of the applications [22].

4. ABMS with ActoDeS

The features of the actor model and the flexibility of its implementation make ActoDeS suitable for building agent-based modelling and simulation (ABMS) applications and for analyzing the results of the related simulations. In fact, the use of active and passive actors allows the development of applications involving large number of actors, and the availability of different schedulers and the possibility of their specialization allow an efficient execution of simulations in application domains that require different types of scheduling algorithms [23].

In particular, ActoDeS offers a very simple scheduler that may be used in a large set of application domains and, in particular, in ABMS applications. Such a scheduler manages agents implemented as passive actors and its execution repeats until the end of the simulation the following operations:

1. Sends a "step" message to all the agents and increments the value of "step";
2. Performs an execution step of all the agents.

In particular, the reception of a "step" message allows agents to understand that they have all the information (messages) for deciding their actions; therefore, they decide, perform some actions and, at the end, send information to other agents.

4.1. Distribution

The modeling and simulation of complex problems can require the use of large number of agents that may determinate unacceptable simulation times, or the impossibility to run the simulation on a single computational node. In such cases, the availability of distributed algorithms and, of course, of an adequate number of computational nodes can help to perform simulations, partitioning the agents among the available computational nodes. Moreover, depending of kind of application, agents may need to communicate with some agents running on different computational nodes. In this case, the execution of the computational nodes must be synchronized to guaranty the processing of the messages in the correct order.

Therefore, a distributed simulation involves a set of computational nodes whose execution is driven by a set of schedulers and managers. In particular, each manager has the duty of creating the subset of agents of its computational node and synchronize the execution of the simulation with the execution of the other computational nodes. Moreover, one of such managers assumes the role of "master" and has the duties of partitioning the agents involved in the simulation and sends the information necessary for the creation of the agents to the other managers.

In particular, the execution of a distributed simulation can be described by the following steps:

1. Master manager partitions the agents and sends the information for creating a subset of agents to each scheduler (including itself).
2. Managers create all the actors of its subset.
3. Repeat until the end of the simulation:
    - Managers send a synchronization message to the other managers and wait for the corresponding messages from them.
    - Schedulers perform an execution step of all their actors.
    - Scheduler send a "end step" message to all their actors and managers.

4.2. Communication

As introduced above, in different kinds of simulations the agents need to exchange data and often such an interaction is not localized, therefore, the partition of agents on several computational nodes may add an important communication cost. An important solution to reduce the cost of communication is to reduce the frequency of interactions, merging multiple interactions to one. In a conservative distributed simulation system, that synchronizes the simulation step of all the computational nodes involved in the simulation, a possible solution is to group all the messages directed to the agents of a specific computational node into the message that identifies the successive synchronization message.

5. COVID-19 epidemic diffusion model

In the last years, several computational models for the simulation of epidemic outbreaks have been used with increased frequency, moreover, the current COVID-19 pandemic has increase the interest and work on the definition and experimentation of such models [24],[25],[26]. The aim of our work is the definition of a model that allows to identify exactly how many people had actually contracted the COVID-19 virus at the beginning of the pandemic and try to simulate the real evolutionary trend of the spread of the virus that has produced such a large number of people positive for the virus. The first experimentation involved the Bergamo province, but our final goal is to simulate the propagation in the Lombardia region.

The basic idea was to model people based on their social interactions, which can vary from person to person, estimated interactions with a particular rate. In our first model, people were identified on the basis of an interaction rate (high, medium and low) and in a random way these people came into contact, creating a network of interactions.

At the beginning, all the people are in a susceptible phase, in this phase each person can be infected by an infected person. When a person is infected, she/he passes from a susceptibility phase to an incubation phase; here she/he remains there for a certain period of time and then passes to another phase in which he is infectious. When a person is in this phase he could infect other people. Once this last phase is over, the infected changes to be positive. After a certain period of time, the person changes phase: heals or dies. When a person heals, she/he can no longer be infected. Moreover, the incubation phase is made to last from 7 to 14 days, that infectious from 3 to 7 and that of positivity from 14 to 30.

In our model, each person is defined by the following attributes:

- Id of the person
- Province of belonging
- Degree of interaction
- Number of people met
- Current phase
- Id of the people met

This model was used for simulating the COVID-19 propagation from January 30, 2020 to March 31, 2020. Of course, the interaction frequency varies for all the people during the simulation because some people become positive at the end of the phase in which they are infectious; therefore, their interactions should collapse enough sudden because they should stay in quarantine. Moreover, from March 8, 2020, people reduced their interaction frequency because of the lockdown rules.

6. Experimentation and Results

In order to test the model for the Bergamo province, multiple tests were carried out. Thanks to them, it was gradually understood which parameters had to be modified to obtain a trend similar to that of the contagion curve in the real case of Bergamo province. In particular, we considered two "key" dates: March 8th (pre-lockdown date) and March 31st 2020. We focused on the parameter that represents the value of the positive swabs made. In summary, there were 997 positives on March 8 and 8803 positives on March 31. Initially, the parameter that regulates infections has been kept fixed at 0.7, given, found in the literature, which should represent the average value of infection without a mask. The use of such parameter allowed to get good results because the average number of positives obtained by ten simulations was 906. Of course

Table 1
Positives numbers for different infection parameters.

			March 8,2020			
Infection value	0.3	0.4	0.5	0.7	1	Real positives
Positives number	563	674	715	906	1297	997
			March 31,2020			
Infection value	0.3	0.4	0.5	0.7	1	Real positives
Positives number	8307	15127	22025	43304	89274	8803

such parameter value is not good for the lookdown period and so a quite good number of positives was obtained fixing the infection parameter at 0.3. In this case, average number of positives obtained by ten simulations was 8370. Table 1 shows the positives number in the two "key" dates for different infection parameters and compares them with the real number of positives.

7. Scaling to large populations

The final goal of our work is the simulation of epidemic outbreaks in the entire Lombardia. However, the number of people living in Lombardia is more than ten million and so its simulation is feasible only with a distributed implementation. The simplest solution is to partition people on the basis of their province. This solution allows to use different parameters in difference provinces, but implies an unbalanced load among the different computational nodes and it can be a big problem if the simulation should require the synchronization of the execution steps.

Therefore, we decided to combine two kinds of partition: i) people is partitioned on the basis of their province and so each person uses the parameter values associated with her/his province, and ii) people is divided in subset of equivalent size by using the number of computational nodes involved in the simulation. While the management of Lombardia population should be possible without problems, thank to the availability of the computational nodes of the High Performance Computing lab of our university, some problems can be raised by the number of the messages necessary to create consistent sets of contacts in each step of the simulation (i.e., when a person A identifies a person B as her/his contact, then she/he needs to inform person B that she/he in her/his contacts).

Currently, we are evaluating several solutions to reduce the communication overhead, some solutions avoid the exchange of messages, by saving for each step, the data of the people of each computational node on a file and then merging the data of the different files associated with this step, other solutions requires the sending by messages, but at each step a computational node sends a message to all the other computational nodes. Each message contains all the contact information for the people of a specific computational node.

8. Conclusions

This paper presented ActoDeS, an actor software library for the development of concurrent and distributed systems, and shows how it can be a suitable mean for building flexible and scalable epidemic forecasting simulations. In particular, the paper presented the first results of the experimentation of ActoDeS for forecasting COVID-19 epidemic diffusion in the Bergamo province and introduced the first work for defining a scalable distributed system able to forecast COVID-19 epidemic diffusion for all the population of Lombardia region. Our current and future work will be dedicated to complete the design and the implementation of such a distributed system, to experiment different communication solutions for the updating of the contacts of each person, and finally, the definition of more accurate person model that replaces the use of the simple interaction frequency to create new contacts, with the use of the person age and the work sector.

References

[1] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan, Mason: A multiagent simulation environment, Simulation 81 (2005) 517–527.
[2] S. Tisue, U. Wilensky, Netlogo: A simple environment for modeling complexity, in: International conference on complex systems, volume 21, Boston, MA, 2004, pp. 16–21.
[3] M. J. North, N. T. Collier, J. R. Vos, Experiences creating three implementations of the repast agent modeling toolkit, ACM Transactions on Modeling and Computer Simulation (TOMACS) 16 (2006) 1–25.
[4] G. Cordasco, R. De Chiara, A. Mancuso, D. Mazzeo, V. Scarano, C. Spagnuolo, Bringing together efficiency and effectiveness in distributed simulations: the experience with d-mason, Simulation 89 (2013) 1236–1253.
[5] F. Cicirelli, A. Furfaro, A. Giordano, L. Nigro, Hla_actor_repast: An approach to distributing repast models for high-performance simulations, Simulation Modelling Practice and Theory 19 (2011) 283–300.
[6] G. A. Agha, Actors: A model of concurrent computation in distributed systems., Technical Report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab, 1985.
[7] D. Kafura, J.-P. Briot, Actors and agents, IEEE concurrency (1998) 24–28.
[8] M.-W. Jang, G. Agha, Agent framework services to reduce agent communication overhead in large-scale agent-based simulations, Simulation Modelling Practice and Theory 14 (2006) 679–694.
[9] J. de Berardinis, G. Forcina, A. Jafari, M. Sirjani, Actor-based macroscopic modeling and simulation for smart urban planning, Science of Computer Programming 168 (2018) 142–164.
[10] F. Bergenti, E. Iotti, A. Poggi, M. Tomaiuolo, Concurrent and distributed applications with actodes, in: MATEC Web of Conferences, volume 76, EDP Sciences, 2016, p. 04043.
[11] F. Bergenti, A. Poggi, M. Tomaiuolo, An actor based software framework for scalable applications, in: International Conference on Internet and Distributed Computing Systems, Springer, 2014, pp. 26–35.

[12] A. Negri, A. Poggi, M. Tomaiuolo, P. Turci, Dynamic grid tasks composition and distribution through agents, Concurrency and Computation: Practice and Experience 18 (2006) 875–885.
[13] G. Adorni, F. Bergenti, A. Poggi, G. Rimassa, Enabling fipa agents on small devices, in: International Workshop on Cooperative Information Agents, Springer, 2001, pp. 248–257.
[14] A. Poggi, M. Tomaiuolo, G. Vitaglione, A security infrastructure for trust management in multi-agent systems, in: Trusting Agents for Trusting Electronic Societies, Springer, 2004, pp. 162–179.
[15] G. Lombardo, P. Fornacciari, M. Mordonini, M. Tomaiuolo, A. Poggi, A multi-agent architecture for data analysis, Future Internet 11 (2019) 49.
[16] F. Bergenti, E. Franchi, A. Poggi, Agent-based social networks for enterprise collaboration, in: 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE, 2011, pp. 25–28.
[17] G. Lombardo, P. Fornacciari, M. Mordonini, L. Sani, M. Tomaiuolo, A combined approach for the analysis of support groups on facebook-the case of patients of hidradenitis suppurativa, Multimedia Tools and Applications 78 (2019) 3321–3339.
[18] G. Lombardo, A. Ferrari, P. Fornacciari, M. Mordonini, L. Sani, M. Tomaiuolo, Dynamics of emotions and relations in a facebook group of patients with hidradenitis suppurativa, in: International Conference on Smart Objects and Technologies for Social Good, Springer, 2017, pp. 269–278.
[19] M. Tomaiuolo, G. Lombardo, M. Mordonini, S. Cagnoni, A. Poggi, A survey on troll detection, Future Internet 12 (2020) 31.
[20] J. Dedecker, T. Van Cutsem, S. Mostinckx, T. D'Hondt, W. De Meuter, Ambient-oriented programming in ambienttalk, in: European Conference on Object-Oriented Programming, Springer, 2006, pp. 230–254.
[21] M. S. Miller, E. D. Tribble, J. Shapiro, Concurrency among strangers, in: International Symposium on Trustworthy Global Computing, Springer, 2005, pp. 195–229.
[22] A. Poggi, Agent based modeling and simulation with actomos., in: WOA, 2015, pp. 91–96.
[23] P. Mathieu, Y. Secq, et al., Environment updating and agent scheduling policies in agent-based simulators., in: ICAART (2), 2012, pp. 170–175.
[24] M. Ajelli, B. Gonçalves, D. Balcan, V. Colizza, H. Hu, J. J. Ramasco, S. Merler, A. Vespignani, Comparing large-scale computational approaches to epidemic modeling: agent-based versus structured metapopulation models, BMC infectious diseases 10 (2010) 190.
[25] D. Chumachenko, V. Dobriak, M. Mazorchuk, I. Meniailov, K. Bazilevych, On agent-based approach to influenza and acute respiratory virus infection simulation, in: 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET), IEEE, 2018, pp. 192–195.
[26] M. Gatto, E. Bertuzzo, L. Mari, S. Miccoli, L. Carraro, R. Casagrandi, A. Rinaldo, Spread and dynamics of the covid-19 epidemic in italy: Effects of emergency containment measures, Proceedings of the National Academy of Sciences 117 (2020) 10484–10491.

Session 2

Organisations, Norms & Argumentation

A Multiagent Framework for Coordinating Industrial Symbiotic Networks

Vahid Yazdanpanah[a], Devrim M. Yazan[b] and W. Henk M. Zijm[b]

[a]*Agents, Interaction and Complexity (AIC) Research Group, School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, United Kingdom*

[b]*Department Industrial Engineering and Business Information Systems, Faculty of Behavioural, Management and Social Sciences, University of Twente, Enschede, 7500 AE, The Netherlands*

Abstract

We present a formal multiagent framework for coordinating a class of collaborative industrial practices called "*Industrial Symbiotic Networks* (ISNs)" as cooperative games. The game-theoretic formulation of ISNs enables systematic reasoning about what we call the ISN implementation problem. Specifically, the characteristics of ISNs may lead to the inapplicability of standard *fair* and *stable* benefit allocation methods. Inspired by realistic ISN scenarios and following the literature on normative multiagent systems, we consider *regulations* and normative socio-economic *policies* as coordination instruments that in combination with ISN games resolve the situation. In this multiagent system, employing Marginal Contribution Nets (MC-Nets) as rule-based cooperative game representations foster the combination of regulations and ISN games with no loss in expressiveness. We develop algorithmic methods for generating regulations that ensure the implementability of ISNs and as a policy support, present the policy requirements that guarantee the implementability of desired ISNs in a *balanced-budget* way.

Keywords

Multiagent Systems, Agent-Oriented Coordination, Normative Methods, Industrial Symbiosis

1. Introduction

Industrial Symbiotic Networks (ISNs) are circular economic networks of industries with the aim to reduce the use of virgin material by circulating reusable resources (e.g., physical waste material and energy) among the network members [1, 2, 3]. In such networks, symbiosis leads to socio-economic and environmental benefits for involved industrial agents and the society (see [4, 5]). One barrier against stable ISN implementations is the lack of frameworks able to secure such networks against unfair and unstable allocation of obtainable benefits among the involved industrial firms. In other words, although in general ISNs result in the reduction of the total cost, a remaining challenge for operationalization of ISNs is to tailor reasonable mechanisms for allocating the total obtainable cost reductions—in a fair and stable manner—among the contributing firms. Otherwise, even if economic benefits are foreseeable, lack of

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy

V.Yazdanpanah@soton.ac.uk (V. Yazdanpanah); D.M.Yazan@utwente.nl (D. M. Yazan); W.H.M.Zijm@utwente.nl (W. H. M. Zijm)

https://www.ecs.soton.ac.uk/people/vy1y20 (V. Yazdanpanah); https://people.utwente.nl/d.m.yazan (D. M. Yazan); https://www.utwente.nl/en/bms/iebis/staff/former/zijm (W. H. M. Zijm)

0000-0002-4468-6193 (V. Yazdanpanah); 0000-0002-4341-2529 (D. M. Yazan); 0000-0002-4227-2624 (W. H. M. Zijm)

stability and/or fairness may lead to non-cooperative decisions. This will be the main focus of what we call the *industrial symbiosis implementation* problem. Reviewing recent contributions in the field of industrial symbiosis research, we encounter studies focusing on the necessity to consider interrelations between industrial enterprises [6, 2] and the role of contract settings in the process of ISN implementation [7, 8]. We believe that a missed element for shifting from *theoretical* ISN design to *practical* ISN implementation is to model, reason about, and support ISN decision processes in a dynamic way (and not by using case-specific snapshot-based modeling frameworks).

For such a multiagent setting, the mature field of cooperative game theory provides rigorous methodologies and established solution concepts, e.g. the core of the game and the Shapley allocation [9]. However, for ISNs modeled as a cooperative game, these established solution concepts may be either non-feasible (due to properties of the game, e.g. being *unbalanced*) or non-applicable (due to properties that the industrial domain asks for but solution concepts cannot ensure, e.g. individual as well as collective rationality). This calls for contextualized multiagent solutions that take into account both the complexities of ISNs and the characteristics of the employable game-theoretical solution concepts. Accordingly, inspired by realistic ISN scenarios and following the literature on normative multiagent systems [10, 11, 12], we consider *regulative* rules and normative socio-economic *policies* as two elements that in combination with ISN games result in the introduction of the novel concept of *Coordinated* ISNs ($\mathcal{C}-$ISNs). We formally present regulations as monetary incentive rules to enforce desired industrial collaborations with respect to an established policy. Regarding our representational approach, we use Marginal Contribution Nets (MC-Nets) as rule-based cooperative game representations. This simply fosters the combination of regulative rules and ISN games with no loss in expressiveness. Accordingly, applying regulatory rules to ISNs enables ISN policy-makers to transform ISN games and ensure the implementability of desired ones in a fair and stable manner.

For the first time, this work provides a multiagent framework—using MC-net cooperative games—for the implementation phase of ISNs.[1] We develop algorithmic methods for generating regulations that ensure the implementability of ISNs (methodological contribution) and as a policy support, present the ISN policy requirements (practical contribution) that guarantee the implementability of all the desired industrial collaborations in a *balanced-budget* way.

2. ISN as a Multiagent Practice

To explain the dynamics of implementing ISNs as multiagent industrial practices, we use a running example. Imagine three industries i, j, and k in an industrial park such that r_i, r_j, and r_k are among recyclable resources in the three firms' wastes, respectively. Moreover, i, j, and k require r_k, r_i, and r_j as their primary inputs, respectively. In such scenarios, discharging wastes and purchasing traditional primary inputs are transactions that incur cost. Hence, having the chance to reuse a material, firms prefer recycling and transporting reusable resources to other enterprises if such transactions result in obtainable cost reductions for both parties—meaning that it reduces the related costs for discharging wastes (on the resource provider side) and

[1]The foundations of the ideas developed in this article were presented at AAMAS'18 [13] and explored further in [14]. Through the text—to appreciate the space limit—we refer the reader to [14] for details and complete proofs.

purchasing cost (on the resource receiver side). On the other hand, the implementation of such an industrial network involves transportation, treatment, and transaction costs. In principle, aggregating resource treatment processes using refineries, combining transaction costs, and coordinating joint transportation may lead to significant cost reductions at the collective level.

What we call the *industrial symbiosis implementation* problem focuses on challenges—and accordingly seeks solutions—for sharing this *collectively obtainable* benefit among the involved firms. Simply stated, the applied method for distributing the total obtainable benefit among involved agents is crucial while reasoning about implementing an ISN. Imagine a scenario in which symbiotic relations ij, ik, and jk, respectively result in 4, 5, and 4 utility units of benefit, the symbiotic network ijk leads to 6 units of benefit, and each agent can be involved in at most one symbiotic relation. To implement the ijk ISN, one main question is about the method for distributing the benefit value 6 among the three agents such that they all be induced to implement this ISN. For instance, as i and k can obtain 5 utils together, they will defect the ISN ijk if we divide the 6 units of util equally (2 utils to each agent). Note that allocating benefit values lower than their "traditional" benefits—that is obtainable in case firms defect the collaboration—results in unstable ISNs. Moreover, unfair mechanisms that disregard the contribution of firms may cause the firms to move to other ISNs that do so. In brief, even if an ISN results in sufficient cost reductions (at the collective level), its implementation and applied allocation methods determine whether it will be realized and maintained. Our main objective in this work is to provide a multiagent implementation framework for ISNs that enables fair and stable allocation of obtainable benefits. In further sections, we review two standard allocation methods, discuss their applicability for benefit-sharing in ISNs, and introduce our normatively-coordinated multiagent system to guarantee stability and fairness in ISNs.

Regulations as Socio-Economic Norms: In real cases, ISNs take place under regulations that concern environmental as well as societal policies. Hence, industrial agents have to comply to a set of rules. For instance, avoiding waste discharge may be encouraged (i.e., normatively promoted) by the local authority or transporting a specific type of hazardous waste may be forbidden (i.e., normatively prohibited) in a region. Accordingly, to nudge the collective behavior, monetary incentives in the form of subsidies and taxes are well-established solutions. This shows that the ISN implementation problem is not only about decision processes among strategic utility-maximizing industry representatives (at a microeconomic level) but in addition involves regulatory dimensions—such as presence of binding/encouraging monetary incentives (at a macroeconomic level). To capture the regulatory dimension of ISNs, we apply a normative policy that respects the socio-economic as well as environmental desirabilities and categorizes possible coalitions of industries in three classes of: *promoted*, *permitted*, and *prohibited*. Accordingly, the regulatory agent respects this classification and allocates incentives such that industrial agents will be induced to: implement promoted ISNs and avoid prohibited ones (while permitted ISNs are neutral from the policy-maker's point of view). For instance, in our ISN scenario, allocating 10 units of incentive to ijk and 0 to other possible ISNs induces all the rational agents to form the grand coalition and implement ijk—as they cannot benefit more in case they defect. We call the ISNs that take place under regulations, *Coordinated* ISN*s* ($\mathcal{C}-$ISNs). Note that the term "coordination" in this context refers to the application and efficacy of monetary incentive mechanisms in the ISN implementation phase, and should not be confused with ISN administration (i.e., managing the evolution of relations).

Dealing with firms that perform in a complex multiagent industrial context calls for implementation platforms that can be tuned to specific settings, can be scaled for implementing various ISN topologies, do not require industries to sacrifice financially, and allow industries to practice their freedom in the market. We deem that the quality of an ISN implementation framework should be evaluated by (1) *Generality* as the level of flexibility in the sense of independence from agents' internal reasoning processes (i.e., how much the framework adheres to the principle of *separation of concerns*), (2) *Expressivity* as the level of scalability in the sense of independence from size and topology of the network, (3) *Rationality* as the level that the employed allocation mechanisms comply to the collective as well as individual rationality axiom (i.e., the framework should assume that no agent (group) participates in a cooperative practice if they expect higher utility otherwise), and (4) *Autonomy* as the level of allowance (i.e., non-restrictiveness) of the employed coordination mechanisms. Then an ideal framework for implementing ISNs should be general—i.e., it should allow for manipulation in the sense that the network designer does not face any re-engineering/calibration burden—sufficiently expressive, rationally acceptable for all firms, and respect their autonomy. The goal of this paper is to develop a multiagent framework with properties close to the ideal one.

Past Work: The idea of employing cooperative game theory for analysis and implementation of industrial symbiosis have been sparsely explored [15, 16, 17]. In [15], Grimes-Casey et al. used both cooperative and non-cooperative games for analyzing the behavior of firms engaged in a case-specific industrial ecology. While the analysis is expressive, the implemented relations are specific to refillable/disposable bottle life-cycles. In [16], Chew et al. tailored a mechanism for allocating costs among participating agents that expects an involved industry to "bear the extra cost". Although such an approach results in collective benefits, it is not in-line with the individual rationality axiom. In [17], Yazdanpanah and Yazan model bilateral industrial symbiotic relations as cooperative games and show that in such a specific class of symbiotic relations, the total operational costs can be allocated fairly and stably. Our work relaxes the limitation on the number of involved industries and—using the concept of Marginal Contribution Nets (MC-Nets)—enables a representation that is sufficiently expressive to capture the regulatory aspect of ISNs.

3. Game-Theoretic Notions

In this work, we build on the *transferable utility* assumption in game-theoretic multiagent settings. This is to assume that the payoff to a group of agents involved in an ISN (as a cooperative practice) can be freely distributed among the group members, e.g., using monetary transactions.[2]

[2]The presented material on basics in cooperative games is based on [9] while for the MC-Net notations, we build on [18, 19].

Cooperative Games: Multiagent cooperative games with transferable utility are often modeled by the tuple (N, v), where N is the finite set of agents and $v : 2^N \mapsto \mathbb{R}$ is the characteristic function that maps each possible agent group $S \subseteq N$ to a real-valued payoff $v(S)$. In such games, the so-called *allocation problem* focuses on methods to distribute $v(S)$ among all the agents (in S) in a reasonable manner. That is, $v(S)$ is the result of a potential cooperative practice, hence ought to be distributed among agents in S such that they all be induced to cooperate (or remain in the cooperation). Various solution concepts specify the utility each agent receives by taking into account properties like fairness and stability. The two standard solution concepts that characterize fair and stable allocation of benefits are the Shapley value and the Core, respectively.

Shapley Value and Fairness: The Shapley value prescribes a notion of *fairness*. It says that assuming the formation of the grand coalition $N = \{1, \ldots, n\}$, each agent $i \in N$ should receive its average marginal contribution over all possible permutations of the agent groups. Let s and n, represent the cardinality of S and N, respectively. Then, the Shapley value of i under characteristic function v, denoted by $\Phi_i(v)$, is formally specified as $\Phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!}(v(S \cup \{i\}) - v(S))$. For a game (N, v), the unique list of real-valued payoffs $x = (\Phi_1(v), \cdots, \Phi_n(v)) \in \mathbb{R}^n$ is called the Shapley allocation for the game. The Shapley allocation have been extensively studied in the game theory literature and satisfies various desired properties in multiagent practices. Moreover, it can be axiomatized using the following properties:

- *Efficiency (EFF):* the overall available utility $v(N)$ is allocated to the agents in N, i.e., $\sum_{i \in N} \Phi_i(v) = v(N)$;
- *Symmetry (SYM):* any arbitrary agents i and j that make the same contribution receive the same payoff, i.e., $\Phi_i(v) = \Phi_j(v)$;
- *Dummy Player (DUM):* any arbitrary agent i of which its marginal contribution to each group S is the same, receives the payoff that it can earn on its own, i.e., $\Phi_i(v) = v(\{i\})$;
- *Additivity (ADD):* for any two cooperative games (N, v) and (N, w), $\Phi_i(u + w) = \Phi_i(v) + \Phi_i(w)$ for all $i \in N$, where for all $S \subseteq N$, the characteristic function $v + w$ is defined as $(v + w)(S) = v(S) + w(S)$.

In the following, we refer to an allocation that satisfies these properties as a *fair* allocation.

Core and Stability: In core allocations, the focus is on the notion of *stability*. In brief, an allocation is stable if no agent (group) benefits by defecting the cooperation. Formally, for a game (N, v), any list of real-valued payoffs $x \in \mathbb{R}^n$ that satisfies the following conditions is a core allocation for the game:

- *Rationality (RAT):* $\forall S \subseteq N : \sum_{i \in S} x_i \geq v(S)$;
- *Efficiency (EFF):* $\sum_{i \in N} x_i = v(N)$.

One main question is whether for a given game, the core is non-empty (i.e., that there exists a stable allocation for the game). A game for which there exist a non-empty set of stable allocations should satisfy the *balancedness* property, defined as follows. Let $1_S \in \mathbb{R}^n$ be the membership vector of S, where $(1_S)_i = 1$ if $i \in S$ and $(1_S)_i = 0$ otherwise. Moreover, let $(\lambda_S)_{S \subseteq N}$ be a vector of weights $\lambda_S \in [0, 1]$. A vector $(\lambda_S)_{S \subseteq N}$ is a *balanced* vector if for all $i \in N$, we have that $\sum_{S \subseteq N} \lambda_S (1_S)_i = 1$. Finally, a game is *balanced* if for all balanced vectors of weights, we have that $\sum_{S \subseteq N} \lambda_S v(S) \leq v(N)$. According to the Bondereva-Shapley theorem, a game has a non-empty core if and only if it is balanced [20, 21]. In the following, we refer to an allocation that satisfies RAT and EFF as a *stable* allocation.

Marginal Contribution Nets (MC-Nets): Representing cooperative games by their characteristic functions (i.e., specifying values $v(S)$ for all the possible coalitions $S \subseteq N$) may become unfeasible in large-scale applications. In this work, as we are aiming to implement ISNs in a scalable manner, we employ a *basic MC-Net* [18] representation that uses a set of rules to specify the value of possible agent coalitions. Moreover, attempting to capture the regulatory aspect of ISNs makes employing rule-based game representations a natural approach. A basic MC-Net represents the cooperative game among agents in N as a finite set of rules $\{\rho_i : (\mathcal{P}_i, \mathcal{N}_i) \mapsto v_i\}_{i \in K}$, where $\mathcal{P}_i \subseteq N, \mathcal{N}_i \subset N, \mathcal{P}_i \cap \mathcal{N}_i = \emptyset$, $v_i \in \mathbb{R} \setminus \{0\}$, and K is the set of rule indices. For an agent coalition $S \subseteq N$, a rule ρ_i is *applicable* if $\mathcal{P}_i \subseteq S$ and $\mathcal{N}_i \cap S = \emptyset$ (i.e., S contains all the agents in $\mathcal{P}_i$ and no agent in $\mathcal{N}_i$). Let $\Pi(S)$ denote the set of rule indices that are applicable to S. Then the value of S, denoted by $v(S)$, will be equal to $\sum_{i \in \Pi(S)} v_i$. In further sections, we present an MC-Net representation of the ijk ISN scenario and show how this rule-based representation enables applying norm-based coordination to ISNs.

4. ISN Games

As discussed in [7, 17], the total obtainable cost reduction—as the economic benefit—and its allocation among involved firms are key drivers behind the stability of ISNs. For any set of industrial agents S, this total value can be computed based on the total *traditional* cost, denoted by $T(S)$, and the total ISN *operational* cost, denoted by $O(S)$. In brief, $T(S)$ is the summation of all the costs that firms have to pay in case the ISN does not occur (i.e., to discharge wastes and to purchase traditional primary inputs). On the other hand, $O(S)$ is the summation of costs that firms have to pay collectively in case the ISN is realized (i.e., the costs for recycling and treatment, for transporting resources among firms, and finally the transaction costs). Accordingly, for a non-empty finite set of industrial agents S the obtainable symbiotic value $v(S)$ is equal to $T(S) - O(S)$. In this work, we assume a potential ISN, with a positive total obtainable value, and aim for tailoring game-theoretic value allocation and accordingly coordination mechanisms that guarantee a fair and stable implementation of the symbiosis.

Our ijk ISN scenario can be modeled as a cooperative game in which $v(S)$ for any empty/singleton S is 0 and agent groups ij, ik, jk, and ijk have the values 4, 5, 4, and 6, respectively. Note that as the focus of ISNs are on the benefit values obtainable due to potential cost reductions, all the empty and singleton agent groups have a zero value because cost reduction is meaningless in such cases. In the game theory language, the payoffs in ISN games are normalized. Moreover, the game is superadditive in nature.[3] So, given the traditional

[3] Superadditivity implies that forming a symbiotic coalition of industrial agents either results in no value or in a positive value. Implicitly, growth of a group can never result in decrease of the value.

and operational cost values for all the possible agent groups S (i.e., $T(S)$ and $O(S)$) in the non-empty finite set of industrial agents N, the ISN among agents in N can be formally modeled as follows.

Definition 1 (ISN Games). *Let N be a non-empty finite set of industrial agents. Moreover, for any agent group $S \subseteq N$, let $T(S)$ and $O(S)$ respectively denote the total traditional and operational costs for S. We say the* ISN *among industrial agents in N is a normalized superadditive cooperative game (N, v) where $v(S)$ is 0 if $|S| \leq 1$ and equal to $T(S) - O(S)$ otherwise.*

According to the following proposition, basic MC-Nets can be used to represent ISNs. MC-Net representations aid combining ISN games with normative coordination rules.

Proposition 1 (ISNs as MC-Nets). *Any* ISN *can be represented as a basic MC-Net.*

Proof. To prove, we do not rely on the expressivity of MC-Nets but provide a constructive proof that respects the context of industrial symbiosis. See [14] for the complete proof. □

Example 1. Our running example can be represented by the basic MC-Net[4] $\{\rho_1 : (ij, k) \mapsto 4, \rho_2 : (ik, j) \mapsto 5, \rho_3 : (jk, i) \mapsto 4, \rho_4 : (ijk, \emptyset) \mapsto 6\}$.

As discussed earlier, how firms share the obtainable ISN benefits plays a key role in the process of ISN implementation, mainly due to stability and fairness concerns. Industrial firms are economically rational entities that defect non-beneficial relations (instability) and mostly tend to reject ISN proposals in which benefits are not shared with respect to their contribution (unfairness). In this work, we focus on Core and Shapley allocation mechanisms as two standard methods that characterize stability and fairness in cooperative games, receptively. We show that these concepts are applicable in a specific class of ISNs but are not generally scalable for value allocation in the implementation phase of ISNs. This motivates introducing incentive mechanisms to guarantee the implementability of "desired" ISNs.

Two-Person Industrial Symbiosis Games: When the game is between two industrial firms (i.e., a bilateral relation between a resource receiver/provider couple), it has additional properties that result in applicability of both Core and Shapley allocations. We denote the class of such ISN games by ISN_Λ. This is, $\mathsf{ISN}_\Lambda = \{(N, v) : (N, v)$ is an ISN game and $|N| = 2\}$. Moreover, the ISN games in which three or more agents are involved will form ISN_Δ. The class of ISN_Λ games corresponds to the so called ISR games in [17]. The difference is on the value allocation perspective as in [17], they assume the elimination of traditional costs (thanks to implementation of the symbiotic relation) and focus on the allocation of operational costs; while we focus on the allocation of the total benefit, obtainable due to potential cost reductions.

Lemma 1 (ISN_Λ Balancedness). *Let (N, v) be an arbitrary ISN_Λ game. It always holds that (N, v) is balanced.*

Proof. See [14] for the complete proof. □

[4] For notational simplicity, we avoid brackets around agent groups, e.g., we write ij instead of $\{i, j\}$.

Relying on Lemma 1, we have the following result that focuses on the class of ISN_Λ relations and shows the applicability of two standard game-theoretic solution concepts for implementing fair and stable industrial symbiotic networks.

Theorem 1. *Let (N, v) be an arbitrary ISN_Λ game. The symbiotic relation among industrial agents in N is implementable in a unique stable and fair manner.*

Proof. See [14] for the complete proof. □

ISN Games: Here, we focus on ISN_Δ games as the class of ISN games with three or more participants and discuss the applicability of the two above mentioned allocation mechanisms for implementing such industrial games.

Example 2. Recall the ijk ISN_Δ scenario from Section 2. To have a stable allocation (x_i, x_j, x_k) in the core, the EFF condition implies $x_i + x_j + x_k = 6$ while the RAT condition implies $x_i + x_j \geq 4 \wedge x_i + x_k \geq 5 \wedge x_j + x_k \geq 4$. As these conditions cannot be satisfied simultaneously, we can conclude that the core is empty and there exists no way to implement this ISN in a stable manner. Moreover, although the Shapley allocation provides a fair allocation $(13/6, 10/6, 13/6)$, it is not rational for firms to implement the ISN. E.g., i and k obtain $30/6$ in case they defect while according to the Shapley allocation, they ought to sacrifice as they collectively have $26/6$.

As illustrated in this example, the Core of ISN_Δ games may be empty which implies the inapplicability of this solution concept as a general method for implementing ISNs.

Theorem 2. *Let (N, v) be an arbitrary ISN_Δ game. The symbiotic relation among industrial agents in N is not generally implementable in a stable manner.*

Proof. See [14] for the complete proof. □

Note that the fair implementation of ISN_Δ games is not always in compliance with the rationality condition. This theorem—in accordance with the intuition presented in example 2—shows that we lack general methods that guarantee stability and fairness of ISN implementations. So, even if an industrial symbiotic practice could result in *collective* economic and environmental benefits, it may not last due to instable or unfair implementations. One natural response which is in-line with realistic ISN practices is to employ monetary incentives as a means of coordination.

5. Coordinated ISN

In realistic ISNs, the symbiotic practice takes place in the presence of economic, social, and environmental *policies* and under *regulations* that aim to enforce the policies by nudging the behavior of agents towards desired ones. In other words, while the policies generally indicate whether an ISN is "good (bad, or neutral)", the regulations are a set of norms that—in case of agents' compliance—result in a spectrum of acceptable (collective) behaviors. Note that the acceptability, i.e., goodness, is evaluated and ought to be verified from the point of view of the policy-makers as community representatives. In this section, we follow this normative approach and aim at using normative coordination to guarantee the implementability of desirable ISNs in a stable and fair manner[5].

Normative Coordination of ISNs: Following [10, 11], we see that during the process of ISN implementation as a game, norms can be employed as game transformations, i.e., as "ways of transforming existing games in order to bring about outcomes that are more desirable from a welfaristic point of view"[11]. For this account, given the economic, environmental, and social dimensions and with respect to potential socio-economic consequences, industrial symbiotic networks can be partitioned in three classes, namely *promoted*, *permitted*, and *prohibited* ISNs. Such a classification can be modeled by a normative socio-economic policy function $\wp : 2^N \mapsto \{p^+, p^\circ, p^-\}$, where N is the finite set of industrial firms. Moreover, p^+, p°, and p^- are labels—assigned by a third-party authority—indicating that the ISN among any given agent group is either promoted, permitted, or prohibited, respectively. The three sets $P^+_\wp$, $P^\circ_\wp$, and $P^-_\wp$ consist of all the $\wp$-promoted, -permitted, and -prohibited agent groups, respectively. Formally $P^+_\wp = \{S \subseteq N : \wp(S) = p^+\}$ ($P^\circ_\wp$ and $P^-_\wp$ can be formulated analogously). Note that $\wp$ is independent of the ISN game among agents in S, its economic figures, and corresponding cost values—in general, it is independent of the value function of the game. E.g., a symbiotic relation may be labeled with p^- by policy $\wp$—as it is focused on exchanging a hazardous waste—even if it results in a high level of obtainable benefit.

Example 3. In our ijk ISN scenario, imagine a policy $\wp_1$ that assigns p^- to all the singleton and two-member groups (e.g., because they discharge hazardous wastes in case they operate in one- or two-member groups) and p^+ to the grand coalition (e.g., because in that case they have zero waste discharge). So, according to $\wp_1$, the ISN among all the three agents is "desirable" while other possible coalitions lead to "undesirable" ISNs.

As illustrated in Example 3, any socio-economic policy function merely indicates the desirability of a potential ISN among a given group of agents and is silent with respect to methods for *enforcing* the implementability of promoted or unimplementability of prohibited ISNs. Note that ISN_Λ games are always implementable. So, ISNs' implementability refers to the general class of ISN games including ISN_Δ games.

The rationale behind introducing socio-economic policies for ISNs is to make sure that promoted ISNs are implementable in a fair and stable manner while prohibited ones are instable. In real ISN practices, the regulatory agent (i.e., the regional or national government) introduces regulations—to support the policy—in the form of monetary incentives[6]. This is to ascribe subsidies to promoted and taxes to prohibited collaborations (see [24] for an implementation theory approach on mechanisms that employ monetary incentives to achieve desirable resource allocations). We follow this practice and employ a set of rules to ensure/avoid the implementability of desired/undesired ISNs among industrial agents in N via allocating incentives. Incentive rules can be represented by an MC-Net $\Re = \{\rho_i : (\mathcal{P}_i, \mathcal{N}_i) \mapsto \iota_i\}_{i \in K}$ in which K is the set of rule indices. Let $\Im(S)$ denote the set of rule indices that are applicable to $S \subseteq N$. Then, the incentive value for S, denoted by $\iota(S)$, is defined as $\sum_{i \in \Im(S)} \iota_i$. This is, a set of incentive rules

[5]In the following, we simply say *implementability* of ISNs instead of *implementability in a fair and stable* manner.
[6]See [22, 23] for similar approaches on incentivizing cooperative agent systems.

can be represented also as a cooperative game $\Re = (N, \iota)$ among agents in N. In the following proposition, we show that for any ISN game there exists a set of incentive rules to guarantee the implementability of the ISN in question.

Proposition 2 (Implementability Ensuring Rules). *Let G be an arbitrary* ISN *game among industrial agents in N. There exists a set of incentive rules to guarantee the implementability of G.*

Proof. Recall that according to Proposition 1, G can be represented as an MC-Net. To prove the claim, we provide Algorithm 1 that takes the MC-Net representation of G as the input and generates a set of rules that guarantee the implementability of G.

Algorithm 1 Generating rule set $\Re$ for ISN game G.

Data: ISN game $G = \{\rho_i : (\mathcal{P}_i, \mathcal{N}_i) \mapsto v_i\}_{i \in K}$ among agents in N; K the set of rule indices for G.
Result: Incentive rule set $\Re$ for G.
$n \leftarrow length(K)$ and $\Re = \{\}$
for $i \leftarrow 1$ **to** n **do**
  if $i \in \Pi(N)$ **then**
    $\Re \leftarrow \Re \cup \{\rho_i : (\mathcal{P}_i, \mathcal{N}_i) \mapsto 0\}$
  else
    $\Re \leftarrow \Re \cup \{\rho_i : (\mathcal{P}_i, \mathcal{N}_i) \mapsto -v_i\}$
  end if
end for

By allocating $-v_i$ to rules that are not applicable to N, any coalition other than the grand coalition will be faced with a tax value. As the original game is superadditive, the agents will have a rational incentive to cooperate in N and the ISN is implementable in a stable manner thanks to the provided incentive rules. □

Till now, we introduced socio-economic policies and regulations as required (but not yet integrated) elements for modeling coordinated ISNs. In the following section, we combine the idea behind incentive regulations and normative socio-economic policies to introduce the concept of *Coordinated* ISN*s* ($\mathcal{C}-$ISNs) as a multiagent system for implementing industrial symbiosis.

Coordinated ISNs: As discussed above, ISN games can be combined with a set of regulatory rules that allocate incentives to agent groups (in the form of subsidies and taxes). We call this class of games, ISNs in presence of coordination mechanisms, or *Coordinated* ISN*s* ($\mathcal{C}-$ISNs) in brief.

Definition 2 (Coordinated ISN Games ($\mathcal{C}-$ISN)). *Let G be an* ISN *and $\Re$ be a set of regulatory incentive rules, both as MC-Nets among industrial agents in N. Moreover, for each agent group $S \subseteq N$, let $v(S)$ and $\iota(S)$ denote the value of S in G and the incentive value of S in $\Re$, respectively. We say the Coordinated* ISN *Game ($\mathcal{C}-$ISN) among industrial agents in N is a cooperative game (N, c) where for each agent group S, we have that $c(S) = v(S) + \iota(S)$.*

Note that as both the ISN game G and the set of regulatory incentive rules $\Re$ are MC-Nets among industrial agents in N, then for each agent group $S \subseteq N$ we have that $c(S)$ is equal to the summation of all the applicable rules to S in both G and $\Re$. Formally, $c(S) = \sum_{i\in\Pi(S)} v_i + \sum_{j\in\Im(S)} \iota_j$ where $\Pi(S)$ and $\Im(S)$ denote the set of applicable rules to S in G and $\Re$, respectively. Moreover, v_i and ι_j denote the value of applicable rules i and j in $\Pi(S)$ and $\Im(S)$, respectively. We sometime use $G+\Re$ to denote the game C as the result of incentivizing G with $\Re$. The next result shows the role of regulatory rules in the enforcement of socio-economic policies.

Proposition 3 (Policy Enforcing Rules). *For any promoted* ISN *game G under policy $\wp$, there exist an implementable $\mathcal{C}-$*ISN *game C.*

Proof. See [14] for the complete proof. □

Analogously, similar properties hold for *avoiding* prohibited ISNs or *allowing* permitted ones. Avoiding prohibited ISNs can be achieved by making the $\mathcal{C}-$ISN (that results from introducing regulatory incentives) unimplementable. On the other hand, allowing permitted ISNs would be simply the result of adding an empty set of regulatory rules. The presented approach for incentivizing ISNs, is advisable when the policy-maker is aiming to ensure the implementability of a promoted ISN in an ad-hoc way. In other words, an $\Re$ that ensures the implementability of a promoted ISN G_1 may ruin the implementability of another promoted ISN G_2. This highlights the importance of some structural properties for socio-economic policies that aim to foster the implementability of desired ISNs. As we discussed in Section 2, we aim for implementing ISNs such that the rationality axiom will be respected. In the following, we focus on the subtleties of socio-economic policies that are enforced by regulatory rules. The question is, what are the requirements of a policy that can ensure the rationality of staying in desired ISNs? We first show that to respect the rationality axiom, promoted agent groups should be disjoint. We illustrate that in case the policy-maker takes this condition into account, industrial agents have no economic incentive to defect an implementable promoted ISN.

Proposition 4. *Let G_1 and G_2 be arbitrary* ISN*s, respectively among promoted (nonempty) agent groups S_1 and S_2 under policy $\wp$ (i.e., $S_1, S_2 \in P^+_\wp$). Moreover, let $\Re_1$ and $\Re_2$ be rule sets that ensure the implementability of G_1 and G_2, respectively. For $i \in \{1, 2\}$, defecting from $\mathcal{C}-$*ISN *$C_i = G_i + \Re_i$ is not economically rational for any agent $a \in S_i$ iff $S_1 \cap S_2 = \emptyset$.*

Proof. See [14] for the complete proof. □

Accordingly, given a set of industrial agents in N and a socio-economic policy $\wp$ we directly have that:

Proposition 5. *For $n = |P^+_\wp|$ if $\bigcap_{i=1}^{n} S_i \in P^+_\wp = \emptyset$ then any arbitrary $S_i \in P^+_\wp$ is minimal (i.e., $S'_i \notin P^+_\wp$ for any $S'_i \subset S_i$).*

Roughly speaking, the exclusivity condition for promoted agent groups entails that any agent is in at most one promoted group. Hence, deviation of agents does not lead to a larger promoted group as no promoted group is part of a promoted super-group, or contains a promoted sub-group. In the following, we show that the mutual exclusivity condition is sufficient for ensuring the implementability of all the ISNs that take place among promoted groups of firms.

Theorem 3. *Let G be an arbitrary* ISN_Δ *game under policy $\wp$ among industrial agents in N and n be the cardinality of $P^+_\wp$. If $\bigcap_{i=1}^n S_i \in P^+_\wp = \emptyset$, then there exists a set of regulatory rules $\Re$, such that all the promoted symbiotic networks are implementable in the coordinated* ISN *defined by $C = G + \Re$. Moreover, any* ISN *among prohibited agent groups in $P^-_\wp$ will be unimplementable.*

Proof. To prove, we provide a method to generate such an implementability ensuring set of rules. We start with an empty $\Re$. Then for all n promoted $S_i \in P^+_\wp$, we call Algorithm 1. Each single run of this algorithm results in a $\Re_i$ that guarantees the implementability of the industrial symbiosis among the set of firms in the promoted group S_i. As the set of promoted agent groups comply to the mutual exclusivity condition, the unification of all the regulatory rules results in a general $\Re$. Formally, $\Re = \bigcup_{i=1}^n \Re_i$. Moreover, as the algorithm applies taxation on non-promoted groups, no ISN among prohibited agent groups will be implementable. □

Example 4. Recalling the ISN scenario in Example 3, the only promoted group is the grand coalition while other possible agent groups are prohibited. To ensure the implementability of the unique promoted group and to avoid the implementability of other groups, the result of executing our algorithm is $\Re = \{\rho_1 : (ij, k) \mapsto -4, \rho_2 : (ik, j) \mapsto -5, \rho_3 : (jk, i) \mapsto -4\}$. In the $\mathcal{C}-$ISN that results from adding $\Re$ to the original ISN, industrial symbiosis among firms in the promoted group is implementable while all the prohibited groups cannot implement a stable symbiosis.

Realized ISNs and Budget-Balancedness: As we mentioned in the beginning of Section 5, regulations are norms that in case of agents' compliance bring about the desired behavior. For instance, in Example 4, although according to the provided tax-based rules, defecting the grand coalition is not economically rational, it is probable that agents act irrationally—e.g., due to trust-/reputation-related issues—and go out of the promoted group. This results in possible normative behavior of a $\mathcal{C}-$ISN with respect to an established policy $\wp$. So, assuming that based on evidences the set of implemented ISNs are realizable, we have the following abstract definition of $\mathcal{C}-$ISN's normative behavior under a socio-economic policy.

Definition 3 ($\mathcal{C}-$ISN's Normative Behavior). *Let C be a $\mathcal{C}-$*ISN *among industrial agents in N under policy $\wp$ and let E be the evidence set that includes all the implemented* ISN*s among agents in N. We say the behavior of C complies to $\wp$ according to E iff $E = P^+_\wp$; and violates it otherwise.*

Given an ISN under a policy, we introduced a set of regulatory rules to ensure that all the promoted ISNs will be implementable. However, although providing incentives makes them implementable, the autonomy of industrial agents may result in situations that not all the promoted agent groups implement their ISN. So, although we can ensure the implementability of all the promoted ISNs, the real behavior may deviate from a desired one. As our introduced method for guaranteeing the implementability of ISNs among promoted agent groups is mainly tax-based, if a $\mathcal{C}-$ISN violates the policy, we end up with collectible tax values. In such cases, our tax-based method can become a *balanced-budget* monetary incentive mechanism (as discussed in [25, 26, 27]) by employing a form of "Robin-Hood" principle and redistributing the collected amount among promoted agent groups that implemented their ISN. In the following, we provide an algorithm that guarantees budget-balancedness by means of a Shapley-based redistribution of the collectible tax value among agents that implemented promoted ISNs. We establish the correctness of Algorithm 2 in Proposition 6.

Algorithm 2 Tax Redistribution for $\mathcal{C}-$ISN game C.

Data: $C = G + \Re$ the $\mathcal{C}-$ISN game among industrial agents in N under policy $\wp$ such that all the ISNs among promoted groups in $P^+_\wp$ are implementable; E the set of implemented ISNs; The collectible tax value τ.
Result: $\Omega_i(C, \wp)$ the distributable incentive value to $i \in N$.
$S^+ \leftarrow E \cap P^+_\wp$, $S^+_u \leftarrow \bigcup_{S \in S^+} S$
for all $i \in (S^+_u, v)$ the sub-game of G **do**
    $k \leftarrow \Phi_i(v)$ the Shapley value of i in (S^+_u, v)
    $\Omega_i(C, \wp) = (1/v(S^+_u)).\tau.k$
end for

We establish the correctness of Algorithm 2 in Proposition 6.

Proposition 6. *Let $C = G + \Re$ be a $\mathcal{C}-$ISN among industrial agents in N under policy $\wp$ such that all the ISNs among promoted groups are implementable (using the provided method in Theorem 3) and let E be the set of implemented ISNs. For any $\mathcal{C}-$ISN, the incentive values returned by Algorithm 2 ensures budget balancedness while preserving fairness (i.e., EFF, SYM, DUM, and ADD).*

Proof. See [14] for the complete proof. □

Note that the redistribution phase takes place after the implementation of the ISNs and with respect to the evidence set E. Otherwise, there will be cases in which the redistribution process provides incentives for agent groups to defect the set of promoted collaborations. Moreover, we highlight that the use of an MC-net enables calculating the Shapley value in a scalable manner (see [18] for complexity results).

6. Conclusions and Future Work

This paper provides a coordinated multiagent system—rooted in cooperative game theory—for implementing ISNs that take place under a socio-economic policy. The use of MC-Nets enables combining the game with the set of policies and regulations in a natural way. The paper also provides algorithms that generate regulatory rules to ensure the implementability of "good" symbiotic collaborations in the eye of the policy-maker. This extends previous work that merely focused on operational aspects of industrial symbiotic relations–as we introduce the analytical study of the regulatory aspect of ISNs. Finally, it introduces a method for redistribution of collectible tax values. The presented method ensures the budget-balancedness of the monetary incentive mechanism for coordination of ISNs in the implementation phase.

In practice, such a framework supports decision-makers in the ISN implementation phase by providing operational tools for reasoning about the implementability of a given ISN in a fair and stable manner. Moreover, it supports policy-makers aiming to foster socio-economically desirable ISNs by providing algorithms that generate the required regulatory rules.

This paper focuses on a unique socio-economic policy and a set of rules to ensure it. One question that deserves investigation is the possibility of having multiple policies and thus analytical tools for policy option analysis (e.g., using [28, 29]) in ISNs. Such a framework assists ranking and investigating the applicability of a set of policies in a particular ISN scenario.

We also aim to focus on the administration of ISNs. Then, modeling the compliance of involved agents to their commitments and capturing trust dynamics [30, 31] will be the main concerns for automated trading in industrial symbiosis systems. For that, we plan to model ISNs as normative organizations [32, 33] and investigate how responsibility reasoning and norm-aware coordination [34, 35, 36] can ensure the robustness and reliability of such organizations.

Acknowledgments

The authors would like to thank the anonymous referees and the conference participants for their valuable comments and helpful suggestions.

SHAREBOX [37], the project leading to this work, has received funding from the European Union's *Horizon 2020* research and innovation programme under grant agreement No. 680843.

References

[1] M. R. Chertow, Industrial symbiosis: literature and taxonomy, Annual review of energy and the environment 25 (2000) 313–337.
[2] V. Yazdanpanah, D. M. Yazan, W. H. M. Zijm, FISOF: A formal industrial symbiosis opportunity filtering method, Engineering Applications of Artificial Intelligence 81 (2019) 247–259.
[3] J. Kirchherr, D. Reike, M. Hekkert, Conceptualizing the circular economy: An analysis of 114 definitions, Resources, conservation and recycling 127 (2017) 221–232.
[4] G. D'Inverno, L. Carosi, G. Romano, Environmental sustainability and service quality beyond economic and financial indicators: A performance evaluation of italian water utilities, Socio-Economic Planning Sciences (2020) 100852.
[5] N. Shen, H. Peng, Can industrial agglomeration achieve the emission-reduction effect?, Socio-Economic Planning Sciences (2020) 100867.
[6] D. M. Yazan, V. A. Romano, V. Albino, The design of industrial symbiosis: An input–output approach, Journal of cleaner production 129 (2016) 537–547.
[7] V. Albino, L. Fraccascia, I. Giannoccaro, Exploring the role of contracts to support the emergence of self-organized industrial symbiosis networks: an agent-based simulation study, Journal of Cleaner Production 112 (2016) 4353–4366.
[8] D. M. Yazan, V. Yazdanpanah, L. Fraccascia, Learning strategic cooperative behavior in industrial symbiosis: A game-theoretic approach integrated with agent-based simulation, Business Strategy and the Environment 29 (2020) 2078–2091.
[9] A. Mas-Colell, M. D. Whinston, J. R. Green, et al., Microeconomic theory, volume 1, Oxford university press New York, 1995.

[10] Y. Shoham, M. Tennenholtz, On social laws for artificial agent societies: off-line design, Artificial intelligence 73 (1995) 231–252.
[11] D. Grossi, L. Tummolini, P. Turrini, Norms in game theory, in: Agreement Technologies, Springer, 2013, pp. 191–197.
[12] G. Andrighetto, G. Governatori, P. Noriega, L. W. van der Torre, Normative multi-agent systems, volume 4, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.
[13] V. Yazdanpanah, D. M. Yazan, H. Zijm, Industrial symbiotic networks as coordinated games, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18, IFAAMAS, 2018, pp. 2145–2147.
[14] V. Yazdanpanah, D. M. Yazan, W. H. Zijm, Coordinating multiagent industrial symbiosis, CoRR abs/2006.01784 (2020).
[15] H. G. Grimes-Casey, T. P. Seager, T. L. Theis, S. E. Powers, A game theory framework for cooperative management of refillable and disposable bottle lifecycles, Journal of Cleaner Production 15 (2007) 1618–1627.
[16] I. M. L. Chew, R. R. Tan, D. C. Y. Foo, A. S. F. Chiu, Game theory approach to the analysis of inter-plant water integration in an eco-industrial park, Journal of Cleaner Production 17 (2009) 1611–1619.
[17] V. Yazdanpanah, D. M. Yazan, Industrial symbiotic relations as cooperative games, in: International Conference on Industrial Engineering and Systems Management, International Institute for Innovation, Industrial Engineering and Entrepreneurship, 2017, pp. 455–460.
[18] S. Ieong, Y. Shoham, Marginal contribution nets: a compact representation scheme for coalitional games, in: Proceedings of the 6th ACM conference on Electronic commerce, ACM, 2005, pp. 193–202.
[19] J. Lesca, P. Perny, M. Yokoo, Coalition structure generation and cs-core: Results on the tractability frontier for games represented by mc-nets, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, IFAAMAS, 2017, pp. 308–316.
[20] L. S. Shapley, On balanced sets and cores, Naval research logistics quarterly 14 (1967) 453–460.
[21] O. N. Bondareva, Some applications of linear programming methods to the theory of cooperative games, Problemy kibernetiki 10 (1963) 119–139.
[22] R. Meir, J. S. Rosenschein, E. Malizia, Subsidies, stability, and restricted cooperation in coalitional games, in: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 2011, pp. 301–306.
[23] Y. Zick, M. Polukarov, N. R. Jennings, Taxation and stability in cooperative games, in: International conference on Autonomous Agents and Multi-Agent Systems, 2013, pp. 523–530.
[24] A. Kakhbod, Resource allocation in decentralized systems with strategic agents: an implementation theory approach, Springer Science & Business Media, 2013.
[25] M. Guo, V. Conitzer, Optimal-in-expectation redistribution mechanisms, in: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2, IFAAMAS, 2008, pp. 1047–1054.
[26] C. Li, U. Rajan, S. Chawla, K. Sycara, Mechanisms for coalition formation and cost sharing in an electronic marketplace, in: international conference on Electronic commerce, ACM, 2003, pp. 68–77.

[27] S. Phelps, P. McBurney, S. Parsons, Evolutionary mechanism design: a review, Autonomous Agents and Multi-Agent Systems 21 (2010) 237–264.
[28] S. Mehryar, R. Sliuzas, N. Schwarz, A. Sharifi, M. van Maarseveen, From individual fuzzy cognitive maps to agent based models: Modeling multi-factorial and multi-stakeholder decision-making for water scarcity, Journal of Environmental Management 250 (2019) 109482.
[29] S. Mehryar, R. Sliuzas, N. Schwarz, M. van Maarseveen, Policy option simulation in socio-ecological systems, in: H. Verhagen, M. Borit, G. Bravo, N. Wijermans (Eds.), Advances in Social Simulation, Springer International Publishing, Cham, 2020, pp. 315–320.
[30] G. Fortino, L. Fotia, F. Messina, D. Rosaci, G. M. Sarné, A meritocratic trust-based group formation in an iot environment for smart cities, Future Generation Computer Systems 108 (2020) 34–45.
[31] P. De Meo, F. Messina, D. Rosaci, G. M. Sarné, An agent-oriented, trust-aware approach to improve the qos in dynamic grid federations, Concurrency and Computation: Practice and Experience 27 (2015) 5411–5435.
[32] V. Yazdanpanah, D. M. Yazan, W. H. M. Zijm, Normative industrial symbiotic networks: A position paper, in: Multi-Agent Systems and Agreement Technologies, Springer, 2016, pp. 314–321.
[33] M. Baldoni, C. Baroglio, F. Capuzzimati, R. Micalizio, Process coordination with business artifacts and multiagent technologies, Journal on Data Semantics 8 (2019) 99–112.
[34] M. Dastani, S. Sardiña, V. Yazdanpanah, Norm enforcement as supervisory control, in: PRIMA 2017: Principles and Practice of Multi-Agent Systems, 2017, pp. 330–348.
[35] M. Baldoni, C. Baroglio, R. Micalizio, Accountability, responsibility and robustness in agent organizations, in: The 1st International Workshop on Responsible Artificial Intelligence Agents, RAIA 2019, IFAAMAS, 2019, pp. 1–8.
[36] V. Yazdanpanah, M. Dastani, N. Alechina, B. Logan, W. Jamroga, Strategic responsibility under imperfect information, in: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019, IFAAMAS, 2019, pp. 592–600.
[37] SHAREBOX, Secure Sharing, http://sharebox-project.eu/, 2020. Accessed: 2020-07-30.

Computable Law as Argumentation-based MAS

Roberta Calegari[a], Andrea Omicini[b] and Giovanni Sartor[a]

[a]*Alma AI – Alma Mater Research Institute for Human-Centered Artificial Intelligence , Alma Mater Studiorum—Università di Bologna, Italy*
[b]*Dipartimento di Informatica – Scienza e Ingegneria (DISI), Alma Mater Studiorum—Università di Bologna, Italy*

Abstract
In this paper we sketch a vision of computable law as argumentation-based MAS, i.e., human-centred intelligent systems densely populated by agents (software or human) capable of understanding, arguing, and reporting, via factual assertions and arguments, about what is happening and what they can make possibly happen. A multi-agent system based on argumentation, dialogue, and conversation is, in this vision, the basis for making the law computable: through argumentation, dialogue, and adherence to social judgment, the behaviour of the intelligent system can be reached, shaped, and controlled with respect to the law. In such a scenario, computable law – and related intelligent behaviour – is likely to become associated with the capability of arguing about state and situation, by reaching a consensus on what is happening around and what is needed, and by triggering and orchestrating proper decentralised semantic conversations to decide how to collectively act in order to reach a future desirable state. Interpretability and explainability become important features for that sort of systems, based on the integration of logic-based and sub-symbolic techniques. Within this novel setting, MAS methodologies and technologies become the starting point to achieve computable law, even if they need to be adapted and extended for dealing with new challenges.

Accordingly, in this paper we discuss how this novel vision can build upon some readily-available technologies, and the research challenges it poses. We analyse a number of approaches and technologies that should be involved in the engineering of systems and services, and become core expertise for distributed systems engineers. Among the others, these include knowledge representation, machine learning, and logic argumentation.

Keywords
computable law, multi-agent system, argumentation, logic, hybrid approaches

1. Introduction

The research field of computable law studies the engineering of the law – i.e., the design of appropriate formal models and the development of the corresponding technology – to allow norms, terms and conditions to be represented in a machine-understandable way [1]. The aim is to enable machine and software agents to process, and reason about, legal abstractions with a certain degree of accuracy, and to take autonomous decisions based on this. In the context of computable law, two key factors have to be considered. The first is that a single or unique way

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
roberta.calegari@unibo.it (R. Calegari); andrea.omicini@unibo.it (A. Omicini); giovanni.sartor@unibo.it (G. Sartor)
http://robertacalegari.apice.unibo.it (R. Calegari); http://andreaomicini.apice.unibo.it (A. Omicini); https://giovannisartor.net/ (G. Sartor)
0000-0003-3794-2942 (R. Calegari); 0000-0002-6655-3869 (A. Omicini); 0000-0003-2210-0398 (G. Sartor)

of modelling legal knowledge cannot be taken for granted—namely, there are multiple ways of identifying and circumscribing the "law" to be modelled, and multiple ways of representing legal contents into automatically processable information structures. Therefore, the computable law model should be able to deal with heterogeneous "legal ontologies".

The second factor is that nowadays application scenarios for computable law are strongly characterised by the same "symbolic vs sub-symbolic" dichotomy that also informs the most promising techniques in artificial intelligence today. Indeed, if, on the one hand, legal intelligence has historically been bound to logic and argumentation (i.e., symbolic approaches), on the other hand, the applications of machine learning algorithms (i.e., sub-symbolic approaches) are increasing today, especially in the field of data analysis and predictive justice. The use of the latter techniques raises ethical and fairness issues linked to the lack of transparency and the possibility of hidden biases. Therefore, models aimed at making the law computable must somehow consider that dichotomy and should try a reconciliation – possibly via a blended integration – so to take advantage of each approach: benefiting from the strengths of each method, and smoothing the corresponding limits.

Along this line, in this paper we sketch a vision of computable law as an argumentation-based multi-agent system. As widely recognised, agent architecture is today the reference for the design of intelligent systems [2, 3, 4], it allows dealing with heterogeneous entities and models, and fits perfectly with the pervasive and distributed scenarios that the computable law aims at addressing. The vision we propose relies on the multi-agent system (MAS) abstractions and architecture [5] expanding them both from the point of view of norms and argumentation (as in existing works on normative MAS, see for instance [6, 7]), and of the integration between symbolic and sub-symbolic techniques.

In this scenario, the very nature of the system actors – intended as agents, but also as surrounding environments – embodies the concepts of computable law implementing and coordinating the activities of distributed processes in order to achieve either individual or common goals. In fact, computable law is likely to become associated with the capability of debating about situations and about the current context, by reaching a consensus on what is happening around and what is needed, and by triggering and directing proper decentralised semantic conversations to decide how to collectively act in order to reach the future desirable state of the affairs. Within this novel setting, interpretability and explainability become a remarkable feature of the multi-agent system.

1.1. Contributions of the Paper

Based on the envisioned scenario, the contributions of this paper are the following:

- We detail the concept of computable law as argumentation-based MAS (i.e., conversational agents), also with the help of a running example, and show how they affect the engineering of intelligent systems, challenging traditional approaches to distributed computing and calling for novel argumentation approaches.
- We investigate a number of approaches and technologies that should be involved in the engineering of systems and services in that original scenario, and should become core expertise for distributed systems engineering. Among the others, those include

knowledge representation and ontologies, machine learning, argumentation models and technologies, human-computer interfaces.

2. Motivating Scenario: self-driving cars

To ground the discussion, we first examine a case study in the area of traffic management, considering the near future of self-driving cars. In that scenario cars are capable of communicating with each other and with the road infrastructure while cities and roads are suitably enriched with sensors and virtual traffic signs able to dynamically interact with cars to provide information and supervision.

The choice of the case study is driven by the fact that both self-driving cars and the whole intelligent transportation domain – including traffic management – are a natural fit for computable law since they need real-time control and feedback – with respect to the norms and the current state of affairs – and should both adapt to legislation as well as to possible contingencies (also involving ethical choices and legal reasoning).

Accordingly, self-driving cars need to *(i)* exhibit some degree of intelligence for taking autonomous decisions, *(ii)* converse with the context that surrounds them, *(iii)* have humans in the loop, and *(iv)* be deeply intertwined with the law setting characterising the environment and the society. Generally speaking, the main success factor to address the goal of the car (i.e., reach a destination) is the capability to converse and dialectically interact with the surrounding environment (other vehicles, infrastructure, humans, etc.), in order to make the best choice and therefore actuate the best action.

Part of this scenario is already a reality in many (smart) cities around the world. There, IT technologies are exploited to improve both the organisation of the transportation infrastructure and the performance of some specific parts. Adaptive traffic lights, dynamic speed and flow metering, urban monitoring stations, and similar tools and algorithms are routinely employed by district administrations as a means to observe and revise traffic situations in (almost) real-time [8, 9], as well as to assist in urban planning.

In the following we analyse the scenario from a computational perspective – i.e., discussing in details actors and actions that come up as well as system requirements – in order to provide the bases for its reification on the most appropriate engineering approaches and corresponding architecture.

2.1. Analysis & requirements

In the envisioned desiderata scenario, passengers simply express their desire (e.g., "bring me to the hospital") and the car starts acting in autonomy, travelling towards the destination and without passengers to worry about the specific actions and decisions to be undertaken. There, autonomous goal-oriented smart agents (either software, objects, humans, etc.) are pervasive since multiple actors come into play, such as other self-driving cars or, for instance, autonomous intersection managers that regulate the flow of cars based on specific goals imposed by the municipality (e.g., reduce circulation in a specific area). Moreover, the environment is one of the main actors of such a system–providing context information, such as rules valid in that specific area or situation, i.e., encapsulating context knowledge and intelligence.

Looking in-depth at the expect behaviour of self-driving cars in an urban environment, it turns out that autonomous cars need to undertake a complex decision-making process, in compliance with norms and social rules, in almost the whole journey, which seamlessly integrates actions (and perceptions) at different levels.

The first action is the trip planning which must be done by considering the *global knowledge*—for instance, local laws, regulations, policies, and average traffic conditions of the intersections and roads along the path toward the destination, influenced by factors such as the day of the week, the hour of the day, the weather, and the like.

Planning is then modulated by considering all contingencies arising during the journey – such as, for instance, a car crash forcing a change path, a protest causing delays in our preferred path, etc. As a consequence, the original plan has to be adapted to the *local knowledge* (and perceptions) cars gather while enacting it; once again, the decision is closely related to legal issues: for instance, what is the best action to perform in order to avoid fines? Or, again, if an accident cannot be avoided, how to choose the ethically-preferable option?

Moreover, there are strict rules on *safety* we would like self-driving cars to automatically enforce, such as slowing down when passing nearby a school, breaking and setting aside the vehicle as soon as the horn of an ambulance is heard, etc. In other words, regardless of what the global and local knowledge may suggest, we abide by a set of general *commonsense rules* orthogonally considered valid.

Generally speaking, compliance with the legal rules by the autonomous cars – including also global, national, state, and local laws, regulations, and policies – must be guaranteed, unless contingent priority situations occur (such as when a norm has to be violated in order to save the life of a passenger/pedestrian).

The analysis carried out so far highlights some key requirements that engineering approaches and techniques have to fulfil. First of all, the envisioned scenario seamlessly integrates perceptions (and actions) at two different scales—namely, the macro and the micro.

The *macro level* of the system includes global knowledge and generally-valid rules, like universal norms and legal conventions, possibly modulated by commonsense reasoning. Moreover, with respect to the surrounding environment, the *macro level* deals with a *mid / long term* horizon and focus on the issue of traffic flow management–including, for instance, traffic flow forecasting and urban planning possibly learned from historical data analysis. On the other hand, the *micro level*, deals with the *short term* horizon, and mostly focuses on intersection management, including a few highly intertwined sub-problems—e.g. collision avoidance, minimisation of average delay, and congestion resolution. The macro-level and the micro-level act synergistically, exploiting some sort of integration in order to achieve individual and social goals.

As the last step, a suitable *infrastructure* for V2I (vehicle-to-infrastructure) and V2V (vehicle-to-vehicle) communication should be considered, for instance through the deployment of Road-side Units (RSU). This infrastructure should make it possible to convey information from the vehicle to the infrastructure and vice-versa. For instance, it should provide information about the road on which the vehicle is travelling and its specific rules, environmental conditions around the vehicle, traffic in the vicinity of the vehicle, and construction in the vicinity of the vehicle.

All the abovementioned ingredients should be mixed consistently in order to achieve the goal of the cars and therefore of the user, respecting current rules and convention. Therefore, the system knowledge base needs to be built by taking into account both macro and micro scales, and agents have to be able to reason and argue over it in order to achieve their goals.

In the following, we show that computable law naturally is an essential ingredient in a distributed multi-party conversation, or dialogue, based on distributed intelligence. It cannot be easily tackled with traditional approaches to distributed computing: instead, different approaches and techniques needs to be put in place and to be fruitfully integrated in order to meet the aforementioned requirements.

3. Computable law as conversation and distributed (micro-)intelligence in MAS

The wide variety of actors and requirements in the above-described scenario recall the MAS model and architecture as intrinsically suitable for addressing issues and challenges lay ahead. Along this line, the vision of computable law described in this paper is based on the fundamental roles of MAS – individuals (agents), society, and environment [10]– according to the Agents and Artefacts (A&A) meta-model [11].

Then, our vision enhances these fundamentals roles taking into account two more key concepts and related abstraction: norms and e-institutions – i.e., proper roles and abstractions for the normative environment – and micro-intelligence—i.e., rational reasoning capabilities to enhance both environment and individuals.

In a nutshell, the system is composed of several agents, each with his own personal goal to achieve. Agents' interaction and dialogue, on normative aspects and surrounding situations, allow them to complete their goals. For that reason, rules and conventions must have an active role in the system (e-institution) in order to be questioned, examined and respected. The system's actors – individuals, environment, e-institution – are intelligent (micro-intelligence) in that they can reason on knowledge and context, argue and explain the rationale behind their decision. The overall behaviour of the system transposes the social behaviour, the debate with the society can act as a social judgment on individuals and therefore can calibrate their individual actions.

Different roles of e-institution and micro-intelligence and their interaction are discussed below.

3.1. e-Institution

In the vision sketched above, particular attention has been given to *normative* concepts since norms and laws are the basic bricks upon which to build the concept of computable law, since individual and collective behaviours are both affected by norms. In particular, we leverage on the well-known concepts of electronic-institution (e-institution) – as in normative MAS [6] – and deliberative agents [12], setting up e-institutions via suitable coordination artefacts exploited as normative abstractions.

Loosely speaking, e-institutions are computational realisations of traditional institutions: i.e., coordination artefacts providing an environment where agents can interact according to stated laws – norms or conventions – in such a way that interactions within the e-institution reproduce norm-based interactions in the actual world. With the term deliberative agents, we emphasise the agents' autonomy stressed both by e-institutions and normative systems. Indeed, in such systems individuals possess the property of normative autonomy—i.e., can decide to violate a norm to achieve their goals, or to change their goals so that they match the existing norms. E-instutions can provide for real-time detection of violations, and norm-enforcement can be envisaged via different enforcement technique. For instance, a simple local blocking rule [13] can be applied (specific actions may be disallowed), or the application of the laws can be prioritised according to the contingent situation (some actions may be discouraged), or norms can be modelled in a game theory framework, where sanctions and rewards are provided for agents [14], or where agents are expected to cooperate according to norms that maximise the society's utility.

From an agent's point of view, two major benefits stem from modeling environments as e-institutions. On the one hand, e-institutions establish conventions – on behaviour, language, and protocols – that inform agents about the law and possibly induce them to comply. In a sense, the environment is given structure, so that the agents have an easy comprehension of its working laws. On the other hand, norm enforcer agents endowed with capabilities for acquiring norms dynamically and enforcing them in uncertain environments can be envisioned and spread all over the system. Recalling our motivating scenario: drivers or traffic wardens know when they can talk, what consequences their acts will have, and what actions are possible at each moment in time. These restrictions contains the set of actions that agents have to consider at each moment in time by limiting the set of options that agents have to think about.

From the systems properties' point of view, e-institution promote a clear embodiment of the laws that govern the system, thus making it observable and more explainable in its autonomous actions—i.e., the explicit role of institution allow to enforce a set of norms whose violation can be perfectly observed. With respect to the case study, e-institution incarnates global system knowledge as global, national, state, and local laws, regulations, and policies.

3.2. Micro-intelligence

In order to ensure different levels of knowledge – and intelligence – as highlighted in Section 2, the MAS model is extended with the concept of *micro-intelligence.* Micro-intelligence is exploited to manage precisely the micro-level of the system intelligence – i.e., local intelligence, as the intelligence of objects, things, but also of the environment – to be integrated with the macro-level intelligence. We recall the micro-intelligence definition from [15, 16] as the *externalised rationality* of cognitive agents, complementing their own in the sense of Clark and Chambers' *active externalism* [17], and under the perspective of Hutchins' *distributed cognition* [18], as depicted in Figure 1. In short, micro-intelligence is *external* because it does not strictly belong to agents, as it is a process independently executed by another entity – namely, an artefact – to whom the agent is (possibly, temporarily) coupled—in the sense of the distributed cognition's "extended mind". Moreover, it is *rational* because it is supposed to convey a sound inference and argumentation process in order to provide an explanation on the decision process. Micro-intelligence complements agents' own cognitive processes because it augments the cognitive capabilities of agents, by embodying situated knowledge about the local environment along with the relative inference and argumentation processes. Agents may even be unaware of the

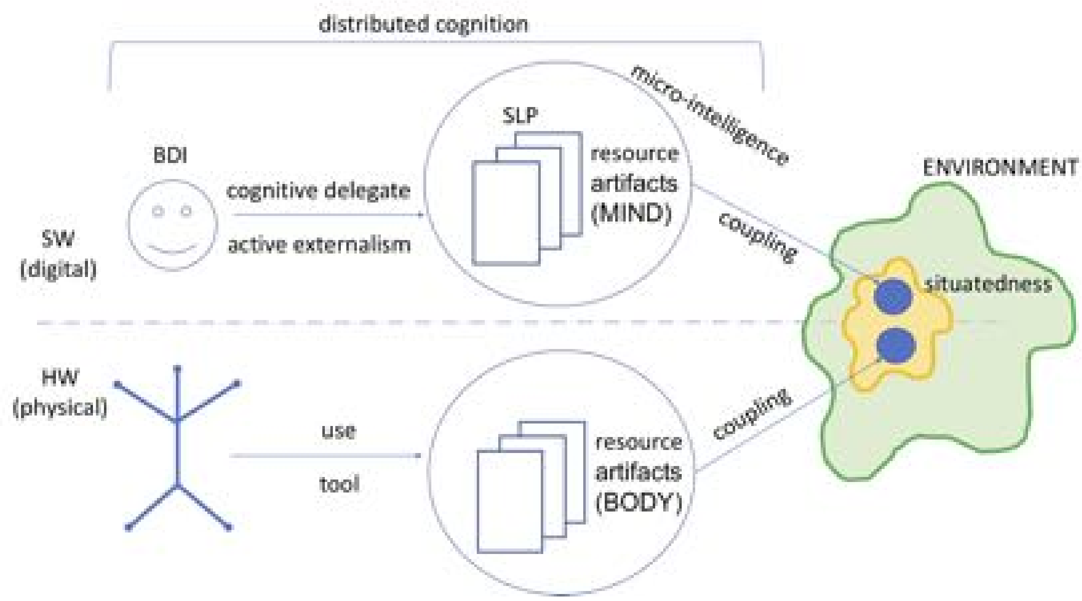


Figure 1: Micro-intelligence under the A&A perspective: distributed cognition is enabled by the active externalism of artefacts acting as cognitive delegates for the agent (extended) mind.

knowledge embodied in the environmental artefact delivering micro-intelligence. Under this perspective, artefacts act as delegates for intelligent (possibly, epistemic, but surely rational) behaviour, since they undertake inference processes on behalf of the interacting agents. Along this lines, our vision stems from two basic premises underpinning the above definition: *(i)* knowledge is locally scattered in a distributed environment, hence its situated nature; *(ii)* inference and argumentation capabilities are admissible and available over this knowledge, with the goal of extending the local knowledge through argumentation, induction, deduction, abduction, and the like.

Operationally, micro-intelligence is about scattering small chunks of machine intelligence all over a distributed and situated system, capable to enable the individual intelligence and the argumentation capability of any sort of devices. Micro-intelligence can be encapsulated in devices of any sort, making them smart, and capable to work together in groups, aggregates, societies. Thus, the micro-intelligence vision promotes ubiquitous distribution of intelligence in large pervasive systems such as those belonging to the pervasive and IoT landscape, in particular as a complement to agent-based technologies and methods, at both the individual and the collective level.

Note the micro-intelligence model – and related technologies – becomes fundamental also for the e-institution since it provides externalised rationality for reasoning and speaking with other system actors as well as the capability to reach properties such as interpretability and explainability, by leveraging on symbolic approach.

Moreover, micro-intelligence, as depicted in this summary, becomes the fundamental engine enabling conversation and argumentation between the system's actors and facilitating the control of contingent situations establishing the possibility to reach a consensus among speaking

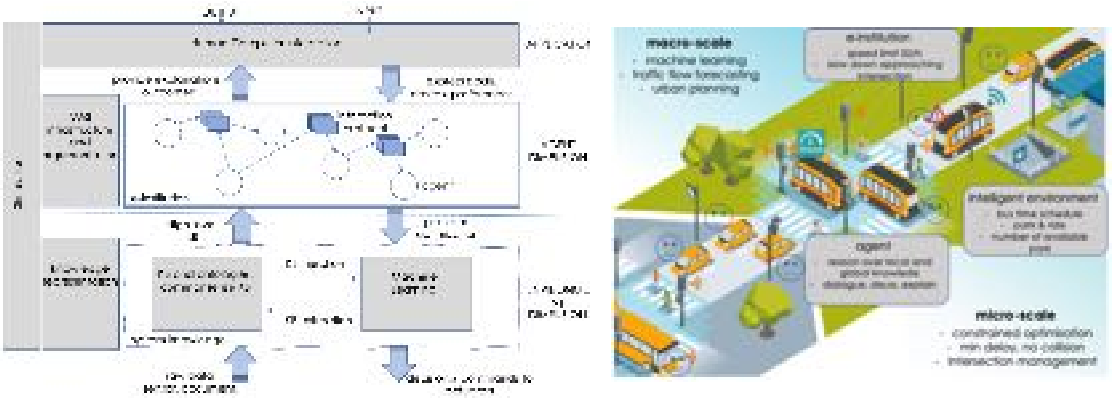


Figure 2: Main architecture components and techniques for realising the vision (left) and an exemplary deployment (right).

agents.

3.3. Overall vision

Mixing up all the above-mentioned ingredients, agents become capable of dialogue, exploiting a type of rational and symbolic intelligence that allows arguing and reasoning on the acquired knowledge. They become aware of the context – normative and social – in which they are immersed and through the argumentative process and the dialogue they can decide the next actions be implemented. The normative context is properly represented by the e-institution abstraction. Distributed (micro-)intelligence embodies the enabler of conversation between entities in the MAS infrastructure, making feasible the computation of the law and its incarnation in a computational system.

Dialogue is not expected only between agents but also with the environment and in particular with the e-institution. The bi-directional dialogue and interaction process, as well as enabling negotiation between system entities, also allow a prompt examination of each agent and of their status in order to detect automatically and real-time a possible deviation from the rules or policies envisaged. The issue of reaching a consensus in an ensemble of cooperating and interacting autonomous components by distributed negotiations has been already investigated in the field of multi-agent systems [19]. In particular, the argumentation-based negotiation area [20, 21] shows how argumentation can help in reaching global and individual goals and solutions, by letting agents converse and motivate their choices.

Figure 2 (left) summarises our vision by highlighting the main roles involved in the system as well as the two main activity flows—from data to users (left) and the opposite, from user goals and desires to activity planning, and lower-level commands (right). The grey boxes, that we describe more in detail in the following section, represent the technologies involved in the vision, while arrows represent the expected provided functionalities.

On one side, knowledge is collected from various sources – e.g., domain-specific knowledge, ontologies, sensors raw data – and is then exploited by agents that live in a normative environment (e-institutions). Agents' activity is then regulated by norms but also addicted to situated knowledge over which agents can reason and discuss in order to achieve goals. The multi-agent system, also thanks to its rational reasoning and argumentation capabilities, can provide outcomes to the users as well as explanations for their behaviours. On the other side, humans can insert input into the system – like desires, preferences, or goal to achieve – and these are transposed into agents' goal, corresponding activity planning, and lower-level commands for actuators.

The law, in this vision, become an internal component of the computational processes: legal norms, values, and principles are mapped and translated into, a computable representation of legal information – i.e., into the system knowledge – that is directly processed by computational entities. Agents become artificial legal agents able to comply with legal requirements and to reason over them.

Computable law assumes a vision of compliance/law by design in this architecture combining top-down compliance with predefined rules and bottom-up learning from cases and experience with capability to address regulatory conflicts according to legal values and principles.

4. Enabling technologies

The above considerations translate into different engineering approaches and therefore different nature of algorithms and techniques that can and must be considered when building the system. In the following, we encapsulate the main technologies involved as well as research challenges and opportunities.

4.1. Knowledge representation

Knowledge representation and related techniques are the cornerstone of the envisioned distributed system, to be able to argue and reason over it.

Of course, system knowledge has to take into account domain-specific knowledge and large-scale ontologies as repositories to interpret the knowledge bases available to the agents and to reason and argument over it. Knowledge could be continuously modified, adapted, and refined by the agents, according to their experience and perception of the environment or to learning from experience.

Accordingly, the knowledge base is plausible that is assembled by two main sources: on the one hand, ontologies and hand-crafted rules, on the other hand, rules learned from big data. Advances in machine learning will allow extracting knowledge from this data and to merge it with the former. Hybrid approach dealing with the integration of symbolic and sub-symbolic approaches becomes of paramount importance.

In this context, there are several issues and challenges to be tackled, to cite few, automatic extraction of knowledge from ML models, extraction of commonsense knowledge from the context, integration of the diverse knowledge in an appropriate logical language that allows argumentation and inference process to be performed. Several research fields are already facing these issues, but the general problem is far from being solved. For sure, we believe that a suitable integration of symbolic and sub-symbolic approaches can help in the achievement of the construction of proper system knowledge.

4.2. Machine learning

In our vision, a fundamental role is played by machine learning involved in different phases—namely, data processing & rule learning, and planning.

Data processing & rule learning. At the most straightforward level, machine learning techniques are clearly involved in raw input data elaboration, coming from sensors and/or documents, into more complex, high-level, structured information. Moreover, agents should be able to learn policies from past experience, by adapting both to the changing environment, and to the continuous progress of the society. Data aggregation, feature extraction, clustering, classification, data and pattern mining techniques are typically employed today to reach these objectives. We believe that hybrid approach could provide promising solutions to these tasks, by merging logic with probabilistic models and statistical learning, so to efficiently handle advantages of both symbolic and sub-symbolic approaches and moving towards explainable systems [22]. As highlighted above, the ML knowledge should somehow be translated into logical knowledge and properly merged with logical knowledge coming from ontologies or domain-expert norm translation or similar.

Planning. Distributed problem solving, planning, reinforcement learning, and cooperation [23] are some of the well-known ML techniques exploited in MAS. Our framework adds the challenge of integrating these techniques in the argumentation setting, so that the planning and cooperation derive from a continuous, natural interaction between agents with the environment. Once the user has specified his desires, the agent must be able to achieve them, interacting and coordinating with other individuals and with the e-institution to define the actions to perform and consequently defining appropriate plans to reify the decisions.

4.3. MAS & Normative MAS - middleware infrastructure

From a more implementation-oriented perspective, given that conversations are a new means of orchestrating the activities of distributed agents, an open research question – and a key one, too – is to understand which services should a middleware provide in order to support such distributed conversations.

The multi-agent infrastructure need not only to allow coordination among system actors but also include the possibility of customisable and reactive artefacts, capable of incorporating regulation and norms and micro-intelligence (possibly in the form of service). Moreover, the middleware should provide support for discussions via an open and shared discussion space, enabling dialogue among components that do not necessarily know each other in advance, and also providing services and or techniques for sharing knowledge, e.g., a tuple space [24]. However, unlike traditional tuple space models, the evolution of the conversation, the argumentation process and the reached consensus should be taken into account, also to be exploited in similar situations and/or to provide explanations. The best way to build such shared dialogue space, also taking into account different source of knowledge (e.g., commonsense kb, ...) and different artefact acting both as law enforcer and intelligence promoter is a fertile ground for research.

4.4. Argumentation & logical reasoning

Argumentation is a necessary feature for agents to talk and discuss to reach an agreement. Several existing works establish the maturity of argumentation models as a key enabler of our vision [25, 26].

Despite the long history of research in argumentation and the many fundamental results achieved, much effort is still needed to effectively exploit argumentation in our envisioned framework. First, research on argumentation has mostly been theoretical, practical applications to real-world scenarios have only recently gained attention and are not yet reified in a ready-to-use technology [27]. Second, many open issues of existing argumentation frameworks regard their integration with contingency situation and situated reasoning to achieve a blended integration of the two concepts. Finally, the fundamental assumption every argumentation framework makes – that is, there must exist either an agreement among agents about the knowledge, or an external judge enacting some form of control over the argumentation process – is quite challenging preserving in our envisioned highly distributed, open, and dynamic scenario. The assumption is somehow related to the requirement of the formal model to have a coherent and logical conclusion, but neither of the assumptions is easy to have in a typical pervasive situation: reaching an agreement among many heterogeneous agents and devices is already a complex task, not obviously easily scalable, an external authority may be an unacceptable centralisation point. The argumentation architecture should be designed in order to be highly scalable, hybrid approaches should be investigated such as:

- provide many external authorities sharing the load of negotiating argumentations among a limited number of participants to enforce shared normative rules, possibly exploiting some notion of proximity;
- base the agreement on temporary agreement valid only for the duration of a conversation, defining somehow the concept of temporal locality;
- dually to the previous one, spatial locality may be the criterion to enforce partial consistency of normative and behavioural rules—according to the concept of "argumentation neighbourhood" where different distributed mediators act to enforce an agreement respecting the law.

In any case, we think that the concept of locality is crucial and should be considered along with distributed argumentation—coherently with the notion of micro-intelligence. In addition, it could be interesting to envision a framework where the specific argumentation and inference process can be swapped at runtime, making consideration on the specific situation, and introducing, for instance, abductive reasoning, or probabilistic argumentation or the algorithm deemed best and necessary for such a situation. The dialogue, therefore, becomes possible by following different rationales that can guide the decision-making process, possibly comparing different perspectives and visions.

4.5. Human-Computer interaction

Finally, techniques coming from natural language processing, computer vision speech recognition – already a reality in most everyday applications (e.g., Amazon Alexa, Google Home,…)

– become essential components of our vision for humans interaction. The challenge in the envisioned scenario is always related to the distributed issues, i.e., making commands possibly understandable to a multitude of agents and vice-versa. Existing algorithms should, therefore, be adapted for dealing with distributed and pervasive environments.

4.6. Simulation

Validation through simulation is of paramount importance when dealing with MAS, and even more when dealing on the integration of many different technologies—as in our vision. Simulation is concerned with the truthfulness of a model with respect to its problem domain and the correctness of its construction. In other words, simulation calls for verification, i.e., “building the system right" and validation, i.e., “building the right system" [28]. So, our vision leverages on consolidated simulation techniques that must become a core technology to validate and verify the final model and its properties.

In the traffic management scenario, for instance, by simulating car-following in continuous traffic flow and comparing simulation data with the data collected from the actual road, the reliability of the model and the architecture. Many simulation models have been proposed on the topic [29, 30, 31].

Also in this field, the simulation models must be able to mix ingredients from the techniques above discussed (symbolic and sub-symbolic) and should be adapted to meet the software engineering requirements these techniques refer to.

5. Conclusions

The heterogeneous nature of intelligence required by pervasive AI systems along with fears related to sub-symbolic techniques more and more exploited in such systems require for models and technologies guaranteeing explainability as one of the main requirement.

Within this context, we believe that argumentation can play a major role, as it allows debates to be studied and analysed, reasoning and persuasion to be exploited in dialogues, with a well-grounded theoretical framework.

In this paper, we show how different dialogues can occur in such pervasive contexts, highlighting the advantages of employing argumentation. Interpretability of decision making, tolerance to uncertainty, adaptiveness, robustness of the system, and improved trust by end-users and amongst interacting components, are the most notable benefits of the proposed approach.

The main limit of the argumentation approach is the typical assumption to have an external judge or authority that has to control the whole argumentation process, but this is very unlikely in a dynamic, distributed scenario like the one we propose. This aspect will certainly be the subject of future work.

However, model and techniques that should play a key role in the engineering of intelligent systems – even if with enhancements and extensions – have been discussed and constitute a starting point for further researches.

Acknowledgments

Roberta Calegari and Giovanni Sartor have been supported by the H2020 ERC Project "CompuLaw" (G.A. 833647). Andrea Omicini has been supported by the H2020 Project "AI4EU" (G.A. 825619).

References

[1] G. Sartor, P. Casanovas, N. Casellas, R. Rubino, Computable models of the law and ICT: State of the art and trends in european research, in: Computable Models of the Law, Springer, 2008, pp. 1–20. doi:10.1007/978-3-540-85569-9_1.

[2] M. J. Wooldridge, N. R. Jennings, Intelligent agents: Theory and practice, The Knowledge Engineering Review 10 (1995) 115–152. doi:10.1017/S0269888900008122.

[3] R. Calegari, G. Ciatto, E. Denti, A. Omicini, Logic-based technologies for intelligent systems: State of the art and perspectives, Information 11 (2020) 1–29. doi:10.3390/info11030167, Special Issue "10th Anniversary of Information—Emerging Research Challenges".

[4] F. Xhafa, S. Patnaik, M. Tavana, Advances in Intelligent Systems and Interactive Applications, volume 1084, Springer International Publishing, 2020. doi:10.1007/978-3-030-34387-3.

[5] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, G. Fortino, Agent-based internet of things: State-of-the-art and research challenges, Future Generation Computer Systems 102 (2020) 1038–1053. doi:10.1016/j.future.2019.09.016.

[6] G. Andrighetto, G. Governatori, P. Noriega, L. W. van der Torre, Normative multi-agent systems, volume 4 of *Dagstuhl Follow-Ups*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013. URL: https://drops.dagstuhl.de/opus/volltexte/dfu-complete/dfu-vol4-complete.pdf.

[7] W. W. Vasconcelos, J. Sabater, C. Sierra, J. Querol, Skeleton-based agent development for electronic institutions, in: 1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2 (AAMAS '02), ACM, New York, NY, USA, 2002, pp. 696–703. doi:10.1145/544862.544911.

[8] J. Rios-Torres, A. A. Malikopoulos, A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps, IEEE Transactions on Intelligent Transportation Systems 18 (2017) 1066–1077. doi:10.1109/TITS.2016.2600504.

[9] V. Gradinescu, C. Gorgorin, R. Diaconescu, V. Cristea, L. Iftode, Adaptive traffic lights using car-to-car communication, in: IEEE 65th Vehicular Technology Conference (VTC2007), 2007, pp. 21–25. doi:10.1109/VETECS.2007.17.

[10] A. Omicini, SODA: Societies and infrastructures in the analysis and design of agent-based systems, in: P. Ciancarini, M. J. Wooldridge (Eds.), Agent-Oriented Software Engineering, volume 1957 of *Lecture Notes in Computer Science*, Springer-Verlag, 2001, pp. 185–193. doi:10.1007/3-540-44564-1_12.

[11] A. Omicini, A. Ricci, M. Viroli, Artifacts in the A&A meta-model for multi-agent systems, Autonomous Agents and Multi-Agent Systems 17 (2008) 432–456. doi:10.1007/

s10458-008-9053-x, Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

[12] R. Rubino, A. Omicini, E. Denti, Computational institutions for modelling norm-regulated MAS: An approach based on coordination artifacts, in: O. Boissier, J. A. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J. S. Sichman, J. Vázquez-Salceda (Eds.), Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems, volume 3913 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2006, pp. 127–141. doi:10.1007/11775331_9.

[13] A. Perreau de Pinninck, C. Sierra, M. Schorlemmer, Distributed norm enforcement: Ostracism in open multi-agent systems, in: P. Casanovas, G. Sartor, N. Casellas, R. Rubino (Eds.), Computable Models of the Law, Springer, 2008, pp. 275–290. doi:10.1007/978-3-540-85569-9_18.

[14] L. Denant-Boemont, D. Masclet, C. N. Noussair, Punishment, counterpunishment and sanction enforcement in a social dilemma experiment, Economic Theory 33 (2007) 145–167. doi:10.1007/s00199-007-0212-0.

[15] A. Omicini, R. Calegari, Injecting (micro)intelligence in the IoT: Logic-based approaches for (M)MAS, in: D. Lin, T. Ishida, F. Zambonelli, I. Noda (Eds.), Massively Multi-Agent Systems II, volume 11422 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 21–35. doi:10.1007/978-3-030-20937-7_2.

[16] R. Calegari, Micro-Intelligence for the IoT: Logic-based Models and Technologies, Ph.D. thesis, Alma Mater Studiorum—Università di Bologna, Bologna, Italy, 2018. doi:10.6092/unibo/amsdottorato/8521.

[17] A. Clark, D. J. Chalmers, The extended mind, Analysis 58 (1998) 7–19. URL: https://www.jstor.org/stable/3328150.

[18] J. Hollan, E. Hutchins, D. Kirsh, Distributed cognition: Toward a new foundation for human-computer interaction research, ACM Transactions on Computer-Human Interaction (TOCHI) 7 (2000) 174–196. doi:10.1145/353485.353487.

[19] S. Kraus, R. C. Arkin, Strategic Negotiation in Multiagent Environments, Intelligent Robotics and Autonomous Agents, MIT Press, 2001. URL: https://mitpress.mit.edu/books/strategic-negotiation-multiagent-environments.

[20] L. Amgoud, Y. Dimopoulos, P. Moraitis, A unified and general framework for argumentation-based negotiation, in: 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), ACM, 2007, pp. 1–8. doi:10.1145/1329125.1329317.

[21] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. Mcburney, S. Parsons, L. Sonenberg, Argumentation-based negotiation, The Knowledge Engineering Review 18 (2003) 343–375. doi:10.1017/S0269888904000098.

[22] R. Calegari, G. Ciatto, A. Omicini, On the integration of symbolic and sub-symbolic techniques for XAI: A survey, Intelligenza Artificiale 14 (2020) 7–32. doi:10.3233/IA-190036.

[23] P. Stone, M. Veloso, Multiagent systems: A survey from a machine learning perspective, Autonomous Robots 8 (2000) 345–383. doi:10.1023/A:1008942012299.

[24] S. Mariani, A. Omicini, Coordination in situated systems: Engineering MAS environment in TuCSoN, in: G. Fortino, G. Di Fatta, W. Li, S. Ochoa, A. Cuzzocrea, M. Pathan (Eds.), Internet and Distributed Computing Systems, volume 8729 of *Lec-*

ture Notes in Computer Science, Springer International Publishing, 2014, pp. 99–110. doi:10.1007/978-3-319-11692-1_9.

[25] J. Hulstijn, L. W. van der Torre, Combining goal generation and planning in an argumentation framework, in: International Workshop on Non-monotonic Reasoning (NMR'04), 2004, pp. 212–218.

[26] S. Modgil, M. Caminada, Proof theories and algorithms for abstract argumentation frameworks, in: Argumentation in Artificial Intelligence, Springer, 2009, pp. 105–129. doi:10.1007/978-0-387-98197-0_6.

[27] R. Calegari, G. Contissa, F. Lagioia, A. Omicini, G. Sartor, Defeasible systems in legal reasoning: A comparative assessment, in: M. Araszkiewicz, V. Rodríguez-Doncel (Eds.), Legal Knowledge and Information Systems. JURIX 2019: The Thirty-second Annual Conference, volume 322 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2019, pp. 169–174. doi:10.3233/FAIA190320.

[28] D. Calvaresi, G. Albanese, J.-P. Calbimonte, M. Schumacher, SEAMLESS: Simulation and analysis for multi-agent system in time-constrained environments, in: Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness. The PAAMS Collection, Springer International Publishing, Cham, 2020, pp. 392–397. doi:10.1007/978-3-030-49778-1_30.

[29] M. Mashayekhi, H. Du, G. F. List, M. P. Singh, Silk: A simulation study of regulating open normative multiagent systems., in: 25th International Joint Conference on Artificial Intelligence (IJCAI-16), 2016, pp. 373–379. URL: https://www.ijcai.org/Proceedings/16/Papers/060.pdf.

[30] J. Dai, X. Li, Multi-agent systems for simulating traffic behaviors, Chinese Science Bulletin 55 (2010) 293–300. doi:10.1007/s11434-009-0230-3.

[31] F. Zhang, J. Li, Q. Zhao, Single-lane traffic simulation with multi-agent system, in: 2005 IEEE Intelligent Transportation Systems, IEEE, 2005, pp. 56–60. doi:10.1109/ITSC.2005.1520219.

A Tool for the Automatic Generation of MOISE Organisations From BPMN

Massimo Cossentino, Salvatore Lopes and Luca Sabatucci

ICAR-CNR, 153, Via La Malfa, 90146 Palermo, Italy

Abstract

Multi-agent systems proved successful in enacting business processes because of their inner properties (distribution of tasks, collaboration and coordination among agents). MAS adoption in enacting processes becomes even more interesting if they exhibit adaptation capabilities. The proposed approach consists in the automatic generation of a MOISE organisation from the BPMN specification of a business process. This organisation is conceived to support adaptation because of the possibility to adapt its configuration at runtime according to emerging needs. Here, we focus on the tool for processing BPMN specification and generating MOISE organization code.

Keywords

Multi-Agent Systems, Business Process, BPMN, Adaptation

1. Introduction

Traditionally, business processes are designed as static, rigid procedures, but in real production environment there are many events and/or exceptions that can not be foreseen at design time and can lead to the failure of the whole process. For instance, some web service could be not reachable, the network could be extremely busy with heavy delay in response etc...However, increasing the agility and the flexibility of business process is not trivial being in contrast to the current trend of over-specifying workflow details with the objective to detail every possible execution branches.

Multi-Agent Systems own many interesting features as autonomy, distribution of tasks, and collaboration/coordination, that proved successfully in a range of application fields. Historically, MASs have good records in implementing workflows because all those features perfectly match with enterprises' needs [1, 2, 3]. Moreover, the Belief Desire Intention (BDI) paradigm[4] allows developers to design applications with practical reasoning, facilitating the definition of business logic.

Most of the agent-based approaches to workflow design are based on services and semantic web description (i.e. DAML-S [5]) for defining the external behaviours of proactive agents, and social and communication abilities of agents to coordinate the flow of tasks. It remains open the issue of coordinating heterogeneous, autonomous agents, whose internal designs is not partially or full known a-priori.

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy

massimo.cossentino@icar.cnr.it (M. Cossentino); salvatore.lopes@icar.cnr.it (S. Lopes); luca.sabatucci@icar.cnr.it (L. Sabatucci)

An important improvement in MAS design and implementation comes out with the JaCaMo platform [6]. It represents an interesting solution because it handle various aspects of agent programming. JaCaMo integrates in an unique platform three multi-agent programming dimensions (agent, environment, and organization levels) and provides Jason [7] for developing BDI agents [4], CArtAgO for defining artifacts [8], and MOISE [9] for specifying agent organizations.

In this paper we propose a tool able to automatically generate organizations of agents to implement a given workflow expressed in Business Process Modeling Notation (BPMN) [10]. A structured organization adds robustness to the MAS system without losing the nice properties of agents like reasoning, communication and coordination abilities. The aim is to produce a MAS implementation of the business process able to adapt itself to various internal/external unexpected conditions (unavailability of services, failure in reaching the expected end-condition, network problem and so on).

The BPMN is a high-level language that allows developer to model all the phases of a planned business process. It focuses on analysis activity and there is not any bounding between tasks and services at design time. A recent result has been that a business process, opportunely designed, may be automatically translated into goals [11]. This translation has the advantage that goals allow for breaking the rigid constraints of a BPMN: whereas sequence flows specify the precise order in which services are invoked, goals can relax the order of task execution, widening the space for adaptation [12]. This goes into the direction of defining several possibilities to attain the desired result.

Having a BPMN as a set of goals, it is possible to conceive a multi agent system able of addressing them [13]. The idea is that, given a set of goals, and a set of available services, it is possible to automatically generate one or more social organizations corresponding to the workflow enactment. Clearly, the number of organizations depends on the availability of services. A redundant service repository allows for different plans to pursue the same set of goals. However, when several organizations can be produced (each one targeted to the same desired result), it is possible to enable run-time adaptation strategies for continuing goal pursuit in case of failure.

The tool we propose uses a three steps procedure:

1. goal extraction from a BPMN definition [11],
2. exploiting a planner for composing available services (stored in a yellow pages service) and obtaining different wokflow solutions to the same set of goals, and
3. employment of each different solution for generating multi-faceted agent organizations voted to achieve the business goal.

The paper is structured as follows: Section 2 presents the theoretical background, providing a brief description of BPMN, MOISE and JaCaMo. Section 3 describes the automatic generation of MOISE organizations; in particular, Section 3.1 presents a running example; Section 3.2 briefly reports the approach by discussing the mapping between BPMN elements and MOISE metamodel elements; Section 3.3 enters into the details, by explaining the algorithm of conversion. Finally, some conclusion is drawn in Section 4.

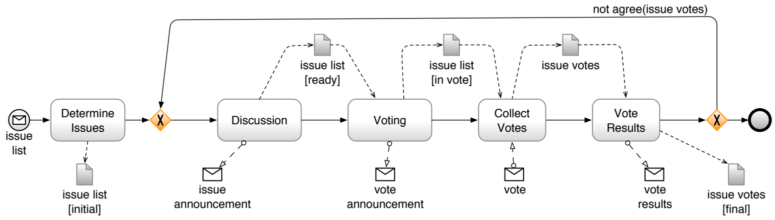


Figure 1: An example of worflow rielaborated from [14]

2. Theoretical Background

In this section we provide a brief description of the main background concepts used along the paper; namely the BPMN notation for the design of business processes and the MOISE framework for multi-agent systems organisations definition.

2.1. Process Modeling with BPMN

The *Business Process Model and Notation* (BPMN) [10, 14] is de-facto standard for business analysts to model a process. It contains a very articulated meta-model and an expressive notation for representing business processes of diverse nature. The graphical notation allows several modelling perspectives [10, 14]; this paper focuses on collaboration diagrams (similar to activity diagrams), in which a process is described as a collection of participants. Processes are composed of five categories of objects: activities, events, messages, data objects, and many kind of gateways. Every participant (each one in a different Swimlane) has her own flow of activities. Coordination occurs when processes exchange messages via message flows.

2.2. Organisations Definitions with MOISE

MOISE defines a collection of elements for modelling an organization. A MOISE organization is a specialized group that is devoted to pursue some goal. An organization is not only a collection of roles. Indeed, defining an organization implies the definition of structural, functional and normative perspectives [9].

The Structural Specification describes who (role) operates in the organization, the organization's hierarchy (groups and subgroups), which role belongs to each group together with its cardinality, and the kind of relationship between roles (authority, communication, acquaintance)

The Functional Specification specifies the scheme of activities associated to each group. A scheme represents the goal decomposition tree. Each scheme is characterized by a goal to attain, and one or more plan. A plan represents the modality (sequence, parallel, choice) to address the inner sub-goals. All the goals are contained into a mission.

The Normative Specification defines obligations and permissions that link roles to missions.

A brief description of the MOISE elements based on definitions proposed by Hubner in [9]. is reported here:

Structural Specification:

Role a role represents a placeholder for an agent that takes in charge to perform some activity. The number of agents that can play a role is constrained by a minimum and maximum number.

Group a group is composed by roles. A group is well formed when all its roles are played by agents.

Link A link specifies the type of relationship between agents playing two roles. It is possible to describe compatibilities between roles.

Functional Specification:

Goal semantic description of a result to obtain; MOISE goals may belong to two different types: achieve and maintain.

Scheme a Scheme is the decomposition of the organization's goals as an articulated gol-tree. It also includes the specification of Missions.

Plan is an operator, used within a Scheme, in order to specify the goal decomposition type (sequence, choice, parallel).

Mission A mission is a set of (sub-)goals.

Normative Specification:

Norm there are two types of norms: obligation and permission. They establish the link between the role and the mission. When an agent plays a role, it should or would commit to missions' goals.

The MOISE organisation (provided as a XML file) is not operational in itself. A set of CArtAgO Artifacts allows all necessary constructs and data structures to make a MAS operative, i.e. allowing agents to adopt ad play a role, to participate to a group, and to commit a mission [6]. For instance, the GroupBoard stores information about the MOISE elements the organization is made of. Once an agent adopt a role, it is subject to the constraints the organization imposes. Depending on the norm, an agent can commit or to be obliged to perform a mission.

3. A Tool for the Automatic Generation of MOISE Organisations from BPMN

This section discusses the tool for the automatic generation of agent organization, by using a running example to present details of the approach.

3.1. Running Example: The Simplified Email Voting Process

In the remaining sections of this paper, we refer to the example workflow reported in Fig. 1. That is a simplification of a well known BPMN example [14]. In this process, members of a

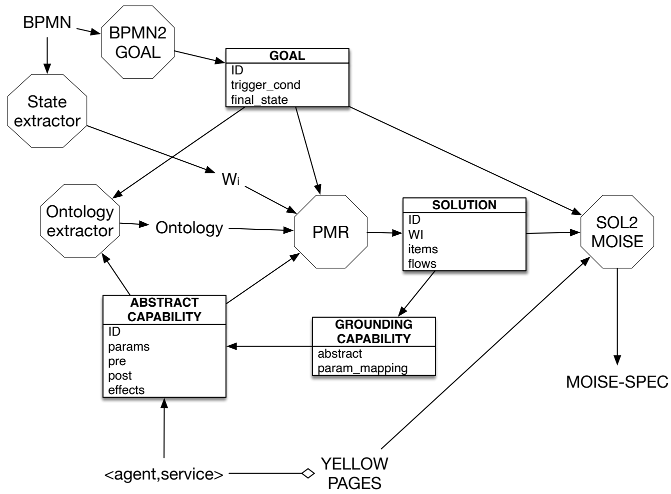


Figure 2: An overview of the main elements involved in the process to generate MOISE organizations from BPMN

committee discuss an issue list, and express a vote: if the majority is not reached, a new iteration of discussion/voting is performed.

The process description reports some events (emails) connecting the reported activities with the external voting members, represented as a swimlane shading the behaviour of the voting members that is of low interest for the current example. Data flow in the process is represented by Data Objects (like the Issue_list) that can evolve their refinement during the process, and that is represented by a state (initial, in_vote,. . .).

3.2. The Proposed Approach

This section describes the proposed approach for generating MOISE organizations starting from the BPMN description of a workflow.

The main tools and artifacts of the architecture are depicted in Fig. 2. They contribute to the generation of the organization according to the following sub-steps:

Goal Extraction The first step consists in the extraction of goals from the BPMN process that is depicted using some BPMN compliant tool (and exported using the XMI format). This becomes the input of the *BPMN2GOAL* module[1]. The tool generates a list of goals by inspecting BPMN elements and their relationships. Indeed, single elements (activities, gateways,. . .) contribute to the advancement in the world state in a way that is dependent on both endogenous factors (the result provided by the work done inside the element) and exogenous ones (the influence that other process elements have on it by creating its input, and constraining its output for compatibility with the remaining part of the process). In other words, a kind of balance of forces is to be solved in order to represent the mutual influence of each BPMN element on the other,

[1]Available online at: http://aose.pa.icar.cnr.it:8080/BPMN2Goal/

GOAL: <*name of the goal (from BPMN activity*>
WHEN: <*conditions for goal triggering, from preconditions of the activity and post-conditions of predecessor BPMN elements*>
THEN: <*intended state of the world, from activity postcondition and preconditions of successor BPMN elements*>

Figure 3: The general structure of a goal extracted from the BPMN process

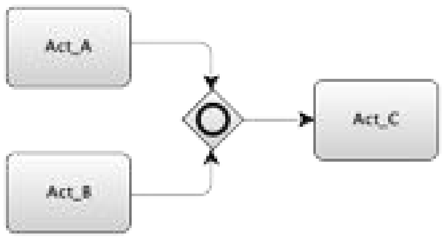


Figure 4: A fragment of BPMN process

and the single element contribution to what becomes the collective outcome of the process execution.

Details about the goals extraction procedure may be found on [11]. Briefly, the structure of a goal is reported in Fig. 3. Usually the goal name is taken from the activity it refers to (GOAL line), while the WHEN clause defines the trigger condition of the goal; this depends on the elements before the activity under analysis. Let us consider the process fragment reported in Fig. 4, an OR gateway merges the control flows of activities Act_A, Act_B that are placed before Act_C: the input of Act_C may be provided by only one of the other two activities feeding the OR gateway or even by both of them. This has to be reported in the WHEN condition by connecting the two output conditions of Act_A, and Act_B with an OR logical operator. We may also suppose Act_C requires some other condition to hold in order to be executed and that is to be expressed in the WHEN specification as well. This may happen when the condition in the original process was generated by other activities/tasks placed before Act_C. Finally, the THEN clause defines the postcondition after the execution of Act_C. This could also depend on the expected input of the following activities as well, in order to ensure the correct execution of the workflow.

State of the World Extraction The current state of the world is of paramount importance for the execution of a MAS solution. This is generated by the *State Extractor* module by processing the input workflow. The method considers the event that is usually sent to the workflow in order to trigger its execution together with any Data Input (an input Data Object for the whole process) as specified in the XMI file.

Ontology Extraction Goal specifications naturally define a vocabulary of terms that may be usefully employed to specify a part of the world where the agents live (predicates in the

WHEN/THEN conditions of goals), and the actions that can be done in it (BPMN activities). This generates a primordial ontology that is generated by the *Ontology Extractor* module. A more accurate processing of some optional details of the BPMN notation greatly contributes to this issue. As an example we could consider Data Objects (that could represent ontology concepts), their states (that could generate predicates), Data type (items) that could contribute to the generation of a IS-A tree. Despite the interesting implications, these features are not relevant to the current work.

Solutions Calculations The possibility to execute the workflow with the available agents in the MAS depends on the capabilities such agents register in the system's Yellow Pages. We suppose each agent, entering the system, registers its own capabilities in such register by using a tuple <*agent, service*>. For each service, an *Abstract Capability* is automatically generated by considering the service interface specifications like pre and post conditions. Of course, services with the same specifications are considered as belonging to the same abstract capability thus creating a redundancy in the capability instantiation that can profitably support some adaptation degree (see [13]). Indeed, the instantiation of the capability also depends on some parameters that are part of the solution and they contribute to the definition of the *Grounding Capability.* A Grounding Capability is the concretization of an Abstract one, and many Grounding capabilities may correspond to one Abstract Capability, according to the available agent-service tuples in the Yellow Pages.

Solutions are computed by the Proactive Means-end Reasoning (PMR) algorithm [15]: each solution is a workflow including capabilities, decision nodes, and so on.

The PMR algorithm, in each solution, defines a world transition system (WTS), like the one shown in Fig. **??** from the input workflow of Fig. 1. The initial state corresponds to the initial event of the workflow (the reception of the issue list in this example). The next state corresponds to the availability of an issue list arranged for the discussion. The PMR identifies the capability *prepare* as the one that could transform the state of the issue list from *[received]* to *[initial]*. This capability is the abstraction of a service registered in the Yellow Pages by one or more agents. Actually this likely means the *prepare* capability proposes the received issue list to the committee chair for editing. Once the chair completes her editing, the list moves to the *[initial]* state and it is ready for discussion in the committee. Again the PMR algorithm identifies one capability (*discussion*) for managing the debate and producing the expected output (the issue list in the *[ready]* refinement state).

Now, if the voting achieves a majority consensus, the final state is reached, otherwise another branch is activated with the possibility to iterate modifications and votes.

This WTS ma be used to deduce the workflow underpinned by the solution. That step is easily done by looking at the capabilities listed in the WTS and reporting them as activities just like it is shown in Fig. 6. Control flows are similarly found in order to complete the design.

It is worth to note that this workflow is equivalent to the input BPMN one in terms of the results it produces but the type and number of services may not (and usually do not) exactly map one-to-one to the initial process activities. This solution is specifically conceived to be executed by the agents in the MAS and it has been designed to only use services that are possessed by those agents.

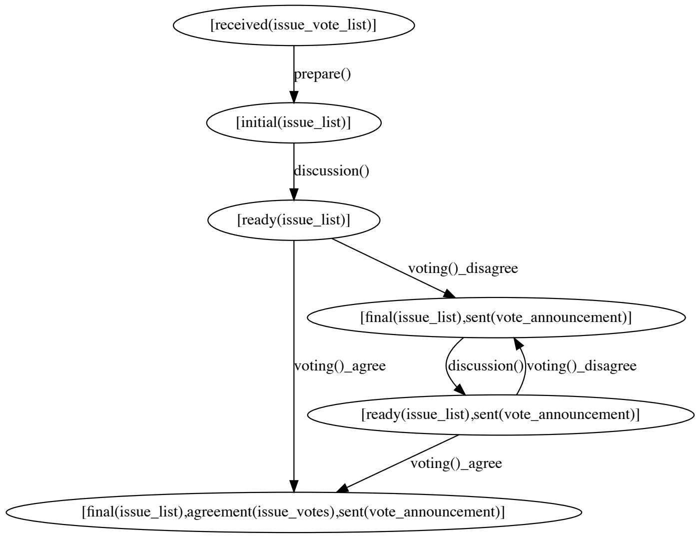


Figure 5: The world transition statechart extracted by the PMR algorithm from the process reported in Fig. 1

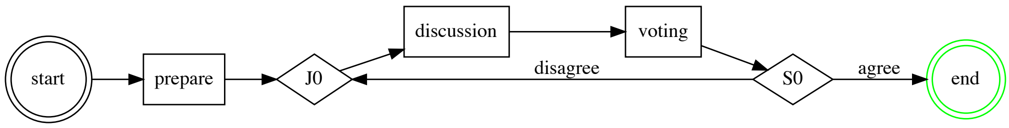


Figure 6: The workflow defined by the PMR algorithm to obtain the prescribed results using the services in the Yellow Pages

Of course the proposed example is very simple and solutions could be easily found even by hand but the approach works for large and complex input BPMN processes that would be very hard to solve using a long list of not exactly one-to-one matching services.

Moreover, the PMR algorithm, if the Yellow Pages repository is huge enough, may produce several different solutions by differently composing all the existing capabilities. This generates an even larger number of concrete solutions (the realization of a solution in terms of concrete capabilities).

MOISE Organization Definition The last step in the adopted process consists in the actual definition of the MOISE organization. The main input for that is: 1) the set of goals, 2) the set of solutions generated by the PMR algorithm and 3) the content of the yellow pages. These are processed according to the algorithm detailed in the next subsection.

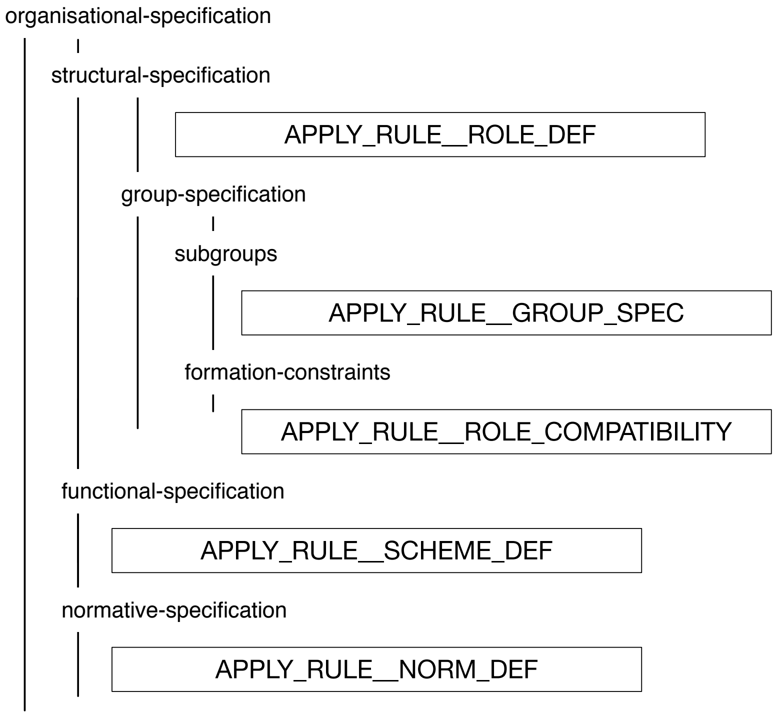


Figure 7: Abstract Representation of the XML output generated by the tool by applying specific rules.

3.3. An Algorithm for MOISE Organisation Definition

The automatic definition of the MOISE organization relies upon the XML template represented in Figure 7, in which XML elements and rules interleave. The algorithm has been written in Scala, a language derived by Java but closely integrated with XML. This way a XML rule is defined as a function that receives some parameter and returns the XML element to be used to complete the schema.

For instance, the "role definition rule" is coded as the following Scala function:

```
def apply_rule__role_def(yp:List[ServiceDescr]): Elem = {
<role-definitions>
  {yp.map(service =>
    <role id={service.id + "_role"}>
            <extends role="worker"/>
    </role>
    )}
</role-definitions>
}
```

The function receives a list of Capabilities (extracted from the system Yellow Pages) and returns a 'role-definitions' tag where children are generated from the list of services (yp parameter). Each service in yp generates (*map* method) a new 'role' xml element extending the 'worker' parent role.

The following function contains the rule for generating the group specification corresponding to a Solution as generated by the PMR algorithm.

```
def apply_rule__group_specification(sol: Solution): Elem = {
  <group-specification id={get_group_id} min="0">
  {apply_rule__group_roles(sol) ++ apply_rule__group_links(sol)}
  </group-specification>
}
def apply_rule__group_roles(s: Solution): Elem = {
  <roles>
    <role id="manager" min="1" max="1"/>
    {solution.wftasks.map(capability =>
      <role id={capability.id + "_role"} min="1" max="1"/>
        )}
  </roles>
}
def apply_rule__group_links(s: Solution): Elem = {
  <links>
    <link from="manager" to="worker" type="authority"
    extends-subgroups="false" bi-dir="false"
    scope="intra-group"/>
    <link from="worker" to="manager" type="acquaintance"
    extends-subgroups="false" bi-dir="false"
    scope="intra-group"/>
  </links>
}
```

Briefly, the rule produces a ‘group-specification’ by applying two sub-rules: the first one is for generating roles of the group; it is done by looking at the specific tasks of the input solution. The PMR solution is an assembly of Capabilities which execution ensures full goal satisfaction. Each of the involved Capabilities produces a new role into the group. Clearly, more agents of the society could own the same Capability, being able of play that role. The second sub-rule is to define structural links among these roles.

An interesting rule is that for generating the scheme, in the functional specification, corresponding to a Solution.

```
def apply_rule__scheme(s: Solution): Elem = {
  val sol_tree = new SolutionPattern(s)
  val opt_tree: Option[WorkflowPattern] = sol_tree.get_tree

  if (opt_tree.isDefined) {
    val tree : WorkflowPattern = opt_tree.get
    val capabilities = get_solution_capabilities(s)
    val scheme_id = get_scheme_id

    <scheme id={scheme_id}>
    {
    apply_rule__plan(tree)++
    apply_rule__management_mission(scheme_id)++
    capabilities.map(c=>apply_rule__mission(c))
    }
    </scheme>
  } else {
    <scheme id={get_scheme_id}>
    </scheme>
  }
}
```

The function uses the SolutionPattern class, responsible of identifying basic workflow patterns[16] in the Solution. Figure 8 shows an example of translation applied to Figure 6. The translation

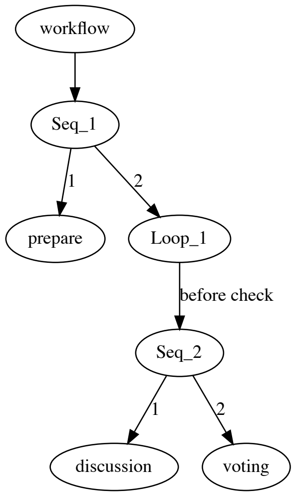


Figure 8: Result of Solution Pattern transformation. This algorithm identifies some of the Workflow Patterns [16] into a Solution, and it renders the workflow as a structured tree.

algorithm exploits the control-flow perspective of the solution in order to identify some basic constructs: Sequence patterns, Choice patterns, Loop patterns.

Sequence Pattern. Description: a 'sequence' models consecutive steps in a workflow. Activities A,B,C represent a sequence if the execution of B is enabled after the completion of A, and, the execution of C is enabled after the completion of B.
Identification. The translation algorithm supposes the sequence is the default approach for implementing a workflow.

Exclusive Choice Pattern. Description: a point in the workflow where the control may pass to one among several branches, depending on the evaluation of a condition.
Identification. The translation algorithm supposes a split exclusive gateway starts an Exclusive Choice pattern. Branches are identified as possible sequences that terminates either with the same join exclusive gateway (Simple Merge pattern) or an end event.

Structured Cycle Pattern. Description: a point in the workflow where one or more activities can be executed repeatedly.
Identification. The translation algorithm supposes a join exclusive gateway possibly begins a loop, whereas it contains a 'before check' sequence of activities until a split exclusive gateway that provides at least one exit condition, and a loop condition that begins one or more 'after check' sequences that rolling back to the first split gateway.

Once the SolutionTree has been built (as in Figure 8), the apply_rule_scheme function easily converts the tree structure into a MOISE 'scheme' via goals and plans. This requires the function below:

```
def apply_rule__plan(pattern: WorkflowPattern) : Elem = {
  pattern match {
    case SequencePattern(children) =>
```

```
      <goal id={get_goal_id}>
        <plan operator="sequence">
          {children.map(c => apply_rule__plan(c))}
        </plan>
      </goal>

    case ChoicePattern(children) =>
      <goal id={get_goal_id}>
        <plan operator="choice">
          {children.map(c => apply_rule__plan(c))}
        </plan>
      </goal>

...
   case ActivityPattern(task) =>
     val id = task.grounding.capability.id
       <goal id={id}></goal>

  }
}
```

It is a recursive rule, that browse the tree structure and incrementally builds the 'goal/plan' decomposition. A SequencePattern is translated into a sequence 'plan', a ChoicePattern is translated into a choice 'plan', whereas the Structured Cycle Pattern is a bit more articulated composition of sequences and choices. The exit condition for the recursion is the Activity (i.e. the leaves of the tree): each activity is translated into a simple 'goal' element.

Just to provide and example, the output of the previous rule application is reported below:

```
<functional-specification>
  <scheme id="scheme1">
    <goal id="root_goal">
      <plan operator="sequence">
        <goal id="prepare"></goal>
        <goal id="goal_1">
          <plan operator="sequence" type="maintain">
            <goal id="goal_2">
              <plan operator="sequence">
                <goal id="discussion"></goal>
                <goal id="voting"></goal>
              </plan>
            </goal>
            <goal id="check_exit_goal_3"></goal>
          </plan>
        </goal>
      </plan>
    </goal>
    <mission id="management_scheme1" min="1" max="1">
      <goal id="root_goal"></goal>
    </mission>
    [...]
  </scheme>
</functional-specification>
```

4. Conclusions and Future Works

The proposed approach consists in the automatic generation of a MOISE organisation from the BPMN specification of a business process with the support of a specific tool. The organisation is conceived to provide system adaptation because it includes several alternative plans (schemes) for pursuing goals. The approach is complemented by a tool for processing BPMN (XMI) code and generating the MOISE organization specification code. The approach is based on a few fundamental steps: 1) the automatic extraction of goals by processing the BPMN activities and their dependencies, 2) the identification of the initial state for agents execution (and solutions computation), 3) the extraction of the ontological vocabulary from goal definitions, 4) the calculation of one or more solutions for the achievement of process objectives by using agents' capabilities, 5) the generation of a MOISE organization composed of several alternative schemes, one for each computed solution. Alternative schemes in the agent organization can be selected according to performance criteria or to overcome a failure thus achieving some system adaptation. The generation of the organization is supported by a tool that uses well known workflow patterns to process the input BPMN XMI code and to generate the MOISE specification.

The availability of different organization's schemes allows to select the scheme (goal decomposition tree and set of missions) that provide the best performance, according to the quality attributes registered in the yellow pages. The approach also allows to replace the scheme that is in execution with another one, in case of agent/service failures thus obtaining a runtime adaptation feature.

So far, the tool supports the whole BPMN 2.0 specification for defining the process, but the Sub-Process element, that is going to be used to create hierarchical groups. The language for specifying Capabilities is a proprietary format. Very soon, we will move towards open-access action specification languages (PPDL for instance). Currently, the tool has some limitations: 1) parallel gateways are understood by the BPMN2Goal parser but not supported in phase of Goal2MOISE generation; 2) timer/clock events are partially supported because they are translated into first-logic predicates that require the agent society uses a specific internal synchronization mechanism.

References

[1] P. A. Buhler, J. M. Vidal, Towards adaptive workflow enactment using multiagent systems, Information technology and management 6 (2005) 61–87.

[2] M. P. Singh, M. N. Huhns, Multiagent systems for workflow, Intelligent Systems in Accounting, Finance & Management 8 (1999) 105–117.

[3] S. Ceri, P. Grefen, G. Sanchez, Wide-a distributed architecture for workflow management, in: Research Issues in Data Engineering, 1997. Proceedings. Seventh International Workshop on, IEEE, 1997, pp. 76–79.

[4] A. S. Rao, M. P. Georgeff, et al., Bdi agents: from theory to practice., in: ICMAS, volume 95, 1995, pp. 312–319.

[5] D.-S. Coalition, A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, et al., Daml-s: Web service description for the semantic web, in: The Semantic Web-ISWC, Springer, 2002, pp. 348–363.

[6] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with jacamo, Science of Computer Programming 78 (2013) 747–761.

[7] R. H. Bordini, J. F. Hübner, M. Wooldridge, Programming multi-agent systems in AgentSpeak using Jason, volume 8, John Wiley & Sons, 2007.

[8] A. Ricci, M. Viroli, A. Omicini, Programming mas with artifacts, in: International Workshop on Programming Multi-Agent Systems, Springer, 2005, pp. 206–221.

[9] M. Hannoun, O. Boissier, J. S. Sichman, C. Sayettat, Moise: An organizational model for multi-agent systems, in: Advances in Artificial Intelligence, Springer, 2000, pp. 156–165.

[10] M. Chinosi, A. Trombetta, Bpmn: An introduction to the standard, Computer Standards & Interfaces 34 (2012) 124–134.

[11] L. Sabatucci, M. Cossentino, Supporting dynamic workflows with automatic extraction of goals from bpmn, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 14 (2019) 1–38.

[12] L. Sabatucci, M. Cossentino, Self-adaptive smart spaces by proactive means–end reasoning, Journal of Reliable Intelligent Environments 3 (2017) 159–175.

[13] M. Cossentino, S. Lopes, L. Sabatucci, Goal-driven adaptation of moise organizations for workflow enactment, in: Proc. of the 8th International Workshop on Engineering Multi-Agent Systems (EMAS 2020), National Research Council of Italy, 2020.

[14] Object Management Group (OMG), Business Process Model and Notation (BPMN 2.0) by Example, Available online at https://www.omg.org/cgi-bin/doc?dtc/10-06-02.pdf, 2010.

[15] L. Sabatucci, M. Cossentino, From means-end analysis to proactive means-end reasoning, in: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), IEEE Press, 2015, pp. 2–12.

[16] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, A. P. Barros, Workflow patterns, Distributed and parallel databases 14 (2003) 5–51.

Session 3

Features of MAS: Robustness, Trust, Explainability

Reinforcement Learning for Autonomous Agents Exploring Environments: an Experimental Framework and Preliminary Results

Nassim Habbash, Federico Bottoni and Giuseppe Vizzari

Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Milan, Italy

Abstract

Reinforcement Learning (RL) is being growingly investigated as an approach to achieve autonomous agents, where the term autonomous has a stronger acceptation than the current most widespread one. On a more pragmatic level, recent developments and results in the RL area suggest that this approach might even be a promising alternative to current agent-based approaches to the modeling of complex systems. This work presents an investigation of the level of readiness of a state-of-the-art model to tackle issues of orientation and exploration of a randomly generated environment, as a toy problem to evaluate the adequacy of the RL approach to provide support to modelers in the area of complex systems simulation, and in particular pedestrian and crowd simulation. The paper presents the adopted approach, the achieved results, and discusses future developments on this line of work.

Keywords

agent-based modeling and simulation, reinforcement learning, complex-systems

1. Introduction

Reinforcement Learning (RL) [1] is being growingly investigated as an approach to implement autonomous agents, where the acceptation of the term "autonomous" is closer to Russell and Norvig's [2] than the most widely adopted ones in agent computing. Russell and Norvig state that:

> *A system is autonomous to the extent that its behavior is determined by its own experience*

A certain amount of initial knowledge (in an analogy to built-in reflexes in animals and humans) is reasonable, but it should be sided by the ability to learn. RL approaches, reinvigorated by the energy, efforts, and promises brought by the *deep learning* revolution, seems one of the most promising ways to investigate how to provide an agent this kind of autonomy. On a more pragmatic level, recent developments and results in the RL area suggest that this approach might even be a promising alternative to current agent-based approaches to the modeling of complex

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
n.habbash@campus.unimib.it (N. Habbash); f.bottoni1@campus.unimib.it (F. Bottoni); giuseppe.vizzari@unimib.it (G. Vizzari)
0000-0002-7916-6438 (G. Vizzari)

systems [3]: whereas currently behavioral models for agents are carefully hand crafted, often following a complicated interdisciplinary effort involving different roles and types of knowledge, as well as validation processes based on the acquisition and analysis of data describing the studied phenomenon, RL could simplify this work, focusing on the definition of an environment representation, the definition of a model for agent perception and action, and defining a reward function. The learning process could, in theory, be able to explore the potential space of the *policies* (i.e. agent behavioral specifications) and converge to the desired decision making model. While the definition of a model of the environment, as well as agent perception and action, and the definition of a reward function are tasks requiring substantial knowledge about the studied domain and phenomenon, the learning process could significantly simplify modeler's work, and at the same time it could solve issues related to model calibration.

The present work is set in this scenario: in particular, we want here to explore the level of readiness of state-of-the-art models to tackle issues of orientation and exploration of an environment [4] by an agent that does not own prior knowledge about its topology. The environment is characterised by the presence of obstacles, generated randomly, and by a target for agent's movement, a goal that must be reached while, at the same time, avoiding obstacles. This represents a toy problem allowing us to investigate the adequacy of the RL approach to support modelers in the area of complex systems simulation, and in particular pedestrian and crowd simulation [5]. We adopted Proximal Policy Optimization (PPO) [6] and trained agents in the above introduced type of environment: the achieved decision making model was evaluated in new environments analogous to the ones employed for training, but we also evaluated the adequacy of the final model to guide agents in different types of environment, less random and more similar to human built environments (i.e. including rooms, corridors, passages) to start evaluating if agents for simulating typical pedestrians could be trained through a RL approach.

2. Problem Statement

The main focus of this paper is automatic exploration of environments without a-priori knowledge of their topology. This is modeled through a single-agent system, where an agent is encouraged to look out for a target placed randomly in a semi-randomly generated environment. This environment presents an arbitrary number of obstacles placed randomly on its space. The environment can be seen as a rough approximation of natural, mountainous terrain, or artificial, post-disaster terrain, such as a wrecked room. The agent can observe the environment through its front-mounted sensors and move on the XY Cartesian plane. In order to solve the problem of automatic exploration in an ever changing obstacle-ridden environment, the main task is to generalize the exploration procedure, to achieve an agent able to explore different environments from the ones it was trained on.

In this paper we develop a framework around this task using Reinforcement Learning. Section 3 provides a definition of the agent, the environment and their interactions. Section 4 goes through Reinforcement Learning and the specific technique adopted for this work. Section 5 provides an architecture to the system, with some details on the tools used. Section 6 reports the experimental results obtained, and Section 7 provides some considerations on the work and possible future developments.

3. Agent and Environment

3.1. Agent model

The agent is modeled after a simplified rover robot with omnidirectional wheels, capable of moving on the ground in all directions. The location of the agent in the environment is described by the triple (x, y, ϑ), where (x, y) denotes its position on the XY plane, and ϑ denotes its orientation. The agent size is 1x1x1 units.

3.2. Observation space

The agent can observe the environment through a set of LIDARs that create a array of surveying rays: these are time-of-flight sensors which provide information on both the distance between the agent and the collided object and the object's type. If a ray is not long enough to reach an object because it is too far away, the data provides the over-maximum-range information to the agent. The standard LIDAR length is 20 units. The agent is equipped with 14 LIDARs, placed radially on a plane starting from the middle of its front-facing side, giving it a field of view of [-$\frac{2}{3}\pi$; $\frac{2}{3}\pi$] for 20 units of range.

More formally, we can define an observation or state as a set of tuples, as follows:

$$s = \{(x_1, o_1), (x_2, o_2), ..., (x_n, o_n)\}, s \in S \tag{1}$$

Where n is the number of LIDARs on the agent, x_i represents the distance from a certain LIDAR to a colliding object in range, and o_i is the type of said object, with $o_i \in \{obstacle, target, \emptyset\}$.

The observations (or state) space is hence defined as S, the set of all possible states s.

3.3. Action space

The possible actions that the agent can perform are:

- Move forward or backward
- Move to the left or to the right, stepping aside (literally), without changing orientation
- Rotate counterclockwise or clockwise (yaw)

The agent can also combine the actions, for example going forward-right while rotating counterclockwise.

More formally, we can define the action space as:

$$A = \{Forward, Side, Rotation\} \tag{2}$$

Where $Forward$, $Side$ and $Rotation$ represent the movement on the associated axes, and their value can be either $\{-1, 0, 1\}$, where 0 represents no movement, and -1 and 1 represent movement towards one or the other reference point on the axis.

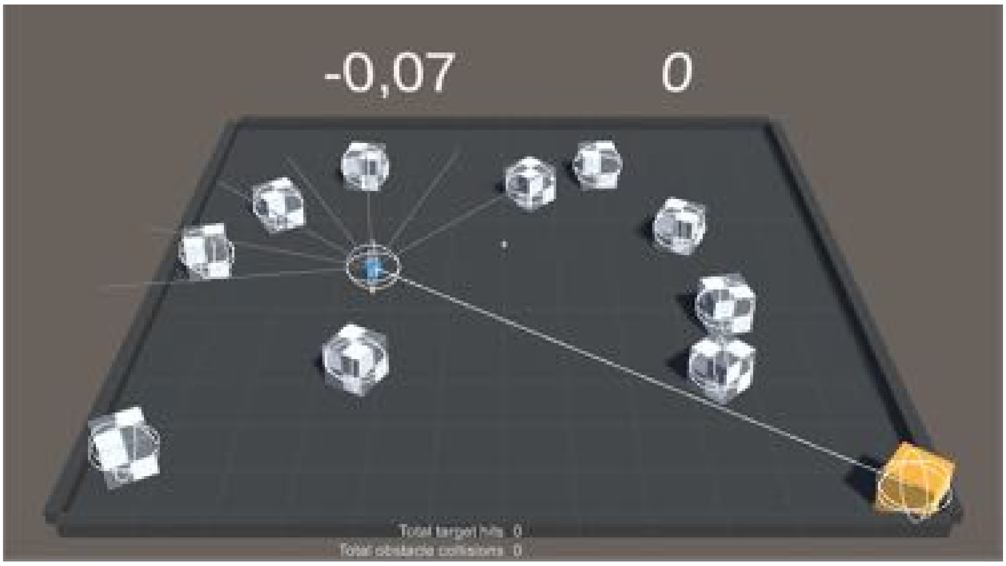


Figure 1: Environment: the blue cube is the agent, the checkerboard pattern cubes are the obstacles and the orange cube is the target. The environment is generated considering the distance between agent and target (white line connecting both) and the minimum spawn distance (the white sphere around objects) to ensure a parametric distribution between objects in the environment. The agent collects observations through its LIDAR array (white rays exiting the agent). On the top are present the episode's cumulative reward on the left and the current number of collisions on the right.

3.4. Environment model

The environment is a flat area of 50x50 units, bounded on its extremities by walls tall 1 unit. A set of gray cubes of 3x3x3 units each are randomly placed on this area as obstacles. The target - the goal the agent must reach - is positioned randomly between this set of obstacles, and is an orange cube of 3x3x3 units.

The provided interactions between agent and environment are collisions. The agent collides with another object in the environment if there's an intersection between the bounding boxes of the two entities. The floor is excluded from collisions. If a collision happens between agent and obstacles, the agent suffers a penalty, while if a collision happens between agent and target, the episode ends successfully, as the agent achieved its goal.

The environment is generated every time the episodes ends, successfully or not, so no two identical episodes are played by the agent. This generation is parametric, allowing for more or less dense obstacle distribution in the environment, or longer or shorter distance between agent and the target.

4. Reinforcement Learning

In the past couple of years Reinforcement Learning has seen many successful and remarkable applications in the robot and locomotive field, such as [7] and [8]. This approach provides many benefits: experimentation can be done in a safe, simulated environment, and it's possible to train models through millions of iterations of experience to learn an optimal behaviour. In some fields - such as robot movement - the RL approach currently outperforms classic heuristic and evolutionary methods [1].

Reinforcement Learning is a technique where an agent learns by interacting with the environment. The agent ought to take actions that maximize a reward, selecting these from its past experiences (Exploitation) and completely new choices (Exploration), making this essentially a trial-and-error learning strategy. After sufficient training, an agent can generalize an optimal strategy, allowing itself to actively adapt to the environment and maximize future rewards. Generally, an RL algorithm is composed of the following components:

1. A policy function, which is a mapping between the state space and the action space of the agent
2. A reward signal, which defines the goal of the problem, and is sent by the environment to the agent at each time-step
3. A value function, which defines the expected future reward the agent can gain from the current and all subsequent future states
4. A model, which defines the behaviour of the environment

At any time, an agent is in a given states of the overall environment $s \in S$ (that it should be able to perceive; from now on, we can consider the state the portion of the environment that is perceivable by the agent), and it can choose to take one of many actions $a \in A$, to cause a change of state to another one with a given probability P. Taken an action a chosen by an agent, the environment returns a reward signal $r \in R$ as a feedback on the goodness of the action. The behaviour of the agent is regulated by what's called a policy function π, which can be defined as

$$\pi_{\Theta}(a|s) = P(A_t = a|S_t = s) \tag{3}$$

and represents a distribution over actions given states at time t with parameters Θ - in this case the policy function is stochastic, as it maps over probabilities. Following is a brief introduction to the two main algorithms used in this work.

4.1. Proximal Policy Optimization

RL presents a plethora of different approaches. Proximal Policy Optimization (PPO) [6], the one used in this work, is a policy gradient algorithm which works by learning the policy function π directly. These methods have a better convergence properties compared to dynamic programming methods, but need a more abundant set of training samples. Policy gradients work by learning the policy's parameters through a policy score function, $J(\Theta)$, through which is then possible to apply gradient ascent to maximize the score of the policy with respect to the policy's parameters, Θ. A common way to define the policy score function is through a loss function:

$$L^{PG}(\Theta) = E_t[log\pi_{\Theta}(a_t|s_t)]A_t \tag{4}$$

which is the expected value of the log probability of taking action a_t at state s_t times the advantage function A_t, representing an estimate of the relative value of the taken action. As such, when the advantage estimate is positive, the gradient will be positive as well; through gradient ascent the probability of taking the correct action will increase, while decreasing the probabilities of the actions associated to negative advantage, in the other case. The main issue with this vanilla policy gradient approach is that gradient ascent might eventually lead out of the range of states where the current experience data of the agent has been collected, changing completely the policy. One way to solve this issue is to update the policy conservatively, so as to not move too far in one single update. This is the solution applied by the Trust Region Policy Optimization algorithm [9], which forms the basis of PPO. PPO implements this update constraint in its objective function through what it calls Clipped Surrogate Objective. First, it defines a probability ratio between new and old policy $r_t(\Theta)$, which tells if an action for a state is more likely or less likely to happen after the policy update, and it is defined as $r_t(\Theta) = \frac{\pi_\Theta(a_t|s_t)}{\pi_{\Theta_{old}}(a_t|s_t)}$. PPO's loss function, is then defined as:

$$L^{CLIP}(\Theta) = E_t[min(r_t(\Theta)A_t, clip(r_t(\Theta), 1 - \epsilon, 1 + \epsilon)A_t)] \tag{5}$$

The Clipped Surrogate Objective presents two probability ratios, one non clipped, which is the default objective as expressed in 4 in terms of policy ratio, and one clipped in a range. The function presents two cases depending on whether the advantage function is positive or negative:

1. $A > 0$: the action taken had a better than expected effect, therefore, the new policy is encouraged to taking this action in that state;
2. $A < 0$: the action had a negative effect on the outcome, therefore, the new policy is discouraged to taking this action in that state.

In both cases, because of the clipping, the actions will only increase or decrease in probability of $1 \pm \epsilon$, preventing updating the policy too much, while allowing the gradient updates to undo bad updates (e.g. the action was good but it was accidentally made less probable) by choosing the non-clipped objective when it is lower than the clipped one. Note that the final loss function for PPO adds two other terms to be optimized at the same time, but we suggest the original paper [6] for a more complete overview of PPO.

4.2. Curiosity

Reward sparseness is one of the main issues with RL. If an environment is defined with a sparse reward function, the agent won't get any feedback about whether its actions at the current time step are good or bad, but only at the end of the episode, where it either managed to succeed in the task or fail. This means that the reward signal is 0 most of the time, and is positive in only few states and actions. One simple example is the game of chess: the reward could be obtained only at the end of the match, but at the beginning, when the reward might be 10, 50, 100 time steps away, if the agent can't receive feedback for its current actions it can only move randomly until, by sheer luck, it manages to get a positive reward; long range dependencies must then be learned, leading to a complicated and potentially overly long learning process. There are many ways to solve the problem of reward sparseness, such as reward shaping, which requires domain-specific knowledge on the problem, or intrinsic reward signals, additional reward signals to mitigate the sparseness of extrinsic reward signals.

Curiosity [10] falls into the second category, and its goal is to make the agent actively seek out and explore states of the environment that it would not explore. This is done by supplying the default reward signal with an additional intrinsic component which is computed by a curiosity module. This module is comprised of a forward model, which takes in s_t and a_t, and tries to predict the features of next state the agent will find itself in $\hat{\Phi}(s_{t+1})$. The more different this value compared to the features of the real next state $\Phi(s_{t+1})$, the higher the intrinsic reward.

To avoid getting stuck into unexpected states that are produced by random processes non influenced by the agent, the module is comprised of an inverse model, which takes $\Phi(s_t)$, $\Phi(s_{t+1})$ and tries to predict the action $\hat{a_t}$ that was taken to get from s_t to s_{t+1}. By training the encoder (Φ) together with the inverse model, it s possible to make it so that the feature extracted ignore those states and events that are impossible to influence, retaining only features actually influenced by the agent's actions.

5. System Architecture

The system has been developed using Unity3D as the main simulation platform and ML-Agents for Reinforcement Learning[1].

Unity3D is a well-established open-source game engine. It provides a lot of out-of-the-box functionalities including tools to assemble a scene, the 3D engine to render it, a physics engine to physically simulate object interaction under physical laws, and many plugins and utilities. In Unity an object is defined as a GameObject, and it can have attached different components according to necessity, such as RigidBodies for physical computations, Controllers for movement, decision-making and elements of the learning system (Agent, Academy and others). An object's life-cycle starts with an initialization and then a cyclic refresh of its state, while the engine provides handler methods for these phases for customizing them through an event-driven programming.

Unity keeps track of time and events on a stratified pipeline: physics, game logic and scene rendering logic are each computed sequentially and asynchronously:

1. Objects initialization.
2. Physics cycle (triggers, collisions, etc). May happen more than once per frame if the fixed time-step is less than the actual frame update time.
3. Input events
4. Game logic, co-routines
5. Scene rendering
6. Decommissioning (objects destruction)

One notable caveat is that physics updates may happen at a different rate than game logic. This, in a game development scenario is sometimes treated by buffering either inputs or events, to result in smoother physical handling, while for more simulations in which a precise correspondence between simulated and simulation time is necessary this might pose a slight inconvenience.

[1]The project's source code is available on Github: https://github.com/nhabbash/autonomous-exploration-agent.

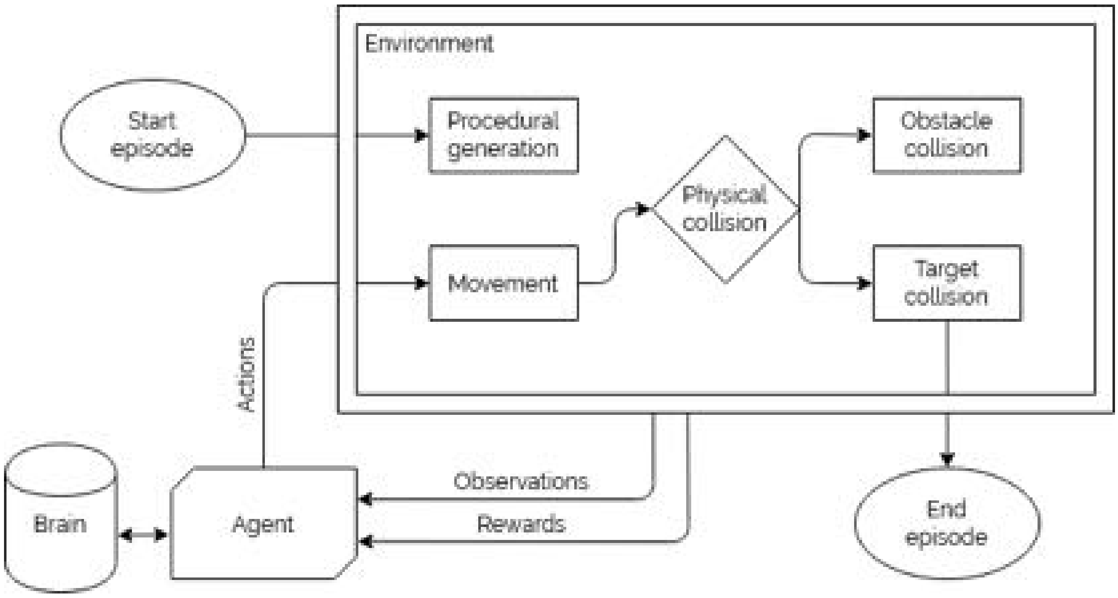


Figure 2: Schematized system architecture, shows the main interactions between the actors in the system

However, for the goals of the present work, this consideration does not represent a crucial problem.

ML-Agents is an open-source Unity3D plugin that enables games and simulations to serve as environments for training intelligent agents. It provides different implementations of state-of-the-art Reinforcement Learning algorithms implemented in TensorFlow, including PPO and Curiosity. ML-Agents comes with an SDK that can be integrated seamlessly into Unity and provides utilities for controlling the agent(s) and environment.

5.1. System design

Figure 2 represents the main actors of the system and their interactions. At the start of each episode the environment is randomly generated according to its parameters, placing obstacles, the target and the agent. The agent can then perform its actions, moving around the environment, while collecting observations through its sensors and receiving rewards according to the goodness of its actions. Physical collisions trigger negative or positive instantaneous rewards according to the time of collision: obstacle collisions produce negative rewards, while target collisions produce positive rewards and end the episode, as the task is successful. The agent class, ExplorationAgent, is responsible of the agent initialization, observation collection, collision detection and physical movement through the decisions received by the Brain interface. The decision-making of the agent is made through its Brain interface, which provides the actions produced by the model to the agent class. The environment is comprised of two classes, ExplorationArea is responsible for resetting and starting every episode, rendering the UI, logging information on the simulation process and placing every object in the environment

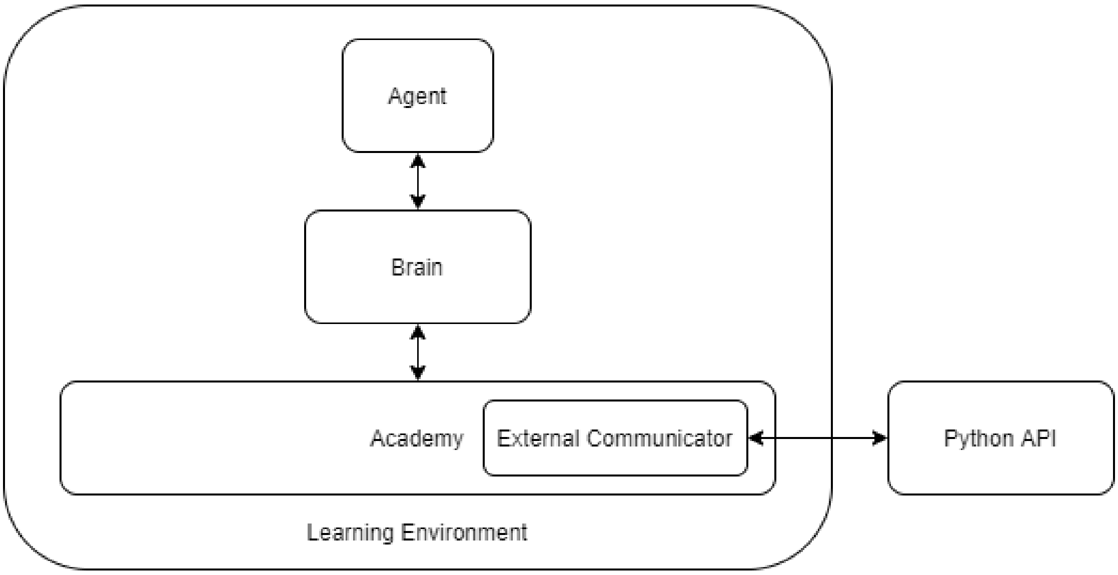


Figure 3: Learning system: the agent receives the actions from the Brain interface, which communicates with the Academy to send and receive observations and actions. The Academy communicates through an external communicator to the ML-Agents back-end, which wraps over Tensorflow and holds the various RL models involved, receiving training/inference data and returning the models outputs

according to its parameters, while the Academy works in tandem with the Brain to regulate the learning process, acting as a hub routing information - whether it's observations from the environment or inferences made by the models - between the system and the RL algorithms under the hood.

Once the training phase ends, ML-Agents generates a model file which can be connected to the brain and used directly for inference. Figure 3 explains how exactly the learning system is structured between Unity and ML-Agents.

5.2. Reward Signal

The main RL algorithm used in this work is PPO. The reward signal is composed of:

- Extrinsic signal: the standard reward produced by the environment according to the goodness of the agent's actions
- Intrinsic signal: the Curiosity of the agent

The extrinsic signal presents some reward shaping, and is defined as: $r = 5 * c_t - p$. The p term stands for penalty, and is a negative reward formulated as $p = c_o * 0.1 + time * 0.001$. Every time the agent reaches a target, indicated by c_t (target collisions), the episode ends, and the positive reward is 5, while if it hits an obstacle, indicated by c_o (obstacle collisions), it receives a penalty of 0.1 for each collision. The agent is also penalized as time passes, receiving a 0.001 negative reward for each timestep, to incentive the agent to solve the task faster.

The intrinsic signal is the Curiosity module, which as section 4.2 goes through, provides an additional reward signal to encourage the exploration of unseen states.

6. Experiments

Different scenarios for the analysis of the effectiveness of the proposed system have been investigated.

The main differences between scenarios consist in:

1. Curriculum environment parameters
2. Penalty function
3. Observation source (LIDAR or camera)
4. Knowledge transferability to structured environments

Comparison between the scenarios is conducted on two aspects: the firsts depends on the canonical RL charts built in training-time in order to define the reward, their trends over times, their balance and other information about the system; the other aspect is an environmental performance comparison, conducted through three performance measures pertaining to the investigated setting, these being CPM (collisions per minute), measuring the mean number of collisions made by the agent on obstacles, TPM (targets per minute), measuring the mean number of goal targets successfully reached by the agent and CPT (collisions per target), measuring the mean number of collisions the agents does before getting to a target. As models between scenarios have been trained with varying curricula and environments, these measures estimate the performance of every model on a shared environment, making comparison between the models happen on the same plane and circumstances. This environmental performances have been measured in a parallel fashion in order to gather more accurate data.

An interactive demo of the system is also available to allow visual comparison of the different scenarios[2].

6.1. Baseline

The first experiment acts as a baseline for other variations, and was conducted on the following parameters:

1. Static parameters:

Number of obstacles	10
Min spawn distance	2
Target distance	45

2. Penalty function: $p = c_o * 0.1 + time * 0.001$
3. Observations source: LIDAR set

[2]Demo at https://nhabbash.github.io/autonomous-exploration-agent/

The parameters in this setting generate fairly open environments with at most 10 obstacles. The minimum distance between obstacles is 2 units, while the target spawns at a distance of 45 units, which is more then half of the environment size.

Table 1 shows how the model collides roughly 5 times per minute and manages to reach a target about 4 and times per minute, making roughly **1.25** obstacles per target reached.

6.2. Curriculum

The second experiment implemented curriculum learning into the training pipeline. The curriculum was structured in seven lessons scaling along with the cumulative reward. Following are its settings:

1. Dynamic parameters:

Reward thresholds	1	2	2.5	2.8	3	3.5	4
Number of obstacles	8	10	13	15	17	18	20
Min spawn distance	6	6	4	4	3	3	2
Target distance	25	28	30	33	35	37	40

2. Penalty function: $p = c_o * 0.1 + time * 0.001$
3. Observations source: LIDAR set

The parameters in this setting generate an increasingly harder environment, with the target getting farther from the agent, and the obstacles getting more cluttered and closer together.

Table 1 shows how the model collides roughly 10.6 times per minute and manages to reach a target about 9.6 and times per minute, making roughly **1.09** obstacles per target reached. This is a significant improvement compared to the baseline, as not only the agent is able to reach the target faster, reaching 2.6 times more targets, but also with less collisions.

6.3. Harder penalty

The third experiment implemented curriculum learning into the training pipeline, and added a harsher penalty to the agent. The curriculum is structured as the above experiment, with the exception of the new parameter Penalty offset. Following are its settings:

1. Curriculum parameters:

Reward thresholds	1	2	2.5	2.8	3	3.5	4
Number of obstacles	8	10	13	15	17	18	20
Min spawn distance	6	6	4	4	3	3	2
Target distance	25	28	30	33	35	37	40
Penalty offset	0.5	1.5	2	2.5	2.5	2.5	2.5

2. Harder penalty function: $p = p_{offset} + c_o + time * 0.001$
3. Observations source: LIDAR set

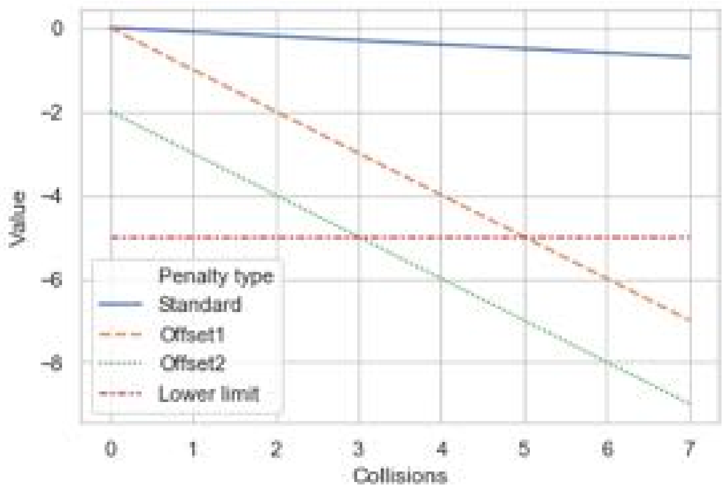


Figure 4: Penalty comparison: standard is the penalty used in most settings in the project, Offset1 and Offset2 are the penalties from the third experiment at different stages of difficulty, and the Cumulative Reward lower limit shows the number of collision before the episode aborts: Offset1 can get out with at most 5 collisions, Offset2 has a maximum of 3

As for the previous situation, the parameters generate a harder environment as the cumulative reward increase, but this time the penalty function too increases in difficulty. The rationale of this experiment is that, as the agent learns how to move to reach the target, the agent should learn to not collide frequently, but instead just search in the environment for the target smoothly. Figure 4 shows how the different penalty relate to each other and the Cumulative Reward lower limit without considering time decrease (same in all penalties).

Table 1 shows how the model collides roughly 4.5 times per minute and manages to reach a target about 3.9 and times per minute, making roughly **1.18** obstacles per target reached. This model obtains results similar to the baseline, while staying below the performances the curriculum model obtained. This may be due to the harsher penalty that does not give the model time to adapt to an optimal policy.

6.4. Camera sensors

The fourth experiment implemented the same curriculum and penalty as the second experiment. The main difference consists in the use of a camera sensor instead of the LIDAR array, thus generating images as observations.

1. Curriculum parameters: same as the second experiment
2. Penalty function: same as the second experiment
3. Observations source: Camera 84x84 RGB

The model performs significantly worse than the others. This is plausibly due to the low number of steps taken by the training of the model (74k) compared to the others (700k), which

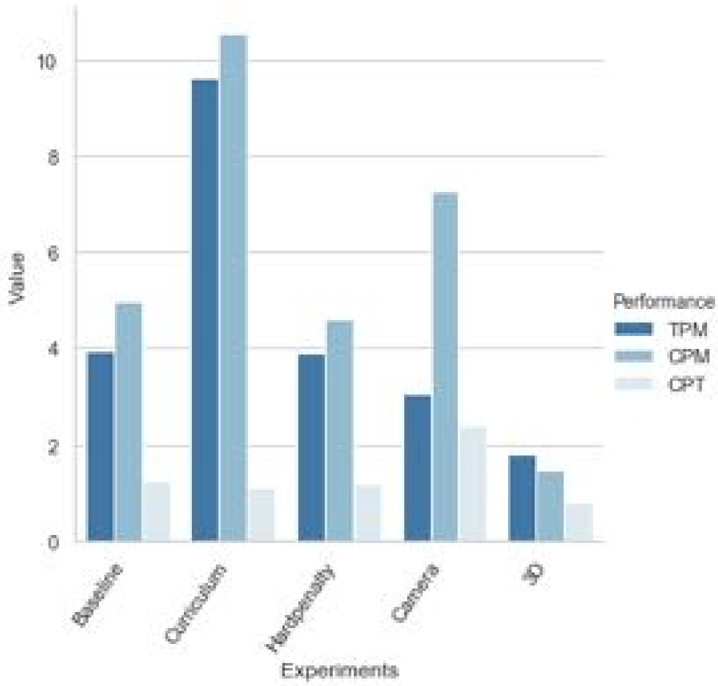


Figure 5: Environmental performances comparisons between the different random experiments

Table 1
Comparison of performance of the different experimented training approaches.

Training approach	CPM	TPM	CPT
Baseline	4,984	3,96	1,25
Curriculum	10,568	9,616	1,09
Hardpenalty	4,592	3,896	1,18
Camera	7,264	3,048	2,38

did not let the algorithm converge to an optimal policy. We must add that the choice not to investigate a longer training phase is due to the fact that the need to analyse this heavier form of input, that nonetheless might not necessarily be more informative, made the training much more expensive in terms of computation time, so whereas the steps are less than in the other experiments, the overall computation time for learning is very similar. The model manages to reach a target with roughly 2.5 obstacles hits each time, but taking roughly 3 times as much as the optimal curriculum model.

6.5. Structured environment transferability

Taking the best performing model (i.e. Curriculum), the experiment consisted in testing how well does the model generalize its task to structured environments - and how well does what

Table 2
Summary of the performance of the curriculum based model (achieved by training in random scenarios) applied to structured environments.

Environment	CPM	TPM	CPT
Rooms	7,2	1,8	4
Corridor	4,8	14,2	0,34
Corridor tight	29,6	2,2	13,45
Turn	10,4	10,4	1
Crossroad	23,4	9,6	2,44

the model learned during training in the chaotic environments transfer to other structured environments, which consist in:

1. "Rooms", two rooms are linked by a tight space in a wall, the agent has to pass through the opening to reach the target;
2. "Corridor" is a long environment - literally a corridor - that the agent has to run across to reach the target;
3. "Corridor tight" is similar to the previous environment but tighter
4. "Turn" is a corridor including a 90° left turn
5. "Crossroad" represents two ways cross and the agent has either to go straight on, to turn left or to turn right.

These environments tested the capability of the agent to follow directions and to look for the target in every space of the scene.

The model does not perform as well as in every environments. This seems to be due the strategy that the model has learned to be optimal for the resolution of the task at hand, seems to be random exploration, and will be addressed later on. The agent seems able to follow a linear path, if the environment is wide enough to allow it to stay away from the borders, it is apparent comparing Corridor and Corridor Tight outcomes: the second experiment has less targets and more collisions per minute then the first one. The model's performances in Crossroad and Turn are similar but, in the first case the agent has to change path more often then in the second one, so collisions happen more frequently. Rooms experiment has low both TPM and CPM because the agent tends to stay in the spawn room, without attempting to pass from the door.

6.6. Performance measures comparison

The models have comparable performances: the curriculum, hardpenalty and camera models reach a similar cumulative reward - with the camera ending on top, followed by the curriculum and then hardpenalty models. This comparison shows the first caveat of the experiments: cumulative reward notwithstanding, the curriculum model achieved way higher environmental performances than the other two models. The comparison makes even more sense if compared to the baseline model, which almost perfectly converged on a higher cumulative reward, but even then, the curriculum model achieved better performances, even with a lower cumulative reward.

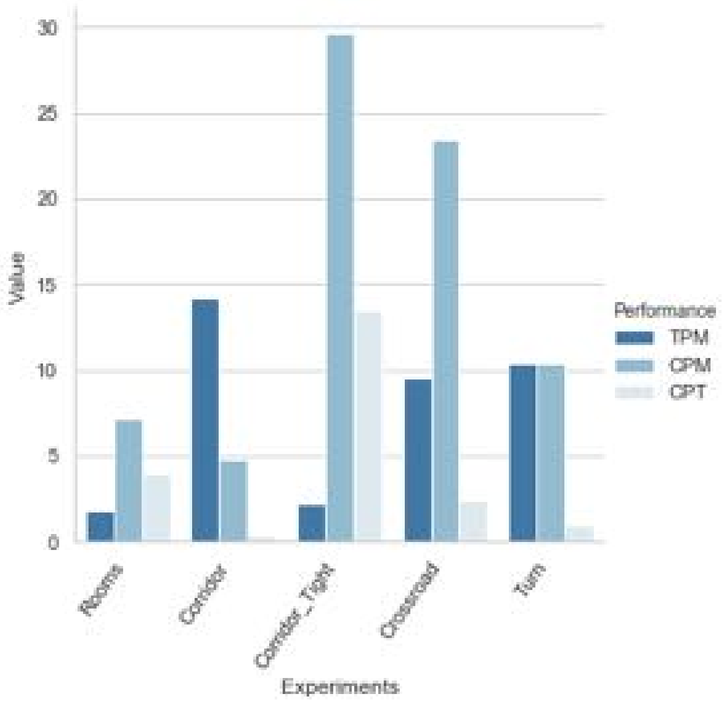


Figure 6: Environmental performances comparisons between the different structured experiments

We also performed experiments in a 3D version of the environment, whose complete description is omitted for sake of space, but in which (intuitively) actions included the possibility to maneuver in the Z axis as well, increasing significantly the dimension of the state space. The measurements of such a 3D maneuvering model took a very long time to converge and its lack of progress in the curriculum shows how the model still hasn't converged to an optimal policy.

For the models employing curriculum learning, we discovered a tendency for the models to reach a plateau early on in terms of steps. This may be due to the manual thresholding setup, which does not accurately increase difficulty. The only model which the learning process managed to reach the last difficulty level is the curriculum model.

It is of note how the distribution of reward of the models capable of obtaining a working policy shifts in time from the curiosity itself to the environmental reward, meaning a significant shift between curiosity-driven exploration and exploitation of the strategy learnt by the policy.

7. Conclusions

The empirical results show a certain decoupling between the environmental measures and the classical performance measures. Measuring performance in reinforcement learning settings is well-known to be tricky, as not always cumulative rewards, policy losses and other low-level measures are able to capture the effectiveness of the behaviour of the agent in the setting.

In the proposed setting, the experiments showed how curriculum learning can be an effective solution for improving the generalizing capabilities of the model, improving significantly how the agent behaves.

The proposed learning system shows that the policy that most commonly is converged to is essentially a random search strategy: the agent randomly explores the environment to find the target. This is demonstrated by the behaviours of the different models - to different levels of performances - which shows the agent randomly moving between obstacles, revisiting previously seen areas until he manages to get the target in the range of its sensors. This is probably due to the random nature of the environment generation, as no two episodes present the same environment, and as such the agent isn't able to memorize the layout of the environment (or portions of it), but is only able to generally try to avoid obstacles until the targets comes into sight.

This consideration represents a reasonable way to interpret results in environments whose structure is closer to the human built environment: whenever looking around to see if the target is finally in sight and moving towards it is possible without excessive risk of hitting an obstacles, the process yields good results, otherwise it leads to a failure. While this does not represent a negative result per se, it is a clear warning that the learning procedure, the environments employed to shape the learning process, can lead to unforeseen and undesirable results. It must be stressed that, whereas this is a sort of malicious exploitation of a behavioural model that was trained on some types of environments and that is being tested in different situations, other state of the art approaches specifically devised and trained to achieve proper pedestrian dynamics still do not produce results that are competitive with hand crafted models [11].

Possible future developments on the RL model side are:

1. **Implement memory**: adding memory to the agent (in the form of a RNN module) might allow it to form a sort of experience buffer for the current episode and allows it to explore the environment in a non-random fashion.
2. **Rework the reward and penalty functions**: the proposed reward and penalty are pretty simplistic, a possible enhancement to the penalty could be implementing soft-collisions, that is, scaling the negative reward obtained by the agent in a collision according to the velocity of the collision - safe, soft touches can be allowed.
3. **Compare different RL algorithms**: different reinforcement learning algorithms (A3C, DQN) might show different insights on the optimal way to implement intelligent agents in the proposed setting.

On the other hand, a different training procedure, including specific environments aimed at representing a sort of *grammar* of the built environment, that could be used to define a sort of curriculum for training specifically pedestrian agents, should be defined to more specifically evaluate the plausibility of the application of this RL approach to pedestrian modeling and simulation.

References

[1] R. S. Sutton, A. G. Barto, Reinforcement Learning: an Introduction, MIT press Cambridge, 2018.

[2] S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach (4th ed.), Pearson, 2020.
[3] S. Bandini, S. Manzoni, G. Vizzari, Agent based modeling and simulation: An informatics perspective, Journal of Artificial Societies and Social Simulation 12 (2009) 4.
[4] D. Weyns, A. Omicini, J. Odell, Environment as a first class abstraction in multiagent systems, Autonomous Agents Multi-Agent Systems 14 (2007) 5–30.
[5] G. Vizzari, L. Crociani, S. Bandini, An agent-based model for plausible wayfinding in pedestrian simulation, Eng. Appl. Artif. Intell. 87 (2020). URL: https://doi.org/10.1016/j.engappai.2019.103241. doi:10.1016/j.engappai.2019.103241.
[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, CoRR abs/1707.06347 (2017). URL: http://arxiv.org/abs/1707.06347. arXiv:1707.06347.
[7] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, D. Silver, Emergence of locomotion behaviours in rich environments, CoRR abs/1707.02286 (2017). URL: http://arxiv.org/abs/1707.02286. arXiv:1707.02286.
[8] W. Yu, G. Turk, C. K. Liu, Learning symmetric and low-energy locomotion, ACM Trans. Graph. 37 (2018) 144:1–144:12. URL: https://doi.org/10.1145/3197517.3201397. doi:10.1145/3197517.3201397.
[9] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, P. Moritz, Trust region policy optimization, in: F. R. Bach, D. M. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, volume 37 of *JMLR Workshop and Conference Proceedings*, JMLR.org, 2015, pp. 1889–1897. URL: http://proceedings.mlr.press/v37/schulman15.html.
[10] D. Pathak, P. Agrawal, A. A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: D. Precup, Y. W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 2778–2787. URL: http://proceedings.mlr.press/v70/pathak17a.html.
[11] F. Martinez-Gil, M. Lozano, F. Fernández-Rebollo, Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models, Simul. Model. Pract. Theory 74 (2017) 117–133. URL: https://doi.org/10.1016/j.simpat.2017.03.003. doi:10.1016/j.simpat.2017.03.003.

Neuro-symbolic Computation for XAI: Towards a Unified Model

Giuseppe Pisano[a], Giovanni Ciatto[b], Roberta Calegari[a] and Andrea Omicini[b]

[a] *Alma AI – Alma Mater Research Institute for Human-Centered Artificial Intelligence, Alma Mater Studiorum—Università di Bologna, Italy*
[b] *Dipartimento di Informatica – Scienza e Ingegneria (DISI), Alma Mater Studiorum—Università di Bologna, Italy*

Abstract
The idea of integrating symbolic and sub-symbolic approaches to make intelligent systems (IS) understandable and explainable is at the core of new fields such as *neuro-symbolic computing* (NSC). This work lays under the umbrella of NSC, and aims at a twofold objective. First, we present a set of *guidelines* aimed at building explainable IS, which leverage on logic *induction* and *constraints* to integrate symbolic and sub-symbolic approaches. Then, we reify the proposed guidelines into a case study to show their effectiveness and potential, presenting a prototype built on the top of some NSC technologies.

Keywords
XAI, Hybrid Systems, Neural Networks, Logical Constraining

1. Introduction

In the last decade, we have witnessed an unprecedented spread of artificial intelligence (AI) and its related technologies [1]. The fields involved are manifold, ranging from autonomous driving systems and expert systems to computer vision and reasoning systems—just to mention a few. In all the aforementioned fields, AI is enabling artificial systems to act in a more intelligent, efficient, and effective way.

Besides the many impactful achievements, some factors have emerged that can slow down the further diffusion of AI technologies. A primary concern is related to the *trustability* of intelligent systems (IS) leveraging on sub-symbolic AI—i.e., exploiting approaches such as deep learning. Indeed, the resulting IS suffers from well-known problems of *opaqueness*, since humans typically find very difficult to understand how sub-symbolic systems work.

The need to overcome opaqueness is one of the main goal of the research in *eXplainable AI* (XAI) [2], essentially aimed at making AI systems more understandable and explainable. Some of the most interesting XAI techniques are deeply rooted in symbolic AI—as representatives of

21st Workshop "From Objects to Agents" (WOA), September 14–16, 2020, Bologna, Italy
g.pisano@unibo.it (G. Pisano); giovanni.ciatto@unibo.it (G. Ciatto); roberta.calegari@unibo.it (R. Calegari); andrea.omicini@unibo.it (A. Omicini)
https://about.me/gciatto (G. Ciatto); http://robertacalegari.apice.unibo.it (R. Calegari); http://andreaomicini.apice.unibo.it (A. Omicini)
0000-0003-0230-8212 (G. Pisano); 0000-0002-1841-8996 (G. Ciatto); 0000-0003-3794-2942 (R. Calegari); 0000-0002-6655-3869 (A. Omicini)

a natural choice for building human-intelligible IS [3]. In fact, symbolic systems offer a human-understandable representation of their internal knowledge and processes: so, integrating them into sub-symbolic models – to promote transparency of the resulting system – is the most prominent stimulus for new research fields such as *neuro-symbolic computing* (NSC) [4].

Our work lays under the umbrella of NSC, and its contribution is twofold. On the one hand, we present a set of *guidelines* aimed at building explainable IS, even when they exploit sub-symbolic techniques. The guidelines leverage on *logic induction* and *logic constraints* as the two main techniques integrating symbolic and sub-symbolic approaches. In particular, logic induction makes it possible to extract the knowledge from black-box ML-based predictors – typically, the sub-symbolic part of an IS – offering a corresponding symbolic, logical representation. Conversely, logic constraints are exploited to inject some logic knowledge into the black box, thus restricting the numerical underlying model.

On the other hand, we reify the proposed guidelines into a case study to show their effectiveness and potential. In particular, we present a prototype built over some promising NSC technologies. The resulting system is then assessed to verify its capability of being adjusted (i.e., debugged and fixed) in case some unexpected behaviour in the sub-symbolic part of the system is revealed. Accordingly, we show how the prototype is correctly performing w.r.t. the proposed guidelines.

The paper is structured as follows. Section 2 provides a brief overview of the field of symbolic and sub-symbolic integration. It also includes the main related works on the use of a hybrid system as a mean for explainability. In Section 3, we first present the guidelines for building explainable systems; then, in Section 4 we discuss a possible instantiation of the proposed guidelines. In Section 5, we proceed with the assessment of our prototype and the discussion of results. Finally, Section 6 concludes the work.

2. Background

In the last years, deep and machine (ML) learning methods have become largely popular and successful in real-world intelligent systems. However, their use raises the issue of understanding and explaining their behaviour to humans. Neural networks in particular – which are the most hyped and widely adopted approach within sub-symbolic AI – mostly suffer from the problem of opaqueness: the way they obtain their results and acquire experience from data is unintelligible to humans.

One of the proposed approaches addressing the explainability problem [2] is the hybridisation of symbolic and sub-symbolic techniques [3]. An increasing number of authors recognises that formal logic is capable of significantly improving humans' understanding of data [5]. In their view, in principle, an opaque system, combined with a symbolic model, can provide a significant result in terms of transparency. Many researches start from these assumptions.

Among the others, the neuro-symbolic computing (NSC) field is a very recent and promising research area whose ultimate goal is to make symbolic and sub-symbolic AI techniques effortlessly work together. Due to the freshness of the topic, however, a well-established and coherent theory of NSC is still missing. For this reason, a variety of methods have been proposed so far, focusing on a multitude of aspects not always addressing interpretability and explainability as their major concern. Nevertheless, some attempts to categorise XAI-related existing works under the NSC umbrella exist [4, 3], which provide for a helpful overview of the topic.

Despite NSC does not explicitly include XAI among its primary goals, in this paper we borrow some ideas from the NSC field to show how the explainability of modern IS may benefit from the integration of symbolic and sub-symbolic AI. In particular, our work focuses on two main NSC sub-fields: namely, the *logic as a constraint*, for the constraining module, and *differentiable programming* for the induction module—since the proposed prototype exploits logic induction via differentiable programming [6]. In short, the former line of research aims at constraining the training process of a sub-symbolic predictor in such a way that it cannot violate the superimposed constraint at runtime. About the latter, differentiable programming is the combination of neural networks approaches with algorithmic modules in an end-to-end differentiable model, often exploiting optimisation algorithms like gradient descent [7].

Within the scope of *logic as a constraint*, most approaches exploit some sort of *logic* formulæ to constrain the behaviour of the sub-symbolic predictor—in most cases, a neural network. This formula is then *vectorised* – i.e. translated into a continuous function over vectors of real numbers – and exploited as a regularisation term in the loss function used for training the sub-symbolic predictor [8, 9, 10]. Different strategies have been proposed to this end. For example, in [11] the symbolic constraints are used to modify the network structure incorporating them into the training process. In the general case, however, logic constraining can be used to fix bias or bugs in the behaviour of a sub-symbolic system, or, it can mitigate the situation in which poor training data is available to correctly train a black-box system on a specific aspect.

With respect to the second research area, some works laying under the umbrella of differentiable programming fruitfully intertwine ML and inductive logic programming (ILP) [12] to provide logic induction capabilities on top of sub-symbolic predictors. ILP is a well established research area, laying at the intersection of ML and logic programming, which is strongly interrelated with NSC. An ILP system is a tool able to induce (i.e., derive) – given an encoding of some background knowledge and a set of positive and negative examples represented logic facts –, a logic program that entails all the positive and none of the negative examples. While traditionally these systems base their operation on the use of complex algorithms as their core component [13] – deeply undermining their efficiency and usability – hybrid approaches exist leveraging NSC to make the induction process more efficient [6]. Furthermore, as we show in this paper, induced logic rules can be used as a means to *inspect* what a black-box predictor has learned—as induction makes the predictor knowledge explicit in symbolic form.

2.1. Related Works

As far as the intersection of logical systems and numerical models is concerned, the main contributions come from [14] and [15]. Their work can be summarised in:

- usage of a knowledge base filled automatically from training data to reason about what has been learned and to provide explanations;
- adoption of logic rules to constrain the network and to correct its biases.

Although these works offer a good starting point in the search of a solution to the transparency problem, some remarks should be pointed out. First, exploiting a knowledge base obtained only from the training data is not sufficient to acquire the knowledge required to explain the entire network behaviour. That would lead to a system giving explanations according to the network optimal functioning, not accounting for the training errors. Moreover, according to these models, also the constraining part should be driven by the rules inferred from the training data, hardly limiting the potential of those techniques. Indeed, the possibility for users to impose their own rules would also give them the ability to mitigate the errors derived from an incomplete or incorrect training set.

The work presented here aims at building a model overcoming both these limitations. As for the explanations' coherence problem, the use of the black box as a data source in the logic induction process should guarantee the correct correlation between the black box itself and the derived logic theory. Furthermore, logic can be leveraged so as to combine the IS with the user experience and knowledge, thus exploiting all the advantages of the constraining techniques.

3. A NSC model for XAI

In this paper, we present a general model for explainable data-driven IS, and a set of guidelines supporting their construction. The novelty of our approach lays in the fruitful intertwining of symbolic and sub-symbolic AI, which aims at providing both predictive accuracy – through the exploitation of state-of-the-art machine learning (ML) techniques – and transparency—through the exploitation of computational logic and logic programming (LP). The proposed model, in particular, aims at overcoming the well-known limitations of ML-based AI w.r.t. interpretability. Accordingly, it leverages on a number of contributions from the NSC and LP research field, as well as two basic techniques—namely, induction and constraining.

The main idea behind our work is that IS should feature both predictive precision and interpretability features. To preserve predictive precision, IS should keep exploiting high-performance, data-driven, black-box predictors such as (deep) neural networks. To overcome the interpretability-related issues, IS should couple sub-symbolic approaches with logic theories obtained by automatically extract the sub-symbolic knowledge of black boxes into symbolic form. This would in turn enable a number of XAI-related features for IS, providing human users with the capabilities of *(i)* inspecting a black box – also for debugging purposes –, and *(ii)* correcting the system behaviour by providing novel symbolic specifications.

Accordingly, in the reminder of this section, we provide further details about the *desiderata* which led to the definition of our model. We then provide an abstract description of our model and a set of guidelines for software engineers and developers. Finally, we provide a technological architecture to assess both the model and the guidelines.

3.1. Desiderata

Regardless of the architectural and technological choices performed by designers and developers, the IS adhering to our model are characterised by several common *desiderata* ($\mathbf{D_i}$) w.r.t. their overall functioning and behaviour. Generally speaking, these are aimed at making IS both prediction-effective and explainable. Indeed, following this purpose, IS should

$\mathbf{D_1}$ attain high predictive performances by leveraging ML and data-driven AI

$\mathbf{D_2}$ provide human-intelligible outcomes / suggestions / recommendations

$\mathbf{D_3}$ acquire knowledge from both data and from high-level specifications

$\mathbf{D_4}$ make their knowledge base inspectable by human experts

$\mathbf{D_5}$ let human experts override / modify their knowledge base

A key enabling point in satisfying these *desiderata* is knowledge representation. While ML and data-driven AI are certainly required to mine effective information from data efficiently, they soon fall short when it comes to satisfying *desiderata* $\mathbf{D_2}$–$\mathbf{D_5}$. This happens because they mostly leverage on a *distributed*, sub-symbolic representation of knowledge which is hard to interpret for human beings. Therefore, to support $\mathbf{D_2}$ and $\mathbf{D_4}$, we need an alternative human-intelligible representation of the sub-symbolic model and a procedure to perform such a representation transformation. Furthermore, to support $\mathbf{D_3}$ and $\mathbf{D_5}$, we also need to link symbolic and sub-symbolic representations in a bidirectional way—meaning that an inverse procedure aimed at converting symbolic information back into sub-symbolic form is needed as well.

Accordingly, the focus of our model is both on the extraction of symbolic representation from the black-box predictor and, vice-versa, on the injection of symbolic representation (constraints) in the corresponding black-box predictor. The purpose of this dichotomy is twofold: guaranteeing the comprehensibility of the black-box model for humans – as symbolic representations are to some extent inherently intelligible –, and enabling debugging and correction of the black-box behaviour, in case some issue is found through inspection.

3.2. Modelling

Generally speaking, we model an explainable, hybrid IS as composed by a black box, a knowledge base (KB) and an automatic logic reasoner, as depicted in Figure 1. Explainable recommendations or suggestions are provided to the end-user via the logic reasoner – based on symbolic rules and facts included in the KB by domain experts – and via black-box predictions—based on data. Accordingly, the reasoner combines several sorts of inference mechanisms (e.g. deduction and induction). Furthermore, to balance the knowledge coming from data with the domain experts knowledge, the model exploits induction and constraining techniques to improve the black box with the logical knowledge and vice-versa.

The black-box module is the core of the IS, making it capable of mining effective information from data. Any sub-symbolic model providing high predictive performances – e.g. neural networks, SVM, generalised linear models, etc. – can be used for the module implementation. This, however, may bring opaqueness-related issues. Thus, the black-box module is complemented with the other two modules to provide explanation and debugging facilities.

The reasoner module is aimed at providing explainable outcomes to the end-users. In particular, explanations are given in terms of logic KB, capable of approximating the black box. The construction of the logic KB relies on the *induction* capabilities offered by this module. More precisely, the outcomes generated by the black box are exploited to build a logic theory, mimicking the work of the black-box predictor with the highest possible fidelity. An inductive

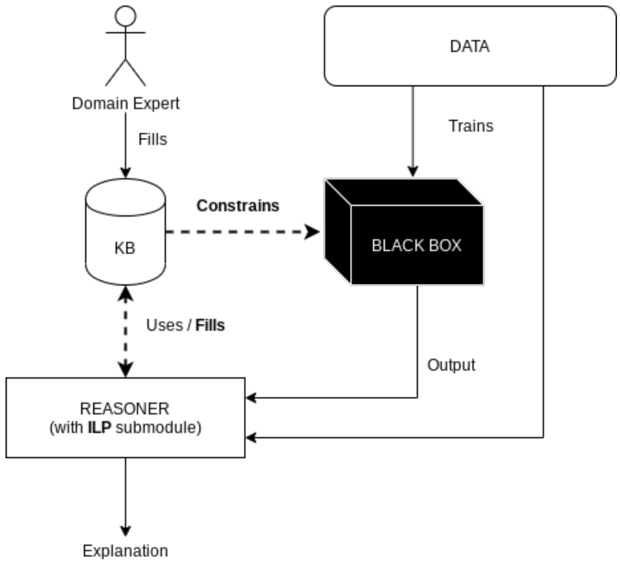


Figure 1: Model schema.

process is then fed with the resulting extended theory. This leads to a theory containing a number of general relations, which can be exploited to provide intelligible information to the end-users. The described workflow supports explainability in two ways: *(i)* it provides a global explanation of how the black box works in terms of general relations that must always hold; *(ii)* it provides local explanations, justifying each black-box conclusion by enabling the use of deductive reasoning on the connected logical knowledge. In other words, this mechanism is what enables IS to be *interpretable* and *debuggable* by users. Finally, the KB module aims at storing the logical knowledge approximating the black-box behaviour. The knowledge can be modified by domain experts. When this is the case, it becomes of paramount importance to keep the black-box module coherent with the human-proposed edits. To this end, constraining is performed to align the black-box behaviour with the KB. This mechanism is what enables IS to be *fixed* by users.

In the reminder of this section, we delve deeper into the details of these mechanisms.

3.2.1. Logic induction

While deductive reasoning moves from universal principles that are certainly true to specific conclusions that can be mechanically derived, inductive reasoning moves from specific instances to general conclusions. Induction is a key mechanism of our model. Assuming IS can transform the data they leverage upon for training black boxes into theories of logic facts, inductive reasoning can be used to extract the rules that best explain that data.

The induction procedure is not meant to replace the black box as a learning or data-mining tool—as it would be quite difficult to obtain the same performance in terms of accuracy and ability to scale over huge amounts of data. Conversely, it aims at "opening the black box", letting humans understand how it is performing and why. In other words, following the abstract framework presented in [16, 17], logic induction is a means for *explaining* the functioning of a black-box predictor via symbolic rules. More precisely, induction is fundamental for the debuggability of our model. For example, it enables the discovery of fallacies in the learning process of a black box even for people not used to the specific domain.

In our model, the induction process is fed with both the raw data and the outcomes of the black box. In the former case, induction leads to the unveiling of latent relations possibly buried in the original data: we refer to the logic theory obtained as the *reference theory*. In the latter case, induction leads to a symbolic approximation of the knowledge the black box has acquired from data via ML: we refer to the logic theory obtained as *explanation theory*. Assuming the soundness of the induction process, discrepancies between these two theories could reveal some possible rules that have not been correctly learned by the sub-symbolic model. It is then possible to fix it by enforcing the correct relations via logic constraining.

Finally, one last point is worth to be discussed. Unless the induction process possesses the same learning capabilities of the black box, it is impossible to detect all its learning errors. In this case, in fact, the reference theory would be the optimal solution itself, and the sub-symbolic model would be useless. As the induction process aims at opening the box, its use on the raw data aims at providing insights on the accuracy of the training phase. Thus, it does not aim at providing an optimal solution.

3.2.2. Logic constraining

While induction is the mechanism aimed at translating sub-symbolic knowledge into symbolic form, constraining is the inverse process aimed at injecting some symbolic knowledge into a black box. In this way, both the induced rules and those coming from domain experts are used to constrain the black box and its outcomes. This is another key mechanism of our model.

In particular, the ability to encode some prior knowledge within a model is interesting for two main reasons. On the one side, it enables a reduction in the data needed to train the black box. In fact, handmade rules may be exploited to model a portion of the domain not included in the training data. So, rather than creating a more exhaustive training set, an expert may directly encode his/her knowledge into rule and train a constrained black box. We call this procedure *domain augmentation*. On the other side, one may exploit the constraining process to guide and support a black-box learning, e.g., helping it to avoid biases. We call this procedure *bias correction*.

3.3. Guidelines

In the following, we discuss the main aspects to be considered when designing a system conforming to our model. As a first step in this direction, we first handle some important aspects concerning data representation.

3.3.1. Type of logic

The first aspect concerns the type of logic used by the IS. Logic, in fact, plays a fundamental role: it heavily affects the explainability properties of the final system. For this reason, the choice of the most appropriate logic formalism is a primary concern.

For our model to be effective, the selected logic formalism should provide high flexibility in the description of the domain. At the same time, to keep a system as simple as possible, it should be coherent in every part of it: from the constraining module to the induction one, every part should share the same representation of the logical knowledge.

3.3.2. Sort of data

The second aspect concerns the sort of data used to feed the IS. For the logical induction process to be carried out on data, the first thing to do is transforming the data itself. In other words, the available data should be translated into a logical theory according to the chosen formalism. For the translation to be effective, the resulting theory should preserve as much as possible the information contained in the original data. Anyway, when dealing with some particular type of data some problems have to be taken into account.

When unstructured data – e.g., images, audio, time-series, etc – come into play, the transformation process may become considerably harder. In that case, a two-step transformation must be performed, involving:

1. extraction of semantically meaningful, structured features from unstructured data;
2. translation of extracted features into the desired logical formalism.

Of course, this procedure adds a whole range of new problems related to the accuracy of the features extracted. In fact, the explanation process is mainly linked to the reliability of the data used as the basis for the induction process. The exploitation of data generated through an automatic process makes a discrepancy between the data extrapolated by the inductive process and the behaviour of the black box more likely.

3.3.3. Sorts of black box

In the general case, we require the architectures adhering our model to be as agnostic as possible w.r.t. the particular sort of black box to be adopted. In fact, the choice of the most adequate sort of black box is strongly affected by *(i)* the nature of the data at hand, *(ii)* the availability of some symbolic extraction technique making the black box less opaque, *(iii)* the possibility of constraining the black box with logic formulæ.

Strictly speaking, the choice of the black box should be delayed to the implementation phase. Here we describe a number of criteria to be taken into account when performing this choice. We recall, however, that the other guidelines described so far are agnostic w.r.t. the sub-symbolic model to be used.

As far as the nature of the data is concerned, traditional ML techniques [18] – e.g., decision trees, generalised linear models, etc. – are usually exploited on structured datasets, whereas deep learning techniques are better suited to deal with unstructured data. However, this is not due to an inadequacy of neural networks for structured data, but rather to the greater simplicity of the learning algorithms used by traditional ML techniques. Structured data delivers a relatively-smaller complexity to deal with, making simpler ML algorithms a more suitable choice for their comprehensibility and usability.

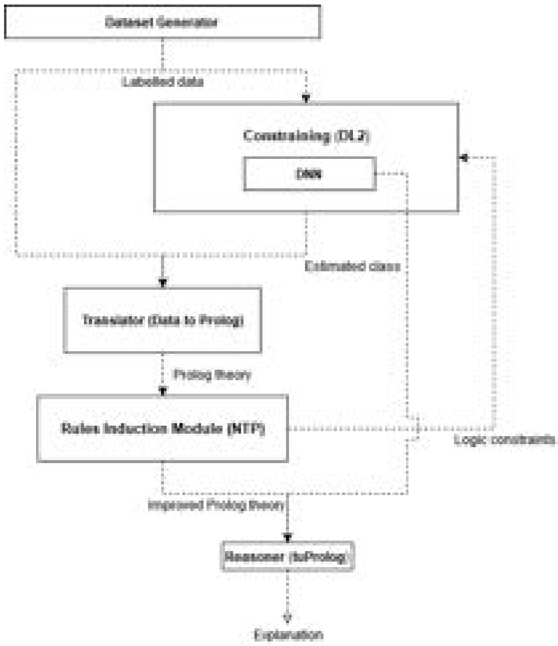


Figure 2: Technologies schema.

As far as opaqueness is concerned, virtually any ML techniques is affected by that to some extent: yet, neural networks remain the most critical from this point of view. Nevertheless, the vast majority of rule induction procedures are either black-box agnostic – a.k.a. pedagogical [19] – or neural-network-specific [3]. In terms of the support for constraining, it should be possible to guide the learning activity trough some regulariser terms attained from the constraints to be enforced.

4. Technological architecture

This subsection provides a technological description of an IS prototype adhering to our model. It is based on the concrete architecture detailed in Figure 2, which specialises the abstract model from Figure 1 through a number of technological commitments.

First, we present our choices in relation to the points examined in Subsection 3.3. As for the type of logic, within the scope of this paper, we adopt first-order logic (FOL) as our formalism of choice, and Prolog as our reference concrete language. FOL is likely to offer the best trade-off between flexibility and expressiveness of the representation language. Furthermore, the choice of FOL enables the exploitation of many existing approaches supporting both conditioning and induction. Finally, the choice of the Prolog syntax enables the direct exploitation of the induced theories within existing automatic reasoners—e.g. tuProlog, SWI-Prolog, etc.

As far as the sort of the data is concerned, in this paper we only describe a prototype based on structured data. In fact, the feature extraction module can be omitted without hindering the generality of our approach. Moreover, the greater simplicity in the data helps to avoid the problems related to the possible information loss due to their transformation. In the future, however, we plan to extend our prototype to support arbitrary datasets including any sort of data. This requires the creation of a module for feature extraction.

In terms of the sort of the black-box used, we focus on a prototype based on neural networks, being them the most critical from the opacity perspective. The remaining technological choices derive consequently.

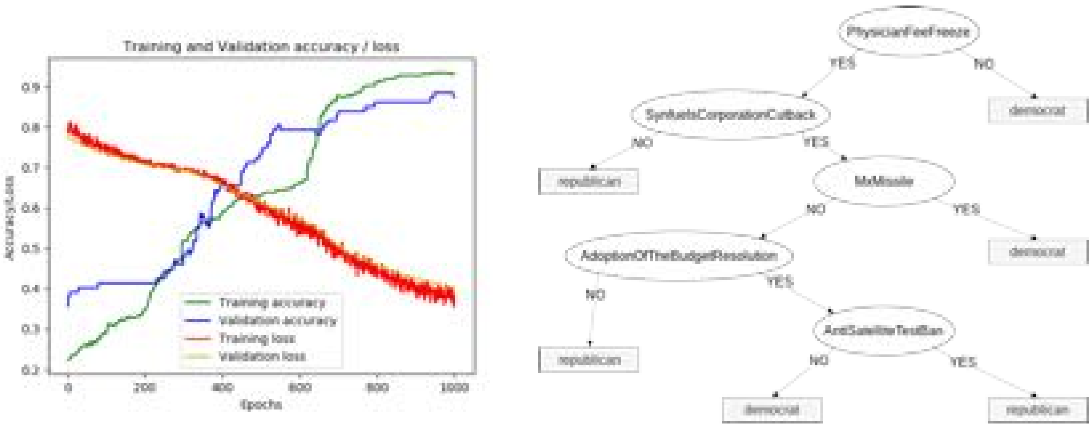
The prototype exploits two main technologies: DL2 [9] for the conditioning part, and NTP [6][20] for the induction part. DL2 is one of those models leveraging on symbolic rules vectorisation as means to constrain the target neural network. We choose DL2 as the most mature and user-friendly technology supporting neural-network constraining through logic rules.

The choice of NTP as the induction engine is more straightforward. Indeed, NTP is among the few ready-to-use technologies offering both deductive and inductive logic capabilities in a differentiable way. On the one hand, *differentiability* is what makes induction more computationally efficient w.r.t. similar technologies—and this is why we choose it. On the other hand, NTP deductive capabilities are not mature enough. In fact, training a model correctly approximating a logic theory in a reasonable amount of time is very challenging—especially when the complexity of the theory grows. While this could be acceptable for the induction process, it is still very limiting for deductive reasoning. Moreover, traditional logic engines combine a very large ecosystem of supporting tools – IDEs, debuggers, libraries – that have potential to hugely improve the effectiveness of the reasoning phase. For this reason, we adopt tuProlog (2P) [21] – a Java-based logic engine built for use in ubiquitous contexts – as the main tool for the manipulation of logic theories, as well for automated reasoning. This choice is motivated by its reliability, modularity, and flexibility.

While the entire system is built around the aforementioned technologies, Python is used as the glue language to keep modules together. In particular, the sub-symbolic module is implemented via the PyTorch learning framework [22], thus ensuring an easy integration with DL2—which is natively built to work with this framework. The modules responsible for the extraction of the Prolog theory from the input dataset (i.e. the Translator block in Figure 2) are created using Python as well. The NTP integration takes place through the Prolog logic language. In fact, NTP easily allows the use of Prolog theories as input for the induction process. The result of the induction phase is also delivered in a logical form, allowing for an easy consultation and analysis through the 2P technology.

5. Assessment

In this section we assess our model from the explainability perspective leveraging on the prototype implementation described in Section 4. Particularly, we train the sub-symbolic module through a real-world ML problem, and we test whether and to what extent our architecture actually provides *(i)* the capability of building a logical representation of sub-symbolic knowledge acquired from data via induction, *(ii)* the capability of altering or fixing the system behaviour via conditioning, and, ultimately, *(iii)* the inspectability and debuggability of the system as a whole. In the following subsections we set up an *ad-hoc* experiment aimed at performing these



(a) Network default training (b) Decision tree built on test data

Figure 3: Network default training (a) and decision tree built on test data (b).

tests. A detailed presentation of the experiments and their results follows.

5.1. Experiment Design

The proposed experiment aims at comprehensively testing the operation of the prototype against some real classification problem. In particular, we consider a binary classification task on structured data. Roughly speaking, the experiment works by artificially constructing a buggy neural network for the classification task, and then showing how our architecture makes it possible to reveal the bug. More precisely, the experiment is structured as follows:

1. a neural network is trained until reaching maximum validation-set accuracy for the classification task;
2. by combining the training data and the predictions of the network, a coherent Prolog theory is extracted;
3. the induction module is used to extract the latent relations from the theory, until at least one relation which properly divides the data is found;
4. through constraining, we inject an error in the network, in such a way that it misclassify some instances;
5. by repeating Item 2 and Item 3, we show that the approximated logic theory reveals the injected error.

This workflow lets us verify the behaviour of our prototype and of all its components. In particular, Item 3 and Item 5 aim at demonstrating that a logic theory that optimally approximates a neural network is actually attainable. In terms of the ability of debugging and correcting a network, the whole procedure aims at demonstrating their feasibility. The extraction of the correct theory in the initial part of the experiment is comparable to the one extracted from a malfunctioning classifier. The inclusion of the fake constraint, as well as its recognition in the theory extracted at the conclusion of the experiment, shows the feasibility of the network correction process.

Two fundamental points are worth to be taken into account for the experiment to be meaningful. The first point is the ability to accurately evaluate the accuracy of the logic theory recovered. In fact, in order to verify that the theory extracted from the neural network is actually the correct one, it is either necessary *(i)* to have an optimal knowledge of the domain, or *(ii)* to use easily analysable and verifiable datasets. As for the first case – being an expert of the analysed domain – it turns out to be very easy, by verifying the correctness of the logical relations. The only alternative comes from the usage of an easily verifiable dataset as the base for the experiment. Indeed, it should be possible also for a domain novice to understand the ratio behind the data. For instance, a simple way to verify the correctness of the rules is to use an alternative ML tool – one that can guarantee a high level of transparency – to analyse the data. In the case of a classifier, the best choice could be a decision tree (DT). In fact, DTs training produces a symbolic model that can be efficiently verified by the user. Hence, DT output can be used as a reference in the evaluation of the induction results.

The second point is the exploitation of constraining as a means to inject bugs into a network in a controlled way. In fact, in order to verify the prototype capability of revealing and correcting bugs in the black-box module, it is first necessary to construct a buggy network. However, in case of a poorly training, as in that case, it can not be clear where the bug is. Hence, to evaluate the actual capabilities of the prototype, the bugs to be spotted must be *a priori* known in order to have a reference.

5.2. Experiment Setup

We base our experiment on the *Congressional Voting Records* data set[1] from the UCI Repository [23]. It consists of table registering the voting intentions – namely, favourable, contrary or unknown – of 435 members of the Congress of the USA on 15 main topics, as well as their political orientation—namely, either Democrat or Republican. The goal of the classification task is to sort a member of the Congress as either democratic or republican depending on his/her intention on those 15 topics.

Given the relatively small amount of instances in the dataset, along with the small number of attributes, an intuitive understanding of the classification problem can be assumed. However, to further simplify the analysis of the results, we use a DT trained over the data as a reference.

A neural network capable of distinguishing between Democrats and Republicans based on voting intentions is trained and assessed as described above. The experiments code is available at the Github repository[2]. We now proceed discussing the results we obtained in each step of the experiment.

[1] https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records
[2] https://github.com/Gilbocc/NSC4ExplainableAI

5.3. Results

The training of the aforementioned neural network is performed via ordinary stochastic gradient descent until reaching a 95% accuracy score on the validation set (containing a randomly selected 20% of whole data). The training process is depicted in Figure 3a.
Through the induction process on the data generated by the network it was possible to recover the following relationships:

$$
\begin{aligned}
democrat(X_1, \ldots, X_N) \ \text{:-}& \\
& opposePhysicianFeeFreeze(X_1, \ldots, X_N), \\
& supportMxMissile(X_1, \ldots, X_N), \\
& supportBudgetResolution(X_1, \ldots, X_N). \\
democrat(X_1, \ldots, X_N) \ \text{:-}& \\
& opposePhysicianFeeFreeze(X_1, \ldots, X_N), \\
& supportBudgetResolution(X_1, \ldots, X_N). \\
republican(X_1, \ldots, X_N) \ \text{:-}& \\
& supportPhysicianFeeFreeze(X_1, \ldots, X_N), \\
& opposeAntiSatelliteTestBan(X_1, \ldots, X_N). \\
republican(X_1, \ldots, X_N) \ \text{:-}& \\
& supportPhysicianFeeFreeze(X_1, \ldots, X_N), \\
& opposeAntiSatelliteTestBan(X_1, \ldots, X_N), \\
& opposeBudgetResolution(X_1, \ldots, X_N), \\
& opposeSynfuelsCutback(X_1, \ldots, X_N).
\end{aligned}
$$

To verify their validity we can examine the DT generated using the original data as source (Figure 3b)—the C4.5 algorithm has been used. As expected, the inductive process managed to recover the relationship that most discriminates between Democrats and Republicans: the tendency of the Democrats to be against the freezing of the cost of medical expenses.

The verification of the correction process is based on the reversal of the relationship retrieved above. Formally, the conditioning of the network occurred on this rule:

$$
\begin{aligned}
republican(X_1, \ldots, X_N) \ \text{:-}& \\
& opposePhysicianFeesFreeze(X_1, \ldots, X_N).
\end{aligned}
$$

In natural language, this Prolog rule expresses that, given a set of votes ($X_1, \ldots, X_N$), if these share the opposition to the freezing of medical expenses, then the voter is likely from the Republican Party. This constraint translates into a very simple result: all the votes should belong to Republicans. Indeed, if initially only the Democrats were against the freezing, by forcing the network to consider them as Republicans, we reach a situation where the whole set of voters is Republican.

More in detail, constraining rules are codified through the DSL offered by the D2L implementation. For example, the above rule can be expressed in the form:

```
dl2.Implication(
    dl2.EQ(x[FeesFreeze], 0),
    dl2.LT(y[Dem], y[Rep]))
```

Table 1
Training with fake constraint results.

	precision	**recall**	**f1-score**	**support**
repubblican	0.42	1.00	0.59	35
democrat	1.00	0.06	0.11	32
accuracy			0.44	87

This rule is then automatically converted in a numerical function. Its evaluation contributes to the final loss function adopted by the target neural network.

For the experiment, a completely-new network is trained considering the new constraint. The results in Table 1 show the effect of the constraint on the network. The Democrats/Republican imbalance – which is made evident by Table 1 – reflects in the result of the induction process on the data generated by the constrained network:

$$
\begin{aligned}
&republican(X_1, \ldots, X_N) \;\texttt{:-} \\
&\qquad supportMxMissile(X_1, \ldots, X_N), \\
&\qquad opposePhysicianFeeFreeze(X_1, \ldots, X_N). \\
&republican(X_1, \ldots, X_N) \;\texttt{:-} \\
&\qquad supportPhysicianFeeFreeze(X_1, \ldots, X_N), \\
&\qquad opposeBudgetResolution(X_1, \ldots, X_N), \\
&\qquad opposeMxMissile(X_1, \ldots, X_N). \\
&republican(X_1, \ldots, X_N) \;\texttt{:-} \\
&\qquad opposePhysicianFeeFreeze(X_1, \ldots, X_N), \\
&\qquad supportBudgetResolution(X_1, \ldots, X_N), \\
&\qquad supportAntiSatelliteTestBan(X_1, \ldots, X_N), \\
&\qquad supportMxMissile(X_1, \ldots, X_N).
\end{aligned}
$$

The inducted knowledge base only contains rules concerning the Republican wing, thus confirming the footprint of the relation imposed during the conditioning process.

Summarising, the results of the assessment are positive. It was first possible to obtain the relationships implied in the operation of the classifier in a logic form. By these rules has been possible *(i)* to identify the more discriminant features in the dataset; *(ii)* to enable the correction of the black box using the user knowledge. The constraining part has also demonstrated effective. Indeed, the imposition of the user-crafted rule has led to a coherent change in the black-box behaviour—enabling its correction. The results demonstrate how the presented guidelines lead to an IS giving explanations about its functioning, thus allowing the user intervention on its behaviour.

6. Conclusions

The solutions proposed so far in the literature to the opaqueness issue – one of the main problem of today AI technologies – are disparate. In this work, we showed how symbolic logic can be a crucial element in this panorama.

On the trails of the NSC models, we presented a series of guidelines aimed at correctly integrating a ML-based predictor – i.e., a black box – with a logic-based subsystem. In particular, our guidelines support the creation of IS exposing clear insights about their own functioning, thus enabling end users to intervene on the IS behaviour via a logical interface. We then tested our guidelines against a prototype IS, in order to study if and to what extent our approach is feasible and useful. Notably, the prototype assessment confirms our approach is feasible exploiting technologies already available in the research scene. Nevertheless, the prototype has been tested only on a single scenario. In order to confirm the efficacy of our approach, we need to perform a more exhaustive range of experiments. Moreover, we plan to extend the prototype assessment with more complex use-cases. For example, as anticipated in Section 4, we intend to enhance the prototype with the support for unstructured data. This extension would considerably improve the applicability of the studied approach, allowing its assessment also on the more complex area of image classification.

Through the above experimental investigation – in the case of more positive results – we aim at introducing a rigorously formalised version of the proposed model—presented in this paper in a more intuitive and preliminary shape. Consequently, we should also better investigate and verify the preliminary guidelines provided. We aim at obtaining an accurate and comprehensive guide that would allow developers efficiently integrating opaque AI systems and logic.

References

[1] J. Bughin, E. Hazan, S. Ramaswamy, M. Chui, T. Allas, P. Dahlstrom, N. Henke, M. Trench, Artificial intelligence: the next digital frontier?, Discussion paper, McKinsey Global Institute, 2017. URL: https://www.mckinsey.com/~/media/mckinsey/industries/advancedelectronics/ourinsights/howartificialintelligencecandeliverrealvaluetocompanies/mgi-artificial-intelligence-discussion-paper.ashx.

[2] A. B. Arrieta, N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion 58 (2020) 82–115. doi:10.1016/j.inffus.2019.12.012.

[3] R. Calegari, G. Ciatto, A. Omicini, On the integration of symbolic and sub-symbolic techniques for XAI: A survey, Intelligenza Artificiale 14 (2020) 7–32. doi:10.3233/IA-190036.

[4] A. S. d'Avila Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, S. N. Tran, Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning, FLAP 6 (2019) 611–632. URL: https://arxiv.org/abs/1905.06088.

[5] S. H. Muggleton, U. Schmid, C. Zeller, A. Tamaddoni-Nezhad, T. Besold, Ultra-strong

machine learning: comprehensibility of programs learned with ILP, Machine Learning 107 (2018) 1119–1140. doi:10.1007/s10994-018-5707-3.

[6] T. Rocktäschel, S. Riedel, End-to-end differentiable proving, in: I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 2017, pp. 3788–3800. URL: http://papers.nips.cc/paper/6969-end-to-end-differentiable-proving.

[7] A. Hernández, J. M. Amigó, Differentiable programming and its applications to dynamical systems, CoRR abs/1912.0 (2019). URL: http://arxiv.org/abs/1912.08168.

[8] L. Serafini, A. S. d'Avila Garcez, Logic tensor networks: Deep learning and logical reasoning from data and knowledge, in: T. R. Besold, L. C. Lamb, L. Serafini, W. Tabor (Eds.), 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016), volume 1768 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016, pp. 23–34. URL: http://ceur-ws.org/Vol-1768/NESY16_paper3.pdf.

[9] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, M. T. Vechev, DL2: training and querying neural networks with logic, in: K. Chaudhuri, R. Salakhutdinov (Eds.), 36th International Conference on Machine Learning, (ICML 2019), volume 97 of *Proceedings of Machine Learning Research*, 2019, pp. 1931–1941. URL: http://proceedings.mlr.press/v97/fischer19a.html.

[10] J. Xu, Z. Zhang, T. Friedman, Y. Liang, G. V. den Broeck, A semantic loss function for deep learning with symbolic knowledge, in: J. G. Dy, A. Krause (Eds.), 35th International Conference on Machine Learning (ICML 2018), volume 80 of *Proceedings of Machine Learning Research*, 2018, pp. 5498–5507. URL: http://proceedings.mlr.press/v80/xu18h.html.

[11] T. Li, V. Srikumar, Augmenting neural networks with first-order logic, in: 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), Association for Computational Linguistics (ACL), 2020, pp. 292–302. doi:10.18653/v1/p19-1028.

[12] S. Muggleton, Inductive logic programming, New Generation Computing 8 (1991) 295–318. doi:10.1007/BF03037089.

[13] A. Cropper, S. Dumancic, S. H. Muggleton, Turning 30: New ideas in inductive logic programming, in: C. Bessiere (Ed.), 29th International Joint Conference on Artificial Intelligence (IJCAI 2020), IJCAI, 2020, pp. 4833–4839. doi:10.24963/ijcai.2020/673.

[14] D. Doran, S. Schulz, T. R. Besold, What does explainable AI really mean? A new conceptualization of perspectives, in: T. R. Besold, O. Kutz (Eds.), 1st International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), volume 2071 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017, pp. 15–22. URL: http://ceur-ws.org/Vol-2071/CExAIIA_2017_paper_2.pdf.

[15] A. Bennetot, J.-L. Laurent, R. Chatila, N. Díaz-Rodríguez, Highlighting bias with explainable neural-symbolic visual reasoning, CoRR abs/1909.0 (2019). URL: http://arxiv.org/abs/1909.09065.

[16] G. Ciatto, M. I. Schumacher, A. Omicini, D. Calvaresi, Agent-based explanations in AI: Towards an abstract framework, in: D. Calvaresi, A. Najjar, M. Winikoff, K. Främling (Eds.), Explainable, Transparent Autonomous Agents and Multi-Agent Systems, volume

12175 of *Lecture Notes in Computer Science*, Springer, Cham, 2020, pp. 3–20. doi:10.1007/978-3-030-51924-7_1, Second International Workshop, EXTRAAMAS 2020, Auckland, New Zealand, May 9–13, 2020, Revised Selected Papers.

[17] G. Ciatto, D. Calvaresi, M. I. Schumacher, A. Omicini, An abstract framework for agent-based explanations in AI, in: 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2019), International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1816–1818. Extended Abstract.

[18] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. F. M. Ng, B. Liu, P. S. Yu, Z. Zhou, M. S. Steinbach, D. J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems 14 (2008) 1–37. doi:10.1007/s10115-007-0114-2.

[19] R. Andrews, J. Diederich, A. B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-Based Systems 8 (1995) 373–389. doi:10.1016/0950-7051(96)81920-4.

[20] M. de Jong, F. Sha, Neural theorem provers do not learn rules without exploration, CoRR abs/1906.0 (2019). URL: http://arxiv.org/abs/1906.06805.

[21] E. Denti, A. Omicini, A. Ricci, tuProlog: A light-weight Prolog for internet applications and infrastructures, in: Lecture Notes in Computer Science, volume 1990, Springer Verlag, 2001, pp. 184–198. doi:10.1007/3-540-45241-9_13.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 2019, pp. 8024–8035. URL: http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.

[23] D. Dua, C. Graff, UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.

Session 4
Healthcare

A Multi-Agent System for Simulating the Spread of a Contagious Disease

Giuseppe Antonio Nanna, Nicola Flavio Quatraro and Berardina De Carolis

Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy

Abstract

Recent events concerning global health risks have made truly evident the importance of advanced strategies and tools to monitor and prevent the spread of new and unpredictable diseases. COVID-19 showed the world that there are many factors that come into play when facing a viral threat, such as politics, social and economic aspects. Taking those into account when trying to deal with such events can make a huge difference in the efficacy and efficiency of the responses to the viruses. In this paper, we propose the use of a Multi-Agent system that extends the previous multi-agent-based approaches by adding a whole new set of features to control the outbreak during the simulation in order to dynamically verify how the government strategies can impact the disease spread.

Keywords

Simulation, Multi-agent systems, Agent-based modeling

1. Introduction

The simulation of an infectious disease spread is complex, there are multiple factors to take into account and it is quite difficult to model all the interactions that occur between them. Some examples of approach to this problem are the mean-field type models [1], the differential equation models [2] and the cellular automata theory [3, 4]. But those do not consider important aspects such as the spatial and temporal variables that describe the setting of the outbreak [5].

Agent-based modeling (ABM) is able to cover effectively all these topics since, for instance, an agent can be easily used to model different categories of individuals constituting the social structure of the population considered, each one with his own behavior, and his movements in a map during time [6]. The model proposed by [5] took into account all those aspects but lacks of features (depending on the government action and on social factors) for managing the disease spread dynamically.

For this reason we developed [1] an ABM simulation, based on the work of [5], enriched with some social, economical and political factors which happened to be highly relevant during the COVID-19 outbreak.

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy

g.a.nanna@inventati.org (G. A. Nanna); nicolaflavioquatraro@gmail.com (N. F. Quatraro); berardina.decarolis@uniba.it (B. De Carolis)

0000-0002-3146-2561 (G. A. Nanna); 0000-0002-2689-137X (B. De Carolis)

[1]The simulation was developed during the lockdown in the midst of the COVID-19 outbreak as a project work for the course on Multi-Agent Systems held by Prof. Berardina De Carolis

The human population, modelled as agents, behaves and interacts in a randomly generated urban setting, where the places with higher risk of contagion are considered. We included households, hospitals, public places, the transportation system, businesses and mass gatherings.

In our simulation humans and businesses dynamically change their behaviour depending on the measures and the restrictions enacted by the local government aimed at slowing down the spread of the disease. In order to simulate a more human behaviour, we also admitted the possibility of reckless actions.

The effectiveness of the government strategies can also be evaluated by monitoring the response of the healthcare system, which can get overwhelmed with tragic consequences on the population.

1.1. Related works

Starting from the early 20th century, even if the first contributions in mathematical modelling of spread of disease were developed by Daniel Bernoulli in 1760 [7], different approaches have been proposed to tackle the complex problem of simulation and prediction of outbreaks such as: the ordinary differential equation model, the discrete differential equation model, the impulsive differential equation model, the differential equation model with time delay, the finite equation theory, the matrix theory, the bifurcation theory, the K-order monotone system theory, the central manifold theory, the Lasalle invariant principle, etc… [8] The models proposed through time can be organized as follows [9]:

Deterministic
In a deterministic mathematical model every individual belongs to a different category, associated with a specific stage of the disease. The model is thus simulated through differential equations because the transition between disease stages can mathematically be represented as derivatives and it's differentiable with respect to time. So, the whole outbreak process can be considered as deterministic, meaning that every step in the simulation can be calculated considering exclusively the previous simulation's step [9].

Stochastic
Stochastic models simulate the possible outcomes of the outbreak considering probability distributions depending on different random variation of variables with respect to time. Those models rely on the probabilistic variations of the disease model variables such as risk of exposure, disease and other illness related events [9].

Some of the most used mathematical model for the simulation of an outbreak are the so called Compartment (or State) Models; those are model in which there's a division in categories between every individual [6].

A clear and simple example of this class of models is the Susceptible-Infected-Recovered (SIR) one [10] in which the whole population can be organized in those three different states. Every individual who gets infected moves from the state Susceptible to the Infected one, and during a fixed amount of time he can infect other Susceptible individuals. After this period, he moves to the Recovered state and, if the immune hypothesis is valid, he can not be infected again [6].

This kind of model has evolved through time getting more complex in relation to the disease considered and the setting in which the simulation holds; for instance, some models take in account the possibility of vaccination and pharmaceutical treatment of the population or, eventually, the measures of quarantine and isolation [6].

More complex models are, for example, the SEQIJR (Susceptible-Exposed-Quarantine-Infective-Isolation-Recovered) model, the SEIRP (Susceptible-Exposed-Infectious-Recovered-Persevered) model and the SLIRDS (Susceptible-Latent-Infected-Recovered-Dead-Susceptible) model [8].

Once the mathematical model is set, an often considered approach to represent it is the Equation-Based Modeling (EBM) in which the relationships among the variables and the states of the simulation are modelled through equations differentiable with respect to time [11]. The SIR model can be formulated with the following equations [12]:

$$\begin{aligned}\frac{dS}{dt} &= -\alpha SI \\ \frac{dI}{dt} &= \alpha SI - \beta I \\ \frac{dR}{dt} &= \beta I \\ S + I - R &= M\end{aligned}$$

where, α is the global transmission rate, β is the recovery (or death) rate, M is the total number of the whole population, $S = S(t)$, $I = I(t)$ and $R = R(t)$ are, respectively, the number of susceptible individuals, the number of infected individuals and the number of recovered individuals at time t [6].

Usually EBM systems are effective to simulate scenarios in which the outbreak can be observed as a mathematical, self-centered event. When this approach fails, other perspectives must be considered. In the ABM, the outbreak is the result of each individual's behaviour and actions that take place in the simulation. This approach allows a better understanding and representation of system's variables such as heterogeneity across individuals, geographical localization and distribution, different forms of interaction between individuals and so on [6]. But, inevitably, it has higher computational and cognitive costs.

These are some other examples of the recent approaches to epidemic spread modelling [6]:

- *Agent-Based Modeling and Simulation of Influenza*:
  In this simulation of influenza [13], the setting is a common Chinese city with a dynamic contact network where every place belongs to one of four categories: workplace, household, place of entertainment and school. Every individual in the simulation lives in a sub-location called mixing group and interacts with other individuals in the same household and mixing group. The interaction types are defined normal, if it occurs in a mixing group, or random, if it occurs between different mixing groups. Every individuals can travel from his house to his assigned location and viceversa. The model has been tested on a city composed by 30000 citizens, 80% of them have been vaccinated, who live in 10252 households. A simulation package QAST verifies the impact of different restriction strategies on the simulation of infectious diseases. The experiment showed that restriction policies can counterattack the outbreak and that agent-based models can represent better models than the mathematical one in studying large-scale population simulations [6].

- *Agent-Based Simulation on Avian Influenza in Vietnam*:
  Using a SIR compartment model [12], combined a EBM and a ABM approach to evaluate the existing restriction measures through a simulation based on the data of the avian influenza contagion. Such model takes into account a well-mixed and homogeneous poultry population where random contacts between infectious and susceptible individuals occur and where the basic reproduction number R_0 of the disease is a mathematically calculated parameter, based on the daily reported individuals' death. The experiments showed, in order to control the outbreak, a strategic campaign of culling, bio-security restriction and large-scale vaccination must been considered [6].
- *Multiagent-Based Simulation of the HIV/AIDS Spatial and Temporal Transmission among Injection Drug Users*:
  In the work [14] a model to simulate the HIV/AIDS transmission model between injecting drug users (IDUs) has been developed. It relies on the multivalent system and geographic information systems (GIS) in an urban setting during a ten years period using the Repast (REcursive Porous Agent Simulation Toolkit) Simphony 1.2 platform. Every individual interacts with other ones through randomly generated social networks, he is described by parameters like gender, education, age and geographic location, and he belongs to one of five different categories: IDU person, HIV-IDU person, HIV person, AIDS person and healthy person. Every day, an individual interacts with five to eight, out of twenty friends. Every individual can affect his friends by encouraging them, if he is healthy, to stop taking drugs or, if he is an IDU, to keep or begin using drugs. The experiment's setting is Kunming, the capital of Yunnan Province, southwest of China. Results showed that individual social influence, the percentage of needle sharing individuals and the starting number of HIV individuals have a major role in the HIV/AIDS outbreak [6].

2. Simulation

The proposed simulation [2] is written in Java and is based on JADE, a software framework for the development of intelligent agents [3].

During the simulation, it is possible to monitor the contagion curves in real-time (figure 1) and to affect the outcome of the simulation by imposing rules and restrictions such as the social distancing (figure 2).

2.1. Agent Based Model

Every individual of the simulated population is modelled through an agent. Every Individual, during daytime, simulate the human behaviour with a list of tasks, which can occur at a certain location of the map, such as staying at home or at work, or can involve a path in the map, such as moving from the supermarket location to home [5].

Thus, each agent periodically performs several possible actions based on his needs:

- going to the supermarket

[2] Simulation repository: https://github.com/h3r0n/sysag_cds
[3] JADE website: https://jade.tilab.com/

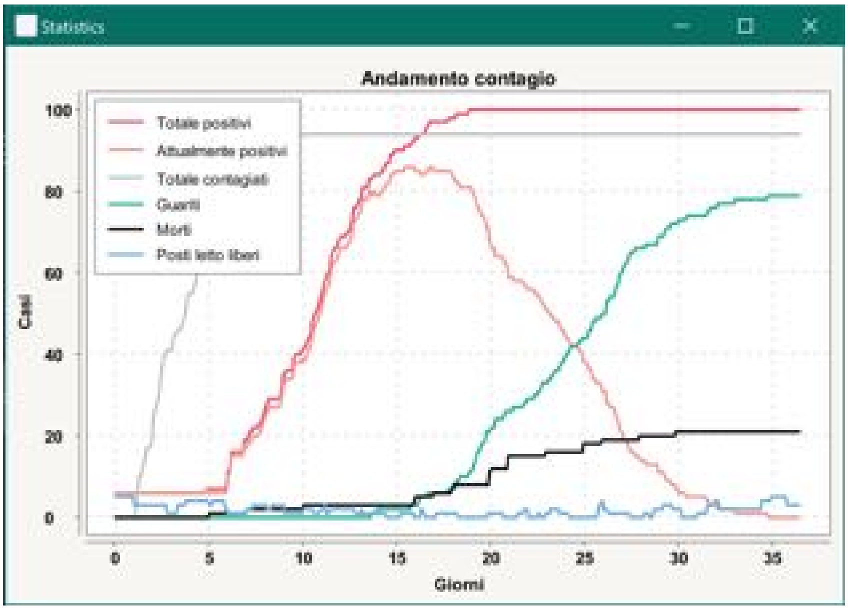


Figure 1: Real-time animated visualization of contagion curves

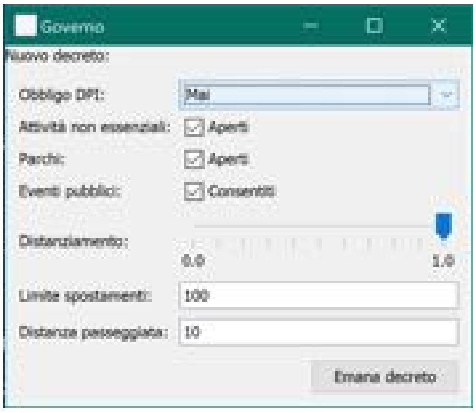


Figure 2: Government GUI

- going to the park
- going to a public event
- going to the hospital if ill
- going outside for a walk

After each action the simulated individual will return to his own house. Since the household transmissibility of diseases like COVID-19 is epidemiologically relevant and it cannot be ignored [15], residences (randomly assigned at the beginning of the simulation) can be shared by agents.

Every agent has some needs to satisfy during its daytime and through its movements it may get in contact with other groups of people who can infect it or that can be infected by it [5]. Some agents simulate workers: in addition to the already specified actions, they will also periodically go to their randomly assigned workplace.

A worldwide study conducted on April 2020 [16] analysed the erroneous conspiracy theory developed by many people during the COVID-19 pandemic: "Those who were more conspiratorial were more likely to report that the government's response was too strong, illogical and that the government hid information to people" [16]. "Some of these beliefs were potentially harmful and some could have led to public rejection of public health measures to suppress the spread of the virus" [16].

For those reasons a number of people who completely ignore the restrictions imposed by the government have been included in the simulation.

2.2. World Model

There are a multitude of factors that can hugely affect the outcome of an outbreak. Different social networks and geographical settings determine a different spreading dynamics and different types of epidemiology [5].

For this reason, in the simulation, agents' activities and tasks take place in geographical locations inside a simplified model of an urban setting in which individuals may interact with each other, this includes engaging in activities related to the daily commuting through the urban transportation network [5]. When an individual leaves a group, he travels through space and time to another destinations to accomplish a different task, which often implies interacting and joining another group of individuals [5]. This approach allows a more realistic simulation of the daily routine of each individual, and, therefore, a more accurate simulation of the population dynamics [5].

The world in which the agents act is modelled as a graph in which the nodes are buildings and the edges are the roads. The graph is randomly generated at the beginning of the simulation. Each component of the geographical world is characterized by a set of coordinates in the graph and by a density factor which describes the average distance allowed in that place, which affects the probability of the contagion.
Roads are used to model a simplified version of the urban transportation network. Therefore, in order to move from a location to another, an individual has to calculate the shortest path to his destination, which is composed by a list of roads the individual must pass through [5].

During the COVID-19 outbreak, the lockdown status imposed the suspension of a great percentage of the commercial and productive activities, which caused a dramatic crisis worldwide [17].
The presence of activities, companies, institutions can affect the spread of the disease and have been used [17] to predict the risk of contagion. Because of their importance on the spread dynamics, businesses were included in the simulation in order to support the local administrations in formulating the best approaches to reduce or restart the local activities during lockdown restrictions [17].

Each business is associated with a specific building, it can be the workplace of one or several workers and can be closed or reopened by the Government. We distinguish different kinds of business:

- supermarket
- park, which represents recreation and entertainment places
- hospital
- non strictly necessary business
- necessary business

The last two categories are needed to differentiate the businesses which are absolutely necessary for the well-being of the population, such as food factories, from those that can eventually be closed to control the outbreak.

Events of large-scale group infection [18] have been considered, they occur in circumstances in which a huge number of individuals are located in the same place, leading to a massive increase in chances to be exposed to the infection,in a short period of time. It has been estimated that 71.7% of the confirmed COVID-19 cases in Korea [18] are related to such events, which usually occur in places like factories, dormitories, schools, companies or places in which individuals are constrained in a limited physical space.

Parks represent all recreation and entertainment places and can also be casually selected as the setting of public events, attended by a large percentage of the population and may be the source of large-scale group infections.

During a pandemic the healthcare system has an higher risk to collapse because of difficulties in triaging, allocation, and a shortage of high-level care beds [19].
Thus, hospitals in the simulation have a limited number of beds, which implies that if the outbreak gets out of control and all the beds get occupied, those disease-ill individual with strong symptoms are much more likely to die due to the lack of proper care.
This allows to further evaluate the success of the Government's strategies to control the spread of the disease.

2.3. Government Model

Social distancing can be defined as "a deliberate effort instituted to stop or slow down the spread of a highly infectious or contagious diseases" [20]. In order control an outbreak, every individual must then reduce the interactions with the rest of the population,which leads in closing partially or wholly social activities specially business and transport because they may enhance social interaction and disease spread [20].

In order to control the epidemic of COVID-19, more than 10 million people in Wuhan were restricted to their home by the Chinese government. By reducing the contact rate of latent individuals, interventions such as quarantine and isolation have effectively reduced the potential peak number of COVID-19 infections and delayed the time of peak infection [21, 22].

At any time during the simulation, the user can pose as the local government and can issue and lift restrictions and social distancing measures with a simple GUI (figure 2). Several ways to try to control and eventually stop the outbreak are provided:

- close the non strictly necessary businesses
- close the parks

- forbid public events
- make the use of PPE mandatory (never, everywhere or just indoors)
- close the buildings with a low average distance between individuals
- reduce the maximum movement range to go for a walk
- reduce the maximum travel distance of every individual

The government can enact those restrictions by issuing a “decree”. After this event, all the agents simulating individuals and businesses get notified and eventually modify their behaviour.

2.4. Infection Model

Our model, based on [5], will consider a SEIR epidemic spread model where every individual (agent) of the simulation can be in one of four different states:

Susceptible
individuals who can contract the disease

Exposed
individuals who have contracted the disease and are not contagious

Infected
individuals who have contracted the disease and are contagious

Recovered
individuals who are healed from the disease and that cannot contract it anymore

The progress of the infection in an individual is then represented through the unidirectional translation between those states, starting from the Susceptible one and ending in the Recovered one, assuming that once an individual is Recovered he becomes immune to the disease.

Once the individual becomes Exposed, there is a latency period before he becomes Infected. This can be represented with a simple formula [5]:

$$I_{pi} = t_i + x_{LP}$$

Where t_i represents the moment in time where the contagion occurs and x_{LP} is the number of days necessary to become infected.

After reaching the Infected state, there is another latency period before the individual heals completely, becoming Recovered:

$$R_{pi} = I_{pi} + x_{IP}$$

Where I_{pi} represents the moment in time where the individual becomes Infected and x_{IP} is the number of days necessary to become Recovered. The translation between Susceptible to Exposed occurs when an Infected individual infects a Susceptible one. The way this happens depends on two conditions:

- The two individual must meet in the same place
- The contagion must occur

The meeting represents a violation of social distancing and is modeled as a probabilistic event. It occurs when the following formula is true:

$$randomvalue < max(infectiousDist, susceptibleDist)$$

where $infectiousDist$ and $susceptibleDist$ are the probabilities (respectively associated to the infectious individual and the susceptible one) of getting close with anyone. Those values are randomly generated every time an individual reaches any geographical location. The values cannot be lower than the density value associated to the location and cannot be higher than the maximum density allowed by the Government (unless the individual is ignoring the decrees issued by the Government).

Similarly, the Contagion event follows a probabilistic distribution and occurs if the following formula is true:

$$randomvalue < P(infectiousPPE, susceptiblePPE)$$

where P is a function depending on whether or not the infectious individual and the susceptible one are wearing personal protection equipment (PPE) like a mask, as according to the World Health Organization Writing Group, wearing a mask can prevent respiratory infectious diseases [23], [24].

This claim has been proven by several studies [24]:

- "In a large systematic review of physical interventions to control spread of infectious diseases, Jefferson et al. concluded from 67 studies that wearing masks is effective as one of the important barriers to transmission of respiratory viruses, and evidences indicate N95 respirators were non-inferior to surgical masks" [25], [24].
- "Other studies also found evidence that wearing masks can significantly reduce the risk of SARS and influenza-related diseases" [26, 27, 28], [24].
- "Aldila et al. constructed MERS determinant mathematical model and found that compared with auxiliary nursing and government publicity, wearing masks is the optimal choice for reducing the number of infections" [29], [24].
- "Barasheed et al. systematically analyzed the utilization and effectiveness of masks by integrating 12,710 samples from more than 50 countries in the world, and found that wearing masks in crowded places could reduce the risk of respiratory infections by 20%" [30], [24].
- "A study in Hong Kong found that the odds ratio (OR) value of wearing masks in public places was only 0.36, which was lower than that of living room disinfection (OR = 0.41) and frequent hand washing (OR = 0.58), indicating that wearing masks effectively restricted the community spread of SARS-CoV in Hong Kong" [31], [24].

Each individual only wears a mask if compulsory as long as he doesn't ignore the Government's dispositions.

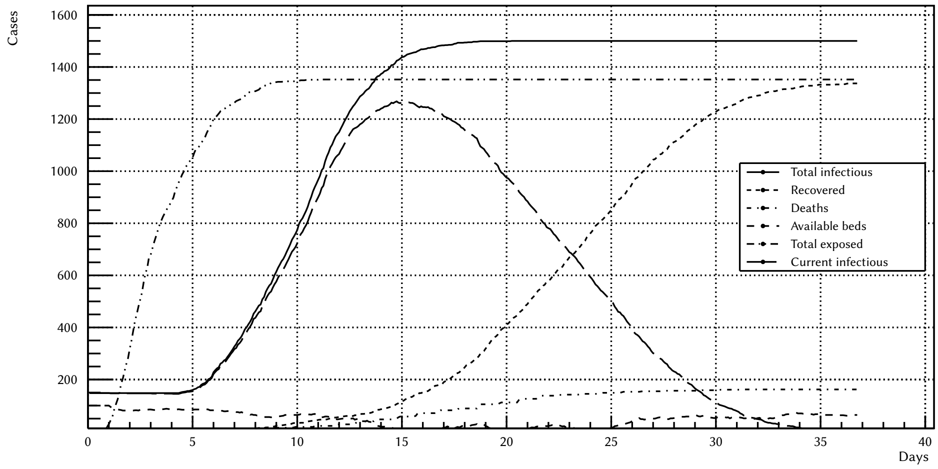


Figure 3: Scenario 1: No precautions - High number of infectious

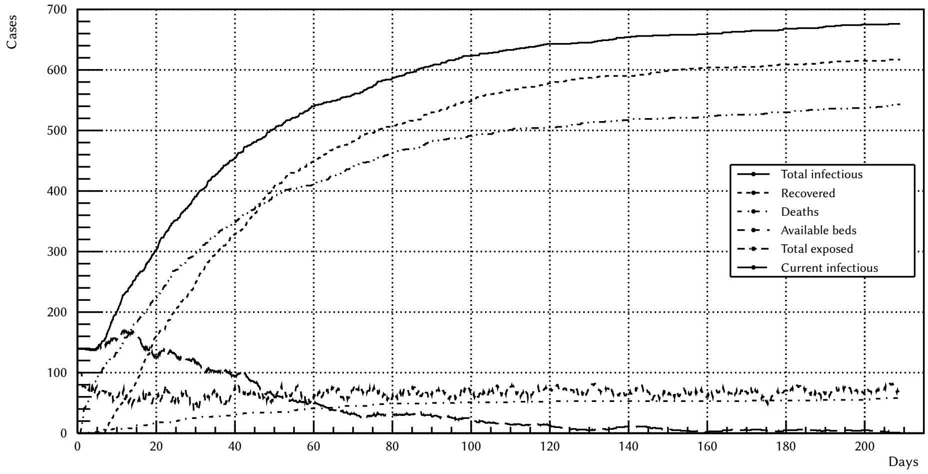


Figure 4: Scenario 2: Precautions - High number of infectious

3. Simulated Scenarios

We chose to simulate 6 different scenarios in order to verify the capabilities of the created model. In each one 1500 people are simulated, of which 20% ignore the Government's restriction and 50% are workers. The map consists of 3600 buildings with 500 businesses. Hospitals have a combined capacity of 100 beds.

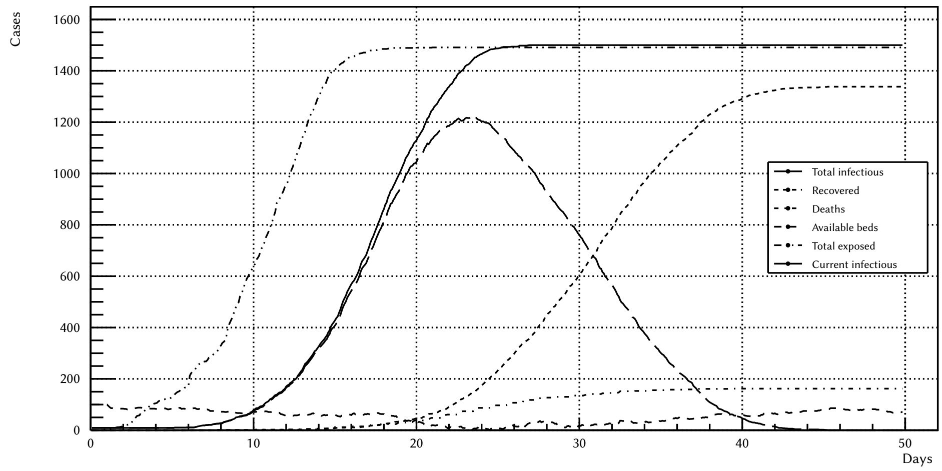


Figure 5: Scenario 3: No precautions - Low number of infectious

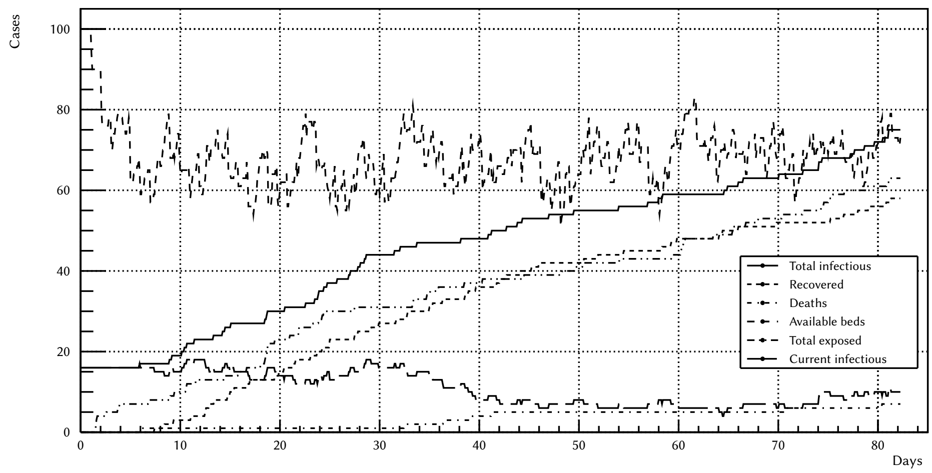


Figure 6: Scenario 4: Precautions - Low number of infectious

- *Scenario 1*: No precautions - High number of infectious
  This scenario simulates a community of 1500 people facing a large wave of contagion (10% of the population is infectious at the beginning, everyone else is susceptible) without adopting any social distancing policy.
  In this scenario, as shown in figure 3, the whole population contracts the disease in a single week and the healthcare system gets overwhelmed a few days later. As a result a

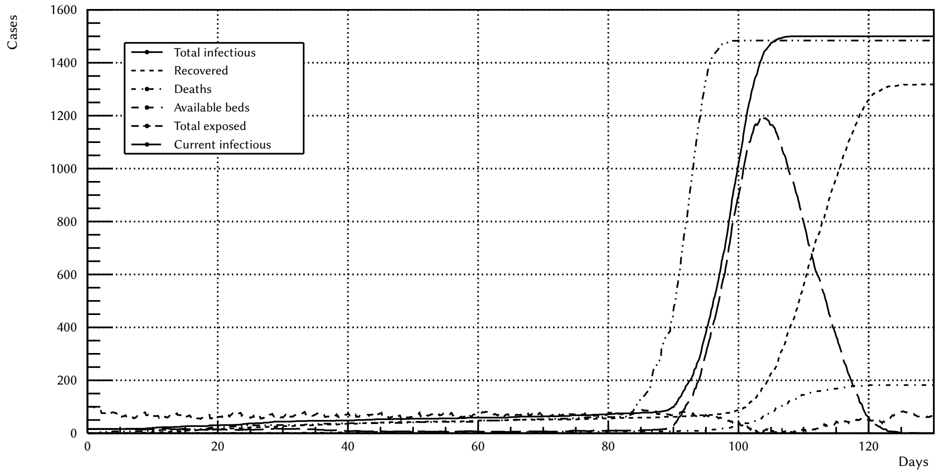


Figure 7: Scenario 5: Premature Reopening

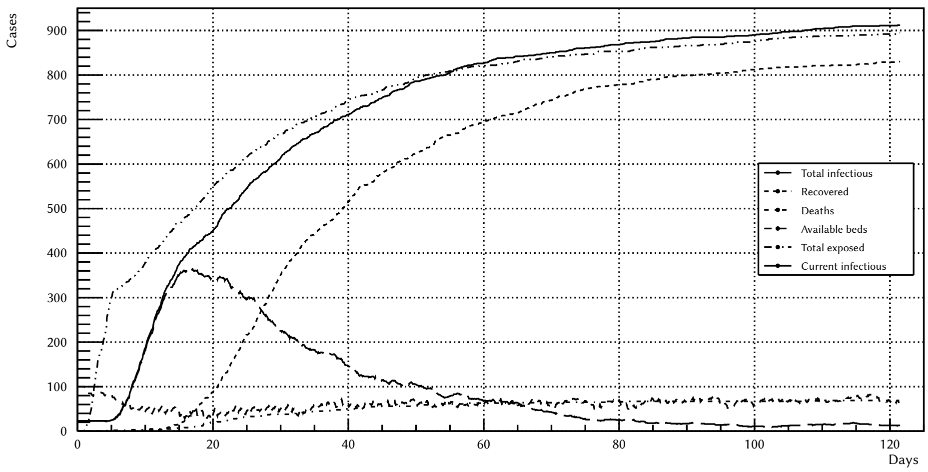


Figure 8: Scenario 6: adopting progressively stricter social distancing measures

relevant number of individuals dies due to the lack of proper care.

- *Scenario 2*: Precautions - High number of infectious
  The starting conditions of this scenario are the same of the previous one, except it simulates the adoption of restrictive social distancing policies.
  The outcome (shown in figure 4) is vastly different as the disease spreads much slower. As a result hospitals don't have to operate at over capacity and less people die.

- *Scenario 3*: No precautions - Low number of infectious
  In this scenario 1% of the population is infectious at the beginning and no social distancing policy is adopted. Despite the different starting conditions, the outcome (shown in figure 5) is close to the *Scenario 1* due to the exponential nature of the infection transmission.
- *Scenario 4*: Precautions - Low number of infectious
  The starting conditions of this scenario are the same of the previous one except it simulates the same social distancing policies adopted in *Scenario 2*.
  As shown in figure 6, the spread of the disease progresses very slowly compared to the previous scenarios, while the number of active cases slowly decreases over time.
- *Scenario 5*: Premature Reopening
  This scenario simulates the non gradual lifting of all restrictions at the end of the previous scenario. The outcome (shown in figure 7) is the same of *Scenario 1* and *3* with the whole population exposed to the disease and a large number of deaths.
- *Scenario 6*: adopting progressively stricter social distancing measures
  This scenario (shown in figure 8) is similar to *Scenario 4*, except the Government doesn't act proactively, but only gradually enacts restrictions after a spike in the number of infectious people.
  The outcome (shown in figure 8) is different because of the large share of the population who contracts the disease, but less people die (compared to *Scenario 1* and *3*) because the healthcare system doesn't get overwhelmed.

4. Conclusions and Future Work

The proposed model is able to simulate a wide range of different situations. It allows to take into account aspects often overlooked by other works, such as the response of local administrations, the capacity of the health system and the presence of local businesses. These factors, as shown in the simulated scenarios, have a not negligible importance on the spread of the disease and its effects on the population.

Even if the proposed approach is general enough to model a generic contagion setting, the examples are shown in the context of the COVID-19 disease. In this context, the contagion curves observed during the COVID-19 pandemic show a strong resemblance to the more realistic scenarios such as Scenarios 4 and 6. Especially in the case of European countries where the adoption of social distancing measures (similar to the simulated ones) allowed the local governments to control the spread of the virus.

The COVID-19 outbreak has given a new perspective on the variables which rule the outbreak response. The economic consequences of the quarantine, isolation and social distancing strategies have been far more worse than expected and every government in such situations has to make very complex choices.

Our goal with this proposed approach is to simulate the effect of such choices on the population and the public health system in order to help making such difficult choices to preserve the interests of the nation and of the population. Even though this simulator relies on a quite simple mathematical formulation, it has been able to properly reproduce the normal distribution of the disease spread curve when no restrictions were imposed while it follows a linear trend when those restrictions come into force. Those results are then matching the expectations and the events that occurred during the COVID-19 outbreak.

However there are many aspects of this tool which can be improved, such as:

- a wider geographical setting where people are able to travel between different communities
- a more heterogeneous social structure with more classes reflecting each individual's age and role
- a more diverse and complex behaviour of each agent allowed by the use of a BDI model

In the future we believe that improving those aspects, with the help of experts in the field, can provide a valid tool that can assist the administrations, before and during an outbreak, in choosing the most optimal strategies to safeguard public health and minimize the consequences of restrictions on the economy at the same time.

References

[1] A. Kleczkowski, B. T. Grenfell, Mean-field-type equations for spread of epidemics: the 'small world' model, Physica A: Statistical Mechanics and its Applications 274 (1999) 355 – 360. doi:https://doi.org/10.1016/S0378-4371(99)00393-3.

[2] S. Fu, G. Milne, Epidemic modelling using cellular automata, in: H. Abbass, J. Wiles (Eds.), The Australian Conference on Artificial Life ACAL 2003, volume N/A, UNSW Press, Australia, 2003, pp. 43–57. Epidemic Modelling Using Cellular Automata ; Conference date: 01-01-2003.

[3] G. Sirakoulis, I. Karafyllidis, A. Thanailakis, A cellular automaton model for the effects of population movement and vaccination on epidemic propagation, Ecological Modelling 133 (2000) 209 – 223. doi:https://doi.org/10.1016/S0304-3800(00)00294-5.

[4] L. Q.-X. Jin Zhen, A cellular automata model of epidemics of a heterogeneous susceptibility, Chinese Physics B 15 (2006) 1248. doi:10.1088/1009-1963/15/6/019.

[5] L. Perez, S. Dragicevic, An agent-based approach for modeling dynamics of contagious disease spread, International Journal of Health Geographics 8 (2009) 50. doi:10.1186/1476-072X-8-50.

[6] M. Shatnawi, S. Lazarova-Molnar, N. Zaki, Modeling and simulation of epidemic spread: Recent advances, 2013 9th International Conference on Innovations in Information Technology, IIT 2013 (2013) 118–123. doi:10.1109/Innovations.2013.6544404.

[7] H. Hethcote, The mathematics of infectious diseases, SIAM Review 42 (2000) 599–653. doi:10.1137/S0036144500371907.

[8] S. Bin, G. Sun, C.-C. Chen, Spread of Infectious Disease Modeling and Analysis of Different Factors on Spread of Infectious Disease Based on Cellular Automata, International Journal of Environmental Research and Public Health 16 (2019) 4683. doi:10.3390/ijerph16234683.

[9] F. Brauer, C. Castillo-Chavez, Mathematical Models in Population Biology and Epidemiology, volume 40 of *Texts in Applied Mathematics*, Springer New York, New York, NY, 2012. URL: http://link.springer.com/10.1007/978-1-4614-1686-9. doi:10.1007/978-1-4614-1686-9.

[10] W. O. Kermack, A. G. McKendrick, G. T. Walker, A contribution to the mathematical theory of epidemics, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character 115 (1927) 700–721. doi:10.1098/rspa.1927.0118.

[11] H. Van Dyke Parunak, R. Savit, R. L. Riolo, Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide, in: G. Goos, J. Hartmanis, J. van Leeuwen, J. S. Sichman, R. Conte, N. Gilbert (Eds.), Multi-Agent Systems and Agent-Based Simulation, volume 1534, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 10–25. doi:10.1007/10692956_2.

[12] D. M. Nguyen, H. Deguchi, M. Ichikawa, Agent-based simulation on avian influenza in Vietnam: Basic characteristics of the epidemic and efficiency evaluation of control measures, in: 2010 7th International Conference on Service Systems and Service Management, 2010, pp. 1–6. doi:10.1109/ICSSSM.2010.5530215, iSSN: 2161-1904.

[13] Zhang Fa, Zhao Qiao-xia, Li Lu, Intervention for contagious disease: Agent-based modeling and simulation, in: 2011 2nd IEEE International Conference on Emergency Management and Management Sciences, 2011, pp. 155–158. doi:10.1109/ICEMMS.2011.6015643.

[14] J. Xiong, J. Wang, K. Yang, S. Peng, Q. Xu, Multiagent-based simulation of the HIV/AIDS spatial and temporal transmission among injection drug users, in: 2010 18th International Conference on Geoinformatics, 2010, pp. 1–6. doi:10.1109/GEOINFORMATICS.2010.5567671, iSSN: 2161-0258.

[15] Q.-L. Jing, M.-J. Liu, J. Yuan, Z.-B. Zhang, A.-R. Zhang, N. E. Dean, L. Luo, M.-M. Ma, I. Longini, E. Kenah, Y. Lu, Y. Ma, N. Jalali, L.-Q. Fang, Z.-C. Yang, Y. Yang, Household Secondary Attack Rate of COVID-19 and Associated Determinants, preprint, Epidemiology, 2020. doi:10.1101/2020.04.11.20056010.

[16] N. Georgiou, P. Delfabbro, R. Balzan, COVID-19-related conspiracy beliefs and their relationship with perceived stress and pre-existing conspiracy beliefs, Personality and Individual Differences 166 (2020) 110201. doi:10.1016/j.paid.2020.110201.

[17] V. Sangiorgio, F. Parisi, A multicriteria approach for risk assessment of Covid-19 in urban district lockdown, Safety Science 130 (2020) 104862. doi:10.1016/j.ssci.2020.104862.

[18] Y.-J. Kang, Lessons Learned From Cases of COVID-19 Infection in South Korea, Disaster Medicine and Public Health Preparedness (2020) 1–8. doi:10.1017/dmp.2020.141.

[19] M. Sorbello, K. El-Boghdadly, I. Di Giacinto, R. Cataldo, C. Esposito, S. Falcetta, G. Merli, G. Cortese, R. M. Corso, F. Bressan, S. Pintaudi, R. Greif, A. Donati, F. Petrini, The Società Italiana di Anestesia Analgesia Rianimazione e Terapia Intensiva (SIAARTI) Airway Research Group, and The European Airway Management Society, The Italian coronavirus disease 2019 outbreak: recommendations from clinical practice, Anaesthesia 75 (2020) 724–732. doi:10.1111/anae.15049.

[20] G. Musinguzi, B. O. Asamoah, The Science of Social Distancing and Total Lock Down: Does it Work? Whom does it Benefit?, Electronic Journal of General Medicine 17 (2020) em230. doi:10.29333/ejgm/7895.

[21] C. Hou, J. Chen, Y. Zhou, L. Hua, J. Yuan, S. He, Y. Guo, S. Zhang, Q. Jia, C. Zhao, J. Zhang, G. Xu, E. Jia, The effectiveness of quarantine of Wuhan city against the Corona Virus Disease 2019 (COVID-19): A well-mixed SEIR model analysis, Journal of Medical Virology

92 (2020) 841–848. doi:10.1002/jmv.25827.

[22] K. Prem, Y. Liu, T. W. Russell, A. J. Kucharski, R. M. Eggo, N. Davies, M. Jit, P. Klepac, S. Flasche, S. Clifford, C. A. B. Pearson, J. D. Munday, S. Abbott, H. Gibbs, A. Rosello, B. J. Quilty, T. Jombart, F. Sun, C. Diamond, A. Gimma, K. van Zandvoort, S. Funk, C. I. Jarvis, W. J. Edmunds, N. I. Bosse, J. Hellewell, The effect of control strategies to reduce social mixing on outcomes of the COVID-19 epidemic in Wuhan, China: a modelling study, The Lancet Public Health 5 (2020) e261–e270. doi:10.1016/S2468-2667(20)30073-6.

[23] World Health Organization Writing Group, Nonpharmaceutical Interventions for Pandemic Influenza, National and Community Measures, Emerging Infectious Diseases 12 (2012) 88–94. doi:10.3201/eid1201.051371.

[24] J. Wang, L. Pan, S. Tang, J. S. Ji, X. Shi, Mask use during covid-19: A risk adjusted strategy, Environmental Pollution 266 (2020) 115099. doi:https://doi.org/10.1016/j.envpol.2020.115099.

[25] T. Jefferson, C. B. Del Mar, L. Dooley, E. Ferroni, L. A. Al-Ansary, G. A. Bawazeer, M. L. van Driel, N. S. Nair, M. A. Jones, S. Thorning, J. M. Conly, Physical interventions to interrupt or reduce the spread of respiratory viruses, Cochrane Database of Systematic Reviews (2011). doi:10.1002/14651858.CD006207.pub4.

[26] T. Jefferson, C. Del Mar, L. Dooley, E. Ferroni, L. A. Al-Ansary, G. A. Bawazeer, M. L. van Driel, R. Foxlee, A. Rivetti, Physical interventions to interrupt or reduce the spread of respiratory viruses: systematic review, BMJ 339 (2009) b3675–b3675. doi:10.1136/bmj.b3675.

[27] C. R. MacIntyre, S. Cauchemez, D. E. Dwyer, H. Seale, P. Cheung, G. Browne, M. Fasher, J. Wood, Z. Gao, R. Booy, N. Ferguson, Face Mask Use and Control of Respiratory Virus Transmission in Households, Emerging Infectious Diseases 15 (2009) 233–241. doi:10.3201/eid1502.081166.

[28] L. Zhang, Z. Peng, J. Ou, G. Zeng, R. E. Fontaine, M. Liu, F. Cui, R. Hong, H. Zhou, Y. Huai, S.-K. Chuang, Y.-H. Leung, Y. Feng, Y. Luo, T. Shen, B.-P. Zhu, M.-A. Widdowson, H. Yu, Protection by Face Masks against Influenza A(H1N1)pdm09 Virus on Trans-Pacific Passenger Aircraft, 2009, Emerging Infectious Diseases 19 (2013). doi:10.3201/eid1909.121765.

[29] D. Aldila, H. Padma, K. Khotimah, B. Desjwiandra, H. Tasman, Analyzing the MERS disease control strategy through an optimal control problem, International Journal of Applied Mathematics and Computer Science 28 (2018) 169–184. doi:10.2478/amcs-2018-0013.

[30] O. Barasheed, M. Alfelali, S. Mushta, H. Bokhary, J. Alshehri, A. A. Attar, R. Booy, H. Rashid, Uptake and effectiveness of facemask against respiratory infections at mass gatherings: a systematic review, International Journal of Infectious Diseases 47 (2016) 105–111. doi:10.1016/j.ijid.2016.03.023.

[31] J. T. Lau, H. Tsui, M. Lau, X. Yang, SARS Transmission, Risk Factors, and Prevention in Hong Kong, Emerging Infectious Diseases 10 (2004) 587–592. doi:10.3201/eid1004.030628.

Transcultural Health-Aware Guides for the Elderly

Rafael H. Bordini[a,b], Viviana Mascardi[b], Stefania Costantini[c], Amal El Fallah-Seghrouchni[d], Yves Lespérance[e] and Alessandro Ricci[f]

[a] *PUCRS, Porto Alegre, Brazil*
[b] *Genova University, Genova, Italy*
[c] *L'Aquila University, L'Aquila, Italy*
[d] *Sorbonne University, Paris, France*
[e] *York University, Toronto, Canada*
[f] *Università di Bologna, Cesena, Italy*

Abstract

In this brief position paper, we present our vision for using software agents and related technologies to address the growing need of transcultural health-aware "Guides" for the elderly, an increasingly important topic given the clear trend of population ageing. Such autonomous intelligent guides are employed in smart living/city infrastructures to give emotional and healthcare support for the elderly wherever they are, whether at home, outdoors, or in hospital. The main purpose is to help ageing people to avoid progressive loss of physical, cognitive, and emotional activity, and most importantly to avoid social exclusion.

Keywords

Population ageing, smart environment, argumentation, ontology

1. Introduction

Most continents, in particular those where the authors live, are exposed to the tangible effects of the ageing of the population, with all the challenges for the society and for the individuals (the ageing people, but also their relatives, caregivers, doctors) that this demographic change raises.

In this paper, we present our vision on how to address some of these challenges; in particular, we investigate how intelligent software agents could mitigate the risk of progressive loss of physical, intellectual, and emotional activity of ageing people, and of their social exclusion,

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
rafael.bordini@pucrs.br (R. H. Bordini); viviana.mascardi@unige.it (V. Mascardi); stefania.costantini@univaq.it (S. Costantini); amal.elfallah@lip6.fr (A. El Fallah-Seghrouchni); lesperan@eecs.yorku.ca (Y. Lespérance); a.ricci@unibo.it (A. Ricci)
https://www.inf.pucrs.br/r.bordini/ (R. H. Bordini); https://person.dibris.unige.it/mascardi-viviana/ (V. Mascardi); http://people.disim.univaq.it/~stefcost/ (S. Costantini); https://webia.lip6.fr/~elfallah/ (A. El Fallah-Seghrouchni); http://www.cse.yorku.ca/~lesperan/ (Y. Lespérance); https://www.unibo.it/sitoweb/a.ricci/en (A. Ricci)
0000-0001-8688-9901 (R. H. Bordini); 0000-0002-2261-9926 (V. Mascardi); 0000-0002-5686-6124 (S. Costantini); 0000-0002-8390-8780 (A. El Fallah-Seghrouchni); 0000-0003-1625-0226 (Y. Lespérance); 0000-0002-9222-5092 (A. Ricci)

in particular when a transition from different situations (moving from home to a protected structure, or from a protected structure to an hospital) takes place.

We believe that one way to cope with the needs of old people, especially during these delicate transitions where they are much more fragile than in stable situations, is to provide them with highly-interactive culturally adaptive "Guides" that not only care for their health but also serve as entertaining companions that interact through spoken dialogues and that stimulate their cognitive abilities. Such Guides, designed and implemented as software agents compliant with the strong notion of agency [1] and also characterised by emotional features à la Bates [2], besides mentalistic ones à la Shoham [3], are meant to become a familiar and trustable point of reference for their users. The Beliefs-Desires-Intentions (BDI) agent model, extended with emotional features as suggested for example by Pereira et al. [4], Alfonso et al. [5], Su et al. [6], would hence represent a suitable architecture for our Guides, and has been recently adopted in healthcare applications [7]. Given the BDI-oriented nature of the Guides[1], we will exploit languages like DALI [8], SR-APL [9], IndiGolog [10], or Jason [11], meant as a standalone infrastructure or, better, integrated in the environment- and organization-oriented JaCaMo framework [12], for their implementation.

We use the word "Guide" to emphasise that such interactive assistants are always available to their users when they need advice or company, wherever they are, be it the comfort of their home, outdoors, but also at difficult times for example when moving to protected structures or hospital is needed. The Guides may provide guidance — following a protocol pre-agreed with doctors, psychologists and caregivers — to the elderly on all those situations which do not require a doctor evaluation and assessment (entertainment, social activity, physical activity, diet, medication to take according to the agreed protocol). By watching over the elderly, Guides can collect precious information on their behaviour and can serve as a source of reliable information for the doctors who care for the elderly. Hence, guidance is bidirectional: caregivers choices and focus of attention can indeed be driven by the Guide, based on what it senses, observes, hears, and deduces.

We expect that Guides should quickly learn the cultural profile of their users and flexibly adapt to cross-cultural interaction. This means adapting the style of conversation to the conversation party: old users may require the adoption of a limited, simple vocabulary, which includes words familiar to them, while doctors may take advantage of a rich technical vocabulary.

By boosting cross-cultural interaction, our Guides will also facilitate people to get in touch and to interact, hence achieving one of the main risks of ageing: social exclusion.

The targets of our investigation are indeed natural language processing, ontologies and argumentation techniques to ensure cross-cultural interaction [13, 14]; agent-oriented approaches to make such cross-cultural interaction intelligent and emotionally realistic and believable, which requires explicit representations of the user's state of mind [15]; and smart-* approaches to make agent-oriented approaches efficient and well integrated with the existing environment and infrastructures.

[1] By "BDI-oriented" here we mean, in a very broad sense, the conceptualization and implementation of the Guides based on explicit knowledge/ beliefs, declarative and rule based reasoning/planning, explicit goals to achieve.

2. The "Guides" and Their Smart Environment

The architectural framework we have devised relies upon a "sensing layer" which is necessary for creating smart environments where people are cared for. Improving the state of the art on the sensing layer falls outside our research investigation: we plan to exploit as many out-of-the-box techniques and tools as possible, among the many available ones [16], to allow the Guides to sense what people are doing in an unobtrusive (or "acceptably obtrusive") way. Among these tools, we will consider cameras tracking people and their actions for detecting falls [17], wearable and IoT devices [18], sleep sensors [19].

We are much more interested in exploring the potential of software agents as the building blocks for analysing, designing, and developing the Guides. In particular, we are interested in argumentation schemes that are specific for the elderly and for each culture, and in their translation into properly formalised argumentation frameworks [20] where the Guides, implemented as agents, can play a role; in the integration of ontologies and ontological reasoning [21] inside such argumentation schemes; in the identification of a way to interact with the user naturally also via voice interaction [22]; in the exploitation of NLP profiling techniques to detect depression and anomalies in the emotional and cognitive status of the user [23].

The adoption of methods grounded on formal techniques throughout all the stages of the Guide engineering will ensure that the actions of the Guide are trustworthy, explainable, and – up to the extent ethics can be formalised and implemented – also ethical [24, 25].

A summary of the main research challenges are:

- developing an intelligent agent approach that supports natural language dialogues with elder users that is suitable for elderly of a particular culture; the evaluation therefore requires human subjects of different cultures to ensure that cultural adaptation works well for different cultures as well as ensuring that the dialogues are suitable specifically for elder users;
- connecting the agent technology with existing smart living environments and smart city infrastructures, so that dialogues are appropriately situated;
- adding features for the intelligent agents to give emotional support for the users as well as care for their overall health (which includes doing physical exercises, taking medication, etc.);
- ensure that access to medical data used in dialogues with the users are ethical and that data about living routines of the users provided to doctors are respectful of privacy.
- verifying formally that dialogues will never lead to unethical or culturally inappropriate interactions.

To address these challenges, our vision builds on a number of technologies:

- **Smart-environment and smart-city infrastructures**. Since the Guides accompany the elderly wherever they are, we need to access data from smart environment sensors as well as inter-operate in "systems of systems" in the context for smart cities, and in particular with hospital systems; we will rely on available standards for this, and connect them to our multi-agent systems infrastructure to allow interconnection of Guides of different people as well as between Guides and existing systems. To this aim we will exploit the lessons we learned while engineering systems that integrate ambient intelligence/IoT on the one hand, and agents/MAS on the other [26, 27, 28]. Besides existing work emerged from the autonomous agents community, in order to immerse the Guides in a smart environment we will also consider the potential offered by open and configurable frameworks like FIWARE[2]. Albeit not being an agent-oriented infrastructure, FIWARE presents many features worth exploring: it has demonstrated its industrial strength in many smart* application domains including healthcare, it implements distributed smart components that interact asynchronously via message passing, and may be in principle integrated with (or "under") JaCaMo, to provide access to the surrounding smart environment via a standard API.

- **NLP profiling techniques and voice-based interaction**. All the interactions between Guides and humans will be through spoken dialogue in natural languages; although existing tools will be used, in our vision we call for a seemless integration between out-of-the-box voice-based human-computer interaction tools and the sophisticated culturally adaptive AI techniques described below, which address the communication level in a broad sense.
- **Big data and computer vision**. We need to have summary information from the relevant data for the various activities the Guides will accompany the user, for supporting medical staff about the ongoing health state of the user, as well as symbolic representation of the surroundings of the user; again we plan to use out-of-the-box techniques for this but connected to our approach on representing environments for autonomous agents.
- **Planning and reasoning**. To provide useful information and support, and to engage in complex dialogues, we rely on various formal techniques such as non-monotonic reasoning, logic programming, and automated planning, adapted to the context of transcultural smart-environment elder care. The literature on adoption of formal techniques for modelling and implementing agent planning and reasoning mechanisms is vast and many recent proposals may be taken under consideration for being integrated into the Guides. They include, for example, extensions of goal-based plans used in BDI programming languages to encapsulate both proactive and reactive behaviour, which supports agent reasoning at runtime [29], contextual planning for multiple agents in ambient intelligence environments, useful for making the plans developed by the Guides aware of the surrounding smart environment [30, 31, 32], dynamic goal decomposition and planning in scenarios characterized by a strong inter-dependency between action and context, needed to cope with unexpected, or even catastrophic, events [33], automated synthesis of a library of template-based process models that achieve goals in dynamic and partially specified environments, which are exactly the kind of environments where the Guides will operate [34].
- **Ontologies**. Ontologies will provide the necessary vocabulary (in various languages and also in accordance with different cultures) to be used in the multi-agent dialogues that the Guides will be able to engage. Normally the Guides only dialogue with their user, but for example in medical consultations the Guides may need to participate in multi-agent

[2] https://www.fiware.org/, accessed on July 2020.

dialogues with the doctor and the user. The interplay between ontologies – and semantic web in general – and BDI-style agents – and declarative agent approaches in general – has been studied for a long time [35]. Our experience ranges from design and development of ontologies in the health domain [36, 37, 38] to their integration into AgentSpeak [13, 39], into MAS via CArtAgO artifacts [40], and into data-aware commitment-based MAS [41]. We will exploit this experience to provide the Guides with the semantic layer required to boost their interaction with users. One further advantage of using ontologies is that they could suitably cope with the dynamism that characterizes the application domain, due not only to the dynamism of the environment, but also to the cultural specificity of the elderly people. Many works on ontology evolution have been proposed in the literature [42], including those connecting ontology evolution and belief revision that seem extremely relevant for our research [43].
Finally, to make ontologies suitable to different cultures, profiles, ages, and genders, but still interoperable, we plan to exploit our background on upper ontologies [44, 45] and design ontologies in such a way that they have some upper layer shared among them, and specialized sub-ontologies for different users and tasks.

- **Argumentation**. The core of the transcultural component of our vision are argumentation protocols based on argumentation schemes (i.e., patterns of reasoning and dialogue exchange); these are used to decide the utterances of the Guides when engaged in dialogues. This is possible in practice given long-term work on the integration of Argumentation Theory techniques into Jason and JaCaMo for both reasoning and communication [14, 46, 47]. Also, the development of an argumentation-based inference mechanism for BDI agents based on Toulmin's model of argumentation [48] recently put forward [49] can be used as an alternative basis for this part of our investigation.
- **Theory of mind**. *"An individual has a theory of mind if he imputes mental states to himself and others. A system of inferences of this kind is properly viewed as a theory because such states are not directly observable, and the system can be used to make predictions about the behavior of others"* [50]. The theory of mind is the ability to attribute mental states – beliefs, intents, desires, emotions, knowledge, etc. – to oneself and to others. Its philosophical roots include the work by Dennet [51] that is very well known to researchers working on BDI-oriented agent theories, languages and architectures. With aging, cognitive abilities including theory of mind seem to decline [52, 53], and cultural factors impact its development as well [54]. Based on these observations, another fundamental aspect of our vision is that we are able to model the minds of users through formal representation of their beliefs and intentions. Current dialogue systems have no such representation and the literature in the area makes it clear that without such a representation, intelligent systems cannot fully address the needs of their users. Our existing framework for theory of mind relies on standard rationality assumptions. Based on our previous, recent investigations [55, 56, 15] we aim at doing pioneering multi-disciplinary work on modelling the minds of elderly and their culture, in particular what are the most appropriate ways to infer beliefs and intentions of users given what they communicate. This is clearly specific to the elderly public and different cultures, specially as the elder might have debilitating diseases that compromise their cognitive processes or even for cultural reasons may want to conceal certain states of mind.

- **Organisations**. Our approach also provides the ability to represent the various organisations that the users are part of (for example elderly clubs, hospitals, former employees, etc.) and this too needs to be adapted to support the different cultural systems where the organisations are situated. By integrating MOISE [57], JaCaMo already supports the specification and implementation of organizations [58], and the research on the exploitation of organizations in MAS is still very active [59, 60].
- **Runtime verification**. Because the basis of our work is formal, this also allows us to employ Runtime Verification (RV) techniques to ensure that Guides never make dialogue utterances that are unethical or inappropriate for the elderly or for a particular culture. RV can also be used at a lower, IoT, level, to check that what sensors transmit is in line with some known pattern recognised as "normal behaviour", to raise an alarm if sensory inputs deviates from that pattern. Runtime verification engines based on computational logic, like RML[3] [61] and the trace expressions formalism it builds on [62, 63, 64, 65] are a promising direction to address this challenge, and are integrated with Jason [66].

Clearly, to evaluate the results of this research, we will need a multidisciplinary team to interact with elderly users, including geriatricians, as well as psychologists, sociologists interested in population aging and philosophers interested in ethical AI systems.

3. Conclusions

This short paper presents the preliminary results of a feasibility study that the authors carried out looking for an answer to the question: "How can we address the growing need of transcultural health-aware tools and technologies for aging people?". We believe that the integration of existing IoT and smart-* approaches can help providing a very effective, pervasive and reliable "sensing layer", on top of which intelligent software agents can be designed and implemented, and can provide the "intelligence layer" needed to implement a cross-cultural Guide. The MAS infrastructure adds a further "intelligent coordination layer" to the architecture. The proper management of emotional aspects is part of this intelligence layer, as the theory of multiple intelligence suggests [67], and a natural interaction interface is the means to reduce the barriers between the users and the Guide.

References

[1] M. Wooldridge, N. R. Jennings, Intelligent agents: theory and practice, Knowledge Eng. Review 10 (1995) 115–152. doi:10.1017/S0269888900008122.
[2] J. Bates, The role of emotion in believable agents, Commun. ACM 37 (1994) 122–125. URL: https://doi.org/10.1145/176789.176803. doi:10.1145/176789.176803.
[3] Y. Shoham, Agent-oriented programming, Artif. Intell. 60 (1993) 51–92. URL: https://doi.org/10.1016/0004-3702(93)90034-9. doi:10.1016/0004-3702(93)90034-9.
[4] D. Pereira, E. C. Oliveira, N. Moreira, Formal modelling of emotions in BDI agents, in: F. Sadri, K. Satoh (Eds.), Computational Logic in Multi-Agent Systems, 8th International

[3] https://rmlatdibris.github.io/.

Workshop, CLIMA VIII, Porto, Portugal, September 10-11, 2007. Revised Selected and Invited Papers, volume 5056 of *LNCS*, Springer, 2007, pp. 62–81. URL: https://doi.org/10.1007/978-3-540-88833-8_4. doi:10.1007/978-3-540-88833-8_4.

[5] B. Alfonso, E. Vivancos, V. J. Botti, Toward formal modeling of affective agents in a BDI architecture, ACM Trans. Internet Techn. 17 (2017) 5:1–5:23. URL: https://doi.org/10.1145/3001584. doi:10.1145/3001584.

[6] Y. Su, B. Hu, Y. Dai, J. Rao, A computationally grounded model of emotional bdi-agents, in: D. Huang, V. Bevilacqua, P. Premaratne, P. Gupta (Eds.), Intelligent Computing Theories and Application - 14th International Conference, ICIC 2018, Wuhan, China, August 15-18, 2018, Proceedings, Part I, volume 10954 of *LNCS*, Springer, 2018, pp. 444–453. URL: https://doi.org/10.1007/978-3-319-95930-6_41. doi:10.1007/978-3-319-95930-6_41.

[7] A. Croatti, S. Montagna, A. Ricci, E. Gamberini, V. Albarello, V. Agnoletti, BDI personal medical assistant agents: The case of trauma tracking and alerting, Artif. Intell. Medicine 96 (2019) 187–197. URL: https://doi.org/10.1016/j.artmed.2018.12.002. doi:10.1016/j.artmed.2018.12.002.

[8] S. Costantini, A. Tocchio, Strips-like planning in the DALI logic programmming language, in: G. Armano, F. D. Paoli, A. Omicini, E. Vargiu (Eds.), WOA 2003: Dagli Oggetti agli Agenti. 4th AI*IA/TABOO Joint Workshop "From Objects to Agents": Intelligent Systems and Pervasive Computing, 10-11 September 2003, Villasimius, CA, Italy, Pitagora Editrice Bologna, 2003, pp. 115–120.

[9] S. M. Khan, Y. Lespérance, SR-APL: a model for a programming language for rational BDI agents with prioritized goals, in: L. Sonenberg, P. Stone, K. Tumer, P. Yolum (Eds.), 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3, IFAAMAS, 2011, pp. 1251–1252. URL: http://portal.acm.org/citation.cfm?id=2034511&CFID=69154334&CFTOKEN=45298625.

[10] S. Sardiña, Y. Lespérance, Golog speaks the BDI language, in: L. Braubach, J. Briot, J. Thangarajah (Eds.), Programming Multi-Agent Systems - 7th International Workshop, ProMAS 2009, Budapest, Hungary, May 10-15, 2009. Revised Selected Papers, volume 5919 of *LNCS*, Springer, 2009, pp. 82–99. URL: https://doi.org/10.1007/978-3-642-14843-9_6. doi:10.1007/978-3-642-14843-9_6.

[11] R. H. Bordini, J. F. Hübner, M. Wooldridge, Programming Multi-Agent Systems in AgentSpeak using Jason, Wiley Series in Agent Technology, John Wiley & Sons, 2007.

[12] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with jacamo, Sci. Comput. Program. 78 (2013) 747–761. URL: https://doi.org/10.1016/j.scico.2011.10.004. doi:10.1016/j.scico.2011.10.004.

[13] V. Mascardi, D. Ancona, M. Barbieri, R. H. Bordini, A. Ricci, CooL-AgentSpeak: Endowing AgentSpeak-DL agents with plan exchange and ontology services, Web Intelligence and Agent Systems 12 (2014) 83–107. URL: https://doi.org/10.3233/WIA-140287. doi:10.3233/WIA-140287.

[14] A. R. Panisson, S. Parsons, P. McBurney, R. H. Bordini, Choosing appropriate arguments from trustworthy sources, in: S. Modgil, K. Budzynska, J. Lawrence (Eds.), Computational Models of Argument - Proceedings of COMMA 2018, Warsaw, Poland, 12-14 September 2018, volume 305 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2018, pp. 345–352. URL: https://doi.org/10.3233/978-1-61499-906-5-345.

doi:10.3233/978-1-61499-906-5-345.
[15] A. R. Panisson, S. Sarkadi, P. McBurney, S. Parsons, R. H. Bordini, On the formal semantics of theory of mind in agent communication, in: M. Lujak (Ed.), Agreement Technologies - 6th International Conference, AT 2018, Bergen, Norway, December 6-7, 2018, Revised Selected Papers, volume 11327 of *LNCS*, Springer, 2018, pp. 18–32. URL: https://doi.org/10.1007/978-3-030-17294-7_2. doi:10.1007/978-3-030-17294-7_2.
[16] A. Lentzas, D. Vrakas, Non-intrusive human activity recognition and abnormal behavior detection on elderly people: a review, Artificial Intelligence Review (2019) 1–47.
[17] A. Hariharan, S. Sourab, V. S. Varshini, I. Rajpal, S. M. George, Remote fall detection system for the elderly, in: 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), volume 1, IEEE, 2019, pp. 870–874.
[18] V. Balas, V. Solanki, R. Kumar, M. Ahad (Eds.), A Handbook of Internet of Things in Biomedical and Cyber Physical System, Springer, 2020.
[19] C. Murphey, D. Um, Development of a sleep monitoring system by using a depth sensor: A pilot study, Advances in Human Factors and Ergonomics in Healthcare and Medical Devices 957 (2019) 191.
[20] S. Modgil, Reasoning about preferences in argumentation frameworks, Artificial intelligence 173 (2009) 901–934.
[21] T. Gruber, Ontologies, Encyclopedia of Database Systems (2008) 1959–1959.
[22] B. Stigall, J. Waycott, S. Baker, K. Caine, Older adults' perception and use of voice user interfaces: A preliminary review of the computing literature, in: Proceedings of the 31st Australian Conference on Human-Computer-Interaction, 2019, pp. 423–427.
[23] Z. Huang, J. Epps, D. Joachim, V. Sethu, Natural language processing methods for acoustic and landmark event-based features in speech-based depression detection, IEEE Journal of Selected Topics in Signal Processing (2019).
[24] High-Level Expert Group on AI, Ethics guidelines for trustworthy AI, 2019. URL: https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.
[25] The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems (Ed.), Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems, IEEE, 2019. URL: https://standards.ieee.org/content/ieee-standards/en/industry-connections/ec/autonomous-systems.html.
[26] A. Casals, A. Belbachir, A. E. Fallah-Seghrouchni, A. A. F. Brandão, Fostering agent cooperation in AmI: A context-aware mechanism for dealing with multiple intentions, in: J. D. Ser, E. Osaba, M. N. Bilbao, J. J. S. Medina, M. Vecchio, X. Yang (Eds.), Intelligent Distributed Computing XII, 12th International Symposium on Intelligent Distributed Computing, IDC 2018, Bilbao, Spain, 15-17 October 2018, volume 798 of *Studies in Computational Intelligence*, Springer, 2018, pp. 225–234. URL: https://doi.org/10.1007/978-3-319-99626-4_20. doi:10.1007/978-3-319-99626-4_20.
[27] A. Freitas, D. Schmidt, A. R. Panisson, R. H. Bordini, F. Meneguzzi, R. Vieira, Applying ontologies and agent technologies to generate ambient intelligence applications, in: F. Koch, F. Meneguzzi, K. Lakkaraju (Eds.), Agent Technology for Intelligent Mobile Services and Smart Societies - Workshop on Collaborative Agents, Research and Development, CARE 2014, and Workshop on Agents, Virtual Societies and Analytics, AVSA 2014, Held as Part of AAMAS 2014, Paris, France, May 5-9, 2014. Revised Selected Papers, volume 498 of

Communications in Computer and Information Science, Springer, 2014, pp. 22–33. URL: https://doi.org/10.1007/978-3-662-46241-6_3. doi:10.1007/978-3-662-46241-6_3.

[28] C. E. Pantoja, H. D. Soares, J. Viterbo, T. Alexandre, A. E. Fallah-Seghrouchni, A. Casals, Exposing IoT objects in the internet using the resource management architecture, Int. J. Softw. Eng. Knowl. Eng. 29 (2019) 1703–1725. URL: https://doi.org/10.1142/S0218194019400175. doi:10.1142/S0218194019400175.

[29] R. H. Bordini, R. Collier, J. F. Hübner, A. Ricci, Encapsulating reactive behaviour in goal-based plans for programming BDI agents: Extended abstract, in: A. E. F. Seghrouchni, G. Sukthankar, B. An, N. Yorke-Smith (Eds.), Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020, International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1783–1785. URL: https://dl.acm.org/doi/abs/10.5555/3398761.3398981.

[30] R. Boukharrou, A. Chaouche, A. E. Fallah-Seghrouchni, J. Ilié, D. Saïdouni, Dealing with temporal failure in ambient systems: a dynamic revision of plans, J. Ambient Intell. Humaniz. Comput. 6 (2015) 325–336. URL: https://doi.org/10.1007/s12652-015-0266-y. doi:10.1007/s12652-015-0266-y.

[31] A. Casals, A. E. Fallah-Seghrouchni, A. A. F. Brandão, C. E. Pantoja, J. Viterbo, Resource-dependent contextual planning in ami, in: E. M. Shakshuki, A. Yasar (Eds.), The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops, April 29 - May 2, 2019, Leuven, Belgium, volume 151 of *Procedia Computer Science*, Elsevier, 2019, pp. 485–492. URL: https://doi.org/10.1016/j.procs.2019.04.066. doi:10.1016/j.procs.2019.04.066.

[32] A. Chaouche, A. E. Fallah-Seghrouchni, J. Ilié, D. Saïdouni, Learning from situated experiences for a contextual planning guidance, J. Ambient Intell. Humaniz. Comput. 7 (2016) 555–566. URL: https://doi.org/10.1007/s12652-016-0342-y. doi:10.1007/s12652-016-0342-y.

[33] S. Costantini, G. D. Gasperis, Dynamic goal decomposition and planning in MAS for highly changing environments, in: P. Felli, M. Montali (Eds.), Proceedings of the 33rd Italian Conference on Computational Logic, Bolzano, Italy, September 20-22, 2018, volume 2214 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018, pp. 40–54. URL: http://ceur-ws.org/Vol-2214/paper4.pdf.

[34] A. Marrella, Y. Lespérance, A planning approach to the automated synthesis of template-based process models, Service Oriented Computing and Applications 11 (2017) 367–392. URL: https://doi.org/10.1007/s11761-017-0215-z. doi:10.1007/s11761-017-0215-z.

[35] V. Mascardi, J. A. Hendler, L. Papaleo, Semantic web and declarative agent languages and technologies: Current and future trends - (position paper), in: M. Baldoni, L. A. Dennis, V. Mascardi, W. W. Vasconcelos (Eds.), Declarative Agent Languages and Technologies X - 10th International Workshop, DALT 2012, Valencia, Spain, June 4, 2012, Revised Selected Papers, volume 7784 of *LNCS*, Springer, 2012, pp. 197–202. URL: https://doi.org/10.1007/978-3-642-37890-4_12. doi:10.1007/978-3-642-37890-4_12.

[36] D. C. Engelmann, J. Couto, V. de Oliveira Gabriel, R. Vieira, R. H. Bordini, Towards an ontology to support decision-making in hospital bed allocation (S), in: A. Perkusich (Ed.), The 31st International Conference on Software Engineering and Knowl-

edge Engineering, SEKE 2019, Hotel Tivoli, Lisbon, Portugal, July 10-12, 2019, KSI Research Inc. and Knowledge Systems Institute Graduate School, 2019, pp. 71–74. URL: https://doi.org/10.18293/SEKE2019-130. doi:10.18293/SEKE2019-130.

[37] A. Ferrando, S. Beux, V. Mascardi, P. Rosso, Identification of disease symptoms in multilingual sentences: An ontology-driven approach, in: D. Ienco, M. Roche, S. Romeo, P. Rosso, A. Tagarelli (Eds.), Proceedings of the First Workshop on Modeling, Learning and Mining for Cross/Multilinguality (MultiLingMine 2016) co-located with the 38th European Conference on Information Retrieval (ECIR 2016), Padova, Italy, March 20, 2016, volume 1589 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016, pp. 6–15. URL: http://ceur-ws.org/Vol-1589/MultiLingMine1.pdf.

[38] L. D. Lauretis, S. Costantini, I. Letteri, An ontology to improve the first aid service quality, in: 2019 IEEE International Conference on Systems, Man and Cybernetics, SMC 2019, Bari, Italy, October 6-9, 2019, IEEE, 2019, pp. 1479–1483. URL: https://doi.org/10.1109/SMC.2019.8914460. doi:10.1109/SMC.2019.8914460.

[39] Á. F. Moreira, R. Vieira, R. H. Bordini, J. F. Hübner, Agent-oriented programming with underlying ontological reasoning, in: M. Baldoni, U. Endriss, A. Omicini, P. Torroni (Eds.), Declarative Agent Languages and Technologies III, Third International Workshop, DALT 2005, Utrecht, The Netherlands, July 25, 2005, Selected and Revised Papers, volume 3904 of *LNCS*, Springer, 2005, pp. 155–170. URL: https://doi.org/10.1007/11691792_10. doi:10.1007/11691792_10.

[40] A. Freitas, A. R. Panisson, L. Hilgert, F. Meneguzzi, R. Vieira, R. H. Bordini, Integrating ontologies with multi-agent systems through CArtAgO artifacts, in: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015, Singapore, December 6-9, 2015 - Volume II, IEEE Computer Society, 2015, pp. 143–150. URL: https://doi.org/10.1109/WI-IAT.2015.116. doi:10.1109/WI-IAT.2015.116.

[41] S. Costantini, G. D. Gasperis, Exchanging data and ontological definitions in multi-agent-contexts systems, in: N. Bassiliades, P. Fodor, A. Giurca, G. Gottlob, T. Kliegr, G. J. Nalepa, M. Palmirani, A. Paschke, M. Proctor, D. Roman, F. Sadri, N. Stojanovic (Eds.), Proceedings of the RuleML 2015 Challenge, the Special Track on Rule-based Recommender Systems for the Web of Data, the Special Industry Track and the RuleML 2015 Doctoral Consortium hosted by the 9th International Web Rule Symposium (RuleML 2015), Berlin, Germany, August 2-5, 2015, volume 1417 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015. URL: http://ceur-ws.org/Vol-1417/paper12.pdf.

[42] F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, Ontology evolution: a process-centric survey, The knowledge engineering review 30 (2015) 45–75.

[43] G. Flouris, On belief change in ontology evolution, AI Communications 19 (2006) 395–397.

[44] V. Mascardi, V. Cordì, P. Rosso, A comparison of upper ontologies, in: M. Baldoni, A. Boccalatte, F. D. Paoli, M. Martelli, V. Mascardi (Eds.), WOA 2007: Dagli Oggetti agli Agenti. 8th AI*IA/TABOO Joint Workshop "From Objects to Agents": Agents and Industry: Technological Applications of Software Agents, 24-25 September 2007, Genova, Italy, Seneca Edizioni Torino, 2007, pp. 55–64. URL: http://woa07.disi.unige.it/papers/mascardi.pdf.

[45] V. Mascardi, A. Locoro, P. Rosso, Automatic ontology matching via upper ontologies: A systematic evaluation, IEEE Trans. Knowl. Data Eng. 22 (2010) 609–623. URL: https:

//doi.org/10.1109/TKDE.2009.154. doi:10.1109/TKDE.2009.154.
[46] A. R. Panisson, R. H. Bordini, Argumentation schemes in multi-agent systems: A social perspective, in: A. E. Fallah-Seghrouchni, A. Ricci, T. C. Son (Eds.), Engineering Multi-Agent Systems - 5th International Workshop, EMAS 2017, Sao Paulo, Brazil, May 8-9, 2017, Revised Selected Papers, volume 10738 of *LNCS*, Springer, 2017, pp. 92–108. URL: https://doi.org/10.1007/978-3-319-91899-0_6. doi:10.1007/978-3-319-91899-0_6.
[47] A. R. Panisson, F. Meneguzzi, M. S. Fagundes, R. Vieira, R. H. Bordini, Formal semantics of speech acts for argumentative dialogues, in: A. L. C. Bazzan, M. N. Huhns, A. Lomuscio, P. Scerri (Eds.), International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014, IFAAMAS/ACM, 2014, pp. 1437–1438. URL: http://dl.acm.org/citation.cfm?id=2617511.
[48] S. E. Toulmin, The uses of argument, Cambridge university press, 2003.
[49] V. de Oliveira Gabriel, A. R. Panisson, R. H. Bordini, D. F. Adamatti, C. Z. Billa, Reasoning in BDI agents using toulmin's argumentation model, Theor. Comput. Sci. 805 (2020) 76–91. URL: https://doi.org/10.1016/j.tcs.2019.10.026. doi:10.1016/j.tcs.2019.10.026.
[50] D. Premack, G. Woodruff, Does the chimpanzee have a theory of mind?, Behavioral and Brain Sciences 1 (1978) 515–526. doi:10.1017/S0140525X00076512.
[51] D. C. Dennett, The Intentional Stance, MIT Press, Cambridge, MA, USA, 1989.
[52] S. Bottiroli, E. Cavallini, I. Ceccato, T. Vecchi, S. Lecce, Theory of mind in aging: Comparing cognitive and affective components in the faux pas test, Archives of Gerontology and Geriatrics 62 (2016) 152–162.
[53] S. Lecce, I. Ceccato, A. Rosi, F. Bianco, S. Bottiroli, E. Cavallini, Theory of mind plasticity in aging: The role of baseline, verbal knowledge, and executive functions, Neuropsychological rehabilitation 29 (2019) 440–455.
[54] A. Shahaeian, M. Nielsen, C. C. Peterson, V. Slaughter, Cultural and family influences on children's theory of mind development: A comparison of australian and iranian school-age children, Journal of Cross-Cultural Psychology 45 (2014) 555–568.
[55] S. Sarkadi, A. R. Panisson, R. H. Bordini, P. McBurney, S. Parsons, Towards an approach for modelling uncertain theory of mind in multi-agent systems, in: M. Lujak (Ed.), Agreement Technologies - 6th International Conference, AT 2018, Bergen, Norway, December 6-7, 2018, Revised Selected Papers, volume 11327 of *LNCS*, Springer, 2018, pp. 3–17. URL: https://doi.org/10.1007/978-3-030-17294-7_1. doi:10.1007/978-3-030-17294-7_1.
[56] S. Sarkadi, A. R. Panisson, R. H. Bordini, P. McBurney, S. Parsons, M. Chapman, Modelling deception using theory of mind in multi-agent systems, AI Commun. 32 (2019) 287–302. URL: https://doi.org/10.3233/AIC-190615. doi:10.3233/AIC-190615.
[57] M. Hannoun, O. Boissier, J. S. Sichman, C. Sayettat, MOISE: an organizational model for multi-agent systems, in: M. C. Monard, J. S. Sichman (Eds.), Advances in Artificial Intelligence, International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI, IBERAMIA-SBIA 2000, Atibaia, SP, Brazil, November 19-22, 2000, Proceedings, volume 1952 of *LNCS*, Springer, 2000, pp. 156–165. URL: https://doi.org/10.1007/3-540-44399-1_17. doi:10.1007/3-540-44399-1_17.
[58] M. R. Zatelli, A. Ricci, J. F. Hübner, Integrating interaction with agents, environment, and organisation in JaCaMo, Int. J. Agent Oriented Softw. Eng. 5 (2016) 266–302. URL: https://doi.org/10.1504/IJAOSE.2016.10001865. doi:10.1504/IJAOSE.2016.10001865.

[59] R. C. Cardoso, T. Krausburg, T. L. Baségio, D. C. Engelmann, J. F. Hübner, R. H. Bordini, SMART-JaCaMo: an organization-based team for the multi-agent programming contest, Ann. Math. Artif. Intell. 84 (2018) 75–93. URL: https://doi.org/10.1007/s10472-018-9584-z. doi:10.1007/s10472-018-9584-z.

[60] A. Ricci, A. Ciortea, S. Mayer, O. Boissier, R. H. Bordini, J. F. Hübner, Engineering scalable distributed environments and organizations for MAS, in: E. Elkind, M. Veloso, N. Agmon, M. E. Taylor (Eds.), Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 790–798. URL: http://dl.acm.org/citation.cfm?id=3331770.

[61] L. Franceschini, RML: runtime monitoring language: a system-agnostic DSL for runtime verification, in: S. Marr, W. Cazzola (Eds.), Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming, Genova, Italy, April 1-4, 2019, ACM, 2019, pp. 28:1–28:3. URL: https://doi.org/10.1145/3328433.3328462. doi:10.1145/3328433.3328462.

[62] D. Ancona, D. Briola, A. E. Fallah-Seghrouchni, V. Mascardi, P. Taillibert, Efficient verification of mass with projections, in: F. Dalpiaz, J. Dix, M. B. van Riemsdijk (Eds.), Engineering Multi-Agent Systems - Second International Workshop, EMAS 2014, Paris, France, May 5-6, 2014, Revised Selected Papers, volume 8758 of *LNCS*, Springer, 2014, pp. 246–270. URL: https://doi.org/10.1007/978-3-319-14484-9_13. doi:10.1007/978-3-319-14484-9_13.

[63] D. Ancona, D. Briola, A. Ferrando, V. Mascardi, Global protocols as first class entities for self-adaptive agents, in: AAMAS, ACM, 2015, pp. 1019–1029.

[64] D. Ancona, A. Ferrando, V. Mascardi, Parametric runtime verification of multiagent systems, in: AAMAS, ACM, 2017, pp. 1457–1459.

[65] A. Ferrando, D. Ancona, V. Mascardi, Decentralizing MAS monitoring with DecAMon, in: AAMAS, ACM, 2017, pp. 239–248.

[66] D. Ancona, S. Drossopoulou, V. Mascardi, Automatic generation of self-monitoring mass from multiparty global session types in jason, in: M. Baldoni, L. A. Dennis, V. Mascardi, W. W. Vasconcelos (Eds.), Declarative Agent Languages and Technologies X - 10th International Workshop, DALT 2012, Valencia, Spain, June 4, 2012, Revised Selected Papers, volume 7784 of *LNCS*, Springer, 2012, pp. 76–95. URL: https://doi.org/10.1007/978-3-642-37890-4_5. doi:10.1007/978-3-642-37890-4_5.

[67] H. Gardner, Frames of Mind: The Theory of Multiple Intelligences, 1983.

Voice assistants in hospital triage operations

Simone Montali, Gianfranco Lombardo, Monica Mordonini and Michele Tomaiuolo

University of Parma, Italy

Abstract

This paper analyzes the creation and usage of a voice assistant for the triage of emergency room patients. This human-centred intelligent system strongly relies on Mycroft, an extensible open source voice assistant. The patients are able to declare their symptoms to the agent, which recognizes the urgency and acts accordingly. The software can even provide useful medical informations to the users.

Keywords

Voice assistant, Mycroft, hospital, E.R., emergency, triage

1. Introduction

While the medical research progressed heavily in the last years, the involved IT infrastructures didn't. This was probably caused by the need of having ultra-reliable software, since no errors are permitted in ER rooms. The use cases of agents in hospitals are numerous. This paper inspects the automation of the triage procedures, usually done by nurses, to help patients faster and more efficiently. The voice assistant is an agent able to connect to Mycroft's cloud services, recognize the patients' symptoms and assign them a priority. It also features a second classifier, based on a deep learning network, trained on a dataset composed of ER rooms interviews. This second classifier is used every time the first one isn't capable of recognizing the symptoms. The intelligent system is even capable of interacting with the italian health ministery's website to retrieve the informations requested by the users. The triage operations produce reports in a standard format, making the integration with existing systems easy, cheap and fast. This article is composed as follows:

- Section 2 introduces this project's context
- Section 3 presents the Mycroft project and its peculiarities
- Section 4 describes the creation of the first module of this project
- Section 5 illustrates the workings of the second classifier
- Section 6 describes the retrieval and conveying of medical informations and tips
- Section 7 describes this project's applicability and requirements

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy

2. Voice Assistants and Triage

2.1. Voice Assistants

Voice assistants are agents capable of interpreting human dialogs and reply through synthetized voices. The most famous examples are Amazon Alexa, Apple Siri, Google Assistant…The assistant waits for a *wake word*, hears the request, transforms it into text, interprets it and *speaks* a reply through a Text To Speech engine. Voice assistants are rarely used in hospitals, but a report from **Voicebot** [1], an organization supporting the spread of these technologies, stated that while just 7.5% of the interviewed people had used a voice assistant in healthcare, 52% would like to do so. Surprisingly, this includes even the higher age groups. Some examples of medical voice assistants have emerged in the last years: for example, *Amazon Alexa* has recently implemented a COVID19 screening skill, triggered by *"Alexa, do I have Coronavirus?"* and followed by questions about the symptoms. Voice based agents are used in another use cases by startup *Orbita, Inc*, which created a voice assistant for hospitals and healthcare workers, handling bedside assistance for hospitalized patients, hospital reception, symptoms screening.

2.2. Triage

Triage is the process in which the patients requesting help in the ER are catalogued by their urgency and their priority [2]. The term was born during wars and maxi-emergencies, but similar operations are done everyday, in hospitals all around the world. The most famous *methods* are the **START** and the **CESIRA**. The Italian health ministery never defined a standard, but generally the patients get divided in 4 groups: red (critical, immediate access), yellow (life threatening, access within 15 minutes), green (access within 60-90 minutes), white[3]. The triage produces a report containing the patient's anagraphic data, the symptoms, the urgency, date and time of access to the hospital. Since these procedures are rather standardized and not much varying, they could be implemented in an autonomous agent, saving time for both the nurse and the patient. A voice assistant could ask the patient all the standard questions, for example the main symptoms, the body temperature, the full name, a pain evaluation…Having done that, the agent could analyze the data and asking for human help only when needed. Having a computer analyze the data instead of a human could, with a sufficient dataset, spot patterns in symptomatologies that a human could not spot. For example, rare diseases are often ignored by healthcare workers because of their rarity, but intersecting biometric data with the patient's declarations could make patterns emerge.

2.3. Triage and COVID19

The COVID19 global pandemic changed the triage procedures heavily: the key is keeping patients with COVID19-compatible symptoms far from the other ones[4]. Usually, this involves the building of *pre-triage* tents, where patients get tested. Integrating a fast COVID19 diagnostic in the agent could reduce the risk of infections.

3. Mycroft

3.1. Why Mycroft?

The most popular voice assistants, like Google Assistant or Amazon Alexa, are proprietary software made by corporations. While this may be good for the usability and functionality, it lacks a critical feature: privacy. Since trust, security and privacy of health data are tremendously precious, we need to protect them, and doing that while using a proprietary voice assistant is kind of impossible. Choosing an open source alternative allows us to guarantee **data privacy**, **extensibility**, **customizability**, and last but not least, **a free tool that every hospital could use**.

3.2. The Mycroft project

Mycroft was born to provide an open source voice assistant to users, developers, researchers.[5] It was initially funded with a crowdfunding in 2015, and obtained great success. The project is based on a subset of tools built by Mycroft, Inc and other companies:

- Precise, a wake word listener based on a *Gated Recurring Unit*, a recurrent neural network similar to a Long Short-Term Memory network with a forget gate [6]. It is trained over audio tracks, not text, making the software compatible with different accents, dialects, languages.
- The Speech To Text component isn't stable enough yet. Because of this, this project uses the Google STT engine [7]. The requests are bucketed and proxied through Mycroft's servers to provide a privacy layer.
- The intent interpretation is based on two tools: **Padatious** and **Adapt**. While Adapt is based on keywords recognition, Padatious is based on a neural network trained over full phrases. Padatious can even extract entities from the intent, like a city name or a disease.
- While Mycroft built a Text To Speech engine, **Mimic**, it isn't currently available in italian. Because of this, we're using the Google STT engine, with proxied requests.
- A graphical user interface was built over the Qt library using KDE Plasmoids, graphical components available in the KDE Plasma desktop environment.

4. Skill Creation

The greatest power Mycroft has to offer is its modularity: every responsibility is tied to a single module, named **skill**, which interprets the related requests and provides an answer. Some examples might be a joke-telling skill, a reminder skill, a timer skill. When Mycroft receives a request, it checks all of the skills' intents for matches. If a match is found, the related function is called. Skills are coded in Python, and have a standard structure. To generate the boilerplate code for a new skill, Mycroft provides a command line interface.

4.1. Skills file structure

Skills have a standard file structure and file types:

- `dialog` files: these files contain the assistants's answer dialogs. Multiple phrases are included and randomly chosen: this makes the assistant more *human*.
- `intent` files: these files are used to train the Padatious' neural network. They contain example phrases for every intent.
- `voc` files: these files contain Adapt keywords. Since this project strictly relies on Padatious, these are almost not used.
- `entity` files: these files contain entities to be extracted by padatious, like city names, diseases…
- `settingsmeta.yaml`: this YAML file defines the skill's settings, to be edited on the Mycroft Home website.
- `__init__.py`: this file contains the skill's main code. It is based on an extension of the `MycroftSkill` class, which contains the classes and decorators needed to define a new skill.

4.2. hospital-triage-skill

The skill is structured as follows: every symptom type is caught in an intent. Basing on the symptom type, different things happen: a urgency code is assigned to the patient (red, yellow, green), anagraphical data is collected, minor symptoms are checked…All the collected data gets saved to a Python dictionary to be exported in JSON. All the *standard* operations, like asking for a pain evaluation or the patient's name, are handled by decorators.

4.3. The `covid_symptom` decorator

The COVID19 global pandemic strongly changed how people behave, interact, get cured. As already discussed in section 2.3, hospitals need to keep COVID19-compatible patients separate from the other ones, and from each other. Automating the COVID19 diagnostic may be crucial now. Therefore, the voice assistant needs to check if the patient may be infected and potentially contagious. Because of this, all of the symptoms that are slightly connected with COVID19 trigger a diagnostic handled by the `covid_symptom` decorator. This asks the patient for known symptoms[8], and multiplies a score (by default equal to 1) by their *multiplier*:

- **Fever**: multiplied by 2.0
- **Sore throat**: multiplied by 1.3
- **Cold**: multiplied by 1.3
- **Breathing fatigue**: multiplied by 1.6
- **Cough**: multiplied by 1.6
- **Contact with infected people**: multiplied by 3.0
- **Missing sense of taste**: multiplied by 1.7

Every suspected patient starts with a score of 1, and if the final score reaches a threshold (by default equal to 15, but it can be adjusted basing on the pandemic's recession) the patient is advised to stay where he is and wait for a doctor to visit him.

4.4. Helpers

The agent needed some helpers to cope with challenges the italian language implies. An example might be the temperature extraction from an utterance: some inputs may be *"trentasette e mezzo"*, *"trentasette punto cinque"*, *"trentasette virgola cinque"*, *"trentasette cinque"*, **all meaning the same thing**. Another example is the number validation: the bot may interpret a "sei" as the verb *essere* instead of a number.

4.5. The exported data

All of the collected and infered data gets exported to a JSON file. Using a standard like JSON makes integrating the bot with existing systems easy: the hospitals don't need to entirely change their informative systems, they just need to integrate some small components. An example of triage report of a COVID19 infected patient may be the following:


```
{
    "can_talk": "yes",
    "main_symptom": "breathing",
    "code": "red",
    "fever": 38.7,
    "has_sore_throat": "yes",
    "has_cold": "no",
    "has_breathing_difficulties": "yes",
    "has_cough": "yes",
    "has_had_contacts": "yes",
    "misses_taste": "yes",
    "covid_score": 22.6304,
    "symptom_declaration": "ho fiatone",
    "age": 21,
    "other_symptoms": "ho mal di testa",
    "pain_index": "8",
    "full_name": "marco rossi"
}
```


4.6. Conversation example

A conversation with the bot may be the following:

- **User**: *Can you help me?*
- **Voice assistant**: *Are you the one seeking medical treatment?*
- **User**: *Yup.*
- **Voice assistant**: *What is your main symptom?*
- **User**: *I have a fever.*
- **Voice assistant**: *Ok, a fever. Have you measured it recently?*
- **User**: *Yes.*

- **Voice assistant**: *Please, tell me your temperature.*
- **User**: *Thirtyseven point eight.*
- **Voice assistant**: *I need some additional informations for the diagnosis. I will now ask you some yes/no questions. Do you have a sore throat?*
- **User**: *Yes.*
- **Voice assistant**: *Do you have a cold?*
- **User**: *Yes.*
- **Voice assistant**: *Do you have any trouble breathing?*
- **User**: *No.*
- **Voice assistant**: *Do you have a cough?*
- **User**: *Yes.*
- **Voice assistant**: *Have you been in contact with people infected by the new coronavirus?*
- **User**: *Yes.*
- **Voice assistant**: *We're notifying a doctor. It is possible that you may be infected by COVID19. Could you tell me how old you are?*
- **User**: *Twentytwo.*
- **Voice assistant**: *How would you define your discomfort from 1 to 10?*
- **User**: *Eight.*
- **Voice assistant**: *If there's any other symptoms I need to know, please tell me*
- **User**: *Nothing else.*
- **Voice assistant**: *Could you please tell me your full name?*
- **User**: *Simone Montali.*
- **Voice assistant**: *Simone Montali, is that right?*
- **User**: *Yes.*
- **Voice assistant**: *Thanks for your patience, go at the RED desk.*

5. Fallback skills and NLP

A problem now arises: what if the patient has a disease which was not covered in the previous symptoms? The intent would not be caught. Sadly, diseases are so vast and numerous that it would be impossible to hardcode them: we need a *statistical* approach. The target is now building a text classifier that is able to classify the patient's declaration to a symptomatology or at least an hospital ward.

5.1. Dataset

Since the medical data is fortunately well protected by privacy laws all around the world, finding medical datasets is not easy. However, creating a dataset from scratch is still possible: this is exactly what **Appen**, an australian start up, did. The dataset used in this project [9] contains audio snippets, transcriptions and symptomatologies of patients requesting help to nurses. Since the Mycroft's STT engine handles the conversion to text, we only need to train a classifier that maps patient requests to symptom types. An example of this dataset's entries may be:

- **Phrase**: *My son had his lip pierced and it is swollen and the skin inside on his lip is grey and looks infected.*
- **Prompt**: *Infected wound*

5.2. Dataset translation

Since the voice assistant was mainly born for Italian users, we need to translate the dataset to Italian. In fact, the approach of creating an Italian dataset from scratch[10, 11] would be impractical in this case. Using Google Translate's APIs would be easy but costly. We could use a scraper to query Google Translate's webpage, but that would be slow. Fortunately, third party Python modules have been made: `googletrans` is the most famous. Using it, we were able to translate the dataset with a simple script.

```
t = translator
for index, row in data.iterrows():
    phrase = translator.translate(
        row["phrase"],
        dest="it").text
    prompt = t.translate(
        row["prompt"],
        dest="it").text
    if (t.detect(phrase).lang == "it" and
        t.detect(prompt).lang == "it"):
        td = translated_data
        td.at[i, "phrase"] = phrase
        td.at[i, "prompt"] = prompt
        i = i+1
```

When Google Translate can't recognize a phrase's meaning, it skips it. The `translator` object, besides translating things, can even detect languages. This allows us to do a little workaround: if the detected language isn't italian in both the phrase and the symptomatology, the entry is skipped. This reduced the dataset from 8000 entries to almost 2000.

5.3. Classifier training

Having an italian dataset, we can now proceed with the classifier's training. To do so, using the **fastai** library simplifies things: it provides the `text.learner` class, which is able to train a model and export it. This enables us to train the model a single time, and simply import it into the voice assistant.

5.4. Data preparation and training

We can now import our dataset in a Jupyter notebook, check for NaNs, and create the `text.learner` with fastai. For the project, we used the AWD-LSTM structure, a recurrent neural network which has really high performance on text processing. This type of network has a **Long Short**

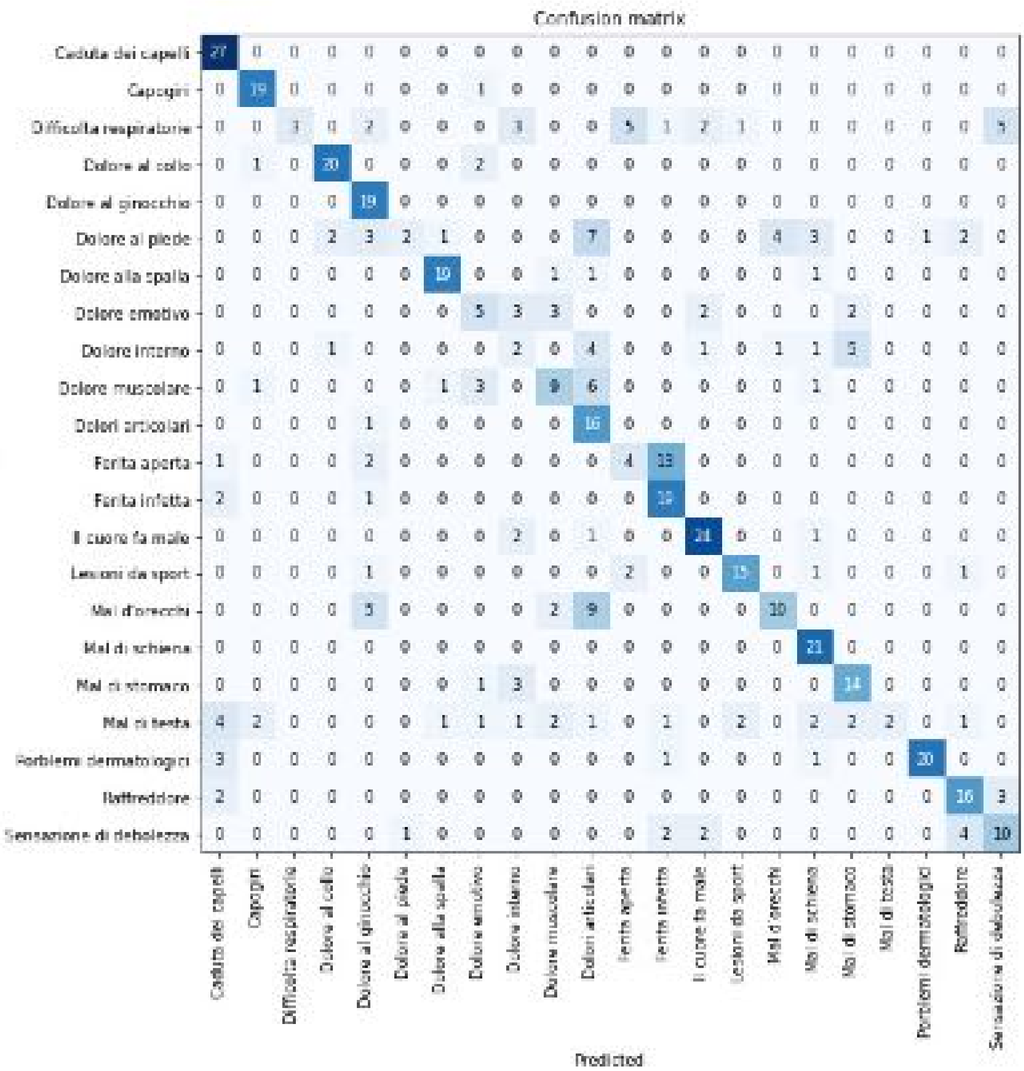


Figure 1: Confusion Matrix

Term Memory[12], which enables the training to find patterns in distant parts of the text, **Average SGD** and **Dropconnect**, a dropout that drops weights instead of activations [13]. We can now train the network and check the confusion matrix: While it isn't perfect, the diagonal in the matrix shows that **the model was trained correctly**. Speaking about accuracy, the model reached **more than 60%**: We can now easily export it and integrate it in the voice assistant by a simple import.

epoch	train_loss	valid_loss	accuracy	time
0	3.092835	3.079816	0.051282	01:43
1	2.957276	2.627386	0.198718	01:42
2	2.600713	1.969201	0.457265	01:32
3	2.192694	1.707950	0.598291	01:38
4	1.923900	1.633354	0.632479	01:35

Figure 2: Accuracy

5.5. Fallback skills creation

As explained earlier, Mycroft works by recognizing intents in the user's requests. These were defined in the last skill, but the symptomatologies are more than the ones coded. However, Mycroft makes it possible to define **fallback skills**, a skill that gets triggered every time the intent is not recognized. By creating one, we're able to get the patient's utterance and analyze it with our classifier. Fallback skills are ordered by a priority, but since our voice assistant will have just one, it doesn't matter. After registering the fallback skill, we can import the model. The symptomatologies inserted in the model are defined in a separate `classes.json` file, containing each symptom's urgency, GUI emoji, name, and its compatibility with COVID19. An example would be:

```json
{
    "name": "Dolore al collo",
    "emoji": "[removed]",
    "code": "yellow",
    "covid": false
}
```

Having done that, the symptoms can now be handled like any other one. Ideally, we'd like to check if the classifier worked right by asking the patient. An example of conversation may be:

- **User**: *I need help.*
- **Voice assistant**: *Are you the one seeking medical treatment?*
- **User**: *Yes.*
- **Voice assistant**: *What is your main symptom?*
- **User**: *My shoulder dislocated while I was playing golf.*
- **Voice assistant**: *So, we're talking about shoulder pains, right?*
- **User**: *Yes.*
- **Voice assistant**: *Could you tell me how old you are?*
- **User**: *Twenty.*
- **Voice assistant**: *How would you define your pain from 1 to 10?*

- **User**: *Nine.*
- **Voice assistant**: *If there's any other symptoms I need to know, please tell me*
- **User**: *Nothing else.*
- **Voice assistant**: *Could you please tell me your full name?*
- **User**: *Mario Rossi.*
- **Voice assistant**: *Mario Rossi, is that right?*
- **User**: *Yes.*
- **Voice assistant**: *Thanks for your patience, go at the YELLOW desk.*

6. Information retrieval

This agents's use cases might not be limited to triage and helping in emergencies. In fact, a patient could just want to know some medical informations about a particular disease without having to talk to a doctor.

6.1. Data source

It is obvious that the data source we're looking for has to be authoritative: Wikipedia is not reliable enough. Luckily, the italian health ministery published a medical encyclopedia on their website. These informations are complete, reliable and interesting. To be able to use them, we can scrape the website using tools like **Selenium**. Selenium is a framework used to test web applications; it provides a simple interface to automate a web browser[14]. We could use two approaches for the data collection: an API approach, which downloads the informations basing on queries, or a periodic one. The first one is possible thanks to the Python `http.server` library, in which we can define a handler for GET requests. This will look for the query on the encyclopedia, download the page, insert the data into a JSON and return it. Since the informations on the website rarely vary, we can download them periodically and keep them saved. By defining a script that iterates over every item in the encyclopedia, we can download all the needed informations. Using **cron**, a job scheduler for Linux, it's possible to automatically download the data periodically.

6.2. Integration of the data into the voice assistant

Having downloaded the data, we can now define a new intent in the voice assistant to request informations. Padatious' entity extraction simplifies this: we can train it with phrases like *"Tell me something about {disease}"*, creating a *disease* entity containing all the known diseases. These are not exact matches: we're looking for the best match, so that if the user asks something about *tumor in the lung*, *lung tumor* is matched. After matching the name, the assistant checks the known data, like *prevention, cure, therapy, symptoms*, and asks the patient to choose. An example of conversation might be the following:

- **User**: *What do you know about celiac disease?*
- **Voice assistant**: *The best match is celiac condition, is that what you're looking for?*
- **User**: *Yes.*

- **Voice assistant**: *What would you like to know? Choose between Description, Symptoms, Causes, Therapy, Complications*
- **User**: *Tell me the causes.*
- **Voice assistant**: *This is what I know: celiac disease is a multifactorial condition…*

7. Applicability and technical requirements

Since this agent could enhance the hospitals' workflow all around the world, it was built to be as cheap and easy as possible. The Text To Speech and Speech To Text engines are computed in cloud: this removes the most computationally heavy module from our responsibility. The intent parser, Padatious, needs an ultra light training just once every start of the bot. Even if the computer is not powerful, it requires just a few minutes. The fastai training is computed in cloud: the model gets exported and used without any further training. This doesn't require any computational power. The only real requirement is defined by the GUI: since it is built on KDE Plasmoids, the desktop environment has to be KDE Plasma. Mycroft recommends using KDE Neon, a Ubuntu LTS distribution with KDE Plasma installed. The devices that support this project are, therefore, almost infinite: even a simple Raspberry Pi or an old computer can run Mycroft.

8. Conclusions

The global pandemic the world had to bear with has highlighted how hospitals, in the last years, have been forgotten by politicians, people, nations. In the last 20 years, in Italy, the number of institutes diminished from 311 thousands to 191 thousands, while the public spending was raised from 60 miliards to 112[15]. With the aid of human-centred intelligent systems[16, 17], health workers could get some relief and concentrate on what matters more: curing people. Voice assistants could *flatten* the *digital divide*, being easy to use for everyone, including the elder. Translation of the agent to multiple languages could be an incredible tool for immigrants, tourists and travelers who can't speak the national language. Right now, the most urgent improvement to make is finding or creating a better dataset: if the classifier worked better, we could skip the hardcoding of the most common symptoms and just leave one skill. The ultra low cost of this project's requirements makes it possible for every hospital in the world to install a tool like this. The open source design (all the source code is licensed with a GPL-3.0 license) guarantees freedom, privacy, expansibility, customizability, and, last but not least, the possibility of saving lives without having to pay a price.

9. Source code

Every single line of code of this project is open source and available on GitHub:

- Mycroft skill for the triage operations and the information requests: montali/hospital-triage-skill

- Fallback skill for the NLP classification and diagnosis of symptoms: montali/hospital-fallback-skill
- Health Ministery's encyclopedia scraper: montali/medical-infos-api

10. Acknowledgements

The authors of this paper would like to thank Prof. Elena Bignami, M.D. in Anesthesiology, Critical Care and Pain Medicine Division, Department of Medicine and Surgery, University of Parma and her team for their collaboration and their precious advices.

References

[1] B. Kinsella, A. Mutchler, Voice assistant consumer adoption in healthcare, Voicebot.AI & Orbita Report (2019). URL: https://voicebot.ai/voice-assistant-consumer-adoption-report-for-healthcare-2019/.

[2] J. Bazyar, M. Farrokhi, A. Salari, H. R. Khankeh, The principles of triage in emergencies and disasters: a systematic review, Prehosp Disaster Med 6 (2020) 1–9.

[3] Linee di indirizzo nazionali sul triage interospedaliero, Ministero Della Salute - Repubblica Italiana, 2019. URL: http://www.salute.gov.it/imgs/C_17_notizie_3849_listaFile_itemName_1_file.pdf.

[4] T. J. Judson, A. Y. Odisho, A. B. Neinstein, J. Chao, A. Williams, C. Miller, T. Moriarty, N. Gleason, G. Intinarelli, R. Gonzales, Rapid design and implementation of an integrated patient self-triage and self-scheduling tool for covid-19, Journal of the American Medical Informatics Association 27 (2020) 860–866.

[5] R. Harrington, Customize an open-source hal for your home, Popular Science (2016). URL: https://www.popsci.com/ultimate-diy-ai/.

[6] F. A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with lstm, Neural Computation 12 (2000) 2451–2471.

[7] Speech-to-text, Google AI and machine learning products (2020). URL: https://cloud.google.com/speech-to-text.

[8] Nuovo coronavirus covid-19, Pagine Mediche (2020). URL: https://www.paginemediche.it/coronavirus.

[9] P. Mooney, Medical speech, transcription, and intent, 2019. URL: https://appen.com/datasets/audio-recording-and-transcription-for-medical-scenarios/.

[10] G. Lombardo, P. Fornacciari, M. Mordonini, L. Sani, M. Tomaiuolo, A combined approach for the analysis of support groups on facebook-the case of patients of hidradenitis suppurativa, Multimedia Tools and Applications 78 (2019) 3321–3339. doi:10.1007/s11042-018-6512-5.

[11] G. Lombardo, A. Ferrari, P. Fornacciari, M. Mordonini, L. Sani, M. Tomaiuolo, Dynamics of emotions and relations in a facebook group of patients with hidradenitis suppurativa, in: International Conference on Smart Objects and Technologies for Social Good, Springer, 2017, pp. 269–278. doi:10.1007/978-3-319-76111-4_27.

[12] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.
[13] S. Merity, N. S. Keskar, R. Socher, Regularizing and optimizing lstm language models, arXiv preprint arXiv:1708.02182 (2017).
[14] Selenium official repository, Last accessed: September 2020. URL: https://github.com/SeleniumHQ/selenium.
[15] La riduzione di ospedali e posti letto negli ultimi 10 anni, Agenzia Giornalistica Italia (2020). URL: https://www.agi.it/fact-checking/news/2020-03-14/coronavirus-rianimazione-posti-letto-7500889/.
[16] G. Cabri, M. Cossentino, E. Denti, P. Giorgini, A. Molesini, M. Mordonini, M. Tomaiuolo, L. Sabatucci, Towards an integrated platform for adaptive socio-technical systems for smart spaces, in: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE, 2016, pp. 3–8.
[17] G. Lombardo, P. Fornacciari, M. Mordonini, M. Tomaiuolo, A. Poggi, A multi-agent architecture for data analysis, Future Internet 11 (2019) 49.

Session 5
Agents & Actors for Data Science

Calibration of an agent-based model for opinion formation through a retweet social network

Loretta Mastroeni[a], Maurizio Naldi[b] and Pierluigi Vellucci[a]

[a]Dept. of Economics, Roma Tre University, Via Silvio D'Amico 77 00145 Rome, Italy
[b]Dept. of Law, Economics, Politics and Modern languages, LUMSA University, Via Marcantonio Colonna 19 00192 Rome, Italy

Abstract

Calibration of agent-based models (ABM) for opinion formation is needed to set their parameters and allow their employment in the real world. In this paper, we propose to use the correspondence between the agent-based model and the social network where those agents express their opinions, namely Twitter. We propose a calibration method that uses the frequency of retweets as a measure of influence and allows to obtain the influence coefficients in the ABM by direct inspection of the weighted adjacency matrix of the social network graph. The method has a fairly general applicability to linear ABMs. We report a sample application to a Twitter dataset where opinions about wind power (where turbines convert the kinetic energy of wind into mechanical or electrical energy) are voiced. Most influence coefficients (76%) result to be zero, and very few agents (less than 5%) exert a strong influence on other agents.

Keywords

Opinion formation, Agent-based models, Twitter, Calibration

1. Introduction

Agent-based models (ABM) are increasingly used to analyse opinion formation (see the survey in [1]), as an alternative to econophysics models [2]. Some examples with general applicability are described in [3, 4, 5, 6]; they are also applied to study specific phenomena such as equality bias [7] or personal finance decisions [8, 9].

However, models with no application to real world data may be too abstract. In the mathematical model describing the interactions among agents, we need to set the parameters governing those interactions, i.e. to calibrate those models. Calibration allows us to obtain realistic expectations about the behaviour of a social group. Very few studies have attempted to calibrate agent-based models. We can group them into two classes, where data are obtained respectively through a laboratory experiment of from the observation of a real social network.

One of the first example of the former class is the celebrated Friedkin-Johnsen model [10], which has been evaluated through data collected on small groups in a laboratory [11], where the authors asked their subjects to estimate the extent to which each other group member influenced their final opinion by means a mechanism based on a reward paid in poker chips. Another example is provided in [12], where participants in the experiment expressed their

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
loretta.mastroeni@uniroma3.it (L. Mastroeni); m.naldi@lumsa.it (M. Naldi); pierluigi.vellucci@uniroma3.it (P. Vellucci)

opinion about the best location for a new leisure center (on the line across two towns), and the influence could be observed through the evolution of one's opinion after examining the opinions of others.

Influence in real contexts has instead been studied through social networks. An advice network inside a manufacturing company was considered in [13] on the basis of the data provided in [14]. The influence of any individual was assumed to be proportional to the number of people who seek advice from him/her. A political context, namely the American Senate, was instead investigated in [15], where co-sponsorship of bills was taken as a measure of influence. The opinion value for each senator was assumed to be the fraction of votes when he/she was present and voted with the majority. Finally, a co-habitation context, namely a university student dormitory, was analysed in [16], where the Social Evolution dataset enclosed in [17] was employed. A similar context is also the stage for the study contained in [17], where surveys were conducted monthly on social relationships, health-related habits, on-campus activities and other issues.

Here we wish to propose a method to calibrate agent-based models for opinion formation, considering data extracted from an online social network, namely Twitter. This contrasts with the contributions appeared in the literature so far, where just physical social networks have been considered. The abundance of data appearing in online contexts makes them a natural choice to look at for our goal. We measure the influence by the frequency of retweeting, which makes our methods applicable to any social network where reposting of opinions is allowed.

Our major original contributions can be summarised as follows:

- we propose a calibration method based on the adjacency matrix in an online social network;
- we provide a systematic assessment of its applicability, considering a taxonomy of agent-based models;
- we demonstrate its application using Twitter opinions on wind power;
- for that specific context, we show that, though most actors exert an influence, very few exert a strong influence (i.e., being retweeted more than once);
- for that specific context, we show that the influence of any agent is limited to one other agent in most cases so that the matrix of influence coefficients is sparse.

2. The retweet network

We base our calibration on Twitter data. In this section, we report some basic information about Twitter and using retweets to calibrate an agent-based model.

Twitter is a popular messaging service, with 330 million monthly active users (see https://www.oberlo.com/blog/twitter-statistics), where messages (aka tweets) are no longer than 140 characters [18]. Message receiving uses an opt-in mechanism: you decide to *follow* somebody and receive all his/her updates. Messages may include *hashtags*, i.e., # symbol terms that associate

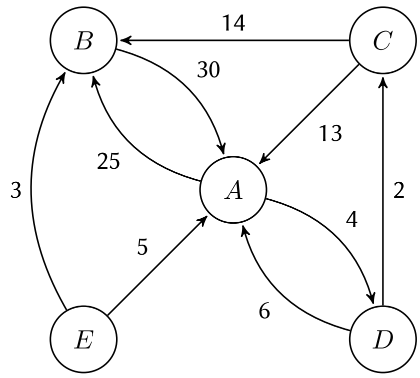


Figure 1: Toy retweet network

a topic to the message: all messages concerning a specific topic (as identified by the hashtags included in those messages) can be retrieved at once by searching for that hashtag.

Twitter allows retweeting a message, i.e., reposting somebody else's tweet. This is a sign of support for somebody else's opinion, stronger than just following him/her because it is an uncritical sharing of his/her opinions (retweeting through the Retweet button does not allow to add comments). It is also specific because it concerns a single message. Being retweeted is a sign of influence. We are using retweets to measure the influence of somebody's opinion on other participants in a social network. In particular, we build a retweet network. Retweet networks have been designated (see Chapter 4 of [19]) as a major tool to analyse the influence of twitterers. In our network each node represents a twitterer (an agent in the corresponding agent-based model) and an edge from node A to node B means that A has been retweeted by B. The weight associated to that edge is the number of retweets: the weighted out-degree of a node is the measure of the associated agent's influence. In the end, we obtain a weighted directed network. It is to be noted that this network allows us to quantify the influence of each agent on each other agent, which is what we need in our any-to-any agent-based model, rather than the overall influence of a twitterer (as in [20]).

In Figure 1, we show a toy retweet network made of five twitterers to illustrate this concept. We see that C has been retweeted 13 times by A, and 14 times by B, but has never retweeted either A or B. Retweeting is not symmetric in general.

3. Calibration methodology

Our aim is to set the parameters defining an agent-based model for opinion formation, i.e., to calibrate the model, exploiting the data retrieved from Twitter. In this section, we describe our calibration method and highlight its applicability.

For the time being, without loss of generality, we refer to an agent-based model such as described in [5], where the generic agent i features a quantity $x_i(t)$ representing its opinion at time t. Then we denote the opinions of all the agents at time t by the vector $\underline{x}(t)$; its evolution over time follows the state updating equation:

$$\underline{x}(t+1) = S\underline{x}(t), \tag{1}$$

with S being the matrix of influence coefficients. Namely, the element s_{ij} of S describes the effect of the opinion of the agent j on the opinion of the agent i.

The process leading from Twitter activity to the matrix of influence coefficients is made of the following phases:

1. scraping Twitter data;

2. building the retweet network and extracting the adjacency matrix;

3. mapping the edge weights on the influence coefficients.

The first phase consists in retrieving the list of tweets/retweets concerning a specific hashtag (or any combination of hashtags and keywords). This phase is often referred to as *Twitter Scraping* and can be easily accomplished using Twitter's API (Application Programming Interface), accessible upon opening a *Twitter developer account.* Tweets were retrieved using the R package twitteR [21]. The search index has a 7-day limit, which means that only tweets posted in the latest seven days will be retrieved. Our inspection interval falls in the week ending on December 9, 2019.

As to Phase 2, we build the retweet network as in Section 2, i.e. a social network where the nodes are the twitterers, who represent the agents in our ABM. That social network is described by its weighted adjacency matrix M, whose generic element m_{ij} is the number of times that the twitterer i has been retweeted by the twitterer j.

For the computation of influence coefficients (Phase 2), we assume that the set of twitterers is the set of agents. If the order of elements in the two sets is not the same, a permutation on either set is needed before applying the calibration method. Since the weights of the edges in the retweet network represent the influence exerted by the twitterers (i.e., the agents) on one another, we obtain the matrix of influence coefficients in the ABM by the simple equation $S = M$, if no normalization is needed. However, it is to be noted that the adjacency matrix elements $m_{ij} \in \mathbb{N}$. A further stage is therefore needed if the influence coefficients belong to a different domain. We can just examine the cases where the domain of the influence coefficients is either $\mathbb{N}$ or $\mathbb{R}^+$. We introduce the set V of values acceptable for the influence coefficients. For example, in [5], we have $V = [0, 1]$. Let's consider first the case where $V \subset \mathbb{R}^+$. Assuming that V always includes the value 0 to describe the case of no influence, defining v as the upper bound of V and $m = \max_{i,j} \; m_{ij}$ we can map the values of the adjacency matrix M into influence coefficients by the following linear scaling

$$s_{ij} = \frac{v}{m} m_{ij} \qquad i, j = 1, 2, \ldots, n. \tag{2}$$

If $V \subset \mathbb{N}$, we have instead either a contraction mapping or a dilation mapping according to whether we have $m > v$ or $m < v$ respectively.

For the applicability of the method, we recall the following categories adopted to classify agent-based models in the survey [1]:

- opinion domain;
- interaction direction;
- interacting agents;
- updating equation;
- updating frequency;
- utility function.

Opinion domain. Though an opinion is intrinsically a qualitative and potentially multifaceted feature, we have to describe it by a numeric variable and choose the domain where that variable can lie. The following three domains have been surveyed in [1]:

- discrete;
- continuous over a bounded interval;
- continuous over $\mathbb{R}$.

In the discrete case, the agent may choose its opinion within a limited set. Discretization lends itself well to represent a qualitative feature through a proper mapping. For example, if we consider the simplest discrete case where we have a binary opinion variable, the two values may represent respectively a positive versus a negative opinion.

If the domain is instead a continuous but bounded interval, a common choice is the $[0, 1]$ interval.

The method proposed here is generally applicable to any opinion domain, since the applicability condition concerns the values of the influence coefficients. However, the closure property may impose some conditions on the influence coefficients. By closure property we mean that the set V is closed with respect to the application of the state updating equation. For example, in [5], where $V = [0, 1]$ and the agents belong to one of c classes (n_i representing the number of agents in class i), and similarly in [4] for the pairwise case, the closure property requires that

$$s_{ii}(n_i - 1) + \sum_{\substack{k=1 \\ k \neq i}}^{c} s_{ik} n_k \leq 1 \qquad i = 1, 2, \ldots, \sum_{j=1}^{c} n_j \tag{3}$$

Interaction direction. That feature considers which way the agents influence each other. again, we refer to the classification established in [1]. In a bilateral interaction, any two agents always influence each other mutually. We have instead a unilateral influence when an agent may influence another agent without being influenced by it. Even in the bilateral case, the influence may not be perfectly symmetrical, since we can have different weights in the opinion updating equations, signalling that the impact of agent X on agent Y is different from that in the reverse direction.

The method is applicable when the interaction direction is bilateral, non symmetric since retweeting may take place in either direction.

Interacting agents. As to the number of agents that interact at each step, the classification adopted in [1] considered the following three:

- pairwise;
- any-to-any;
- closest neighbours.

In the pairwise case, just two agents interact at any single opinion updating round. Since the pairs may change at each round, any agent may interact (influencing or being influenced by) with any other agent in the long run. The any-to-any interaction case is obviously the case where all the agents change their opinion at each time step, since they are influenced by the opinions of the other agents at the previous time step. Finally, in the closest neighbour case, any agent interacts just the closest agents (where the notion of closest involves the use of some distance metric, which is natural in a social network).

In our method, since the interaction is associated with retweeting, and any twitterer can retweet any other twitterer, the calibration we propose applies to all three categories. For the case of pairwise interaction, the weights will be employed in pairs at each round, though they have been estimated considering the embedding social network as a whole. As to the closest neighbour case, if the distance employed for agents is that established on the embedding retweet network, there is actually no difference, since the closest neighbour are those retweeting and actually the only ones exhibiting a non-zero weight.

Updating equation. The updating equation is the function that relates the opinion of an agent to the opinions of the other agents. Here, the classification proposed in [1] is quit simple, considering a linear vs a nonlinear model.

Here we have described the method considering just linear updating equations so far, but it could also be applied to non-linear updating equations, though requiring a more complex mapping from M to V.

Updating frequency. This parameter can be considered as the speed of the opinion formation process. The survey in [1] considers periodic and aperiodic updating. In the periodic setting, each time step involves a change of opinions for all the agents. On the other hand, we fall in the aperiodic case when just a couple of agents changes their opinion at each time step (and it is not known in advance when their turn comes again), or opinions are updated just after a triggering event, or opinion change takes place for a random selection of agents at each time.

Our calibration method is agnostic to the choice of updating frequency, since it can be applied as many times as desired.

Utility function. Again, our calibration method is agnostic to the choice of utility function, as long as that does not impact on the influence coefficients.

4. The dataset

As recalled in the Introduction, for our calibration method we have chosen an application example concerning the influence of people' opinions about wind power. In this section, we describe our dataset and the procedure we have adopted to build it.

The procedure goes along the following four phases:

1. tweet retrieval;

2. duplicate removal;
3. selection of relevant tweets;
4. retweet network building.

In order to retrieve all relevant tweets, we exploit the Twitter API by searching for all the tweets containing either of the following word combinations:

- *wind* AND *power*;
- *wind* AND *energy*.

We deem those words to be fairly representative of the tweets associated to wind power for our calibration demo. Of course, if our aim were to go beyond a demo, we could devise an ampler set of words to obtain a dataset as exhaustive as possible. In this paper, we consider the tweets posted in the week ending on December 9, 2019. Again, the calibration could be made more accurate by extending the analysis horizon over several weeks or even longer periods.

Since many tweets contain both the above combinations, our basket after the retrieval phase may contain duplicates. In the second phase of our procedure, we remove all duplicates. The unique tweets after Phase 2 are 36539.

We must however recognize the possibility of including tweets that, despite binding the word *wind* to *power* or *energy* terms, are not relevant to our actual theme, i.e. the use of wind to get electrical power. Examples of such tweets are shown in Figure 2.

We need to eliminate as many as possible non-relevant tweets. In order to arrive at a set of relevant tweets, we employ a semi-automatic procedure, based on hashtags and the co-occurrence principle. Our procedure, which makes up Phase 3 of the overall procedure mentioned above, goes through the following steps:

1. select the k most frequent hashtags in the dataset of interest;
2. identify the hashtags that are surely relevant with out topic and form a group with them (say Group X);
3. form a group with all the other hashtags (say Group Y);
4. examine all tweets containing Group Y hashtags but not Group X ones, and move their hashtags to Group X if those tweets are relevant;
5. assign to Group X all the hashtags co-occurring with Group X hashtags (this is not done iteratively, but just once for each Group X hashtag).

The last step of the procedure is equivalent to:

1. building the network of hashtags, where an edge is drawn between two hashtags if those two hashtags co-occur in at least one tweet;
2. identifying Group X hashtags;

(a) Tweet 1

(b) Tweet 2

(c) Tweet 3

Figure 2: Example of non-relevant tweets.

3. adding neighbours of Group X hashtags to Group X.

A subset of the hashtag network is shown in Figure 3, where the nodes (hashtags) belonging to Group X are highlighted in dark colour.

At this point we have the Group X of relevant hashtags. We can consider a tweet as relevant if it contains any hashtag included in Group X and build the retweet network on the basis of those tweets as described in Section 3. The resulting network is built out of 4739 tweets and is made of 3528 nodes and 3617 edges.

Figure 3: Network of hashtags

5. Experimental calibration results

After proposing the calibration method in Section 3, in this section we apply it to the dataset described in Section 4.

In Figure 4, we show the resulting retweet network. In order to avoid excessive garbling, we have drawn just the nodes with degree larger than one and have arranged it by degree, so that the most central nodes are located in the inner core of the graph. We see that there are just a few very much retweeted twitterers.

Aside from the sheer demonstration of the applicability of our calibration method, we exploit this application to investigate the following research questions (RQ):

1. how frequent is the retweet phenomenon?
2. how widespread is the influence of any single agent upon the community?
3. how heavy is the influence of any single agent on another specific agent?

As to RQ1, we already have a partial answer from Figure 4. There are wide imbalances in the connectivity of individual nodes. The sparsity of the adjacency matrix is 99%; this means that the network is extremely far from a fully-meshed network where every twitterer retweets all his/her fellows. However, this should be better investigated over time, since we considered a single week, and retweeting relationship accrue over time. In fact, though older tweets are often quickly forgotten, most users tend to "live in the present", forgetting or abandoning topics they followed just some hours or days after, the retweeting relationship lasts over time if the retweeting user is a convinced follower/supporter of the retweeted one.

In order to answer RQ2, we show the distribution of nodes by their degree in Figure 5(a). We see that 89.6% of the agents influences just another agent (the corresponding nodes have degree 1), though there is a very small number of agents exerting their influence on a large

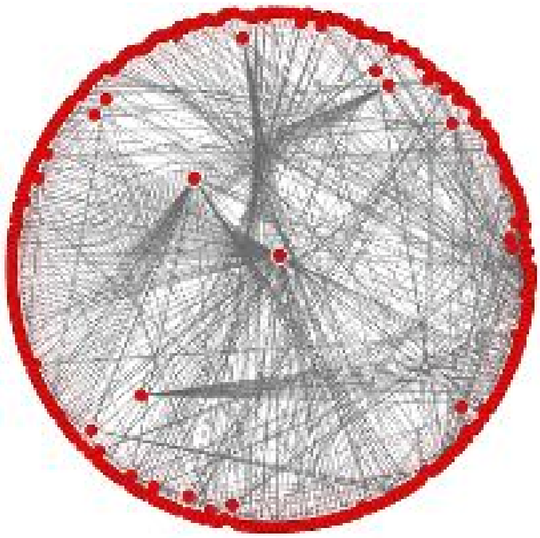


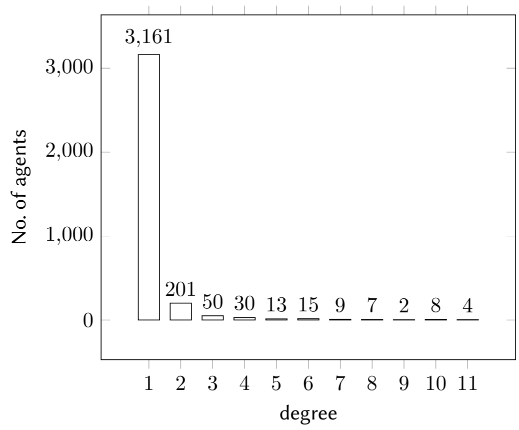
Figure 4: Retweet network arranged by degree centrality

number of other agents, even more than 10. An overall measure of the imbalance in influence is given by the graph centralization measure (see Chapter 5.3.1 of [22]), which takes values in the [0,1] range, with 0 corresponding to a network where all the agents have the same influence, and 1 corresponding to a star network (maximum possible imbalance). The centrality measure we consider is based on the degree (i.e., the number of links incident upon a twitterer in the embeddding retweet network). Mathematically, if g_v is the degree of node v, and v^* is the node exhibiting the highest degree the *graph centralization* is:

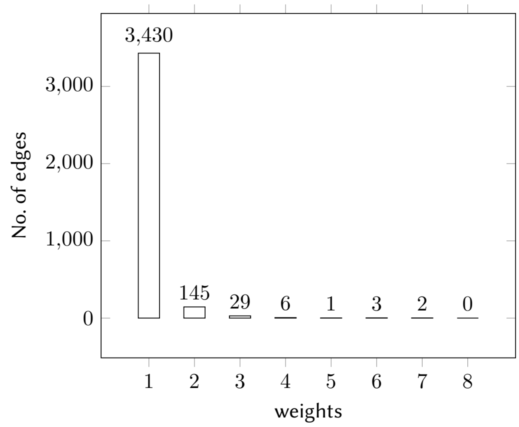
$$G := \frac{1}{(n-1)(n-2)} \sum_{i=1}^{n} [g_{v^*} - g_{v_i}] \,. \tag{4}$$

In our case, the graph centralization measure achieves an intermediate value, namely 0.4275. Though this could appear lower than expected, given the emergence of a few dominant twitterers, those large imbalances that inflate the sum in Equation (4) are probably countered by the large size of the network (hence, large value of n that enlarges the denominator in Equation (4)).

As to RQ3, in the correspondence between the retweeting network and the agent-based model, we have set the influence coefficients proportional to the edge weights. A measure of the level of influence exerted by twitterers on their fellows is therefore the weight: if some edge is associated to a large weight, that relationship bears a heavy influence. If we look at the distribution of edge weights in Figure 5(b), we see that most twitterers (94.8%) exert just a small influence on others (i.e., the weight of their edges is just 1), though there are a small minority that are retweeted more frequently (even 7 times over the week of observation) and therefore exert a heavier influence.



(a) Distribution of nodes by their degree



(b) Distribution of edges by their weight

Figure 5: Influence by agents

6. Conclusions

Our paper deals with a critical issue in the development of agent-based models, i.e., setting the parameters that govern the model and allow to apply such models in the real world (what we call the calibration of the model). Our method can be applied whenever the agents act in a social network. Though we showed an example using Twitter data, any social network allowing an opinion reposting mechanism can be used. Also, the class of agent-based models to which it can be applied is fairly large. The only significant limitation of the current approach is the linear form of the equations that govern the interaction among agents. The possible limitations on the correspondence between the ranges of influence coefficients in the agent-based model on one side and edge weights in the social network on the other side could be addressed easily by suitable operations, e.g., by translation and rescaling to realign the two ranges or possibly by nonlinear transformations.

We therefore envisage this calibration model to fill the gap between the theoretical analysis of an agent-based model and its applicability in a real world context. We wish to address its limitations as to the applicability to nonlinear models and different parameter ranges in our future work.

References

[1] L. Mastroeni, P. Vellucci, M. Naldi, Agent-based models for opinion formation: a bibliographic survey, IEEE Access (2019) 1–12.
[2] P. Vellucci, M. Zanella, Microscopic modeling and analysis of collective decision making: equality bias leads suboptimal solutions, Annali dell'Università di Ferrara (2017).
[3] Y. Shang, An agent based model for opinion dynamics with random confidence threshold, Communications in Nonlinear Science and Numerical Simulation 19 (2014) 3766–3777.
[4] S. Monica, F. Bergenti, An analytic study of opinion dynamics in multi-agent systems, Computers & Mathematics with Applications 73 (2017) 2272 – 2284.
[5] L. Mastroeni, M. Naldi, P. Vellucci, Opinion dynamics in multi-agent systems under proportional updating and any-to-any influence, in: M. Paolucci et al. (eds.), Advances in Optimization and Decision Science for Society, Services and Enterprises, AIRO Springer Series 3, Springer Nature Switzerland., 2019. URL: https://www.springer.com/gp/book/9783030349592#aboutBook.
[6] E. Dacrema, S. Benati, The mechanics of contentious politics: an agent-based modeling approach, The Journal of Mathematical Sociology 44 (2020) 163–198.
[7] L. Mastroeni, M. Naldi, P. Vellucci, Individual competence evolution under equality bias, in: European Modelling Symposium (EMS), Manchester, 2017.
[8] T. Lux, Estimation of an agent-based model of investor sentiment formation in financial markets, Journal of Economic Dynamics and Control 36 (2012) 1284–1302.
[9] L. Mastroeni, M. Naldi, P. Vellucci, An agent-based model on scale-free networks for personal finance decisions, in: Proceedings of the 20th Workshop "from Objects to Agents", 2019.
[10] N. E. Friedkin, E. C. Johnsen, Social influence and opinions, The Journal of Mathematical Sociology 15 (1990) 193–206.
[11] N. E. Friedkin, E. C. Johnsen, Social influence networks and opinion change, Advances in Group Processes 16 (1999) 1–29.
[12] M. Mäs, A. Flache, Differentiation without distancing. Explaining bi-polarization of opinions without negative influence, PLOS ONE 8 (2013) 1–17.
[13] P. Jia, A. MirTabatabaei, N. E. Friedkin, F. Bullo, Opinion dynamics and the evolution of social power in influence networks, SIAM Review 57 (2015) 367–397.
[14] D. Krackhardt, Cognitive social structures, Social Networks 9 (1987) 109 – 134.
[15] K. R. Varshney, Bounded confidence opinion dynamics in a social network of Bayesian decision makers, IEEE Journal of Selected Topics in Signal Processing 8 (2014) 576–585. doi:10.1109/JSTSP.2014.2309945.
[16] L. Li, A. Scaglione, A. Swami, Q. Zhao, Consensus, polarization and clustering of opinions in social networks, IEEE Journal on Selected Areas in Communications 31 (2013) 1072–1083.

[17] A. Madan, M. Cebrian, S. Moturu, K. Farrahi, A. Pentland, Sensing the "health state" of a community, IEEE Pervasive Computing 11 (2012) 36–45.
[18] T. O'Reilly, S. Milstein, The twitter book, O'Reilly Media, Inc., 2011.
[19] S. Kumar, F. Morstatter, H. Liu, Twitter data analytics, Springer, 2014.
[20] E. Bakshy, J. M. Hofman, W. A. Mason, D. J. Watts, Everyone's an influencer: quantifying influence on twitter, in: Proceedings of the fourth ACM international conference on Web search and data mining, ACM, 2011, pp. 65–74.
[21] J. Gentry, Package 'twitteR', CRAN repository., 2016. URL: https://cran.r-project.org/web/packages/twitteR/twitteR.pdf.
[22] S. Wasserman, K. Faust, et al., Social network analysis: Methods and applications, volume 8, Cambridge university press, 1994.

Actor-based architecture for Cloud Services Orchestration: the case of social media data extraction

Stefano Cavalli, Stefano Cagnoni, Gianfranco Lombardo and Agostino Poggi

University of Parma

Abstract
In this paper we present a distributed system for social media scraping which aims to acquire an arbitrarily large number of information from social networks, by exploiting an actor-based solution able to orchestrate efficiently several services on cloud. Our goal is to ensure that correct operations among actors occur, thanks to a master node, based on the ActoDeS architecture, which takes care of managing communications, interface and messages exchanged by client nodes. As a use case, we consider Twitter as social media platform for the key role that is playing in the modern society, as shown by Google Trends data. However, Twitter's search API have many limitations and there is definitely no way to make it work when it comes to obtaining millions of records within a monthly or annual time range. Thus, we have designed a distributed solution that is able to overcome these constraints without breaking the current laws on this subject and the policies of Twitter.

Keywords
Actor-based systems, Cloud computing, Web-scraping, Data Analysis

1. Introduction

Nowadays, web scraping is one of the most widely used techniques to extract any type of data from a web page, thanks to multiple software that automatize this process. Over the past few years, it has been the subject of many controversies but on September 9th 2019, the Appeal from the United States District Court for the Northern District of California officially declared [1] that web scraping is not illegal, as are not methodologies that try to prevent users from working on it, provided that no access is made by the software within the platform itself. Therefore, in the United States, as well as in the rest of the world, laws are changing and more than ever the knowledge represents a truly competitive key for companies and industries.

With over 500 million tweets per day, according to their 2020 Q1 report [2], Twitter has over 166 million daily active users who generate a quantity of information that can certainly be defined as big data. In light of this, the interest in big data generated over social media is increasing and, at the same time, software solutions to perform data extraction are more and

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
stefano.cavalli1@unipr.it (S. Cavalli); stefano.cagnoni@unipr.it (S. Cagnoni); gianfranco.lombardo@unipr.it (G. Lombardo); agostino.poggi@unipr.it (A. Poggi)
0000-0002-3505-0556 (S. Cavalli); 0000-0003-4669-512X (S. Cagnoni); 0000-0003-1808-4487 (G. Lombardo); 0000-0003-3528-0260 (A. Poggi)

Figure 1: Google Trends data shows interest over time in Twitter (blue) and Facebook (red) expressed by people from all over the world, within a time range which goes from July 2013 to July 2020

more demanded. Several research works have also highlighted the importance of collecting contents from the main social media platforms, such as Facebook and Twitter, to perform various kinds of analysis. For example, in [3], data from Facebook are used to analyze a rare disease. In [4], social media data are used to study collaboration in firms and organizations. In [5] and [6], data from Twitter are used to train classifiers able to detect troll users.

Moreover, different sentiment analysis techniques perform really well and obtain successful results when applied to social media [7, 8, 9]. These applications are improving day by day and they are moving to cloud and to distributed systems, to better tackle countless performance issues, software and hardware conflicts, backup and replication problems. Cloud orchestration manages and coordinates all the activities coming from cloud automation, i.e., those processes that run without the need for human interaction. Nowadays, cloud orchestration brings many advantages depending on the application context: cost reduction and scaling, efficiency, improved security and enhanced visibility into resources.

When it comes to distributed computing and actor-model, one of the software frameworks for multi-agent and actor architecture modelling is ActoDeS [10]: developed entirely in Java, it takes advantage of the actor model by delegating the management of events to the execution environment. Moreover, thanks to the use of different implementations of components that manage the execution of the actors, it is suitable for the development of efficient and scalable applications in particular in the data mining domain [11]. In this environment, messages, actors and message patterns are all instances of Java classes. Every message is an object containing a set of fields, a message pattern is an object defined as a combination of constraints on the value of some message field and the actor address acts as both a system-wide unique identifier and a local and remote proxy of an actor.

In this article, we present an actor-based architecture for orchestrating cloud services. In particular, as use case, the system aims at easing the execution of data extraction from Twitter. The following tasks are available: retrieving data for a long period of time, without any limitation; distributing the computing power and computational logic on different machines and making sure that a master process will orchestrate all services.

2. Related Work

Examining the state of the art of web scraping [12], we confirm that this term is always considered in two different contexts: (*i*) data collection, in which the main focus is not a real-time system, therefore speed performance is not an important factor and it does not affect results; (*ii*) work within a real-time environment, where it is important to provide all the information required and where a millisecond error could be very critical.

Nowadays, web scraping is widely used. Some of its applications include: job search [13], weather data [14], advertisement [15], journalism [16], and health sector [17]. Financial trading [18] is also important and requires real-time systems to be very performing. News, along with political-economic decisions, may drastically influence a price stock within a very short time range, where web scraping automation plays a fundamental role. In this case, web scraping is used to design and implement automated bots that act as trading decision support systems [19].

In 2017, Trifa *et al.* [20] described a Personalized Multi-Agent Systems (PMAS) on a distributed and parallel architecture in order to predict users' topics of interest using data coming from web scraping tools, focusing on Twitter, Facebook and LinkedIn. As previously mentioned, we have chosen to evaluate Twitter as social network because it is still increasing its user base against Facebook, which instead has set an interest decrease for seven consecutive years, based on Google trends data [21] (Figure 1) which is frequently used in data analysis and statistical topics.

Building a cloud system, in order to analyze an unlimited amount of data, avoids running into several problems and leads to multiple advantages. Integrity, which refers to data completeness, accuracy and consistency, must be kept safe; a cloud system always helps to provide data integrity [22]. Along with it, connectivity, speed, scalability and store availability are also features of cloud or cloud-hybrid systems. Working with services on a cloud-based environment can be risky and drive into several problems [23]: large computational load can be required and nodes could take too long for a decision-making process; recovery from mistakes that can arise with a centralized decision making process would not be managed by the system. By using an actor model, all these problems can be solved. This is due to the fact that each actor is able to make decisions in this cloud-based environment, where only partial information may be available. If one or more nodes fail, a readjustment process will take place in order to prevent any related problem. Dealing with actor-model also brings some advantages: the use of simple and high-level abstractions for distribution, concurrency and parallelism; an asynchronous, non-blocking and highly performing message-driven programming model can be easily designed; take advantage of a very lightweight event-driven processes.

Cloud orchestration and automation are well-known terms in tech industries. Based on their context, researchers have different opinions about them, but when web services come up, orchestration has been extensively discussed [24] along with concurrency programming. Concurrency in today's middleware is almost always thread-based and hardware still evolves towards more parallel architectures. Threads are the most used way to execute code concurrently and they should manage possible conflicts of their execution, avoiding synchronization that seems to help on correct data ordering and consistency. However, this paradigm raises several problems when it comes to deadlocks, maintaining data consistency and liveness. Most of the problems with concurrency, such as deadlocks and data corruption, result from having shared state. That is where the actors come into play: the actor model, where independent processes exchange immutable messages, is an interesting alternative to the shared state model.

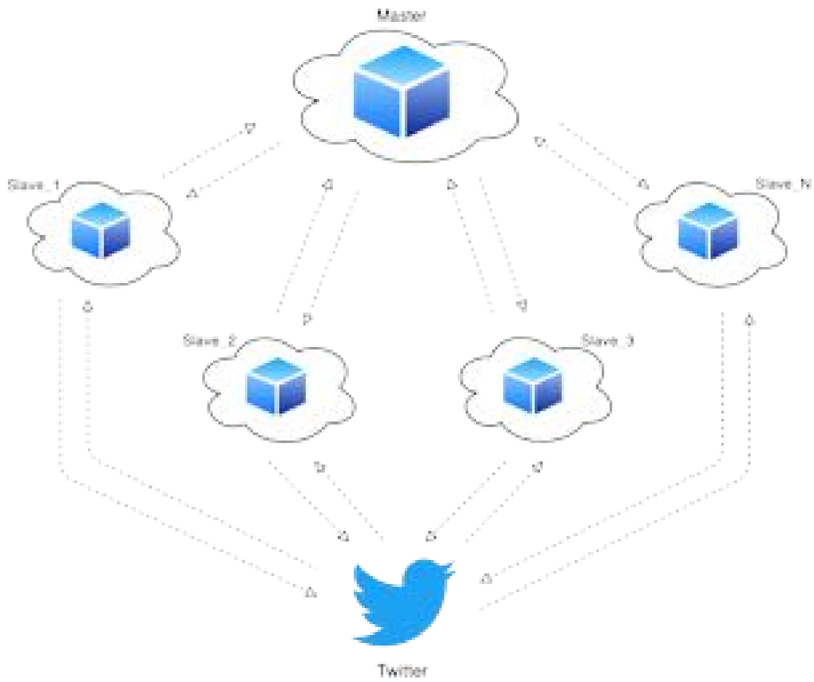


Figure 2: Actor model cloud-based architecture: each node represents an actor enclosed in its cloud environment; it can communicate exchanging messages with the master node, retrieving data coming from Twitter. *N* stands for the number of actors the system is designed for.

3. System Architecture

In this section, we describe the cloud system architecture and its components, as shown in Figure 2. The solution is composed by multiple actors that play different roles depending on their behavior and act with different logics. The overall system combines several existing technologies, which have strongly evolved during the last decades such as server, databases, web services, actor model, networks and cloud. The main purpose of our work is to obtain any number of tweets, specifying a time interval together with a language tag (i.e. "EN" for English) and a keyword which is part of the tweets themselves.

3.1. Multi-Actor model

As previously mentioned, a multi-actor model represents the optimal architectural choice, to describe the distributed cloud system we have worked on. Using a master-slave connection, we have built different nodes: the master node is designed using ActoDeS and its events management, while slave nodes are Python services running concurrently. ActoDeS allows actors to work on indipendent but communicating actor spaces. In order to make this communication available ActoDeS implements a Dispatcher which takes care of messages forwarding between different actor spaces. First of all, rather than explicit sleeping or waking up, actors react to events that come from interactions with other actors (i.e. receiving a message). If they worked

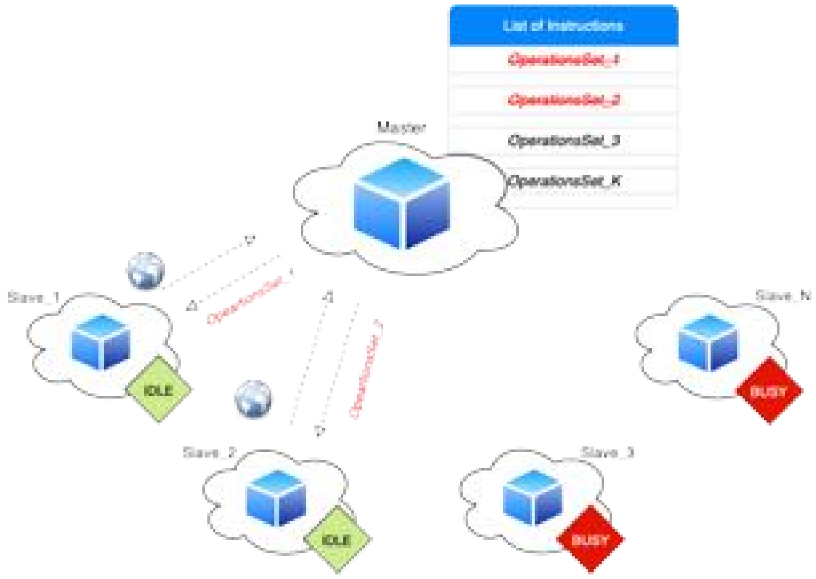


Figure 3: When clients are in IDLE state, they send a message to the master trough the WebSocket and it replies with a set of operations coming from the List of Instruction, it than update the list removing this specific item. In this example, two clients are busy for some other reasons and they are not communicating with the server.

out their previous tasks and nobody send them new messages, these actors get passive and wait for another message to receive. In this architecture, actors communicate asynchronously with each other by sending immutable messages following a specific pattern which is later discussed. A FIFO (First-In-First-Out) order is respected when an actor receive multiple messages from others. Receiving messages is always a blocking operation from the actor point of view, however, sending a new message must be non-blocking. In particular, when a slave wants to send data to the master, it keeps going on working as a web scraping service: in this case, a non-blocking operation cannot be performed at all.

The master node acts as a system orchestrator (Figure 3): firstly, it sets all the communication channels between itself and all the other nodes, obviously located on separate servers. According to the user's requests, a list of instructions is made by the master node, which keeps it up-to-date as needed during the execution time. The list of instructions contains a set of operations that must be executed by the clients. At a certain point, while it is not working, a client can get one of these instructions by establishing a WebSocket connection with the master and sending a message to it. A WebSocket is a communication protocol, located at layer seven in the OSI model and standardized by IETF as RFC 6455 in 2011, that provides full-duplex communications channels over a single TCP connection. The master, which instantly reacts to this event and wakes up, replies with a random message taken from the list of instructions and updates the list, removing the instruction itself. While multiple clients may simultaneously connect to the master, a FIFO (First-In-First-Out) strategy is applied in order to maintain consistency and optimize the workload on all clients, avoiding that some of them work more than others or stay inactive for too long, causing inefficiency. Once the client has done with its set of operations (a

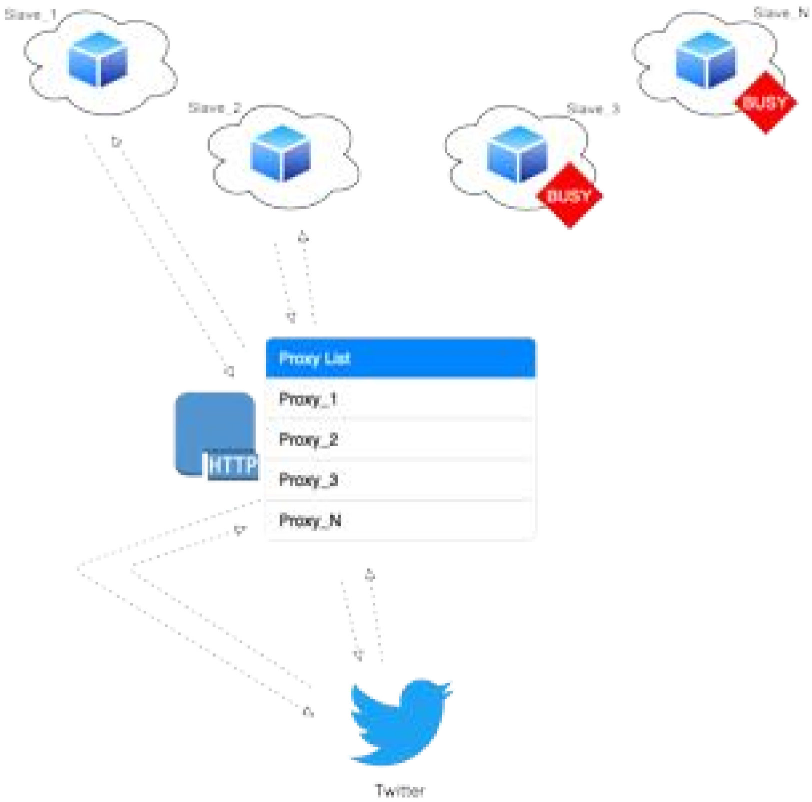


Figure 4: Once server sends a message to the clients, they send an http request to Twitter using a proxy list, in order to retrieve tweet's information. In this example, *Slave_3* and *Slave_N* are busy and they are not communicating with Twitter.

process which is extensively discussed below), it sends another message to the master using the Remote Copy Protocol (RCP) to transfer files and Secure Shell (SSH) to provide authentication and encryption. These files are all stored in the master's server and they will be removed from client's one as soon as the sending process is successfully completed. When the list of instructions is empty, it means there is no need to keep the client alive again and the server interrupts its WebSocket.

3.2. The Twitter case study

In this section we describe the structure of the instruction list, which we have introduced previously, and how the clients behave with Twitter 4. The so-called instruction list is a set of operations that clients execute, regardless of the others' behavior. In particular, each of them receives a time range together with a language and a keyword and will take care of downloading all the tweets that fall within those parameters listed in Table 1.

Each client operates asynchronously and executes instructions received from the server sequentially. By the time a client receives a server's message, containing a set of operations, it starts its job making an HTTP request, thanks to the Python3 Requests library [25], in order to

Table 1
Search parameters *Lan, Interval, Kw*

Parameter	Description
Lan	The language of a tweets..
Interval	A specific interval that includes a list of tweets.
Kw	A word, or sentence, that must be part of the tweets.

Table 2
Search parameters *Lan, Interval, Kw*

Parameter	Description
Username	Username of the tweet's creator.
FullName	Full name of the tweet's creator.
User_id	User id of the tweet's creator.
Tweet_id	Tweet's id.
Tweet_url	Tweet's url.
Timestamp	Tweet's timestamp.
Replies	Number of tweet's replies.
Retweets	Number of tweet's retweets.
Likes	Number of tweet's likes.
Text	Plain text of the tweet.
Html	Html extracted from the tweet.

Table 3
Effective parameters *Lan, Interval, Kw* used while testing the entire system.

Parameter	Value
Lan	**English**
Interval	**2009-01-01 / 2019-01-01**
Kw	**Bitcoin**

retrieve tweets from Twitter. We use a free proxy list [26] available from the web, which is a list of open HTTP/HTTPS/SOCKS proxy servers that allow clients to make indirect network connections to Twitter. This approach prevents from being banned by Twitter, once too many http requests are made.

Once an http request is made, the Python Beautifulsoup4 library is used in order to parse the content, retrieving all possible information a tweet may hold (Table 2).

4. Experimental Results

We have implemented the previously described system and a set of experiments has been performed. Using twenty-two different clients and one server, while working on a distributed and cloud environment, we have chosen the parameters described in Table 3.

The first problem we faced on is that when dealing with such systems, we will not have the data size, nor the execution time, until the execution itself is finished. Nobody has any idea about how long it could takes to retrieve all those tweets, since we do not know how many tweets may have been posted for a specific term in a given time interval. Choosing the parameters mentioned above, we have realized that the execution time took 8 hours and 23 minutes (in a distributed system, it is the maximum execution time of the node that takes the longest time to complete its work), and 53,353,764 tweets have been collected in multiple files with comma-separated values (CSV) format, for a total physical size of 73 GB.

While working in a distributed cloud system the workload is shared among the nodes, choosing a sequential approach within a non-distributed actor-model it requires a longer execution time. The process includes one server which involves just one client. The client receives all the information coming from the server and works on each request by processing the data received from Twitter. However, this architecture shows no advantages at all, delaying the execution time up to 193 hours and 52 minutes using the same parameters we already focused on in the actor-model system.

5. Conclusion

The proposed actor-based approach shows how it is possible to obtain a large amount of data coming from Twitter, using a distributed cloud-based system. The API limits imposed by Twitter are incresingly difficult to avoid and restrict the scientific research progress in many field, such as the Natural Language Processing (NLP). The actor-model we choose optimizes every single action in order to guarantees a well distributed workload between all nodes, drastically reducing the possibility that an IP address may be banned by Twitter using a list of free proxies available on the web.

References

[1] Appeal from the United States District Court for the Northern District of California, https://parsers.me/appeal-from-the-united-states-district-court-for-the-northern-district-of-california, last accessed October 2020.

[2] Twitter Q1 2020 Financial Report, https://s22.q4cdn.com/826641620/files/doc_financials/2020/q1/Q1-2020-Earnings-Press-Release.pdf, last accessed October 2020.

[3] G. Lombardo, P. Fornacciari, M. Mordonini, L. Sani, M. Tomaiuolo, A combined approach for the analysis of support groups on facebook-the case of patients of hidradenitis suppurativa, Multimedia Tools and Applications 78 (2019) 3321–3339.

[4] E. Franchi, A. Poggi, M. Tomaiuolo, Social media for online collaboration in firms and organizations, International Journal of Information System Modeling and Design (IJISMD) 7 (2016) 18–31.

[5] P. Fornacciari, M. Mordonini, A. Poggi, L. Sani, M. Tomaiuolo, A holistic system for troll detection on twitter, Computers in Human Behavior 89 (2018) 258–268.

[6] M. Tomaiuolo, G. Lombardo, M. Mordonini, S. Cagnoni, A. Poggi, A survey on troll detection, Future Internet 12 (2020) 31.

[7] G. Lombardo, A. Ferrari, P. Fornacciari, M. Mordonini, L. Sani, M. Tomaiuolo, Dynamics of emotions and relations in a facebook group of patients with hidradenitis suppurativa, in: International Conference on Smart Objects and Technologies for Social Good, Springer, 2017, pp. 269–278.

[8] G. Angiani, S. Cagnoni, N. Chuzhikova, P. Fornacciari, M. Mordonini, M. Tomaiuolo, Flat and hierarchical classifiers for detecting emotion in tweets, in: Conference of the Italian Association for Artificial Intelligence, Springer, 2016, pp. 51–64.

[9] P. Fornacciari, M. Mordonini, Social network and sentiment analysis on twitter: Towards a combined approach., in: KDWeb, 2015.

[10] F. Bergenti, A. Poggi, M. Tomaiuolo, An actor based software framework for scalable applications, in: International Conference on Internet and Distributed Computing Systems, Springer, 2014, pp. 26–35.

[11] G. Lombardo, P. Fornacciari, M. Mordonini, M. Tomaiuolo, A. Poggi, A multi-agent architecture for data analysis, Future Internet 11 (2019) 49.

[12] R. Diouf, E. N. Sarr, O. Sall, B. Birregah, M. Bousso, S. N. Mbaye, Web scraping: State-of-the-art and areas of application, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 6040–6042.

[13] A. Phaphuangwittayakul, S. Saranwong, S. Panyakaew, P. Inkeaw, J. Chaijaruwanich, Analysis of skill demand in thai labor market from online jobs recruitments websites, in: 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2018, pp. 1–5.

[14] Fatmasari, Y. N. Kunang, S. D. Purnamasari, Web scraping techniques to collect weather data in south sumatera, in: 2018 International Conference on Electrical Engineering and Computer Science (ICECOS), 2018, pp. 385–390.

[15] A. Maududie, W. E. Y. Retnani, M. A. Rohim, An approach of web scraping on news website based on regular expression, in: 2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT), 2018, pp. 203–207.

[16] E. N. SARR, O. SALL, A. DIALLO, Factextract: Automatic collection and aggregation of articles and journalistic factual claims from online newspaper, in: 2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS), 2018, pp. 336–341.

[17] A. Amalia, R. M. Afifa, H. Herriyance, Resource description framework generation for tropical disease using web scraping, in: 2018 IEEE International Conference on Communication, Networks and Satellite (Comnetsat), 2018, pp. 44–48.

[18] B. B. P. Maurya, A. Ray, A. Upadhyay, B. Gour, A. U. Khan, Recursive stock price prediction with machine learning and web scrapping for specified time period, in: 2019 Sixteenth International Conference on Wireless and Optical Communication Networks (WOCN), 2019, pp. 1–3.

[19] K. Salah-ddine, K. Abouloula, E. Brahim, Money management limits to trade by robot trader for automatic trading, International Journal of Engineering 7 (2018).

[20] A. Trifa, A. H. Sbaï, W. L. Chaari, Evaluate a personalized multi agent system through social networks: Web scraping, in: 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2017, pp. 18–20.

[21] R. Kruthika, P. Balasubramanian, V. Sureshkumar, Relationship between google trends

data and index returns, in: 2018 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC), 2018, pp. 042–045.
[22] A. M. Talib, R. Atan, R. Abdullah, M. Azrifah, Cloudzone: Towards an integrity layer of cloud data storage based on multi agent system architecture, in: 2011 IEEE Conference on Open Systems, 2011, pp. 127–132.
[23] F. De la Prieta, S. Rodríguez, J. Bajo, J. M. Corchado, A multiagent system for resource distribution into a cloud computing environment, in: Y. Demazeau, T. Ishida, J. M. Corchado, J. Bajo (Eds.), Advances on Practical Applications of Agents and Multi-Agent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 37–48.
[24] X. Kang, C. Zhou, X. Liu, H. Sun, Y. Huang, Improving performance for decentralized execution of composite web services, in: 2013 IEEE Ninth World Congress on Services, IEEE Computer Society, Los Alamitos, CA, USA, 2010, pp. 582–589.
[25] Python Requests Library, https://requests.readthedocs.io/, last accessed October 2020.
[26] Free Proxy List, https://free-proxy-list.net/, last accessed October 2020.

An Early Warning System for Seismic Events based on the Multi-Agent Model

Roberto Spina[a,b], Andrea Fornaia[a] and Emiliano Tramontana[a]

[a]Department of Mathematics and Computer Science, University of Catania, Italy
[b]National Council of Geologists (CNG), Rome, Italy

Abstract

When a disastrous earthquake is about to occur in a specific territory, there are a series of anomalies that alter the pre-existing natural balances. Seismic swarms, ground deformation, bright flashes, emissions of various gas types (radon, CO_2,..), changes in the composition and flow rate groundwater are just some physical-chemical perturbations induced by the growing stress condition borne by the crustal masses. Dilatancy theory and asperity model allow us to interpret the dynamic mechanisms to which the seismic precursors are due: the development of a network of cracks and the sliding of areas with less mechanical resistance are in agreement with seismic, mechanical and geochemical anomalies that occur before to high magnitude earthquakes. In areas with high seismic risk, constant monitoring of geophysical parameters is frequent, carried out using different types of sensors.

The MAS (Multi-Agent System) model is one of the most suitable choices for efficiently implementing a seismic alert system, based on the interpretation of experimental data obtained from the sensor network. Using this type of approach, a Seismic Early Warning (SEW) has been created that according to the data acquired by the sensors and through the activities carried out by agent clusters, define the risk of seismic events having magnitude at least six. The SEW system aims to interpret, in real-time, the variations of an adequate number of seismic precursors for specific threshold values, calculated statistically. The integrated and complementary analysis of them, using several specific Boolean expressions, assesses the contribution provided by each parameter for computing the level of risk, divided into soft, medium and hard. The model has been tested with data gathered in New Zealand, a nation with a high seismic and volcanic risk which offers free access to some seismic precursors.

Keywords

MAOP, MAS, multi-agent system, seismic precursors, earthquakes, geophysical parameters

1. Introduction

Software applications based on MAOP (Multi-Agent Oriented Programming) are widely used in various fields and have taken on an increasingly important role thanks to the use of artificial intelligence (AI) techniques [1, 2, 3]. The adoption of centralized methods presents intrinsic difficulties due to the growing complexity of the systems, the dimensions of which continue to increase: in this context, the architectural solutions proposed by the MAS (Multi-Agent System) offer different advantages and a good solution for the modelling of complex distributed

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
roberto.spina@phd.unict.it (R. Spina); fornaia@dmi.unict.it (A. Fornaia); tramontana@dmi.unict.it (E. Tramontana)
0000-0001-7393-7249 (R. Spina); 0000-0001-6034-855X (A. Fornaia); 0000-0002-7169-659X (E. Tramontana)

systems [3, 4, 5]. One of the fundamental characteristics of the MAS paradigm is the interaction between agents, independent and autonomous software modules that perform specific propaedeutics activities for the development of one or more system functions. The innumerable properties that distinguish the agents (communication, persistence, reactivity, proactivity, etc.) make them potentially suitable for monitoring natural phenomena, especially those capable of producing catastrophic events.

The different activities required by a seismic early warning system, summarized in the acquisition, interpretation and formatting of geophysical data with the addition of real-time user assistance, make the MAS model one of the most appropriate systems for real-time monitoring of precursor parameters (seismic swarms, ground deformation, soil temperature, radon gas emission, etc.) aimed at predicting earthquakes potentially destructive.

Concerning the previous observations, a software system called SEW (Seismic Early Warning) has been developed, based on a dashboard that, in real-time, shows updates based on alert level (soft, medium and hard) that characterizes a specific area at high seismic risk. The system of agents, operating in the background, allows comparing the experimental data of seismic precursors with the corresponding threshold values, obtained statistically from seismic swarms which, in the past, have produced a destructive earthquake. Through simple Boolean expressions, it will then be possible to automatically establish a certain alarm level in the places surrounding the seismic source.

The remainder of this document is organized as follows. A literature survey is discussed in Sections II. Section III presents the seismic domain of the application and the methodologies used. Section IV defines the characteristics of the MAS and the collaborative interaction between agents. Section V illustrates the case study of New Zealand, a region with high seismic and volcanic risk in which a SEW prototype has been implemented and tested and the results of which are discussed in section VI. And in conclusion, the MAS approach advantages and all the new features presented were analyzed in Section VII.

2. Related Work

Many applications that combine artificial intelligence and MAS technologies, in various sectors, have been developed: some examples are represented by the new opportunities created respectively for traffic control at intersections[1], for monitoring and improving the cloud performance and security [2], or for alerting people about crowded destinations [6]. In the last years, a group of scientists presented a Multi-Agent System paradigm and discuss how it can be used to design intelligent and distributed systems [3]. Next, a decentralized approach of MAS has been developed using a distributed simulation kernel to solve partitioning, load balancing and interest management problems in an integrated, transparent and adaptive manner [4].

Different works have been produced on the implementation of multi-agent systems relating to coordination and rescue in the stages following the occurrence of a high-intensity seismic event. A multi-agent system for the evacuation of people in immediate post-emergency situations has been implemented for the city of Iaşi (Romania) [7]. A series of simulators using a MAS architecture, following the damage caused by the 1999 earthquakes in Turkey and Pakistan in 2005, have been developed: damage, victims and other auxiliary simulators [8]. A Disaster

Management System (DMS) developed with the multi-agent model has been proposed to adequately manage a multi-risk situation consisting of two or more disasters occur at the same time, such as, for example, the combination of earthquake and tsunami [9].

Other systems for the management of the pre-post seismic phases have been developed by the authors in various ways: (i) through a Seismic and Volcanic Early Warning System in the Etna area based on specific threshold values for each geophysical precursor [10]; (ii) with an approach based on the coupling of multi-agent systems and intelligent systems (cellular automata) for simulation on rescue in the event of an earthquake disaster [11]; (iii) through simulations of various post-seismic evacuation scenarios for people using a multi-agent system [12]; (iv) integrating GIS with multi-agent seismic disaster simulations to investigate factors significantly affecting rescue efforts, and to clarify countermeasures for saving lives [13].

3. Approach

This section describes the model on which SEW is based and the methodologies used to implement the proposed system. The first activity concerns the extraction of useful information from the databases of seismic precursors (seismic swarms, soil deformations, ...) and their visualization in a SPA (Single Page Application), based on Angular. The next phase is the implementation of a multi-agent system defining the alert level of the seismic territory based on the result obtained from a set of Boolean expressions founded on the seismic precursors.

3.1. Methodology

Within each seismic zone, there are one or more seismogenic structures (faults) which, with their displacement, can produce the vibrations that generate the earthquake. Sequences of seismic events that, in some cases, may prelude to a major magnitude earthquake (mainshock) are called seismic swarms. Every single event (foreshock) belonging to the sequence often occurs a short time from the previous one.

Suppose we consider all the seismic swarms that in the past have given rise to mainshocks of medium-high magnitude (above 6) which, about to the characteristics of the territory concerned, can produce serious damage to people and things. We denote with S_1, S_2, S_3 three classes of seismic swarms which as a final result gave an earthquake of magnitude $\mathrm{M}_w \geq 6$ and with $(P_1, \bar{S}_1)$, $(P_2, \bar{S}_2)$, $(P_3, \bar{S}_3)$, the ordered pairs where P corresponds to the number of S elements and $\bar{S}$ to the arithmetic mean for each class S_1, S_2, S_3. The average of the averages for the three classes of seismic swarms that prelude to an earthquake of Magnitude M_w will be given by:

$$SS_{T_h} = \frac{1}{N} \sum_{i=1}^{3} P_i \bar{S}_i \tag{1}$$

$$N = P_1 + P_2 + P_3. \tag{2}$$

The SS_{T_h} value obtained corresponds to the most probable value for seismic swarms with $\bar{M}_w < 6$ for that specific seismogenic structure, and therefore a threshold value for the mainshock of magnitude $\mathrm{M}_w \geq 6$.

The same procedure is performed for geophysical precursors for which a consistent database of measurements is available. Consequently, three further threshold values (RC_{T_h}, GD_{T_h} and ST_{T_h}) will be obtained referring respectively to *Radon Concentration (RC)*, *Ground Deformation (GD)* and *Soil Temperature (ST)* for earthquakes with $\mathrm{M}_w \geq 6$. The four previously threshold values (T_h) will be associated with the respective standard deviations (σ) expressed by:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{3}(\bar{S}_i - T_h)^2}{3}} \tag{3}$$

and the achievement of the threshold value (T_h) will occur when:

$$[\bar{M_w} \pm \sigma_w] \cap [SS_{Th} \pm \sigma_{th}] \neq \emptyset \tag{4}$$

where $\bar{M}_w$ represents the average magnitude of the current seismic swarm and σ_w the standard deviation associated with it. In real conditions in presence of an extensive seismic swarm (SS), the system will calculate the average magnitude value ($\bar{SS}$) of the seismic sequence in progress and the other three seismic precursors ($\bar{RC}$, $\bar{GD}$ and $\bar{ST}$). In the next step, it will compare the mean value of the four precursors with the respective threshold values.

The definition of the alarm level will be based on the evaluation of a series of Boolean expressions and conditional instructions, arranged in sequence, which allows defining the type of alarm based on the number of precursor parameters that have reached or exceeded the threshold value, going from *hard* (all variables are true) to *soft* (only two variables have exceeded the threshold value). Among the four Boolean variables, *SS* (*Seismic Swarm*) has a fundamental role in defining any alarm level:

if ($SS \wedge RC \wedge GD \wedge ST$) (5)
    return "hard";
if ($SS \wedge ((RC \wedge GD) \vee (RC \wedge ST) \vee (GD \wedge ST))$)
    return "medium";
if ($SS \wedge (RC \vee GD \vee ST)$)
    return "soft".

3.2. Theoretical Background of the SEW

Forecasting of high-magnitude seismic events has as its foundation some theories that, since the last century, have been proposed by various authors to explain the phenomena that determine earthquakes.

Dilatancy theory [14] foresees that before an earthquake the seismogenic area is subject to an increase in stress with an expansion of the crustal volume due to a substantial cracking of the rocks. Consequently, the rocks undergo a variation of their physical characteristics and from the external regions, the fluids are attracted by this extensive fracturing phenomenon. Both the gases and the liquids circulating within the crustal volume change their paths and upon contact with different rocks and/or fluids change their geochemical composition. The interpretation of the phenomena that prelude and follow an earthquake is the basis of what is proposed by Aki (1979) and Kanamori (1981) called respectively *barrier model* and *asperity model.*

In the *barrier model* [15, 16] it is assumed that, before the earthquake, the stress on the fault is uniform. The earthquake is produced by the sliding of the weakest area, while the most resistant area (*barrier*) is opposed to dislocation. In this way, there is an increase in barrier stress. Consequently, after the earthquake, the barrier may be affected by seismic (aftershock) or aseismic sliding episodes.

The *asperity model* [15, 16] considers that the sliding that generates the earthquake concerns the most resistant area, i.e. the asperity. Before the mainshock, the stress on the fault is not uniform because aseismic sliding and preliminary shocks (*foreshock*) have reduced stress in the weakest areas of the fault, concentrating it on asperities. When stress reaches a critical value, the asperity yields giving rise to the earthquake.

The three models agree with the physico-chemical anomalies related to the extensive fracture affecting the seismogenic area (seismic precursors) and with the long sequences of earthquakes preceding (foreshock) and subsequent (aftershock) at the mainshock, providing a valid interpretative key.

4. Framework

The SEW dashboard consists of a series of software components (agents) capable of performing simple tasks, but unable to perform a complex task individually. Recall that each task can be decomposed into simpler parts, to be performed by individual agents or groups of them who cooperate. By planning their interaction through a multi-agent architecture, based on collaboration and the exchange of information, it is possible to achieve the common goal.

A more detailed analysis of the individual responsibilities attributed to each agent and the proposed architectures are described in the following sections.

4.1. Hardware and software architecture

A fundamental prerequisite for the implementation of SEW is the presence, in the seismic territory, of a capillary network of seismometers and GPS sensors, recently replaced by GNSS (*Global Navigation Satellite System*). By this acronym we mean a constellation of satellites that, by sending a signal from space, allow specific receivers to determine their geographical coordinates (longitude, latitude and altitude) on any point on the earth's surface: any ground deformation, before, simultaneously or after a seismic event, will be highlighted by deviations from the original positions.

The test of the system, carried out in the Experiments section, was based on the available datasets, i.e. *Seismic Swarms* and *Ground Deformation*. Additional precursor parameters, in seismic areas where they are available, could significantly improve the results obtainable from SEW: *concentration of Radon*, CO_2, *Arsenic* and *Iron*, *soil temperature* are some of the many precursors that give significant anomalies before a destructive earthquake. The sensors network, arranged optimally for the seismogenic structures, must guarantee monitoring of the precursor parameters with measurements carried out continuously through a Repeater-Gateway transmission system, as in the case of *ground deformation*, *earthquakes* and *soil temperature*. For other precursors (*Radon*, *Iron*, CO_2 and *Arsenic*) the data acquisition can take place directly with on-site sampling.

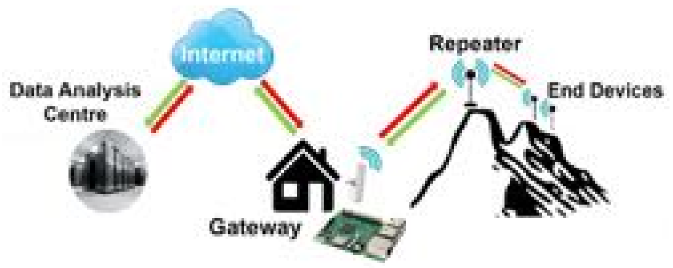


Figure 1: Repeater-Gateway transmission system [17]

Figure 1 shows the acquisition-transmission scheme of the wireless network, consisting of three main components: the gateway, the repeater, and the end devices [17]. GNSS receivers, seismometers and geochemical sensors acquire the experimental data and send them to a repeater which amplifies the signal strength to be transmitted to the gateway, equipped with an internet connection, which routes them to the respective servers of the data processing center. And from this moment on, software agents come into play, carrying out a series of sub-activities to achieve the final goal corresponding to the definition of the current alert level. The main features of the Multi-Agent System is based on some assumptions: (i) no agent can solve a problem on his own but must make use of the collaboration of the others to achieve the intended purpose; (ii) each agent differs from the others in the properties that distinguish it and the tasks it can perform; (iii) agents are divided and associated in a congregation, i.e. groupings of them that perform a series of semantically similar tasks.

With reference to the third point, we can consider that each group of agents acts in parallel and independently from the others, even if they share the same final objective. E.g., the cluster of agents SS (Seismic Swarm) acts in parallel with the clusters GD (Ground Deformation), RC (Radon Concentration) and ST (Soil Temperature): each group carries out similar activities to determine if there is an overlap between your current experimental data range and that of the corresponding threshold value, expressed by the relation (4). The interpretation of the data obtained from the n agent clusters and the definition of the alert level is the exclusive relevance of agent A. To verify that no malfunctions have occurred, a group of three demon agents (X_1, X_2 and X_3), periodically and alternately, checks whether the state of A is consistent, by sending it a message to which a response must follow. In case of no-confirmation, the role of the main agent will be assigned to one of the two "A substitutes" (A_1 or A_2) who will assume the same functions performed by A.

The detail of the interactions between the agents relating to different clusters is described in the following section.

4.2. Collaborative interaction between agents

The description of the interactions in the SEW system is based on the assumptions the MAS implementation concerns earthquakes with a magnitude greater than six and each agent is characterized by its internal state, that is, by variables and data structures which, at a given instant, contain specific values. Agents are server-side back-end components queried by the

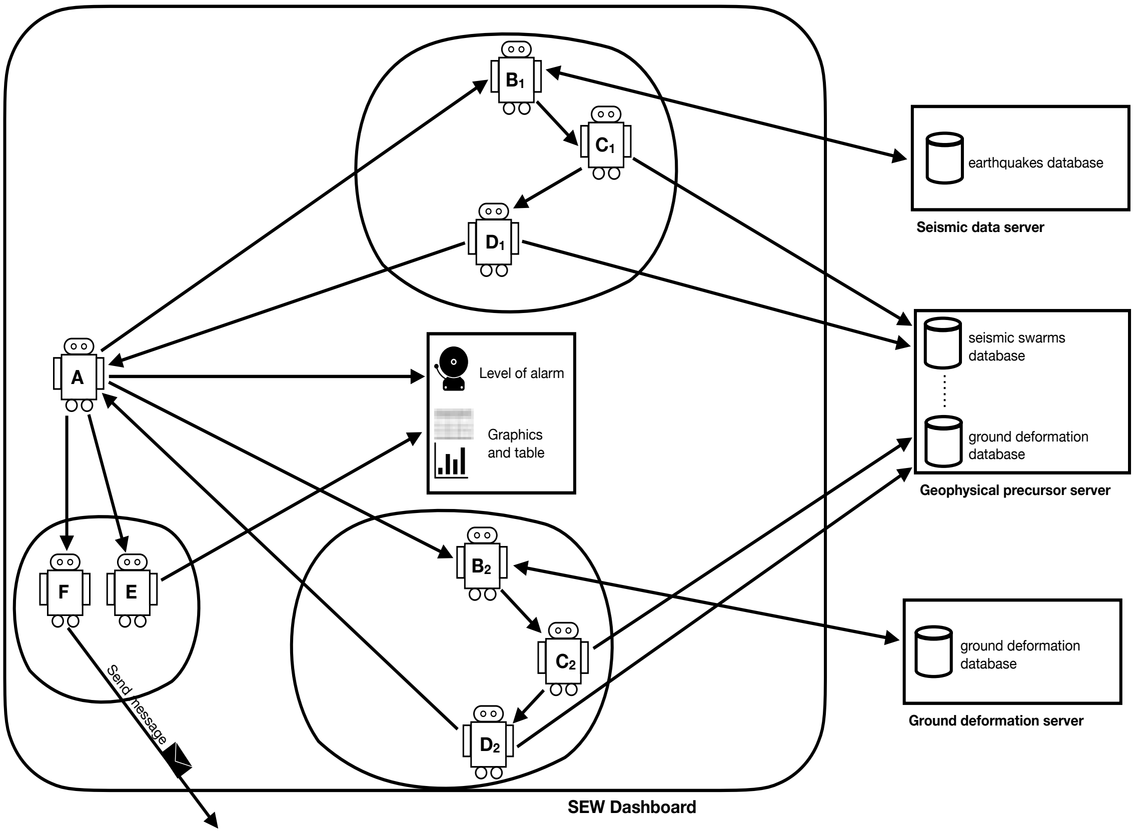


Figure 2: Agent interactions scheme

front-end. The system, still under development, uses Java Agent Development Framework (JADE), a network-oriented framework that guarantees very efficient communication. The example shown refers to the geophysical parameter "Seismic Swarm", but the actions and operations carried out can also be considered substantially equivalent for the other geophysical parameters.

Collaboration and exchange of information can be summarized with the following activities, distributed over a series of agents: (i) download of experimental data from the corresponding servers where they have been stored by the sensor network; (ii) filtering according to certain rules that establish whether they are suitable for registration or not; (iii) analysis of current data and comparison with statistically calculated threshold values; (iv) establish if the alarm level must be updated defining its criticality; (v) formatting and display of data to be presented to the user; (vi) notification of a warning to a select group of scientists on any existing critical problems.

Figure 2 highlights the different roles assumed by agents B_1, C_1, D_1, belonging to the same group of agents, while A belongs to a hierarchically higher level. Every 10 minutes agent A sends a notification to agent B_1 that queries the internal server for the latest updates on seismic events occurred in that source area. In case of a positive response, it sends a message to agent

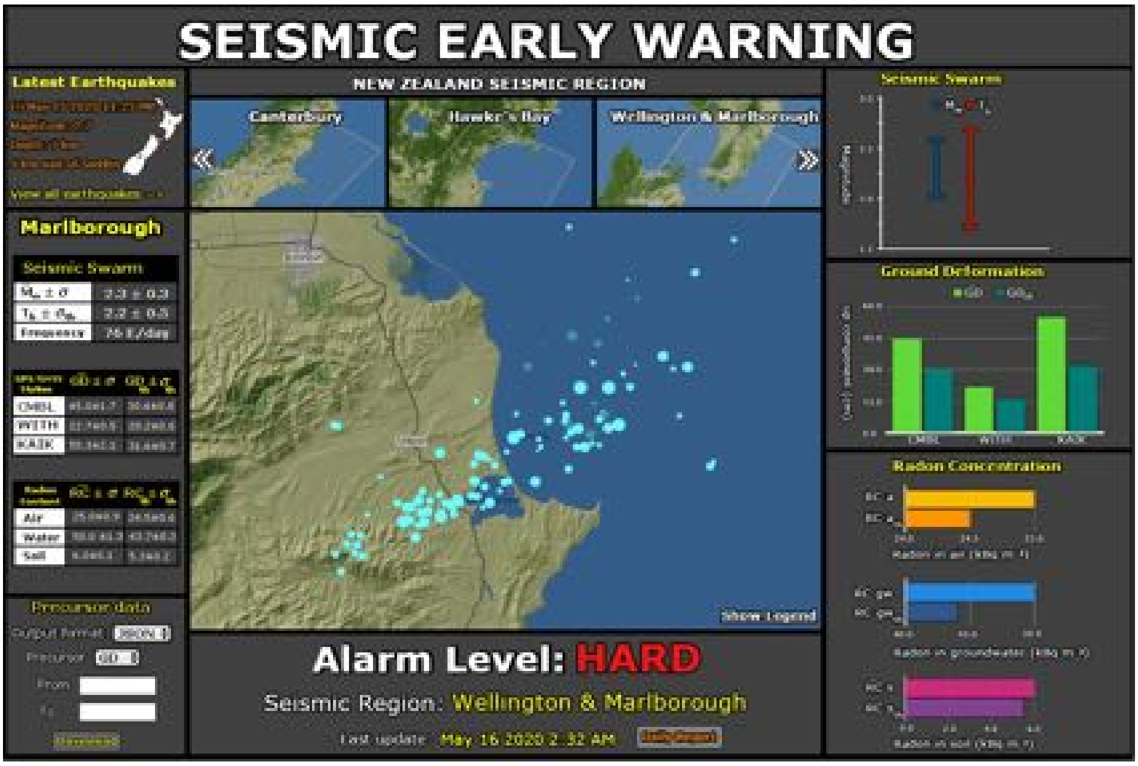


Figure 3: A mockup of the SEW dashboard

C_1 which includes the magnitude (M_w), the hypocenter (H_p) and the date/hour (D) in which the seismic event occurred. Received the message, the agent C_1 compares the data received with those of the previous earthquake, stored in its internal state: the earthquake will be entered in the seismic swarm database only if it has $M_w \geq 1$ and occurred within 24 hours from the previous one, otherwise it is discarded. If the earthquake is inserted in the current seismic sequence, C_1 sends a notification signal to agent D_1 which activates and checks the earthquake frequency (F_E) in its own state in the last seven days, with the specifications defined previously (hypocenter, magnitude). If the frequency is sufficiently high (e.g. $F_E \geq 5$ earthquakes/day), D_1 calculates the average magnitude and the associated σ for the current seismic sequence and compares it with the corresponding threshold value. In case it reaches or exceeds the threshold value, D_1 updates the value of $\bar{M}_w$ in the database and sends a message to agent A, whose evaluation will take into account the frequency of the seismic swarm in the last days.

Next, based on the result obtained from the Boolean expression (5), it will decide whether to activate an alarming level and of which type (soft, medium or hard), sending a notification to the E and F agents, "specialized" in user assistance. In particular, the E agent will update the table and the respective graphs (histograms, box-and-whisker diagrams, ...), while the F agent will send, via e-mail, a report to a small group of scientists. The document, created in an automated way, will report the experimental data that determined the activation of the specific alert level. At the end of the activity cycle, the clusters of agents listen for new notifications that can re-trigger the sequence of activities listed above. A mockup of the SEW interface, currently under development, is shown in Figure 3.

4.3. User assistance

Within the MAS, the purpose of Agents E and F is to assist users for the interpretation of experimental data and the notification of system status information documents. There are two degrees of access with level 2 users (scientists) who have more rights than level 1 (normal user). The main activities carried out by agent E can be summarized in: i) facilitating the interpretation of experimental data, showing them in real-time in the form of graphs and tables; ii) make them available in various formats, via download, for further research activities.

At the end of the activity cycle, carried out by the various clusters of agents, which aim at determining the alert level, A transmits the updates that have occurred: upon receipt of the notification, agent E is activated instantly by refreshing the SEW dashboard, which will show updated graphs and tables of each seismic precursor. For both user levels, there is a button pointing to the precursor databases which contains recent and historical experimental data. Through a multiple-choice menu, it is possible to select one of the following possible formats: JSON, CSV and KML. When the user clicks on the “download” button, reactively and according to the selected choice, Agent E will take care of data extraction, formatting according to the selected format and starting the download process.

At the same time, Agent F takes care of creating a report, in pdf, to be sent via e-mail to a small group of scientists whose e-mail addresses have been stored. The document will be sent only in the presence of a hard level alarm and will present several standard fields: (i) the geographical coordinates of the area in which the seismic swarm occurred and the hypocentral depth; (ii) the frequency of earthquakes in the last two days; (iii) the average values and the relative standard deviation of the seismic precursors; (iv) further technical information on the instrumentation used, the seismogenic structure affected by the seismic swarm, etc. The information reported in the document have been extracted from the databases and system variables in which they are stored and assembled in a specific template, used for the realization of the report. In the case of the other alarm levels, no notification will be sent to the scientists, however it will always be possible to access an updated report, once a day, directly from the dashboard whose access is limited to level 2 users only.

5. Case study: New Zealand, a land with high seismic and volcanic risk

New Zealand is a region characterized by a high seismic and volcanic risk due to the presence of a fair number of active volcanoes and the particular geodynamic location, in the collision zone between the Australian Plate and Pacific Plates. For this reason, the area is covered by a dense network of sensors, some of which have only recently been operating, which allow continuous monitoring of different seismic and volcanic precursors. The data are made available to users through the GeoNet project (Geological hazard information for New Zealand) at https://www.geonet.org.nz.

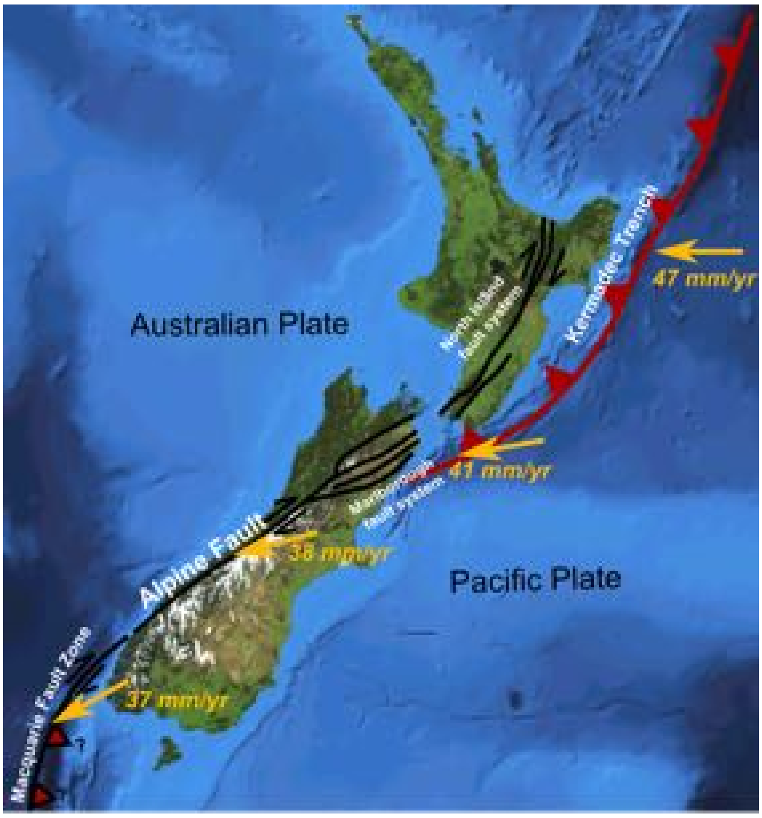


Figure 4: Scheme of the subduction area (Kermadec Trench) and the transform zone (Alpine fault) with the relative displacement speeds of the Pacific plate in collision with the Australian plate. *(Credit: Mikenorton via Creative Commons https://commons.wikimedia.org/w/index.php?curid=10735284)*

5.1. Seismotectonic overview

Within the GeoNet Quake Search section, New Zealand is divided into 10 seismic regions, from Auckland & Northland to Wellington & Marlborough. The intense tectonic and seismic activity is attributable to the presence of the Alpine fault, a large dextral transform structure, which crosses the southern part and marks the contact between the Pacific and the Australian plate. In the eastern off-shore area of the north island, the Pacific plate dips below the Australian plate: the phenomenon of subduction continues also at the Cook Strait and is the cause of deep earthquakes and the presence of active volcanism in the island of North. There are also a series of active secondary faults kinetically connected with the Alpine one, like Marlborough fault system, a set of four major faults which transfer displacement between Alpine fault and the Kermadec Trench (see Figure 4).

5.2. Experiments

The network of seismometers and GPS/GNSS sensors is well developed and represents a good way to test the seismic alert system. Each seismic region is covered by a fair number of

GPS/GNSS stations for the measurement of the ground deformation, even if for some stations the operativeness has occurred only in the last years and for others, the first registrations are from 1999. Of the three components that relate to displacement from the initial position (east, north, and up) only the up component was taken into consideration, relative to the vertical displacements of the ground. And this because the other two components, east and north, are mainly attributable to the displacement of the two plates.

The seismic data available on the "GeoNet Quake Search" page of the geonet.org.nz site were filtered by geographic coordinates, region and depth and downloaded in CSV format: the threshold value and the relative standard deviation were then calculated for two high magnitude seismic events. To download the data relating to the ground deformation, the GeoNet API was used, which allows the experimental data to be downloaded quickly, using special queries carried out in GET mode.

The SEW test was performed on the northern segment of the Marlborough fault system of the Wellington & Marlborough seismic region. The GPS/GNSS stations used for the calculation of the threshold values are those closest to the seismogenic structure analyzed, in which experimental data were available from 2004. Seismic events occurred on 2013-07-21 and 2016-11-14 were considered, respectively of $M_w = 6.5$ and $M_w = 6.2$. Only two mainshocks have been considered, although they are made up of more than 600 seismic events in total, because catastrophic earthquakes of high magnitude, over the last twenty years, are quite limited in number.

For the ground deformation, the registrations made up to four months before the mainshock was considered and the threshold values for each of the three stations were obtained using the data relating to the two seismic events of 2013 and 2016. In reality, by restricting the datasets to one month before the seismic event, the variation in the values obtained for the three stations is negligible and falls within the order of a tenth of a millimetre.

The *2013-08-16 earthquake* of $M_w = 6.5$ was used to test the correspondence between seismic swarms in progress and statistically calculated threshold values. A further test was performed on the *seismic swarm of May 2018* which as a final result did not give a mainshock.

6. Results

The data of the seismic swarms relating to earthquakes occurred on 2013-07-21 and 2016-11-14 are shown in Table 1: M_{ms} indicates the magnitude value of the mainshock. The threshold value obtained for the northern segment of the Marlborough fault system is of 2.2 ± 0.2. Table 2 shows the threshold values (T_h) and the respective standard deviations (σ), expressed in centimetres, relating to the 2013 and 2016 earthquakes for the three stations CMBL, WITH and KAIK. All stations are characterized by negative ground displacements which denote land subsidence before the mainshock.

The 2013-08-16 earthquake of $M_w = 6.5$, which occurred about a month later after the strong earthquake of July 2013, was used as a sequence to test the system. Figure 3 shows one of the seismic swarms, in the Marlborough fault system, which preceded the mainshock: you can see the alignment of the hypocenters along a preferential direction that corresponds to the direction of development of the fault system that generated it (see Marlborough fault system of Figure 4).

Table 1
Seismic swarms before the mainshock (Marlborough fault system)

Seismic swarms	M_{ms}	$\bar{M}$	N_{se}
2013-07-21	6.5	2.4	340
2016-11-14	6.2	2.0	290
Threshold value: 2.2			
Standard deviation: 0.2			
Total number of seismic events: 630			

Table 2
Ground deformation before the mainshock (Marlborough fault system)

GPS/GNSS Stations	$T_h(cm)$	$\sigma(cm)$	N_m
CMBL	-29	4.1	400
WITH	-4.6	0.8	400
KAIK	-17.2	0.4	400
Total number of measurements: 1200			

Table 3 reports the average magnitude value and the relative standard deviation of the seismic swarm before the mainshock which is in the range of 2.3 ± 0.4. The fields relating to the three ground deformation measuring stations show the values $\bar{GD} \pm \sigma_{gd}$. It can be seen that in all stations the intervals of the ground deformation in progress fall within the intervals of the threshold values $T_h \pm \sigma$. Hence, condition (4) is verified for both seismic precursors (SS and GD):

$$([\bar{M_w} \pm \sigma_w] \cap [SS_{Th} \pm \sigma_{th}] \neq \emptyset) \wedge ([\bar{GD} \pm \sigma_{gd}] \cap [GD_{Th} \pm \sigma_{th}] \neq \emptyset). \quad (6)$$

The evaluation of the Boolean expression for ground deformation corresponds to a logical AND between the three GNSS/GPS stations:

$$(CMBL) \wedge (WITH) \wedge (KAIK). \quad (7)$$

Figure 5 shows that in the three stations considered, before the event of August 2013, the ground deformation intervals intersects that of the respective threshold values and therefore, according to the final result, the evaluation of the GD parameter returns true. A similar result is also obtained for the seismic swarm parameter with an almost complete overlap between the confidence interval in progress and that relating to the threshold value. Table 3 also shows the results of the seismic swarm of May 2018 (about 115 foreshocks), indicated as two asterisks,

Table 3
Test 2013-08-16 earthquake* and seismic swarm** on May 2018

	Seismic Swarm	CMBL	WITH	KAIK
$\bar{GD}$*	-	-24.6 $\pm$ 3.5	-4.9 $\pm$ 0.3	-16.8 $\pm$ 0.4
$\bar{GD}$**	-	94.7 $\pm$ 0.5	9.4 $\pm$ 0.5	57.6 $\pm$ 0.5
$\bar{M_w}$*	2.3 $\pm$ 0.4	-	-	-
$\bar{M_w}$**	2.3 $\pm$ 0.7	-	-	-

which affected the same fault system. It can be observed that although the average value of the seismic swarm falls within the confidence interval of the respective threshold value, the expression (7) returns false because this does not happen for the ground deformation which has an inverse (positive) sign with respect to the corresponding (negative) threshold values.

7. Conclusions

An integrated, complementary and real-time analysis of a series of precursor parameters to establish of the alert level of a seismic risk territory, is the idea on which the SEW forecasting system is based. With the integrated analysis, we aim to simultaneously analyze the experimental data of the physico-chemical precursors for which an adequate network of sensors is available. Acting in a complementary way means considering the results obtained by each parameter not disjoint from the others but which contribute, in different ways, to the achievement of the final objective. The innovation of the proposed model lies precisely in these short and simple concepts and the final evaluation of Boolean expressions made up of representative variables of each precursor allows each of them to make their contribution. In this way, it is possible to assess whether the transformation that a seismic territory is undergoing is on average attributable to those that occurred in the past in the periods preceding earthquakes of equal magnitude ($M_w \geq 6$).

According to the *theory of dilatancy and asperity*, the transformations that a territory undergoes before a strong seismic event produce ground deformations which, by fracturing, generates foreshock and catalyzes fluids from the surrounding areas, making their geochemical properties vary. If we consider that the entity of the deformations depends on the mechanical characteristics of the rocks present in each seismogenic area, we can consider that before each "characteristic earthquake" [15], physico-chemical anomalies, on average similar to those that occurred in the past, can be generated.

The SEW system has been implemented using the MAS approach, hence ensuring modularity, efficiency and maintainability. The choice to use a multi-agent structure greatly facilitates the process of acquiring and processing experimental data carried out in parallel by the agent clusters, each of which deals with a specific precursor. Two further agents, specialized in user

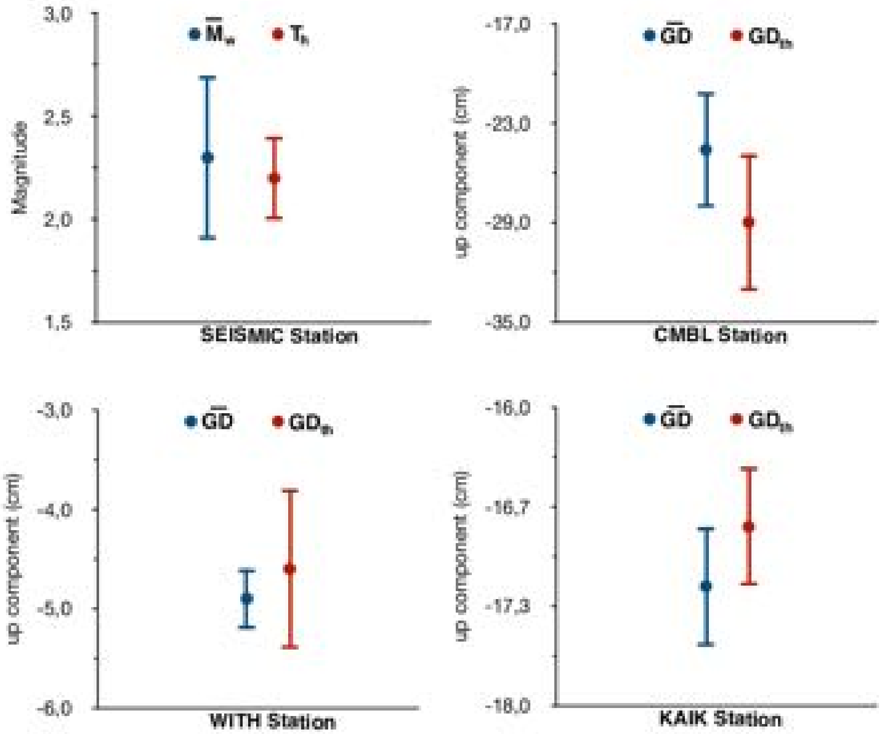


Figure 5: Comparison between the ranges of the data in progress and the threshold values for the ground deformation and the seismic swarms related to the August 2013 earthquake. Note that in all cases there is an overlap of the intervals.

assistance, take care of adequately formatting diagrams, tables and reports to be presented to users or sent to specific scientific groups to alert them of any critical states.

The results of the $M_w = 6.5$ earthquake test of 2013-08-16 on one of the seismic regions of New Zealand show an extensive overlap of the ranges $[\bar{M_w} \pm \sigma_w]$ and $[\bar{GD} \pm \sigma_{gd}]$ of both precursors with their respective confidence intervals of the threshold values and only small parts of the left interval are external to them. The ground deformation indicates that the areas surrounding the seismogenic structure undergo pronounced subsidence in the period before the seismic event. The choice of this datasets is due to the possibility of using both seismic swarms and ground deformations starting from 2004, a combination not possible for surface earthquakes in the other seismic regions of New Zealand.

On the contrary, the seismic swarm of May 2018, which affected the same area, shows that even if the seismic swarms in progress meet the threshold intervals, the deformation values in the three stations are largely outside the intervals $[GD_{Th} \pm \sigma_{th}]$: the absence of the mainshock is therefore in agreement with the result of the expression (7) which returns false. The results, therefore, confirm that a forecast based on a fair number of precursors can be a good solution for the implementation of a seismic alert system.

Acknowledgement

We acknowledge the New Zealand GeoNet project and its sponsors EQC, GNS Science and LINZ, for providing data/images used in this study, and project TEAMS–TEchniques to support the Analysis of big data in Medicine, energy and Structures–Piano di incentivi per la ricerca di Ateneo 2020/2022.

References

[1] M. Krzysztoń, B. Śnieżyński, Combining machine learning and multi-agent approach for controlling traffic at intersections, in: Computational Collective Intelligence, LNCS, volume 9329, 2015, pp. 57–66. doi:10.1007/978-3-319-24069-5_6.

[2] D. Grzonka, A. Jakóbik, J. Kołodziej, S. Pllana, Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security, Future Generation Computer Systems, Elsevier 86 (2018) 1106–1117. doi:10.1016/j.future.2017.05.046.

[3] A. E. F. Seghrouchni, A. M. Florea, A. Olaru, Multi-agent systems: A paradigm to design ambient intelligent applications, in: Studies in Computational Intelligence, SCI, volume 315, 2010, pp. 3–9. doi:10.1007/978-3-642-15211-5_1.

[4] V. Suryanarayanan, G. Theodoropoulos, M. Lees, PDES-MAS: Distributed simulation of multi-agent systems, Procedia Computer Science, Elsevier 18 (2013) 671–681. doi:10.1016/j.procs.2013.05.231.

[5] A. Calvagna, E. Tramontana, Delivering dependable reusable components by expressing and enforcing design decisions, in: Proc. of IEEE Computer Software and Applications Conference (COMPSAC), IEEE, 2013, pp. 493–498.

[6] C. Cavallaro, G. Verga, E. Tramontana, O. Muscato, Suggesting just enough (un)crowded routes and destinations, in: Proc. of 21st Workshop 'From Objects to Agents' (WOA 2020), Bologna, Italy, 2020, pp. 493–498.

[7] G. Bunea, F. Leon, G. Atanasiu, Postdisaster evacuation scenarios using multiagent system, Journal of Computing in Civil Engineering, American Society of Civil Engineers 30 (2016). doi:10.1061/(ASCE)CP.1943-5487.0000575.

[8] F. Fiedrichn, An hla based multiagent system for optimized resource allocation after strong earthquakes, in: J. Cohen (Ed.), Proceedings of the 2006 Winter Simulation Conference, Monterey, CA, 2006, pp. 486–492. doi:10.1109/WSC.2006.323120.

[9] D. Moser, D. Pinto, A. Cipriano, Developing a multiagent based decision support system for realtime multi-risk disaster manage- ment, International Journal of Environmental and Ecological Engineering 9 (2015) 831–835. doi:doi.org/10.5281/zenodo.1099832.

[10] R. Spina, A. Fornaia, E. Tramontana, VSEW: an early warning system for volcanic and seismic events, in: Proceedings of IEEE International Conference on Smart Computing, SMARTCOMP, Bologna, Italy, 2020.

[11] A. Tani, T. Yamamura, Y. Waridashi, H. Kawamura, A. Takizawa, Simulation on rescue in case of earthquake disaster by multi-agent system, in: Proc. of World Conference on Earthquake Engineering, Vancouver, BC, Canada, 2004, pp. 1–6.

[12] G. Bunea, G. M. Atanasiu, F. Leon, The effect of information on the performance of a multiagent system for post-disaster evacuation, in: Proc. of International Symposium on Life-Cycle Civil Engineering, Delft, The Netherlands, 2016, pp. 2053–2059.
[13] T. Furuya, S. Sadohara, Modeling and simulation of rescue activity by the local residents in the seismic disaster, in: Proc. of ESRI International User Conference, 2004.
[14] I. G. Main, A. F. Bell, P. G. Meredith, S. Geiger, S. Touati, The dilatancy-diffusion hypothesis and earthquake predictability, Geological Society, London, Special Publications 367 (2012) 215–230. doi:https://doi.org/10.1144/SP367.15.
[15] K. Aki, Asperities, barriers, characteristic earthquakes and strong motion prediction, Journal of Geophysical Research 89 (1984) 5867–5872. doi:10.1029/JB089iB07p05867.
[16] M. Béjar-Pizarro, et al., Asperities and barriers on the seismogenic zone in north chile: State of the art after the 2007 mw 7.7 tocopilla earthquake inferred by GPS and InSar data, Geophysical Journal International 183 (2010) 390–406. doi:10.1111/j.1365-246X.2010.04748.x.
[17] S. Awadallah, D. Moure, P. Torres-González, An internet of things (IoT) application on volcano monitoring, Sensors 19 (2019) 4651. doi:10.3390/s19214651.

Session 6

Tools & Applications

A Reactive Cognitive Architecture based on Natural Language Processing for the task of Decision-Making using a Rich Semantic

Carmelo Fabio Longo[a], Francesco Longo[b] and Corrado Santoro[a]

[a]Department of Mathematics and Computer Science, University of Catania, Viale Andrea Doria, 6, 95125 Catania, Italy
[b]Department of Engineering, University of Messina, Contrada di Dio, S. Agata, 98166 Messina, Italy

Abstract

The field of cognitive architectures is rich of approaches featuring a wide range of typical abilities of human mind, like perception, action selection, learning, reasoning, meta-reasoning and others. However, those leveraging Natural Language Processing are quite limited in both domain and reasoning capabilities. In this work, we present a cognitive architecture called CASPAR, based on a Belief-Desire-Intention framework, capable of reactive reasoning using a highly descriptive semantic made of First Order Logic predicates parsed from natural language utterances.

Keywords

Cognitive Architecture, Natural Language Processing, Artificial Intelligence, First Order Logic, Internet of Things

1. Introduction

In the last decade, a large number of devices connected together and controlled by AI has entered in millions of houses: the pervasive market of Internet of Things (IoT). Such a phenomenon is extended also in domains other than the domestic one, such as smart cities, remote e-healthcare, industrial automation, and so on. In most of them, especially the usual domestic ones, *vocal assistants* assume an important role, because voice is the most natural way to give the user the feeling to deal with an intelligent sentient being who cares about the proper functioning of the home environment. But how *intelligent* are these vocal assistants actually? Although there can be more definitions of *intelligence*, in this work we are interested only in those related to autonomous agents acting in the scope of decision-making.

Nowadays, companies producing vocal assistants aim more at increasing their pervasiveness than at improving their native reasoning capabilities; with *reasoning capabilities*, we can intend not only the ability to infer the proper association *command* $\rightarrow$ *plan* from utterances, but also to be capable of combining facts with rules in order to infer new knowledge and help the user in decision-making tasks.

Except the well known cloud-based vocal assistants [1], other kind of solutions [2, 3, 4] are based on neural models exclusively trained on the domotic domain; or they exploit chat

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
fabio.longo@unict.it (C. F. Longo); flongo@unime.it (F. Longo); santoro@dmi.unict.it (C. Santoro)
0000-0002-2536-8659 (C. F. Longo); 0000-0001-6299-140X (F. Longo); 0000-0003-1780-5406 (C. Santoro)

engines [5, 6] whose understanding skills are strictly depending on syntax. This makes the range of their capabilities quite limited.

In light of the above, in this paper our aim is the design of a cognitive architecture, called CASPAR, based on Natural Language Processing (NLP), that makes it possible the implementation of intelligent agents able to outclass the available ones in performing deductive activities. Such agents could be used for both domotic purposes and any other kind of applications involving common deductive processes based on natural language. As a further motivation, we have to highlight that, as claimed in [7], cognitive architectures have been so far mainly used as research tools, and very few of them have been developed outside of academia; moreover, none of them has been specifically designed for IoT. Of course, most of them have features and resources which could be exploited in such a domain, but the starting motivations were different from ours.

Although cognitive architectures should be distinguished from models that implement them, our architecture can be used as domotic agent *as is*, after the definitions of both the involved entities and the I/O interfaces.

This paper is structured as follows: Section 2 describes the state of the art of related literature; Section 3 shows in detail all the architecture's components and underlying modules; Section 4 shows the architecture reasoning heuristic in the presence of clauses made of composite predicates, taking into account possible argument substitutions as well; Section 5 summarizes the content of the paper and provides our conclusions, together with future work perspectives. A Python implementation of CASPAR is also provided for research purposes in a Github repository[1].

2. Related work

The number of existing cognitive architectures has reached several hundreds according to the authors of [7]. Among the most popular ones, which also influenced several subsequent works, there are SOAR, CLARION and LIDA, mentioned in a theoretical comparison in [8]. Most of them got inspired either by neuroscience or psychanalysis/philosophy studies; the former are surely less fancy, being supported by scientific data regarding functions of brain modules in specific conditions and their interactions. The Integrated Information Theory [9] provides even a metric *Phi* to evaluate the consciousness level of a cognitive system, which would be proportional to those overall interactions. In this section, we will focus mostly on those architectures implementing Reasoning/Action Selection, Natural Language Processing and Decision-Making, being the main basis on which CASPAR has been built.

In [10] the authors describe three different spoken dialog systems, one of them based on the FORR architecture and designed to fulfill the task of ordering books from the public library by phone. All the three dialog systems are based on a local Speech-to-Text engine called PocketSphinx which is notoriously less performing than cloud-based systems [11]. This leads to a greater struggle to reduce the bias between user's request and result.

The authors of [12] present a computational model called MoralIDM, which integrates multiple AI techniques to model human moral decision-making, by leveraging a two-layer

[1]http://www.github.com/fabiuslongo/pycaspar

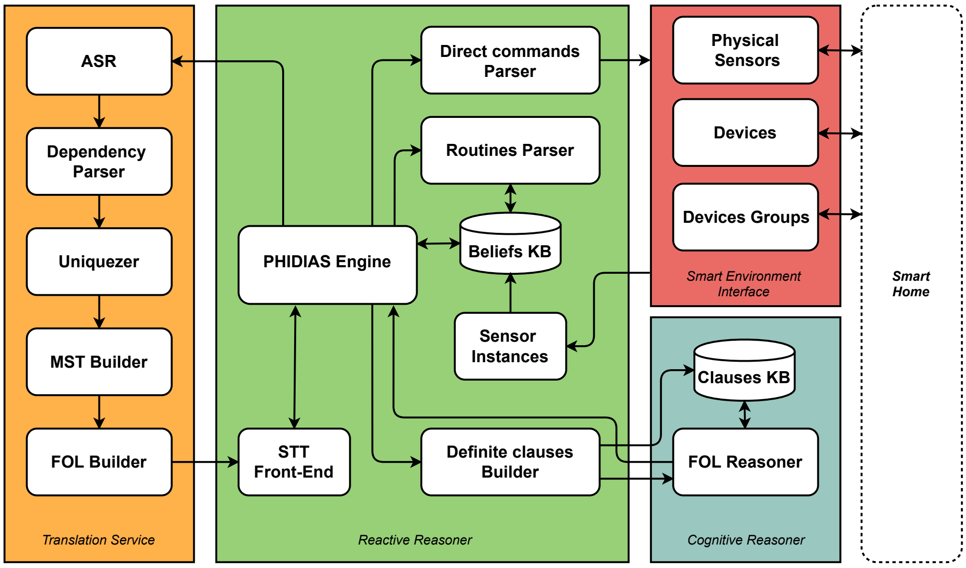


Figure 1: The Software Architecture of CASPAR

inference engine which takes into account prior cases decisions and a knowledge base with a formal representation of moral quality-weighted facts. Such facts are extracted from natural language by using a semi-automatic translator from simplified English (which is the major weakness of such approach) scenarios into predicate calculus.

The DIARC architecture [13] has been designed for addressing the issue of recognizing morally and socially charged situations in human-robot collaborations. Although it exploits several well known NLP resources (such as Sphinx, Verbnet, and Framenet), it has been tested only on trivial examples in order to trigger robot reactions, using an ad-hoc symbolic representation of both known and perceived facts.

In general, probing the existing cognitive architectures leveraging NLP, we have found that most of them are limited in both domain of application and in term of semantic complexity.

3. The Architecture

The name that has been chosen for the architecture presented in this paper is CASPAR. It derives from the following words: **C**ognitive **A**rchitecture **S**ystem **P**lanned **a**nd **R**eactive, whom summarize its two main features. In Figure 1, all interacting components are depicted, filled with distinct colours.

The main component of this architecture, namely the *Reactive Reasoner*, acts as "core router" by delegating operations to other components, and providing all needed functions to make the whole system fully operative.

This architecture's Knowledge Base (KB) is divided into two distinct parts operating separately, which we will distinguish as *Beliefs KB* and *Clauses KB*: the former contains information of physical entities which affect the agent and which we want the agent to affect; the latter contains conceptual information not perceived by agent's sensors, but on which we want the agent to make logical inference.

The Beliefs KB provides exhaustive cognition about what the agent could expect as input data coming from the outside world; as the name suggests, this cognition is managed by means of proper beliefs that can - in turn - activate proper plans in the agent's behaviour.

The Clauses KB is defined by the means of assertions/retraction of *nested* First Order Logic (FOL) definite clauses, which are possibly made of composite predicates, and it can be interrogated providing answer to any query (*True* or *False*).

The two KBs represent, somehow, two different kinds of human being memory: the so called *procedural memory* or *implicit memory*[14], made of thoughts directly linked to concrete and physical entities; the *conceptual memory*, based on cognitive processes of comparative evaluation.

As well as in human being, in this architecture the two KBs can interact with each other in a very reactive decision-making process.

3.1. The Translation Service

This component (left box in Figure 1) is a pipeline of five modules with the task of taking a sound stream in natural language and translating it in a *neo-davidsonian* FOL expression inheriting the shape from the event-based formal representation of Davidson [15], where for instance the sentence:

$$\textit{Brutus stabbed suddenly Caesar in the agora} \tag{1}$$

is represented by the following notation:

$$\exists e\ \texttt{stabbed(e, Brutus, Caesar)} \land \texttt{suddenly(e)} \land \texttt{in(e, agora)}$$

The variable e, which we define *davidsonian* variable, identifies the verbal action related to *stabbed*. In the case a sentence contains more than one verbal phrases we'll make usage of indexes for distinguish e_i from e_j with $i \neq j$.
As for the notation used in this work, it does not use ground terms as arguments of the predicates, in order to permit the sharing of different features related to the same term like it follows, whether we include the adjective *evil*:

$$\exists e\ \texttt{stabbed(e, Brutus(x), Caesar(y))} \land \texttt{evil(x)} \land \texttt{suddenly(e)} \land \texttt{in(e, agora(z))}$$

which can also be represented, *ungrounding* the verbal action arguments, as it follows:

$$\exists e\ \texttt{stabbed(e, x, y)} \land \texttt{Brutus(x)} \land \texttt{Caesar(y)} \land \texttt{evil(x)} \land \texttt{suddenly(e)} \land \texttt{in(e, z)} \land \texttt{agora(z)}$$

Furthermore, in the notation used for this work each predicate label is in the form `L:POS(t)`, where `L` is a lemmatized word and `POS` is a Part-of-Speech (POS) tag from the Penn Treebank tagset[16].

The first module in the pipeline, i.e., the *Automatic Speech Recognition* [17, 18, 19] (ASR), allows a machine to understand the user's speech and convert it into a series of words.

The second module is the *Dependency Parser*, which aims at extracting the semantic relationships, namely *dependencies*, between all words in a utterance. In [20], the authors present a comparative analysis of ten leading statistical dependency parsers on a multi-genre corpus of English.

The third module, the *Uniquezer*, aims at renaming all the entities within each dependency in order to make them unique. Such a task is mandatory to ensure the correctness of the outcomes of the next module in the pipeline (the *Macro Semantic Table*), whose data structures need a distinct reference to each entity coming from the dependency parser.

The fourth module, defined as *MST Builder*, has the purpose to build a novel semantic structure defined as *Macro Semantic Table* (MST), which summarizes in a canonical shape all the semantic features in a sentence, starting from its dependencies, in order to derive FOL expressions.

Here is a general schema of a MST, referred to the utterance u:

```
MST(u) = {ACTIONS, VARLIST, PREPS, BINDS, COMPS, CONDS}
```

where

$$
\begin{aligned}
\texttt{ACTIONS} &= [(\texttt{label}_k,\ \texttt{e}_k,\ \texttt{x}_i,\ \texttt{x}_j),\ldots] \\
\texttt{VARLIST} &= [(\texttt{x}_1,\ \texttt{label}_1),\ldots(\texttt{x}_n,\ \texttt{label}_n)] \\
\texttt{PREPS} &= [(\texttt{label}_j,\ (\texttt{e}_k \mid \texttt{x}_i),\ \texttt{x}_j),\ldots] \\
\texttt{BINDS} &= [(\texttt{label}_i,\ \texttt{label}_j),\ldots] \\
\texttt{COMPS} &= [(\texttt{label}_i,\ \texttt{label}_j),\ldots] \\
\texttt{CONDS} &= [\texttt{e}_1,\ \texttt{e}_2,\ldots]
\end{aligned}
$$

All tuples inside such lists are populated with variables and labels whose indexing is considered disjoint among distinct lists, although there are significant relations which will be clarified later. The MST building takes into account also the analysis done in [21] about the so-called *slot allocation*, which indicates specific policies about entity's location inside each predicate, depending on verbal cases. This is because the human mind, in the presence of whatever utterance, is able to populate implicitly any semantic role (identified by subject/object slots) taking part in a verbal action, in order to create and interact with a logical model of the utterance. In this work, by leveraging a step-by-step dependencies analysis, we want to create artificially such a model, to give an agent the chance to make logical inference on the available knowledge. All the dependencies used in this paper are part of the ClearNLP[22] tagset, which is made of 46 distinct entries. For instance, considering the dependencies of 1:

```
nsubj(stabbed, Brutus)
ROOT(stabbed, stabbed)
advmod(stabbed, suddenly)
dobj(stabbed, Caesar)
prep(stabbed, In)
det(agora, The)
pobj(in, agora)
```

from the couple nsubj/dobj it is possible to create new a tuple inside ACTIONS as it follows, taking also in account of variables indexing counting:

(stabbed, e_1, x_1, x_2)

and inside VARLIST as well:

(x_1, Brutus)
(x_2, Caesar)

Similarly, after an analysis of the couple prep/pobj it is possibile to create further tuples inside PREPS like it follows:

(in, e_1, x_3)

and inside VARLIST:

(x_3, agora)

The dependency advmod contains informations about the verb (*stabbed*) is going to modify by means the adverb *suddenly*. In light of this, a further tuple inside VARLIST will be created as it follows:

(e_1, suddenly)

As for the BINDS list, it contains tuples with a quality modifiers role: in the case the 1 had *the brave Caesar* as object, a further dependency amod will be created as it follow:

amod(Caesar, brave)

In this case, a *bind* between *Caesar* and *brave* will be created inside BINDS as it follows:

(Caesar, brave)

As with BINDS, COMPS contains tuples of terms related to each other, but in this case they are part of multi-word nouns like *Barack Hussein Obama*, whose nouns after the first will be classified as compound by the dependency parser.

As for the CONDS lists, it contains davidsonian variable whose related predicates subordinate the remaining others. For instance in the presence of utterances like:

if the sun shines strongly, Robert drinks wine

or

while the sun shines strongly, Helen smiles in both cases, the dependency mark will give information about subordinate conditions related to the verb *shines*, which are mark(shines, If) and mark(shines, while). In those cases, the davidsonian variable related to *shines* will populate the list CONDS. In the same way, in presence of the word *when*, a subordinate condition might be inferred as well; but since any adverbs are classified as advmod (as we have seen for *suddenly* before), it will be considered as subordinate condition only when its POS is WRB and not RB; the former denotes a wh-adverb, the latter a qualitative adverb.

The fifth and last module, defined as *FOL Builder*, aims to build FOL expressions starting from the MSTs. Since (virtually) all approaches to formal semantics assume the Principle of Compositionality[2], formally formulated by Partee [23], every semantic representation can be incrementally built up when constituents are put together during parsing. In light of the above, it is possible to build FOL expressions straightforwardly starting from a MST, which summarizes all semantic features extracted during a step-by-step dependencies analysis.
For the rest of the paper, the labels inside the MST tuples will be in the form of lemma:POS. Then, for instance, instead of stabbed we'll have stab:VBD, where *stab* is the lemmatization of *stabbed* and VBD is the POS representing a past tense.
For each tuple (var, lemma:POS) in VARLIST the following predicate will be created:

$$\texttt{lemma:POS(var)}$$

which represents a noun, such as tiger:NN(x_1) or Robert:NNP(x_1)[3]. var can also be a davidsonian variable when POS has the value of RB. In such cases, the tuples represent adverbs, such as Hardly:RB(e_1) or Slowly:RB(e_2).
For each tuple (lemma:POS, dav, subj, obj) in ACTIONS, the following predicate will be created:

$$\texttt{lemma:POS(dav, subj, obj)}$$

representing a verbal action, such as be:VBZ(e_1, x_1, x_2) or shine:VBZ(e_2, x_3, x_4).
For each tuple (lemma:POS, dav/var, obj) in PREPS the following predicate will be created:

$$\texttt{lemma:POS(dav/var, obj)}$$

where dav/var is a variable either in a tuple of ACTIONS or of VARLIST, respectively, while obj is a variable in a tuple of VARLIST. Such predicates represent verbal/noun prepositions.
For each tuple (lemma:POS_1, lemma:POS_2) in COMPS, whose first entity lemma:POS_1 is in a tuple of VARLIST, a predicate will be created as follows:

$$\texttt{lemma:}POS_2\texttt{(var)}$$

where var is the variable of a the tuple in VARLIST with lemma:POS_1 as second entity. In case of multi-word nouns, each further noun over the first of them in VARLIST will be encoded

[2]"The meaning of a whole is a function of the meanings of the parts and of the way they are syntactically combined."
[3]Without considering entities enumeration.

within COMPS.
As for CONDS, its usage will be explained next with an example. Let the sentence in exam be:

$$\textit{When the sun shines strongly, Robert is happy} \tag{2}$$

the related MST is:

```
ACTIONS = [(shine01:VBZ, e1, x1, x2),
           be01:VBZ(e2, x3, x4)]
VARLIST = [(x1, sun01:NN), (x2, ?), (x3, Robert01:NNP), (x4,
           happy01:JJ)]
CONDS = [e1]
```

It has to be noticed the numeration of the entities within each list, as effect of the Uniquezer processing before the MST building. As final outcome we'll have an implication like the following:

$$\texttt{shine01:VBZ}(e_1,\ x_1,\ \_) \wedge \texttt{sun01:NN}(x_1) \implies \texttt{be01:VBZ}(e_2,\ x_3,\ x_4) \wedge \texttt{Robert01:NNP}(x_3) \wedge \texttt{happy01:JJ}(x_4)$$

3.2. The Reactive Reasoner

As already mentioned, this component (central box in Figure 1) has the task of letting other modules communicate with each other; it also includes additional modules such as the Speech-To-Text (SST) Front-End, IoT Parsers (Direct Command Parser and Routine Parser), Sensor Instances, and Definite Clauses Builder. The Reactive Reasoner contains also the Beliefs KB, which supports both Reactive and Cognitive reasoning.

The core of this component processing is managed by the Belief-Desire-Intention Framework Phidias[24], which gives Python programs the ability to perform logic-based reasoning (in Prolog style) and lets developers write reactive procedures, i.e., pieces of program that can promptly respond to environment events.

The agent's first interaction with the outer world happens through the STT Front-End, which is made of production rules reacting on the basis of specific words asserted by an Instance Sensor; the latter, being instance of the superclass *Sensor* provided by Phidias, will assert a belief called STT(X) with X as the recognized utterance, after the sound stream is acquired by the microphone and translated in text by means of the ASR.

The Direct Command Parser has the task of combining FOL expressions predicates with common variables coming from the Translation Services, via a production rules system. The final outcome of such rules is a belief called INTENT, which might trigger another rule in the Smart Environment Interface. A similar behaviour is reserved to the Routine Parser, when subordinating conditions within an IoT command are detected; it produces two types of beliefs: ROUTINE and COND, linked together by a unique code. The belief ROUTINE is a sort of *pending* INTENT, which cannot match any production rule and execute its plan until the content of its related COND meets those of another belief asserted by a Sensor Instance and called SENSOR. Then, the ROUTINE belief will be turned into INTENT and get ready for the execution as direct command, as shown in lines 2, 3, 5, 7, 8 of Listing 1 in the Appendix

The Definite Clauses Builder is responsible of combining FOL expression predicates with common variables, through a production rules system, in order to produce nested definite clauses. Considering the 2 and its related FOL expression producted by the Translation Service, the production rule system of the Definite Clauses Builder, taking in account of the POS of each predicate, will produce the following *nested* definite clause:

$$\texttt{shine01:VBZ(sun01:NN(x}_1\texttt{), \_)} \implies \texttt{be01:VBZ(Robert01:NNP(x}_3\texttt{),}$$
$$\texttt{happy01:JJ(}x_4\texttt{))}$$

The rationale behind such a notation choice is explained next: a definite clause is either atomic or an implication whose antecedent is a conjunction of positive literals and whose consequent is a single positive literal. Because of such restrictions, in order to make MST derived clauses suitable for doing inference with the Backward-Chaining algorithm (which works only with KB made of definite clauses), we must be able to incapsulate all their informations properly. The strategy followed is to create composite terms, taking into account of the POS tags and applying the following hierarchy to every noun expression as it follows:

$$\texttt{IN(JJ(NN(NNP(x))), t)} \tag{3}$$

where IN is a preposition label, JJ an adjective label, NP and NNP are noun and proper noun labels, x is a bound variable and t a predicate.
As for the verbal actions, the nesting hierarchy will be the following:

$$\texttt{ADV(IN(VB(t}_1\texttt{, t}_2\texttt{), t}_3\texttt{))}$$

where ADV is an adverb label, IN a preposition label, VB a verb label, and t_1, t_2, t_3 are predicates; in the case of intransitive or imperative verb, instead of respectively t_2 or t_1, the arguments of VB will be left void. As we can see, a preposition might be related either to a noun or a verb.

3.3. The Smart Environment Interface

This component (upper right box in Fig.1) provides a bidirectional interaction between the architecture and the outer world. In Listing 1 in the Appendix, a simple example is shown, where a production rules system is used as reactive tool to trigger proper plans in the presence of specific asserted beliefs. In [25] we have shown the effectiveness of this approach by leveraging the Phidias predecessor Profeta[26], even with a shallower analysis of the semantic dependecies, as well as an operations encoding via WordNet[27] in order to make the operating agent multi-language and multi-synonimous.

Such an interface includes a production rules system containing different types of entities definitions and operation codes involving the entities themself, which trigger specific procedures containing high level language (e.g., lines 11 and 12 in Listing 1 in the Appendix). The latter should contain all required functions for driving each device in order to get the desired behaviour, whose implementation in this work is left to the developer. Each production rule contains also subordinating conditions defined as *Active Beliefs*: `lemma_in_syn(X, S)` checks the membership of the lemma X to the synset S, to make the rule multi-language and multi-synonimous (after having defined the entities depending on the language); while, the Active

Belief `eval_cls(Y)` lets Belief KB and Clauses KB interact with each other in a very decision-making process, where the agent *decides* either to execute or not the related plan within the square brackets, accordingly to the reasoning of the query `Y`; the latter in line 12-13 of Listing 1 in the Appendix is the representation of the sentence *an inhabitant is at home*.

Finally, this module contains also production rules to change routines into direct commands according to the presence of specific belief related to conditionals, which might be asserted or not by some Sensor Instance (see lines 2, 3, 5, 8, 9 of Listing 1 in the Appendix).

3.4. The Cognitive Reasoner

This component (bottom right box in Figure 1) allows an agent to assert/query the Clauses KB with *nested* definite clauses, where each predicate argument can be another predicate and so on, built by the *Definite Clauses Builder* module (within the *Reactive Reasoner*).

Beyond the nominal FOL reasoning with the known Backward-Chaining algorithm, this module exploits also another class of logical axioms, the so-called *assignment rules*. We refer to a class of rules of the type "P is-a Q" where P is a predicate whose variable travels across one hand-side to another, with respect to the implication symbol. For example, if we want to express the concept: *Robert is a man*, we can use the following closed formula:

$$\forall \texttt{x}\ \texttt{Robert(x)} \implies \texttt{Man(x)} \tag{4}$$

but before that, we must consider a premise: the introduction of such rules in a KB can be possible only by shifting all its predicates from a strictly semantic domain to a pure conceptual one, because in a semantic domain we have just the knowledge of morphological relationships between words given by their syntactic properties. Basically, we need a medium to give additional meaning to our predicates, which is provided by WordNet [27]. This allows us to make logical reasoning in a conceptual space thanks to the following functions:

$$F_I : P_S \longrightarrow P_C \quad F_{Args(F_I)} : X_S^n \longrightarrow Y_C^n \tag{5}$$

F_I is the *Interpreter Function* between the space of all semantic predicates which can be yield by the `MST` sets and the space of all conceptual predicates P_C; it is not injective, because a single semantic predicate might have multiple corrispondences in the codomain, one for each different synset containing the lemma in exam. $F_{Args(F_I)}$ is between domain and codomain of all predicate's argument of F_I, which have equal arity. For instance, considering the FOL expression of (4):

$$\texttt{be:VBZ}(\texttt{e}_1,\ \texttt{x}_1,\ x_2) \land \texttt{Robert:NNP}(\texttt{x}_1) \land \texttt{man:NN}(x_2)$$

After an analysis of *be*, we find the lemma within the WordNet synset encoded by `be.v.01` and defined by the gloss: *have the quality of being something*. This is the medium we need for the domain shifting which gives a common sense meaning to our predicates.

In light of above, in the new conceptual domain given by (5), the same expression can be rewritten as:

$$\texttt{be\_VBZ}(\texttt{d}_1,\ \texttt{y}_1,\ \texttt{y}_2) \land \texttt{Robert\_NNP}(\texttt{y}_1) \land \texttt{man\_NN}(\texttt{y}_2)$$

where `be_VBZ` is fixed on the value which identify y_1 with y_2, `Robert_NNP(x)` means that x identify *Robert*, and `man_NN(x)` means that x identify *a man*.

Considering the meaning of `be_VBZ`, it does make sense also to rewrite the formula as:

$$\forall y \ \texttt{Robert\_NNP(y)} \implies \texttt{man\_NN(y)} \tag{6}$$

whrere y is a bound variable like x in (4).

Having such a rule in a KB means that we can implicitly admit additional clauses having `man_NN(y)` as argument instead of `Robert_NNP(y)`.

The same expression, of course, in a conceptual domain can also be rewritten as a composite fact, where `Robert_NNP(x)` becomes argument of `man_NN(x)` as it follows:

$$\texttt{man\_NN(Robert\_NNP(y))} \tag{7}$$

which agrees with the hierarchy of 3 as outcome of the Definite Clauses Builder.

As claimed in [28], not every KB can be converted into a set of definite clauses, because of the single-positive-literal restriction, but many KB can, like the one related to this work for the following reasons:

1. No clauses made of one single literal will ever be negative, due to the closed world assumption. Negations, initially treated like whatever adverb, when detected and related to *ROOT* dependency are considered as polarity inverter of verbal phrases; so, in this case, the *assert* will be turned into *retract*.
2. When the right hand-side of a clause is made by more than one literals, it is easy to demonstrate that, by applying the implication elimination rule and the principle of distributivity of $\vee$ over $\wedge$, a non-definite clause can be splitted into n definite clauses (where n is the number of consequent literals).

4. Nested Reasoning and Clause Conceptual Generalizations

The aim of the Cognitive Reasoner is to query a KB made of nested clauses that are also made closer to any possible related query, thanks to an appropriate pre-processing at assertion-time. Such a pre-processing, which creates a runtime *expansion* of the KB for every asserted clause, takes advantage of assignment rules for derivation of new knowledge.

The Backward-Chaining algorithm, as is, in presence of clauses where argument manipulation is required, might not be effective. To achieve such a goal, when required clauses are not present in the KB but deductible by proper arguments substitutions, the clauses evaluations at reasoning-time can be quite heavy and not feasible in term of complexity, because the process requires unifications at every single step. Instead, we will show how, by expanding properly the KB at assertion-time, the reasoning itself can be achieved acceptably. In order to obtain such a goal, CASPAR extends the radius of the nominal Backward-Chaining through the expansion of the Clauses KB with new knowledge generated starting from arguments substitutions on copies of specific clauses already asserted before.

For instance, let us consider a KB made at most of one-level[4] composite predicates as follows:

[4]Supposing a zero-level composite predicate be `P(x)`.

$$P_1(G_1(x_1)) \land P_2(G_2(x_2)) \implies P_3(F_3(x_3))$$
$$P_1(F_1(x_1))$$
$$P_2(F_2(x_2))$$
$$F_1(x) \implies G_1(x)$$
$$F_2(x) \implies G_2(x)$$
$$H_3(x) \implies F_3(x)$$

Querying such a KB with $P_3(H_3(x))$, for instance, using the Backward-Chaining algorithm, it will return *False* because there are neither any unifiable literals present nor as consequent of a clause. Instead, by exploiting $H_3(x) \implies F_3(x)$, we can also query the KB with $P_3(F_3(x))$ which is present as consequent of the first clause and it is surely satisfied together with $P_3(H_3(x))$: that's what we define as *Nested Reasoning*.

Now, to continue the reasoning process, we should check about the premises of such clause, which is made of the conjunction of two literals, namely $P_1(G_1(x_1))$ and $P_2(G_2(x_2))$. The latters, although not initially asserted, can be obtained starting by argument substitution on copies of other clauses from the same KB. Such a process is achieved by implicitly asserting the following clauses together with $P_1(F_1(x_1))$ and $P_2(F_2(x_2))$:

$$P_1(F_1(x_1)) \implies P_1(G_1(x_1))$$
$$P_1(F_1(x_1)) \implies P_2(G_2(x_2))$$

Since we cannot know in advance what a future successful reasoning requires, considering all possible nesting levels, along with the previous clauses also the so-called *Clause Conceptual Generalizations* will be asserted:

$$P_1(G_1(x_1)) \land P_2(G_2(x_2)) \implies F_3(x_3)$$
$$F_1(x_1)$$
$$F_2(x_1)$$

where the antecedent of the implication is unchanged to hold the quality of the rule, while $F_1(x_1)$, $F_2(x_1)$, $F_3(x_3)$, as satisfiability contributors of respectively $P_1(F_1(x_1))$, $P_2(F_2(x_2))$, $P_3(F_3(x_3))$, are assumed asserted together with the latters. In other terms, the predicates: P_1, P_2, P_3 can be considered as *modifiers* of respectively F_1, F_2, F_3.

A generalization considering also the antecedent of the implicative formula is possible only through a weaker assertion of the entire formula itself, by changing $\implies$ with $\land$ as it follows:

$$\exists\ x_1,\ x_2,\ x_3 \mid G_1(x_1) \land G_2(x_2) \land F_3(x_3)$$

which is not admitted as definite clause, being not a single positive literal. In any case, the mutual existence of x_1, x_2, x_3 which satisfies such a conjunction, is already subsumed by the implication.

After such a theoretic premise, let's make a more practical example considering the following natural language utterance:

When the sun shines hard, Barbara drinks slowly a fresh lemonade

Table 1
Clause Generalizations Constituents Table (A=Applied, NA=Not Applied)

Hard	Slowly	Fresh
A	NA	NA
A	A	NA
A	NA	A
A	A	A

The corresponding definite clause will be (omitting the POS tags for the sake of readability):

```
Hard(Shine(Sun(x1), __)) ⟹ Slowly(Drink(Barbara(x3),
                  Fresh(Lemonade(x4))))
```

Considering as *modifiers* adjectives, adverbs and prepositions, following the schema in Table 1: all the clauses generalization (corresponding to the first three rows of the table, while the forth is the initial clause) can be asserted as it follows:

```
Hard(Shine(Sun(x1), __)) ⟹ Drink(Barbara(x3), Lemonade(x4)))
Hard(Shine(Sun(x1), __)) ⟹ Slowly(Drink(Barbara(x3), Lemonade(x4)))
Hard(Shine(Sun(x1), __)) ⟹ Drink(Barbara(x3), Fresh(Lemonade(x4)))
```

As said before, the antecedent (whether existing) of all generalizations remains unchanged to hold the quality of the triggering condition, while the consequent shape will range on all possible variations of its modifiers, which will be 2^n with n as number of modifiers. Here the adverb *Hard*, being common part of all the antecedents composition, is always *Applied*.

Although in such a case the number of generalizations is equal to 4, in general it might be quite higher: it has been observed, after an analysis of more text corpus from the Stanford Question Answering Dataset[29], that the average number of modifiers in a single non-implicative utterance is equal to 6. In such cases the number of generalizations would be equal to 64, but greater numbers of modifiers would make the parsing less tractable, considering also arguments analysis for possible substitutions. In order to limit such a phenomenon, depending on the domain, CASPAR gives the chance to limit the number of generalizations by a specific parameter which modifies the policies of selective inclusion/exclusion of modifiers categories (adjectives, adverbs or prepositions).

In such a scenario, of course, the more the combinatorial possibilities, the more the number of clauses in the Clauses KB. It will appear clear, for the reader, that this approach sacrifices space for a lighter reasoning, but we rely on a three distinct points in favor of our choice:

1. An efficient indexing policy of the Clauses KB, for a fast retriving of any clause.
2. The usage of the class *Sensor* of Phidias for every clauses assertion, which works asynchronously with respect to the main production rules system, will make the agent immediately available after every interrogation without any latency, while additional clauses will be asserted in background.

3. We point to keep the Clauses KB as small as possible, in order to limit the combinatorial chances. In this paper we assume the assignment rules properly chosen among the most likely which can get the query closer to a proper candidate. As future works, a reasonable balancing between two distinct Clauses KB working on different levels might be a good solution: in the lower level (long-term memory) only clauses pertinent with the query will be searched, then put in the higher one (short-term memory) for attempting a successful reasoning. Similar approaches have been used with interesting outcomes in some of the widespread Cognitive Architectures[8].

As result evaluation, we consider a slightly rephrased KB (*Colonel West*) treated in [28], showing how CASPAR is able to make a successful reasoning for a question requiring a non-trivial deduction. Although this architecture is designed to work as vocal assistant, one can alike verify the reasoning by asserting manually the same belief STT asserted by the Sensor Instance as shown in Listing 2 in the Appendix. In light of this, after each assertion (lines 1, 8, 13, 18, 30) the new asserted clauses are shown, and it appears clear how the agent *expands* the Clauses KB considering generalizations and argument substitutions. After the query is given (line 45), is shown how the nominal Backward-Chaining algorithm is not enough for achieving a successful reasoning, while it happens using the Nested Reasoning.

In Section 3.3 we have also shown how a direct command or routine can be subordinated by a clause. Although in the example (see lines 12-13 of Listing 1 in the Appendix) the production rule contains the representation of *An inhabitant is at home*, even a clause involving the Nested Reasoning might trigger such a rule; for instance, a simple toy scenario could include a facial recognizer among the domotic devices, which obtains information about known/unknown faces when someone is detected in the environment. Such a recognition could generate a clause representing (for instance) *Robert is at home*, which, combined with another clause representing *Robert is an inhabitant*, will produce the representation of *An inhabitant is at home*; the latter will trigger the production rule (related to a direct command or routine) that will turn off the alarm in the garage. This will not happen whether a thief or a domestic animal is detected, thus it provides indeed a valid example about how Beliefs KB and Clauses KB interact with each other, in a non-trivial process of deduction.

5. Conclusions and Future Work

In this paper, we have presented the design of a cognitive architecture called CASPAR able to implements agents capable of both reactive and cognitive reasoning. Nevertheless we want to mark a way towards a comprehensive strategy to make deduction on Knowledge Bases whose content is parsed directly from natural language. This architecture works by using a Knowledge Base divided into two distinct parts (Beliefs KB and Clauses KB), which can also interact with each other in decision-making processes. In particular, as long as the Clauses KB increases, its cognitive features improve due to an implicit and native capability of inferring combinatorial rules from its own Knowledge Base. Thanks to the Nested Reasoning and the Clause Conceptual Generalizations, CASPAR is able to transcend the limit of the known Backward-Chaining algorithm due to the nested semantic notation; the latter is as highly descriptive as compact. Furthermore, agents based on such an architecture are able to parse complex direct IoT commands and routines, letting the user customize with ease his own Smart Environment Interface and Sensors, with whatever Speech-to-Text engine.

As future works, we want to test CASPAR capabilites with other languages than english and evaluate other integrations, like Abductive Reasoning and Argumentations. Even chatbots applications can take advantage of this architecture's features.

Finally, we want to exploit Phidias multiagent features by implementing standardized communication protocols between agents and exploit other ontologies as well.

References

[1] V. Këpuska, G. Bohouta, Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home), in: 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), 2018, pp. 99–103.

[2] H. Jeon, H. R. Oh, I. Hwang, J. Kim, An Intelligent Dialogue Agent for the IoT Home, in: AAAI Workshops, 2016. URL: https://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/view/12596.

[3] E. V. Polyakov, M. S. Mazhanov, A. Y. Rolich, L. S. Voskov, M. V. Kachalova, S. V. Polyakov, Investigation and development of the intelligent voice assistant for the Internet of Things using machine learning, in: 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT), 2018, pp. 1–5.

[4] M. Mehrabani, S. Bangalore, B. Stern, Personalized speech recognition for Internet of Things, in: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 369–374.

[5] R. Kar, R. Haldar, Applying Chatbots to the Internet of Things: Opportunities and Architectural Elements, International Journal of Advanced Computer Science and Applications 7 (2016). URL: http://dx.doi.org/10.14569/IJACSA.2016.071119. doi:10.14569/IJACSA.2016.071119.

[6] C. J. Baby, F. A. Khan, J. N. Swathi, Home automation using IoT and a chatbot using natural language processing, in: 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), 2017, pp. 1–6.

[7] I. Kotseruba, J. K. Tsotsos, 40 years of cognitive architectures: core cognitive abilities and practical applications, Artificial Intelligence Review (2018) Rev 53, 17–94 (2020). doi:https://doi.org/10.1007/s10462-018-9646-y.

[8] D. F. Lucentini, R. R. Gudwin, A comparison among cognitive architectures: A theoretical analysis, in: 2015 Annual International Conference on Biologically Inspired Cognitive Architectures, 2015.

[9] T. Giulio, Consciousness as integrated information: A provisional manifesto, The Biological bulletin (2008) 215(3), 216–242.

[10] S. Epstein, R. Passonneau, J. Gordon, T. Ligorio, The role of knowledge and certainty in understanding for dialogue, in: AAAI Fall Symposium Series, 2011. URL: https://www.aaai.org/ocs/index.php/FSS/FSS11/paper/view/4179.

[11] V. Këpuska, G. Bohouta, Comparing speech recognition systems (microsoft api, google api and cmu sphinx), Int. Journal of Engineering Research and Application Vol. 7, Issue 3, (Part -2) (March 2017) 20–24.

[12] M. Dehghani, E. Tomai, K. Forbus, M. Klenk, An integrated reasoning approach to moral decision-making, in: Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3, AAAI'08, AAAI Press, 2008, p. 1280–1286.

[13] M. Scheutz, P. Schermerhorn, J. Kramer, D. Anderson, First Steps toward Natural Human-like HRI, Auton. Robots 22 (2007) 411–423. URL: https://doi.org/10.1007/s10514-006-9018-3. doi:10.1007/s10514-006-9018-3.

[14] D. L. Schacter, Implicit memory: history and current status, Journal of Experimental Psychology: Learning, Memory, and Cognition vol. 13, 1987 (1987) 501–518.

[15] D. Davidson, The logical form of action sentences, in: The logic of decision and action, University of Pittsburg Press, 1967, p. 81–95.

[16] L. D. Consortium, Treebank-3, 2017. URL: https://catalog.ldc.upenn.edu/LDC99T42.

[17] X. Huang, L. Deng, An Overview of Modern Speech Recognitiong, Microsoft Corporation, 2009, pp. 339–344.

[18] R. Rajan Mehla, Mamta, Automatic speech recognition: A survey, International Journal of Advanced Research in Computer Science and Electronics Engineering Volume 3, Issue 1 (January 20147) 20–24.

[19] A. D. Saliha Benkerzaz, Youssef Elmir, A study on automatic speech recognition, Journal of Information Technology Review Volume 10, Number 3 (August 2019).

[20] A. S. Jinho D. Choi, Joel Tetreault, It depends: Dependency parser comparison using a web-based evaluation tool, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015, p. 387–396.

[21] S. Anthony, J. Patrick, Dependency based logical form transformations, in: SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, 2015.

[22] ClearNLP, Clear nlp tagset, 2015. URL: https://github.com/clir/clearnlp-guidelines.

[23] B. H. Partee, Lexical Semantics and Compositionality, volume 1, Lila R. Gleitman and Mark Liberman editors, 1995, p. 311–360.

[24] C. S. Fabio D'Urso, Carmelo Fabio Longo, Programming intelligent iot systems with a python-based declarative tool, in: The Workshops of the 18th International Conference of the Italian Association for Artificial Intelligence, 2019.

[25] C. F. Longo, C. Santoro, F. F. Santoro, Meaning Extraction in a Domotic Assistant Agent Interacting by means of Natural Language, in: 28th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE, 2019.

[26] L. Fichera, F. Messina, G. Pappalardo, C. Santoro, A python framework for programming autonomous robots using a declarative approach, Sci. Comput. Program. 139 (2017) 36–55. URL: https://doi.org/10.1016/j.scico.2017.01.003. doi:10.1016/j.scico.2017.01.003.

[27] G. A. Miller, Wordnet: A lexical database for english, in: Communications of the ACM Vol. 38, No. 11: 39-41, 1995.

[28] P. N. Stuart J. Russel, Artificial Intelligence: A Modern Approach, Pearson, 2010.

[29] stanford, The stanford question answering dataset squad2.0, 2018. URL: https://rajpurkar.github.io/SQuAD-explorer/.

In this appendix, a simple instance of Smart Environment Interface is provided (Listing 1) together with an example of how Clauses Knowledge Base changes, after assertions (Listing 2).

```
# Routine conditionals management
+SENSOR(V, X, Y) >> [check_conds()]
check_conds() / (SENSOR(V, X, Y) & COND(I, V, X, Y) & ROUTINE(I, K, J, L, T)) >> [-COND(I, V, X, Y), +START_ROUTINE(I), check_conds()]
check_conds() / SENSOR(V, X, Y) >> [-SENSOR(V, X, Y)]

# Routines execution
+START_ROUTINE(I) / (COND(I, V, X, Y) & ROUTINE(I, K, J, L, T)) >> [show_line("routine not ready!")]
+START_ROUTINE(I) / ROUTINE(I, K, J, L, T) >> [-ROUTINE(I, K, J, L, T), +INTENT(K, J, L, T), +START_ROUTINE(I)]

# turn off
+INTENT(X, "light", "kitchen", T) / lemma_in_syn(X, "change_state.v.01") >> [exec_cmd("change_state.v.01", "light", "kitchen", T)]
+INTENT(X, "alarm", "garage", T) / (lemma_in_syn(X, "change_state.v.01") &
    eval_cls("At_IN(Be_VBP(Inhabitant_NN(x1), __), Home_NN(x2))")) >> [exec("change_state.v.01", "alarm", "garage", T)]

# any other commands
+INTENT(V, X, L, T) >> [show_line("Result: failed to execute the command: ", V)]
```

Listing 1: A simple instance of Smart Environment Interface

```
> +STT("Nono is an hostile nation")

Be(Nono(x1), Nation(x2))
Be(Nono(x1), Hostile(Nation(x2)))
Nono(x) ==> Nation(x)
Nono(x) ==> Hostile(Nation(x))

> +STT("Colonel West is American")

Be(Colonel_West(x1), American(x2))
Colonel_West(x) ==> American(x))

> +STT("missiles are weapons")

Be(Missile(x1), Weapon(x2))
Missile(x) ==> Weapon(x)

> +STT("Colonel West sells missiles to Nono")

Sell(Colonel_West(x1), Missile(x2)) ==> Sell(American(v_0), Missile(x4))
Sell(Colonel_West(x1), Missile(x2)) ==> Sell(American(x3), Weapon(v_1))
Sell(Colonel_West(x1), Missile(x2)) ==> Sell(Colonel_West(x1), Weapon(v_2))
Sell(Colonel_West(x1), Missile(x2))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(Colonel_West(x1), Missile(x2)), Nation(v_4))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(American(v_5), Missile(v_6)), Nation(v_4))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(American(v_7), Weapon(v_8)), Nation(v_4))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(Colonel_West(v_9), Weapon(v_10)), Nation(v_4))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(Colonel_West(x1), Missile(x2)), Hostile(Nation(v_11))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(American(v_12), Missile(v_13)), Hostile(Nation(v_11))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(American(v_14), Weapon(v_15)), Hostile(Nation(v_11))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(Colonel_West(v_16), Weapon(v_17)), Hostile(Nation(v_11))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(American(v_18), Missile(v_19)), Nono(x3))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(American(v_22), Weapon(v_23)), Nono(x3))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3)) ==> To(Sell(Colonel_West(v_26), Weapon(v_27)), Nono(x3))
To(Sell(Colonel_West(x1), Missile(x2)), Nono(x3))

>+STT("When an American sells weapons to a hostile nation, that American is a criminal")

To(Sell(American(x1), Weapon(x2)), Hostile(Nation(x3))) ==> Be(American(x4), Criminal(x5))

>+STT("reason")

Waiting for query...

> +STT("Colonel West is a criminal")

Reasoning..............

Query: Be_VBZ(Colonel_West(x1), Criminal(x2))

 ---- NOMINAL REASONING ---

Result: False

 ---- NESTED REASONING ---

Result:  {v_211: v_121, v_212: x2, v_272: v_208, v_273: v_209, v_274: v_210, v_358: v_269, v_359: v_270, v_360: v_271}
```

Listing 2: CASPAR Clauses Knowledge Base changes and reasoning, after assertions

2P-Kt: logic programming with objects & functions in Kotlin

Giovanni Ciatto[a], Roberta Calegari[b], Enrico Siboni[c], Enrico Denti[a] and Andrea Omicini[a]

[a]*Dipartimento di Informatica – Scienza e Ingegneria (DISI), Alma Mater Studiorum—Università di Bologna, Italy*
[b]*Alma Mater Research Institute for Human-Centered Artificial Intelligence, Alma Mater Studiorum—Università di Bologna, Italy*
[c]*University of Applied Sciences and Arts of Western Switzerland (HES-SO), Sierre, Switzerland*

Abstract

Mainstream programming languages nowadays tends to be more and more *multi-paradigm* ones, by integrating diverse programming paradigms—e.g., object-oriented programming (OOP) and functional programming (FP). Logic-programming (LP) is a successful paradigm that has contributed to many relevant results in the areas of symbolic AI and multi-agent systems, among the others. Whereas Prolog, the most successful LP language, is typically integrated with mainstream languages via foreign language interfaces, in this paper we propose an alternative approach based on the notion of *domain-specific language* (DSL), which makes LP available to OOP programmers straightforwardly within their OO language of choice. In particular, we present a *Kotlin DSL* for Prolog, showing how the Kotlin multi-paradigm (OOP+FP) language can be enriched with LP in a straightforward and effective way. Since it is based on the interoperable 2P-Kt project, our technique also enables the creation of similar DSL on top of other high-level languages such as Scala or JavaScript—thus paving the way towards a more general adoption of LP in general-purpose programming environments.

Keywords

logic programming, object-oriented programming, multi-paradigm languages, domain-specific languages, Kotlin

1. Introduction

Logic Programming (LP) [1, 2] is a programming paradigm based on formal logic, inspired to the idea of declaratively specifying a program semantics via logic formulæ, so that automatic reasoners can then prove such formulæ by leveraging on different *control* strategies. In the years, LP has contributed to the development of a number of diverse research fields laying under the umbrella of symbolic AI—such as automatic theorem proving, multi-agent systems, model checking, research optimisation, natural language processing, etc.

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
giovanni.ciatto@unibo.it (G. Ciatto); roberta.calegari@unibo.it (R. Calegari); enrico.siboni@hevs.ch (E. Siboni); enrico.denti@unibo.it (E. Denti); andrea.omicini@unibo.it (A. Omicini)
https://about.me/gciatto (G. Ciatto); http://robertacalegari.apice.unibo.it (R. Calegari); http://enricodenti.apice.unibo.it (E. Denti); http://andreaomicini.apice.unibo.it (A. Omicini)
0000-0002-1841-8996 (G. Ciatto); 0000-0003-3794-2942 (R. Calegari); 0000-0003-3584-8637 (E. Siboni); 0000-0003-1687-8793 (E. Denti); 0000-0002-6655-3869 (A. Omicini)

Nowadays, LP is one of the major programming paradigms available for software development, along with the imperative, functional, and object-oriented ones. In particular, LP is today one of the best-suited choices for tackling problems involving knowledge representation, logic inference, automated reasoning, search in a discrete space, or meta-programming [3]. Moreover, today LP supports the core of AI components in pervasive and distributed systems, providing intelligence where and when it is needed [4].

This is why the *integration* of LP within the main programming languages is nowadays more interesting than ever. However, to make it actually work, integration should be designed – from a linguistic standpoint – so as to reduce both the development time and the learning curve of developers, as well as the psychological and cultural resistances against the adoption of new paradigms—thus, basically, moving LP up to a manageable level by the OOP developer.

Most mainstream programming languages – such as Java, Kotlin, Scala, Python, JavaScript, C# – have recognised the added value of the integration of diverse programming paradigms under a unique syntax, a coherent API, and a rich standard library. In fact, they all already support both the object-oriented (OOP) and functional (FP) programming paradigms. We believe that the same languages would largely benefit from extending their support to LP languages as well, making them usable in the OOP context in the same way as FP features already are.

Interoperability among Prolog [5] (as the first and most prominent LP language) with other languages from different paradigms is not new: the Prolog community has actually been studying this theme for years [6], as shown by the many forms of integration historically present in most Prolog systems, providing either *(i)* logic-to-OOP interoperability, making object orientation available within Prolog scripts, or *(ii)* OOP-to-logic interoperability, making logic programming exploitable within object-oriented code—or both, as in the case of tuProlog Java Library [7]. It is worth noting here that existing integrations make quite strong assumptions. For instance, item *(i)* assumes the main application is written in Prolog and the interoperation is given via a suitable interface allowing the injection of (small) parts written in other languages, while item *(ii)* implicitly assumes LP and Prolog to be somehow *harmonised* with the language(s) hosting them, at the paradigm, syntactical, and technological levels. Both these assumptions can actually create a barrier against an effective adoption of LP in today applications despite its huge potential.

Therefore, the approach adopted in this work is to devise instead a form of LP-FP-OOP *blended integration*, such that the key features of LP paradigm can be exposed and made available to mainstream language developers in a way that is the most natural in that context. The key requirement, therefore, is the "making LP easy to adopt for OOP programmers", in order to break down the learning curve for non-experts, and pursuit the typical developers' mindset. Going beyond the mere interoperability intended as simple technical habilitation, our approach is meant to embrace the perspective of the OOP developer aiming at exploiting the potential of LP.

To this end, we show how designing Prolog as a *domain-specific language* (DSL) for an OOP language such as Kotlin can make LP close enough to the OOP developers' mindset to overcome most cultural barriers, while at the same mimicking the Prolog syntax and semantics close enough to make its use straightforward for LP developers.

Generally speaking, the integration of Prolog with other paradigms aims at bringing the effectiveness and declarativeness of LP into general-purpose OOP framework. In particular, OOP designers and programmers can be expected to benefit from LP features for *(i)* data-driven or data-intensive computations, such as in the case of complex-event-processing frameworks; *(ii)* operation research or symbolic AI algorithms, or more generally other algorithms involving a search space to be explored; *(iii)* multi-agent systems, and the many areas in that field leveraging on LP, such as knowledge representation, argumentation, normative systems, etc.

The proposed solution is based on the 2P-Kt technology [8], a re-engineering of the tuProlog project [9, 10] as a Kotlin multi-platform library supporting the JVM, JS, Android, and Native platforms. The case of a Kotlin-based Prolog DSL is presented and discussed.

Accordingly, the paper is organised as follows. Section 2 briefly introduces LP, providing details about tuProlog, 2P-Kt, and Kotlin as well. It also summarises the current state of the integration between LP and mainstream programming languages. Section 3 discusses the rationale, design, and architecture of our Prolog-Kotlin DSL along with some examples. Section 4, briefly illustrates a case study where our DSL is used in combination with functional programming to solve a simple AI task in an elegant way. Finally, in section 5, concludes the paper by providing some insights about the possible future research directions stemming from our work.

2. State of the Art

2.1. Logic programming

Logic programming (LP) is a programming paradigm based on computational logic [11, 12]. In LP languages, a program is typically a set of logical relations, that is, a set of sentences in logical form, expressing *facts* and *rules* about some domain. In Prolog [5] , in particular, both facts and rules are written in the form of *clauses*, i.e. H `:-` B_1, ..., B_n, which are read declaratively as logical implications (H is true if all B_i are true). While both H and B_i are *atomic* formulæ, H is called the *head* of the rule and $(B_1, ..., B_n)$ is called the *body*. Facts are rules without body, and they are written in the simplified form: H.

An atomic formula is an expression in the form $P(t_1, ..., t_m)$ where P is a predicate having m arguments, $m \geq 0$, and $t_1, ..., t_m$ are *terms*. Terms are the most general sort of data structure used in Prolog. A term is either a constant (either a number or an atom), a variable, or a (possibly recursive) function of terms. *Variables* are named placeholders which may occur within clauses to reference (yet) unknown objects. When reading clauses declaratively, variables are assumed to be universally quantified if occurring into a rule head, existentially quantified if in the body.

A logic program is executed by an inference engine that answers a query by trying to prove it is *inferable* from the available facts and rules. In particular, the Prolog interpreter exploits a *deduction* method based on the SLD resolution principle introduced in [2]. The SLD principle, in turn, heavily leverages on the unification algorithm [13] for constructing a most general unifier (MGU) for any two suitable terms. Provided that such a MGU exists, the subsequent application of the resulting substitution to the terms, renders them syntactically equal. These two mechanisms – resolution and unification – constitute the basis of any Prolog solver. In particular, SLDNF is an extension of SLD resolution for dealing with *negation as failure* [14]. In SLDNF, goal clauses can contain negation as failure literals—i.e. the form $not(p)$.

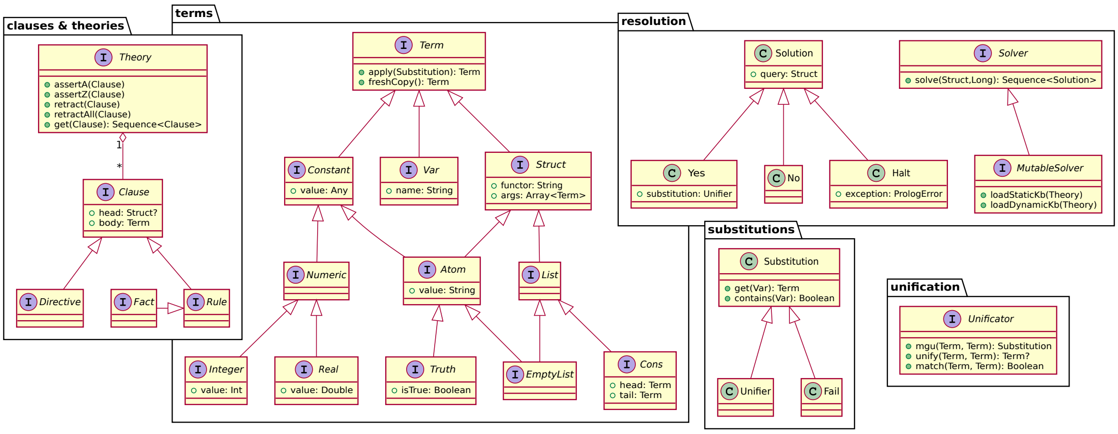


Figure 1: Overview on the public API of 2P-Kt: a type is provided for each relevant concept in LP

2.2. tuProlog and 2P-Kt

tuProlog [10] (2P for short) is a logic programming framework supporting multi-paradigm programming via a clean, seamless, and bidirectional integration between the logic and object-oriented paradigms. 2P-Kt is a Kotlin project for LP, representing a complete rewriting of the tuProlog project, whose ultimate purpose is to provide an open LP ecosystem supporting multiple platforms and programming paradigms.

More importantly, 2P-Kt includes a rich API providing many logic-programming facilities to developers. For instance, it enables the exploitation of clauses and terms to represent symbolic knowledge via FOL in Kotlin. Clauses and terms can be manipulated via logic unification, possibly producing substitutions to be applied to other clauses and terms. Logic theories can be created as indexed and ordered sequences of clauses, enabling the exploitation of classical `assert` or `retract` operations – as well as bare clause retrieval via pattern-matching – in an efficient way. Finally, the full potential of Prolog SLDNF resolution strategy can be exploited, if needed, via lightweight solvers which could be instantiated and queried on the fly, by lazily consuming their solutions.

Accordingly, the 2P-Kt API – sketched in fig. 1 – exposes a Kotlin type for each relevant LP concept: *(i)* logic `Terms` (as well as for each particular sort of term, e.g. `Variables`, or `Structures`, etc.), *(ii)* logic `Substitutions` and `Unification`, *(iii)* Horn `Clauses` (there including `Rules`, `Facts`, and `Directives`), *(iv)* knowledge bases and logic theories (e.g. the `Theory` type), and *(v)* automatic reasoning via Prolog's SLDNF resolution strategy, as provided by the `Solver` interface.

2.3. Prolog integration

In order to integrate Prolog with high-level languages, many Prolog implementations expose a foreign language interface (FLI, table 1). In most cases, the target language is Java, and the FLI is bi-directional—meaning that it supports both "calling Prolog from the target language" and vice versa. In a few particular cases, however, the JavaScript and C# languages are also supported.

Table 1
Prolog implementations and their foreign language interfaces (FLI)

Prolog Implementation	Platform	FLI towards	Nature of FLI	Paradigm of FLI	Source
BProlog [15]	C	Java	JNI	imperative	Official BProlog doc.
Ciao! [16]	C	Java	TCP/IP	imperative	Official Ciao Prolog doc.
ECLiPSe [17]	C	Java	JNI	imperative	Official ECLiPSe doc.
SICStus [18]	C	Java, C#	TCP/IP	object-oriented	Jasper Library
SWI [19]	C	Java	JNI	object-oriented	JPL API
τProlog [20]	JS	JavaScript	Native	object-oriented	Project homepage
tuProlog [9]	JS, JVM Android	Kotlin, Java, JavaScript	Native	object-oriented, functional	Project homepage
XSB [21]	C	Java	TCP/IP	imperative	Interprolog Java Bridge

As we are mostly interested in discussing how and to what extent Prolog exploitation is possible within the target languages, in the reminder of this section we only focus on those FLI allowing to "call Prolog from the target language".

Each FLI mentioned in table 1 is characterised by a number of features that heavily impact the way the users of the target languages may actually exploit Prolog. These are, for instance, the nature of the FLI – which is tightly related to the target platform of the underlying Prolog implementation – and its reference programming paradigm.

By "nature" of the FLI, we mean the technological mechanism exploited by a particular Prolog implementation to interact with the target language. There are some cases – like tuProlog and τProlog – where this aspect is trivial, because the target language is also the implementation language—so Prolog is considered there as just another library for the target language. More commonly, however, Prolog is implemented in C, and the Java Native Interface (JNI) is exploited to make it callable from Java. This is for instance the case of SWI- and ECLiPSe-Prolog. While this solution is very efficient, it hinders the portability of Prolog into particular contexts such as Android, and its exploitation within mobile applications.

Another viable solution is to leverage on the TCP/IP protocol stack. This is for instance the case of SICStus- and Ciao-Prolog. The general idea behind this approach is that the Prolog implementation acts as a remote server offering logic-programming services to the target language via TCP/IP, provided that a client library exists on the Java side making the exploitation of TCP/IP transparent to the users. While this solution is more portable – as virtually any sort of device supports TCP/IP –, it raises efficiency issues because of the overhead due to network/inter-process communications. A more general solution of that sort is instead offered by the LPaaS architecture [22], where the fundamental idea is to have many Prolog engines distributed over the network, accessed as logic-based web services. By the way, the implementation of LPaaS[1] is based on tuProlog.

By "reference programming paradigm" of the FLI we mean the specific programming style proposed by a Prolog implementation to the target language users through its API. Some FLI are conceived to be used in a strictly imperative way: this is e.g. the case of BProlog or Ciao! Prolog, where a Java API is available for writing Prolog queries and issue them towards the underlying Prolog system in an imperative way. Other Prolog implementations, such as SICStus- and

[1] http://lpaas.apice.unibo.it

SWI-Prolog, offer a more object-oriented API which let developer not only represent queries but also terms, and solutions (there including variable bindings) which can be consumed through ordinary OOP mechanisms such as iterators.

In most cases, however, the code to be produced in the target language is far less concise and compact than pure Prolog – unless strings and parsing are extensively adopted –, especially when the size of the Prolog code to be represented grows. So, for instance, the simple Prolog query `?- parent(adam, X)` (aimed at computing who Adam's son is) in Java through SWI-Prolog's JPL interface would be written as:

```
Query query = new Query("parent", new Term[] { new Atom("adam"), new
 ↪  Variable("X") } );
Map<String,Term> solution = query.oneSolution();
System.out.println("The child of Adam is " + solution.get("X"));
```

While this a totally effective solution on the technical level, we argue that a more convenient integration among LP and the other paradigms is possible. In particular, in this paper we show how 2P-Kt allows for a finer integration at the paradigm level through *domain-specific languages.*

2.4. Kotlin Domain-Specific Languages (DSL)

Domain-specific languages (DSL) are a common way to tackle recurrent problems in a general way via software engineering. When a software artefact (e.g. library, architecture, system) is available to tackle with some sort of problems, designers may provide a DSL for that artefact to ease its usage for practitioners. There, the exploitation of a DSL hides the complexity of the library behind the scenes, while supporting the usage of the artefact for non-expert users too. This is, for instance, the approach of the Gradle build system[2]—which is nowadays one of the most successful build automation systems for the JVM, Android, and C/C++ platforms. It is also the approach followed by the JADE agent programming framework, which is nowadays usable through the Jadescript DSL [23].

Building a DSL usually requires the definition of a concrete syntax, the design and implementation of a compiler or code generator, and the creation of an ecosystem of tools—e.g., syntax highlighters, syntax checkers, debuggers. In particular, compilation or code generation are fundamental as they are what makes a DSL machine-interpretable and -executable.

Some high-level languages, however, such as Kotlin, Groovy, or Scala, enable a different approach. They come with a flexible syntax which *natively* supports the definition of new DSL with no addition to the hosting language required. For instance, Gradle consists of a JVM library of methods for building, testing, and deploying sources codes, plus a Kotlin- or Groovy-based DSL allowing developers to customise their particular workflows.

The advantages of this approach are manifold. First, no compiler or code generator has to be built, as the DSL is already part of the hosting language and it is therefore machine-interpretable and -executable by construction. Second, the DSL automatically inherits all the constructs,

[2]https://gradle.org

API, and libraries of the hosting language—there including conditional or iterative constructs, string manipulation API, etc., which are common and useful for many DSL, regardless of their particular domain. Third, the ecosystem of tools supporting the hosting language – e.g. compilers, debuggers, formatters, code analysers, IDE, etc. – can be reused for the DSL as well, easing its adoption and making it more valuable.

When Kotlin is the hosting language of choice, DSL leverage on a small set of features making the Kotlin syntax very flexible, described below:

operator overloading[3] — allowing ordinary arithmetic, comparison, access, and function invocation operators to change their ordinary meaning on a per-type basis;

block-like lambda expressions[4] — including a number of syntactic sugar options such as *(i)* the possibility to omit formal parameters in case of a single-argument lambda expression, and *(ii)* the possibility to omit the round parentheses in case of a function invocation having a lambda expression as a last argument;

function types/literals with receiver[4] — allowing functions and methods to accept lambda expressions within which the `this` variable references a different object than the outer scope;

extension methods[5] — allowing pre-existing types to be extended with new instance methods whose visibility is scope-sensible.

Of course, Kotlin-based DSL automatically inherit the full gamma of facilities exposed by the Kotlin language and standard library—there including support for imperative, and object-oriented programming, as well as a rich API supporting functional programming through most common high-order operations. Furthermore, if properly engineered, these DSL may be executed on all the platforms supported by Kotlin—which commonly include, at least, the JVM, JavaScript, and Android. This is for instance the case of our DSL proposed in section 3.

While this paper focuses on Kotlin-based DSL, similar results could be achieved in other languages by means of equivalent mechanisms. For instance, in Scala, extension methods have to be emulated via implicit classes, and DSL, in particular, can be built via the "Pimp My Library" pattern [24].

3. A domain-specific language for LP

3.1. Design Rationale

Regardless of the technological choices, the design of our DSL leverages on a small set of principles ($\mathbf{P_i}$) briefly discussed below. In fact, our aim is to ($\mathbf{P_1}$) provide a DSL that is a *strict* extension of its hosting language, meaning that no feature of the latter language is prevented by the usage of our DSL. Dually, we require ($\mathbf{P_2}$) our DSL to be fully interoperable and finely integrated with the hosting language, meaning that all the features of the latter language can be

[3] https://kotlinlang.org/docs/reference/operator-overloading.html
[4] https://kotlinlang.org/docs/reference/lambdas.html
[5] https://kotlinlang.org/docs/reference/extensions.html

exploited from within our DSL. However, we also require ($\mathbf{P_3}$) the DSL to be well encapsulated and clearly identifiable within the hosting language, in order to prevent unintended usage of the DSL itself. Finally, we require ($\mathbf{P_4}$) our DSL to be as close as possible to Prolog, both at the syntactic and semantic level, to ease its exploitation for logic programmers.

In order to accomplish to the aforementioned principles, we choose the Kotlin language as the technological reference for prototyping our proposal. This is because it *(i)* comes with a flexible syntax supporting the definition of DSL, *(ii)* supports a considerable number of platforms (JVM, JavaScript, Android, Native) and therefore enables a wide exploitation of LP – for instance, on smart devices –, *(iii)* includes a number of libraries supporting LP-based applications, such as 2P-Kt.

3.2. The Kotlin DSL for Prolog

Before dwelling into the details of our proposal, we provide an overview of what a Kotlin DSL for Prolog has to offer.

Let us consider the following Prolog theory describing a portion of Abraham's family tree:

```
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).

parent(abraham, isaac).
parent(isaac, jacob).
parent(jacob, joseph).
```

It enables a number of queries, e.g. `ancestor(abraham, X)`, which can be read as "Does there exist some `X` which is a descendant of Abraham?". According to the Prolog semantics, this query may have a number of solutions, enumerating all the possible descendants of Abraham that can be deduced from the above theory—i.e., Isaac, Jacob, and Joseph.

The same result may be attained through the following Kotlin program, which leverages on our DSL for Prolog:

```
prolog {
  staticKb(
    rule {
      "ancestor"("X", "Y") `if` "parent"("X", "Y")
    },
    rule {
      "ancestor"("X", "Y") `if` ("parent"("X", "Z") and
       ↪  "ancestor"("Z", "Y"))
    },
    fact { "parent"("abraham", "isaac") },
    fact { "parent"("isaac", "jacob") },
    fact { "parent"("jacob", "joseph") }
  )
```

```
  for (sol in solve("ancestor"("abraham", "X")))
    if (sol is Solution.Yes)
      println(sol.substitution["X"])
}
```

The program creates a Prolog solver and initialises it with standard built-in predicates. Then it loads a number of facts and rules representing the aforementioned Prolog theory about Abraham's family tree. Finally, it exploits the solver to find all the descendants of Abraham, by issuing the query `ancestor(abraham, X)`.

It is worth noting how the simple code snippet exemplified above is adherent w.r.t. our principles. In fact, the whole DSL is *encapsulated* ($\mathbf{P_3}$) within the

$$\texttt{prolog \{ } \langle \textit{DSL block} \rangle \texttt{ \}}$$

In there, LP facilities can be exploited in combination with the imperative and functional constructs offered by the Kotlin language and its standard-library ($\mathbf{P_1}$ and $\mathbf{P_2}$)—such as the `for-each`-loop used to print solutions in the snippet above. Furthermore, within `prolog` blocks, both theories and queries are expressed via a Kotlin-compliant syntax mimicking Prolog ($\mathbf{P_4}$). The main idea behind such syntax is that each expression in the form

$$\texttt{"functor"(}\langle e_1 \rangle\texttt{, } \langle e_2 \rangle\texttt{, ...)}$$

is interpreted as a compound term (a.k.a. structure) in the form $\texttt{functor}(t_1, t_2, \ldots)$, provided that each Kotlin expression e_i can be recursively evaluated as the term t_i—e.g. capital strings such as `"X"` are interpreted as variables, whereas non-capital strings such as `"atom"` (as well as Kotlin numbers) are interpreted as Prolog constants. In a similar way, expressions of the form[6]

$$\texttt{rule \{"head"(}\langle e_1 \rangle\texttt{, ..., } \langle e_N \rangle\texttt{) `if` (}\langle e_{N+1} \rangle \texttt{ and ... and } \langle e_M \rangle\texttt{)\}}$$

are interpreted as Prolog rules of the form

$$\texttt{head}(t_1, \ldots, t_N) \texttt{ :- } t_{N+1}, \ldots, t_M$$

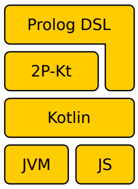
provided that $M > N$ and each Kotlin expression e_i can be recursively evaluated as the term t_i. A similar statement holds for fact expressions of the form `fact { ... }`.

To support our DSL for Prolog, a number of Kotlin classes and interfaces have been designed on top of the 2P-Kt library, exploiting manifold extensions methods, overloaded operators, etc. to provide the syntax described so far. The details of our solution – there including its architecture, design, and implementation – are discussed in the reminder of this section.

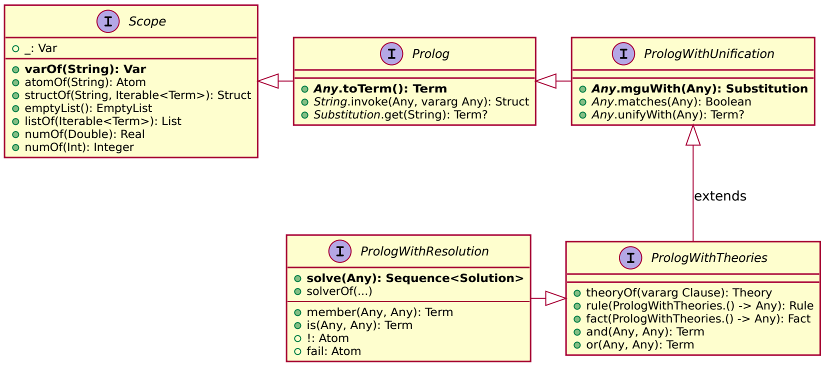
3.3. Architecture, Design, Implementation

The Kotlin DSL for Prolog is essentially a compact means to instantiate and use objects from the 2P-Kt library, through a Prolog-like syntax. More precisely, it consists of a small software layer

[6] backticks make Kotlin parse words as identifiers instead of keywords



(a) Layered view showing the modules of 2P-KT and our Kotlin DSL for Prolog



(b) Structural view showing the API of our Kotlin DSL for Prolog, leveraging on the 2P-KT API

Figure 2: Architectural view of our Kotlin DSL for Prolog

built on top of Kotlin and 2P-KT, as represented by fig. 2a. In particular, the DSL is enabled by the five *factory interfaces* depicted in fig. 2b. We call *factory interface* a type definition whose methods are aimed at *instantiating* objects of related types, as dictated by the Gang of Four' abstract factory pattern [25].

The five factory interfaces are: `Scope`, `Prolog`, `PrologWithUnification`, `PrologWithTheories`, and `PrologWithResolution`. Each factory type extends the previous one with more LP-related functionalities. So, for instance, while instances of `Scope` simply provide the basic bricks to create logic terms, instances of `Prolog` leverage on these bricks to enable the exploitation of the Prolog-like syntax exemplified above. `PrologWithUnification` extends `Prolog` with unification-related facilities, and it is in turn extended by `PrologWithTheories`, which lets developers create both clauses and theories via a Prolog-like syntax. Finally `PrologWithResolution` extends `PrologWithTheories` by adding resolution-related facilities, plus some syntactic shortcuts for writing rules exploiting Prolog standard predicates such as `member/2`, `length/1`, etc.

In the following we provide further details about how each factory contributes to our DSL.

Scope The simplest factory type is `Scope`. As suggested by its name, it is aimed at building terms which must share one or more variables. For this reason, it exposes a number of factory methods – roughly, one for each sub-type of `Term` –, some of which are mentioned in fig. 2b. The main purpose of a scope, however, is to enable the creation of objects reusing the same logic `Variables` more than once. Thus, it includes a method – namely, `varOf(String)` – which always returns the same variable if the same name is provided as input.

For example, to create the term `member(X, [X | T])`, one may write:

```
val m = Scope.empty {
  structOf("member", varOf("X"), consOf(varOf("X"), varOf("T")))
} // m references the term member(X, [X | T])
```

There, the two calls to `varOf("X")` actually return the same object, if occurring within the same scope—whereas they would return different variables if invoked on different scopes. This mechanism is what enables developers to instantiate terms and clauses without having to explicitly refresh variables among different clauses: it is sufficient to create them within different scopes.

Prolog The `Prolog` factory type extends `Scope` by adding the capability of creating terms through a Prolog-like syntax. To do so, it exposes a number of extension methods aimed at automatically converting Kotlin objects into Prolog terms or using Kotlin objects in place of Prolog terms. The most relevant extension methods are mentioned in fig. 2b. These methods are:

- `fun Any.toTerm(): Term`, which is an extension method aimed at making any Kotlin object convertible into a Prolog term, through the syntax `obj.toTerm()`. To convert an object into a term, it leverages on the following type mapping: *(i)* a Kotlin number is either converted into a Prolog real or integer number, depending on whether the input number is floating-point or not, *(ii)* a Kotlin string is either converted into a Prolog variable or atom, depending on whether the input string starts with a capital letter or not, *(iii)* a Kotlin boolean is always converted into a Prolog atom, *(iv)* a Kotlin iterable (be it an array, a list, or any other collection) is always converted into a Prolog list, provided that each item can be recursively converted into a term, *(v)* a Kotlin object remains unaffected if it is already an instance of `Term`, *(vi)* an error is raised if the input object cannot be converted into a `Term`.

- `operator fun String.invoke(vararg Any): Struct`, which is an extension method aimed at overloading the function invocation operator for strings. Its purpose is to enable the construction of compound terms through the syntax `"f"(arg`$_1$`, ..., arg`$_N$`)`, which mimics Prolog. Its semantics is straightforward: assuming that each `arg`$_i$ can be converted into a term via the `Any.toTerm()` extension method above, this method creates a N-ary `Structure` whose functor is `"f"` and whose i^{th} is `arg`$_i$`.toTerm()`. So, for instance, the expression

  ```
  "member"("X", arrayOf(1, true))
  ```

  creates the Prolog term `member(X, [1, true])`.

- `fun Substitution.get(String): Term?`, which is an extension method aimed at overloading the `get` method of the `Substitution` type in such a way that is can also accept a string other than a `Variable`. This enables DSL users to write expressions such as

  ```
  substitution.get("X")
  ```

  instead of having to create a variable explicitly via `varOf("X")`. While we only discuss this method, the main idea here is that every method in the 2P-KT API accepting some basic Prolog type – such as `Var`, `Atom`, or `Real` – as argument should be similarly overloaded to accept the corresponding Kotlin type as well—e.g. `String` or `Double`. This is what enables a fine-grained integration of our DSL with the Kotlin language and the 2P-KT library.

It is worth highlighting that every `Prolog` object is also a particular sort of `Scope`. So, converting the same string into a `Variable` twice or more times, within the same `Prolog` object, always yields the exact same variable.

Prolog with unification The `PrologWithUnification` factory type extends `Prolog` by adding the capability of *(i)* computing the most general unifier (MGU) among two terms, *(ii)* checking whether two terms match or not according to logic unification – i.e., checking if a MGU exists unifying the two terms –, and *(iii)* computing the term attained by unifying two terms—assuming an MGU exists for them. To do so, it exposes a number of extension methods aimed at providing unification-related support to Kotlin objects, provided that they can be converted into terms. The most relevant extension methods are mentioned in fig. 2b.

Prolog with theories The `PrologWithTheories` factory type extends `PrologWithUnification` by adding the capability of creating logic clauses (e.g. rules and facts) and theories. To do so, it exposes a number of methods aimed supporting the conversion of Kotlin objects into Prolog clauses – through the syntactic facilities presented so far –, and their combination into theories. The most relevant methods, mentioned in fig. 2b, are the following:

- `fun theoryOf(vararg Clause): Theory` , an ordinary method aimed at creating a logic `Theory` out of a variable amount of clauses.
- `infix fun Any.`if`(Any): Rule` , which is an extension method aimed at creating logic rules via a Prolog-like syntax in the form `head `if` body` . Its semantics is straightforward: assuming that both `head` and `body` can be converted into logic goals via the `Any.toTerm()` extension method above, this method creates a binary `Structure` whose functor is '`:-`' and whose arguments are `head.toTerm()` and `body.toTerm()`. Similar methods exists – namely, `and`, `or`, etc. – to create conjunctions or disjunctions of clauses.
- `fun rule(PrologWithTheories.() -> Any): Rule` , an ordinary method aimed at creating a rule in a separate scope, thus avoiding the risk of accidentally referencing the variables created elsewhere. It creates a fresh, empty, and nested instance of `PrologWithTheories` and accepts a function with receiver to be invoked on that nested instance. The function is expected to return a Kotlin object which can be converted into a Prolog rule. Any variable possibly created within the nested scope is guaranteed to be different than any homonymous variable defined elsewhere.
- `fun fact(PrologWithTheories.() -> Any): Fact` , analogous to the previous method, except that it is aimed at creating Prolog facts. So, for instance, one may write

```
val r1 = fact {
    "member"("X", consOf("X", `_`))
}
val r2 = rule {
```

```
        "member"("X", consOf(`_`, "T")) `if` "member"("X", "T")
    }
```

while being sure that the X variable used in r1 is different than the one used in r2.

Prolog with resolution The PrologWithResolution factory type extends PrologWithUnification by adding the capability of performing logic queries and consuming their solutions attained through the standard Prolog semantics. To do so, it exposes a number of methods aimed at supporting *(i)* the instantiation of Prolog Solvers, *(ii)* the loading of Prolog theories, either as static or dynamic knowledge bases (KB), and *(iii)* the invocation of Prolog queries on those KB. The most relevant methods, mentioned in fig. 2b, are the following:

- `fun solve(Any, Long): Sequence<Solution>` , which is an ordinary method aimed at executing Prolog queries without requiring a new solver to be explicitly created. It accept a Kotlin object as argument – which must be convertible into a Prolog query –, and an optional timeout limiting the total amount of time the solver may exploit to compute a solution. If the provided arguments are well formed, this method returns a Sequence of Solutions which *lazily* enumerates all the possible answers to the query provided as input, using Prolog's SLDNF proof procedure.

- `fun staticKb(vararg Clause)` , which is an ordinary method aimed at loading the *static* KB the solve method above will operate upon.

- `fun dynamicKb(vararg Clause)` , which is analogous to the previous method, except that it loads the *dynamic* KB the solve method above will operate upon.

- `fun member(Any, Any): Struct` which is an ordinary method aimed at easily creating invocations of the member/2 built-in predicate. While we only discuss this method, the main idea here is that every standard built-in predicate in Prolog has a Kotlin counterpart in PrologWithResolution accepting the proper amount or arguments and returning an invocation to that built-in predicate. This is aimed easing the exploitation of the standard Prolog predicates to developers leveraging our DSL. So, for instance, we also provide facility methods for built-in such as is/2, +/2, -/2, !/0, fail/0, append/3, etc.

Instances of PrologWithResolution are created and used via the

```
fun <R> prolog(PrologWithResolution.() -> R): R
```

static method, which accepts a lambda expression letting the user exploit the DSL on the fly, and returns the object created by this lambda expression. This is for instance what enables developers to write the code snippet discussed in section 3.2.

4. Case study: N-Queens

We present now a brief example demonstrating how, by integrating multiple programming paradigms, developers may easily produce compact and effective solutions. Suppose one is willing to implement the following Kotlin method

```
fun nQueens(n: Int): Sequence<List<Position>>
```

aimed at computing all the possible solutions to the N-Queens problem. More precisely, the method is expected to enumerate all the possible dispositions of n queens on a $n \times n$ chessboard, such that no queen can be attacked by others. Each solution can be represented by a list of queen positions, in the form $[(1, Y_1), \ldots, (n, Y_n)]$, where each Y_i represent the row occupied by the ith queen—i.e., the one in column i.

Computing all solutions for the N-Queens problem may require a lot of code in most programming paradigms. However, in LP, the solution is straightforward and compact. One may, for instance, leverage the following Prolog code:

```
no_attack((X1, Y1), (X2, Y2)) :-
  X1 =\= X2, % infix operator
  Y1 =\= Y2,
  (Y2 - Y1) =\= (X2 - X1),
  (Y2 - Y1) =\= (X1 - X2). % arithmetic expression

no_attack_all(_, []).
no_attack_all(C , [H | Hs]) :-
  no_attack(C, H),
  no_attack_all(C, Hs).

solution(_, []).
solution(N, [(X, Y) | Cs]) :-
  solution(N, Cs),
  between(1, N, Y), % built-in predicate
  no_attack_all((X, Y), Cs).
```

which can satisfy queries in the form

```
?- solution(N, [(1, Y1), ..., (N, YN)])
```

(provided that some actual N is given) by instantiating each variable `Yi`.

Thanks to our Kotlin DSL for Prolog, one may exploit the elegance of LP in implementing the aforementioned `nQueens` method. For instance, one may implement it as follows:

```
fun nQueens(n: Int) = prolog {
  staticKb(
    rule {
```

```
      "no_attack"(("X1" and "Y1"), ("X2" and "Y2")) `if` (
          ("X1" `=!=` "X2") and // infix operator
          ("Y1" `=!=` "Y2") and
          (("Y2" - "Y1") `=!=` ("X2" - "X1")) and
          (("Y2" - "Y1") `=!=` ("X1" - "X2")) // arithmetic expr
        )
    },
    fact { "no_attack_all"(`_`, emptyList) },
    rule {
      "no_attack_all"("C", consOf("H", "Hs")) `if` (
          "no_attack"("C", "H") and
          "no_attack_all"("C", "Hs")
        )
    },
    fact { "solution"(`_`, emptyList) },
    rule {
      "solution"("N", consOf(("X" and "Y"), "Cs")) `if` (
          "solution"("N", "Cs") and
          between(1, "N", "Y") and // built-in predicate
          "no_attack_all"(("X" and "Y"), "Cs")
        )
    }
  )
  return solve("solution"(n, (1 .. n).map { it and "Y$it" }))
}
```

This implementation produces a *lazy* stream of solutions to the N-Queens problem, given a particular value of n. In doing so, it takes advantages not only of logic- but also of functional-programming paradigm. For instance, on the last line, it exploits the `map` high-order function to build a list in the form $[(1, Y_1), \ldots, (n, Y_n)]$, to be provided as argument of `solution/2`.

A number of minutiæ may be noted as well by comparing the Prolog code with its Kotlin counterpart. For instance, Prolog operators (such as `=\=/2`, `-/2`, etc.) retain their infix notations in the Prolog DSL. This is possible because they are part of the DSL, in the same way as Prolog built-in predicates (such as `between/3`). This implies the Kotlin compiler can prevent standard operators and built-in from being silently mistyped.

5. Conclusions and future works

In this paper we propose a novel way of integrating the logic, object-oriented, functional, and imperative programming paradigms into a single language, namely Kotlin. More precisely, we describe how a domain-specific language (DSL) for Prolog can be built on top of the Kotlin language by exploiting the 2P-Kt library. The proposed solution extends the Kotlin language with LP facilities by only relying on its own mechanisms – therefore no external tool is necessary apart from Kotlin itself and 2P-Kt –, even if, however, analogous extensions can in principle be constructed for other high-level languages as well—such as Scala.

Our DSL is currently implemented as part of the 2P-Kt project – namely, within the `dsl-*` modules –, and it is hosted on both GitHub [8] and Maven Central. Other than being available to the public for general purpose usage – under the terms of the Apache License 2.0[7] open-source license –, the DSL is currently extensively exploited to implement the unit tests of 2P-Kt itself.

While in this paper we discuss the design rationale and architecture of our DSL by explicitly focusing on Kotlin as our target technology, in the future we plan to generalise our approach, possibly tending to technology independence. Furthermore, we plan to provide other implementations of our DSL, targeting other high-level languages and platforms, in order to make logic programming and its valuable features available in general-purpose programming languages and frameworks.

As concerns possible research directions and applications, in the future, we plan to exploit our DSL in several contexts. For instance, we argue our DSL may ease the usage of the logic tuple spaces offered by the TuSoW technology [26]. Similarly, we believe our DSL may be used as a basic brick in the creation of logic-based technologies for MAS, as the MAS community is eager of general-purpose, logic-based technologies targetting the JVM platform [27]. Along this line, we argue logic-based languages for MAS may benefit from the integration of our DSL – or a similar one – to support the reasoning capabilities of agents. Finally, we plan to exploit our DSL within the scope of XAI [28], to ease the creation of hybrid (i.e., logic + machine-learning) systems, and management on the symbolic side.

Acknowledgments

R. Calegari has been supported by the H2020 ERC Project "CompuLaw" (G.A. 833647). A. Omicini has been partially supported by the H2020 Project "AI4EU" (G.A. 825619).

References

[1] K. R. Apt, The Logic Programming Paradigm and Prolog, Cambridge University Press, 2001, pp. 475–508. doi:10.1017/CBO9780511804175.016.

[2] R. A. Kowalski, Predicate logic as programming language, in: J. L. Rosenfeld (Ed.), Information Processing, Proceedings of the 6th IFIP Congress, North-Holland, 1974, pp. 569–574.

[3] R. Calegari, G. Ciatto, E. Denti, A. Omicini, Logic-based technologies for intelligent systems: State of the art and perspectives, Information 11 (2020) 1–29. doi:10.3390/info11030167, special Issue "10th Anniversary of Information—Emerging Research Challenges".

[4] A. Omicini, R. Calegari, Injecting (micro)intelligence in the IoT: Logic-based approaches for (M)MAS, in: D. Lin, T. Ishida, F. Zambonelli, I. Noda (Eds.), Massively Multi-Agent Systems II, volume 11422 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 21–

[7] https://www.apache.org/licenses/LICENSE-2.0

35. doi:10.1007/978-3-030-20937-7_2, international Workshop, MMAS 2018, Stockholm, Sweden, July 14, 2018, Revised Selected Papers.
[5] A. Colmerauer, P. Roussel, The birth of prolog, in: J. A. N. Lee, J. E. Sammet (Eds.), History of Programming Languages Conference (HOPL-II), ACM, 1993, pp. 37–52. doi:10.1145/154766.155362.
[6] R. Bagnara, M. Carro, Foreign language interfaces for Prolog: A terse survey, ALP Newsletter 15 (2002). URL: https://dtai.cs.kuleuven.be/projects/ALP/newsletter/may02/index.html.
[7] E. Denti, A. Omicini, A. Ricci, Multi-paradigm Java-Prolog integration in tuprolog, Science of Computer Programming 57 (2005) 217–250. doi:10.1016/j.scico.2005.02.001.
[8] 2P-Kt, Github page, https://github.com/tuProlog/2p-kt, Last access: October 22, 2020.
[9] tuProlog, home page, http://tuprolog.unibo.it, Last access: October 22, 2020.
[10] E. Denti, A. Omicini, A. Ricci, tuProlog: A light-weight Prolog for Internet applications and infrastructures, in: I. Ramakrishnan (Ed.), Practical Aspects of Declarative Languages, volume 1990 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2001, pp. 184–198. doi:10.1007/3-540-45241-9_13, 3rd International Symposium (PADL 2001), Las Vegas, NV, USA, 11–12 March 2001. Proceedings.
[11] J. W. Lloyd (Ed.), Computational logic, Springer, 1990.
[12] A. Nerode, G. Metakides, Principles of Logic and Logic Programming, Elsevier Science Inc., USA, 1996.
[13] A. Martelli, U. Montanari, An efficient unification algorithm, ACM Transactions on Programming Languages and Systems 4 (1982) 258–282. doi:10.1145/357162.357169.
[14] K. L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), Symposium on Logic and Data Bases, Advances in Data Base Theory, Plemum Press, New York, 1977, pp. 293–322. doi:10.1007/978-1-4684-3384-5_11.
[15] BProlog, home page, http://www.picat-lang.org/bprolog, Last access: October 22, 2020.
[16] Ciao! Prolog, home page, https://ciao-lang.org, Last access: October 22, 2020.
[17] ECLiPSe Prolog, home page, https://eclipseclp.org, Last access: October 22, 2020.
[18] SICStus Prolog, home page, https://sicstus.sics.se, Last access: October 22, 2020.
[19] SWI Prolog, home page, https://www.swi-prolog.org, Last access: October 22, 2020.
[20] τProlog, home page, http://tau-prolog.org, Last access: October 22, 2020.
[21] XSB Prolog, home page, http://xsb.sourceforge.net, Last access: October 22, 2020.
[22] R. Calegari, E. Denti, S. Mariani, A. Omicini, Logic programming as a service, Theory and Practice of Logic Programming 18 (2018) 846–873. doi:10.1017/S1471068418000364, special Issue "Past and Present (and Future) of Parallel and Distributed Computation in (Constraint) Logic Programming".
[23] F. Bergenti, G. Caire, S. Monica, A. Poggi, The first twenty years of agent-based software development with JADE, Autonomous Agents and Multi Agent Systems 34 (2020) 36. doi:10.1007/s10458-020-09460-z.
[24] B. C. Oliveira, A. Moors, M. Odersky, Type classes as objects and implicits, in: Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications, OOPSLA '10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 341–360. doi:10.1145/1869459.1869489.
[25] E. Gamma, R. Helm, R. E. Johnson, J. Vlissides, Design Patterns: Elements of Reusable

Object-Oriented Software, Addison-Wesley Professional Computing Series, Addison-Wesley, Reading, MA, 1995. URL: https://www.safaribooksonline.com/library/view/design-patterns-elements/0201633612/.

[26] G. Ciatto, L. Rizzato, A. Omicini, S. Mariani, TuSoW: Tuple spaces for edge computing, in: The 28th International Conference on Computer Communications and Networks (ICCCN 2019), IEEE, Valencia, Spain, 2019, pp. 1–6. doi:10.1109/ICCCN.2019.8846916.

[27] R. Calegari, G. Ciatto, V. Mascardi, A. Omicini, Logic-based technologies for multi-agent systems: A systematic literature review, Autonomous Agents and Multi-Agent Systems 35 (2021) 1:1–1:67. doi:10.1007/s10458-020-09478-3, collection "Current Trends in Research on Software Agents and Agent-Based Software Development".

[28] R. Calegari, G. Ciatto, A. Omicini, On the integration of symbolic and sub-symbolic techniques for XAI: A survey, Intelligenza Artificiale 14 (2020) 7–32. doi:10.3233/IA-190036.

Suggesting Just Enough (Un)Crowded Routes and Destinations

Claudia Cavallaro, Gabriella Verga, Emiliano Tramontana and Orazio Muscato

Department of Mathematics and Computer Science, University of Catania, Italy

Abstract

Though people like to visit popular places, for health-related concerns and due to the recent restrictions adopted around the world, gatherings should be avoided. When planning a trip, one has to consider both attractiveness in terms of general interest for the destinations, and the density of people gathering there. In this work, we propose a recommendation system aiming at offering users some suggestions on useful routes and destinations that balance both liveliness and overcrowding. Firstly, we use datasets storing GPS positions as a basis for the statistics on routes and destinations. Then, we use an accurate probability algorithm that estimates the number of people moving from one place to another in the city and accordingly we show a list of destinations to users. The destination points are filtered based on the user's preference on the density of people. A multi-agent system is used to handle the user requests to find a route for a trip, statistics on possible destinations, and suggestions to users. Thanks to our solution we can inform users on suitable routes and destinations, as well as alert them when a preferred destination is overcrowded.

Keywords

GPS trajectory, Recommendation systems, Movement predictions, Multi-agent system

1. Introduction

Currently, organising a trip should take into account the number of people that will gather in the chosen destination points, since it is necessary to avoid visiting a place that will become overcrowded to comply with the restrictions due to the Covid-19 influenza pandemic. Hence, an estimate of the number of people that will be in some place in a future time can be valuable for people moving and in situations where they could choose visiting some other place.

In previous works, the statistics accumulated over time are used to estimate a measure of traffic or gatherings [1, 2, 3, 4, 5]. Moreover, both popular online services, and other apps just count the number of people currently present in some place [6, 7, 8, 9]. However, statistics gathered in the past cannot be a reliable indication for the current situation that has to cope with e.g. restrictions on gatherings, lower capacity of public transport means, etc. due to the influenza pandemic. Additionally, a kind of real time measures of gatherings do not let other people plan their trip, hence understanding whether e.g. one hour later when arriving at the destination, the place will still be (un)crowded. A better estimate is therefore needed which

WOA 2020: Workshop "From Objects to Agents", September 14–16, 2020, Bologna, Italy
claudia.cavallaro@unict.it (C. Cavallaro); gabriella.verga@unict.it (G. Verga); tramontana@dmi.unict.it (E. Tramontana); muscato@dmi.unict.it (O. Muscato)
0000-0002-7169-659X (E. Tramontana)

takes into account: (i) the current amount of people in some place, and (ii) the statistics on the number of people that being in some origin place typically flow to another place to visit later on. Moreover, an app behaving as an assistant agent is needed to timely inform interested people.

This paper proposes an approach to determine the probability that users are moving along some routes. Given the recordings of several user positions, we compute the probability for a user that being in some place A will move to another place B (i.e. a possible destination), hence when arriving in place B he will contribute to the number of people gathering there. By computing beforehand the probability that he will go to place B in a future time, we can guess whether a place will become overcrowded. Our proposed estimation of people destinations is based on the analysis of the co-occurrence of places statistically visited by an amount of people greater than a threshold. Moreover, we propose to give users an app that will let them know the amount of people that will gather to some areas that can be reached from the point where the user is. The app provides us support for suggesting the user possible destinations that are viable both in terms of distance, usefulness, and gathering. Moreover, the app provides means to collect data on the current amount of people in some place and then their trajectories. When collecting user data by means of the app we make sure that user privacy is preserved by providing only an approximate location to a central server.

Our approach can be useful in many contexts where estimating the number of people beforehand can be a crucial factor for a better service, such as e.g. when organising public transport, or for retailer, etc. Moreover, it could be enriched with data, coming from proper authorities, that reveal some places where a Covid-19-positive has been found. Then, by using our computed trajectories, we could give for other places the probabilities of having infection spreads.

The paper is organised as follows. Next section describes the related work. Section 3 explains our proposed solution. Section 4 illustrates the experiments and shows the viability of our approach. Finally, conclusions are drawn in Section 5.

2. Related work

This section offers an overview of the studies on multi-agent systems and the analysis of the movement of people.

Multi Agent System: a multi-agent system is a system with a significant number of independent agents that interact with each other [10]. In recent years, multi-agent systems have been widely used as being regarded suitable for systems with a modular architecture, thanks to their independence [10]. Generally, agents interact in three ways [11]: (i) each agent can communicate directly with any other agents ("autonomous agents"); (ii) agents communicate indirectly with each other via an intermediary ("facilitator"); (iii) all agents communicate with each other via an intermediary, however the agents can communicate with each other after the communication has been set up by the intermediary ("Mediator"). In the second case, the robustness can be poor and the overhead is relatively high but the intermediary acts as a protective wall for users privacy because agents do not communicate directly and it processes the information received from the users, decreasing their work [12]. On the other hand, the use of an intermediary has several advantages in terms of synchronization, reusability, scalability and modularity [13, 14].

Mobility monitoring: an accurate monitoring of user mobility provides support for efficient resource usage. E.g., it could help avoid traffic congestion [1, 2, 3], give warnings or make targeted advertising by discovering the next place to visit [15], or simply study user behaviour [16]. The approach described in [4] identifies the routine behaviour of two sample of people by using a probabilistic approach (Latent Dirichlet Allocation) to extract 10 different positions shared by multiple users. Other approaches identify people movements. In [5], the proposed system observes GPS data in the urban area of Milan and creates an origin-destination matrix, then provides the similarity between two trajectories, leading to the discovery of mobility behaviours.

Moreover, to find typical trajectories of a group of people, the approach in [17] identifies flows using a grid with the Apriori algorithm [18], however noting only flows and not the frequently visited points of interests, nor the probability of some people moving to another place. In [19], the flows are identified by associating them with spatio-temporal trajectories shared by multiple users heading in the same direction. Finally, other approaches use machine learning techniques, e.g. the approach in [20] is based on a clustering algorithm, and uses an unsupervised learning solution for an automatic lane detection in multidirectional pedestrian flows.

Our proposed work uses a multi-agent system communicating with a server that acts as an intermediary and predicts movements by means of an innovative and reliable mathematical solution. Unlike the work presented in [4], our goal is to detect the foreseeable routes by computing their probability, instead their method determines the probability that a group of users is moving together. The work presented in [20] uses an unsupervised learning approach and aggregates instantaneous information on the position and speed of pedestrians to form clusters, calculated on short time windows. We use a fixed grid to group people positions into cells, and then compute the probabilities using the data arriving from the agents in real time. The paper [21] presents personalised recommendations for guiding tourists through the city of Melbourne by observing their actions. This system is modelled as a Markov decision process that recommends the user in sequence the next place to visit. However, unlike the StayPoint analysis presented here, it does not consider the stationary nature of visitors over a period of time and this is a key element in avoiding overcrowding. In [17] the frequent *corridors*, i.e. routes, were found on a grid through Apriori algorithm [18], while in this work we initially find the areas having highly visited points of interest, hence giving great importance to the user stay time. Moreover, we consider the Confidence and Lift metrics used in the Market Basket Analysis to know the probability of displacement and therefore predict the contagion areas that are continuously updated.

3. Proposed Approach

We aim at determining the probability that users are moving from one point to another point. Such a probability is then used to provide recommendations accordingly. Recommendations, alerts, or user requests, are communicated by means of a smartphone app. Therefore, our proposed solution comprises two parts: (i) an algorithm that determines the probability of people movements, and (ii) an app on the users device to track movements and suggest destinations.

3.1. Determining Probability of Movement

To determine the probability of movement we perform two steps. The first step consists in determining shared people flows from the points recorded during the previous movements of each user. Then, the second step consists in obtaining statistics on the amount of people that being in point A subsequently go to point B.

By analysing every *GPS trajectory*, i.e. the set of recorded GPS points temporally ordered, we extracted its *StayPoints* (SPs). They are the centres of the areas within which a user stays for more than a certain time: for some reason that area is of interest. Then, the geographical area where the SPs of all dataset are located has been discretised by means of a grid, made up of equal *Square Cells*. Each determined SP has been associated with a single square cell if it is contained in that space. Sure, a cell could contain multiple SPs if these are close enough, depending on the width of the cell. Cells that did not contain any SP have not been considered.

We then determined the subset of *frequently visited cells* consisting of all the cells that having at least one SP within them have been visited at least by 10% of the people. For the sake of reliability, we compute only the statistics between the frequently visited cells, and we consider the Confidence as used by the Market Basket Analysis. Confidence denotes the percentage of trajectories frequently visiting a cell B which also frequently visit cell A. I.e. for a value of Confidence higher than a threshold (set as 60% in our experiments), we can assert that a large group of people having visited cell A moves together to cell B. Confidence is an estimation of conditioned probability. Two or more cells for which there is a Confidence higher than 60% that have been visited by a large group of people are dubbed *co-visited cells*.

Then, we check the reliability of the association rules obtained ($A \Rightarrow B$) through *Lift*, which will confirm that the transition of a user from the *SP* in A to the SP in B has a positive correlation.

3.2. A Multi-Agent Recommendation System

In general, an agent, according to Wooldridge [22], is merely *"a software (or hardware) entity that is situated in an environment and is able to autonomously react to changes in that environment"*. Each agent has the basis to learn and communicate, and in our case, learning takes place by capturing the user GPS positions and, communication is realised by connecting to a centralised server, which alerts all agents when needed and stores the geographical coordinates of the points visited by users. Figure 1 shows the main components of the proposed multi-agent system.

An agent runs on a smartphone as an app in order to receive suggestions on possible destination. The agent offers recommendations highlighting any 'warm', that is very crowded, or 'cold', that is uncrowded, place, using the statistics gathered as described in the previous section. For this, the agent periodically reads the user position and checks whether a known StayPoint (*SP*) is nearby. Then, the agent communicates to the server whether it is close to a *SP*. This lets the agents contribute in determining the number of people close to a *SP*, rather than giving their actual position, hence preserving the user's privacy. In this context, the privacy protection is intended to prevent the disclosure of information relating to the exact location of the user. Figure 2 shows the app providing information to the user.

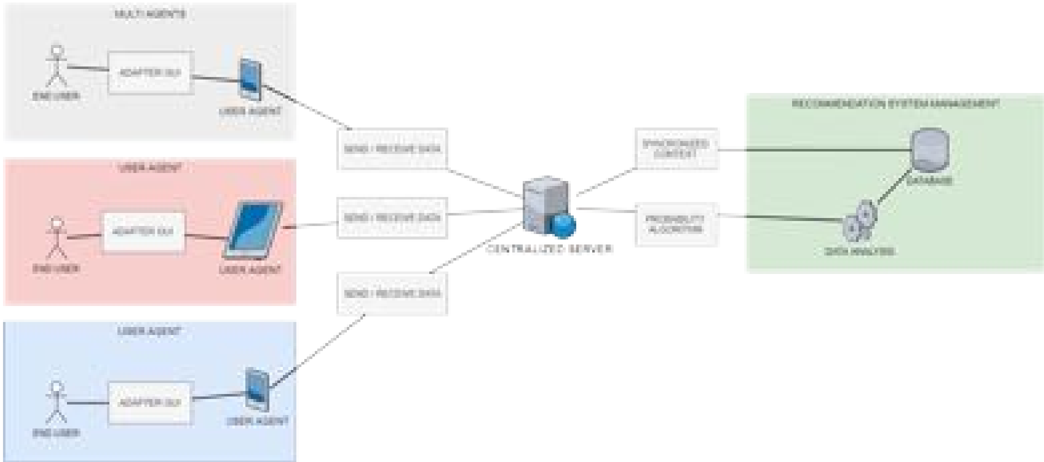


Figure 1: System architecture showing the interaction between agents and server: each agent sends his preference for crowded places and where he is, the server gathers data and creates recommendations.

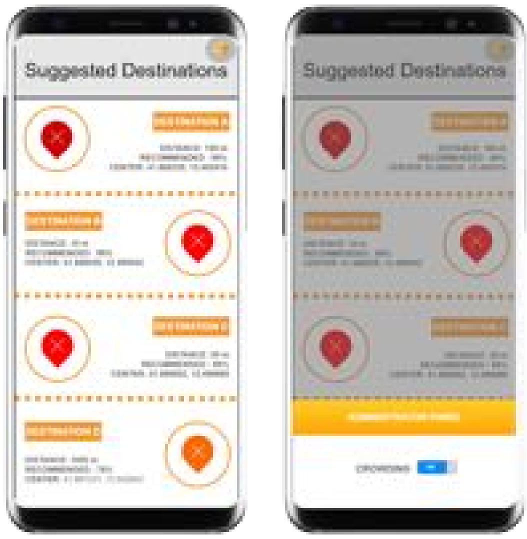


Figure 2: User communicates with agents via application GUI. The left panel shows the list of destinations suggested by the multi-agent system and the right panel is the administration view where the user gives his preferences on (un)crowded places. The colour of the icons represents the intensity of crowding, that is, more (less) red equals more (less) crowded.

The server, having acquired by the user the position of the nearest *SP*, returns the list of other *SP*s that could be visited according to the probability estimate of passing through that point (0 equals low probability, 1 equals high probability). In this way, we create a Collaborative Filtering based recommendation system [23], as it is based on the choices of other users.

Finally, the user, through an administration panel, can set with a flag, if he prefers 'warm' or 'cold' places. Thus, the agent, based on the users choice and the list received from the server, determines information to show and then suggest. Destinations are displayed as a map or a list. All agents are independent of each other and since they extrapolate data directly from the device they are reliable, making the architecture stable and trustworthy.

4. Experiments

This section describes the experiments carried out using a dataset that collects real movements from one part of the city to another by taxis and/or people. Positions have been gathered by periodically reading geo-coordinates from tablets or smartphones. The experiments focus on data analysis for determining the probabilities of moving from one StayPoint to another as described in Section 3. This approach is used in our centralized server in order to select the list of suggestions to send to the agents. The used dataset allows us to simulate the behaviour of a reasonable number of users, showing the useful of the app. Over time, data are updated as provided by agents. Below we describe the dataset used in our experiments, then the tests carried out, and the results that have been found.

4.1. Dataset

The dataset used to perform our tests is *Cabspottingdata* [24] and includes the trajectories collected in May 2008 by 536 taxis, for a total of $11,219,424$ GPS points. Cab mobility traces are provided by the Exploratorium—the museum of science, art and human perception through the cabspotting project[1]. To gather data each vehicle was outfitted with a GPS tracking device that was used by dispatchers to efficiently reach customers. Data were sent from each cab to a central receiving station, and then delivered in real-time to dispatch computers via a central server. Each mobility trace file, associated to a taxi ID, contains in each line: latitude, longitude, occupation, timestamp. Where latitude and longitude are in decimal degrees, the occupation indicates whether a taxi has a fare (1 = busy, 0 = free) and the time is in the UNIX era format. The area covered by these routes corresponds to the county of San Francisco of USA and its surroundings in California, with maximum and minimum longitude and latitude $= [-127.08143; 32.8697]$ x $[-115.56218; 50.30546]$. The total size of the trajectories registered with a customer on the taxi consists of $5,017,659$ points.

4.2. Tests carried out to find flows and StayPoints

A trajectory T is an ordered sequence of GPS points, in which the positions occupied (for example by a vehicle) and the timestamps associated with them are recorded chronologically. Each position is represented by latitude and longitude of the geographical point. $T = \{p_0 = (x_0, y_0, t_0), p_1 = (x_1, y_1, t_1), \ldots, p_n = (x_n, y_n, t_n)\}$, where $\forall i \in [0, n],\ p_i = (x_i, y_i, t_i)$ with $t_i < t_{i+1}$, and x_i, y_i and t_i represent longitude, latitude, and timestamp, respectively.

The first step was the data cleaning in order to eliminate noise, due for example to GPS errors. It was performed by computing the instantaneous speed of each point of the rides recorded on the taxi. The maximum acceptable speed threshold has been set for 150 $\frac{km}{h}$. The trajectories have been summarised for the comparison of the distance, considering for each

[1] http://cabspotting.org

path two successive points in temporal order only if they were at a minimum distance of 140 m. This was done to decrease the size of the dataset and therefore will allow a reduction in the execution time of the algorithm.

To carry out a statistical analysis, 90% of trajectories were randomly selected, and this set was the *Train* set for the the flow detection algorithm. The complementary set, that is the remaining 10% of the trajectories, consists in the *Test* set, that is the verification set. We considered 6 time slots of 4 hours each, to visualise the movement of the vehicles at different times and the trajectories were therefore split according to the 6 time slots. To identify the sub-trajectories common to different users in the same time slot, a maximum tolerance distance was set between two different points of different users as 280 m. The distance between two points was computed by using the *Haversine distance*, which given two points $P_i(lt_i, lg_i)$ and $P_j(lt_j, lg_j)$ characterised by latitude and longitude in decimal degrees returns their distance in meters considering the curvature of the earth:

$$d(P_i, P_j) = 2R \arcsin \sqrt{\sin^2 \frac{lt_i - lt_j}{2} + \cos lt_i \cos lt_j \sin^2 \frac{lg_i - lg_j}{2}}$$

where R is the mean radius of the earth.

We define *flows* as close sub-trajectories, belonging to different users, spatially similar and recorded in the same time slot. The density of a flow is the number of users that pass through it. For detecting flows in this dataset, the minimum density threshold was set to 25. According to these parameters, 12 flows were identified, ranging from 1 to 2 km in length. The minimum density of the flows found is 26 taxis, while the maximum density found is 192 taxis. Then, by taking the complement of the trajectory sample (10% of the taxis, as the test set) we checked where their GPS points were compared to the previous train set. We found that the points of the test set intersect with the 12 paths identified on the train set. Another check was carried out by confirming the correspondence of the points of the flows on a map. It consists of the process of matching the coordinates of the obtained flows and the road segments, and assessing that there are no external points with respect to road segments (see Figure 3).

We apply the StayPoint detection algorithm to each trajectory (more details can be found in [25]) with time threshold, *TimeThr*, equal to 10 minutes, and distance threshold, *DistThr*, 100 meters. Such thresholds should suffice to select the positions in which a user dwells (in several *SPs*) as he finds the place interesting, and removes the locations where a user is stopping because e.g. he is blocked at the traffic light.

The execution time for the StayPoints detection algorithm on the whole Cabspotting dataset (536 taxis and more than $11M$ points) was 36 minutes and 54 s. We obtained a total of 4261 SPs, which is an average of 8 SPs per vehicle journey. The results show that 98% of users have at least one StayPoint associated with their trip (523 users out of 536). The implementation of StayPoints detection algorithm used Python and the experiments were executed in a host having an Intel Xeon CPU E5-2620 v3 2.40GHz, with RAM 32 GB.

Figure 4 shows the recorded trace for each trip in blue, and the detected *SPs* in yellow. Figure 5 shows the detected flows in magenta and the *SPs* in that area in green.

Figure 3: Flows detected for the Cabspotting dataset.

4.3. Movement prediction

For predicting the movements of people, firstly a grid was built, which covers the map, made up of square cells with a side of 1 km. Such a grid lets us discretise the data and estimate the probability of movement from a cell having some *SPs* inside it to another cell also having at least one *SP*. Two distinct geographical areas comprising some *SPs* are represented as two square cells without intersection, therefore a space partition is formed. Figure 6 shows such a grid, having size 80×46 cells (latitude by longitude), and the obtained *SPs* are mapped in to the grid

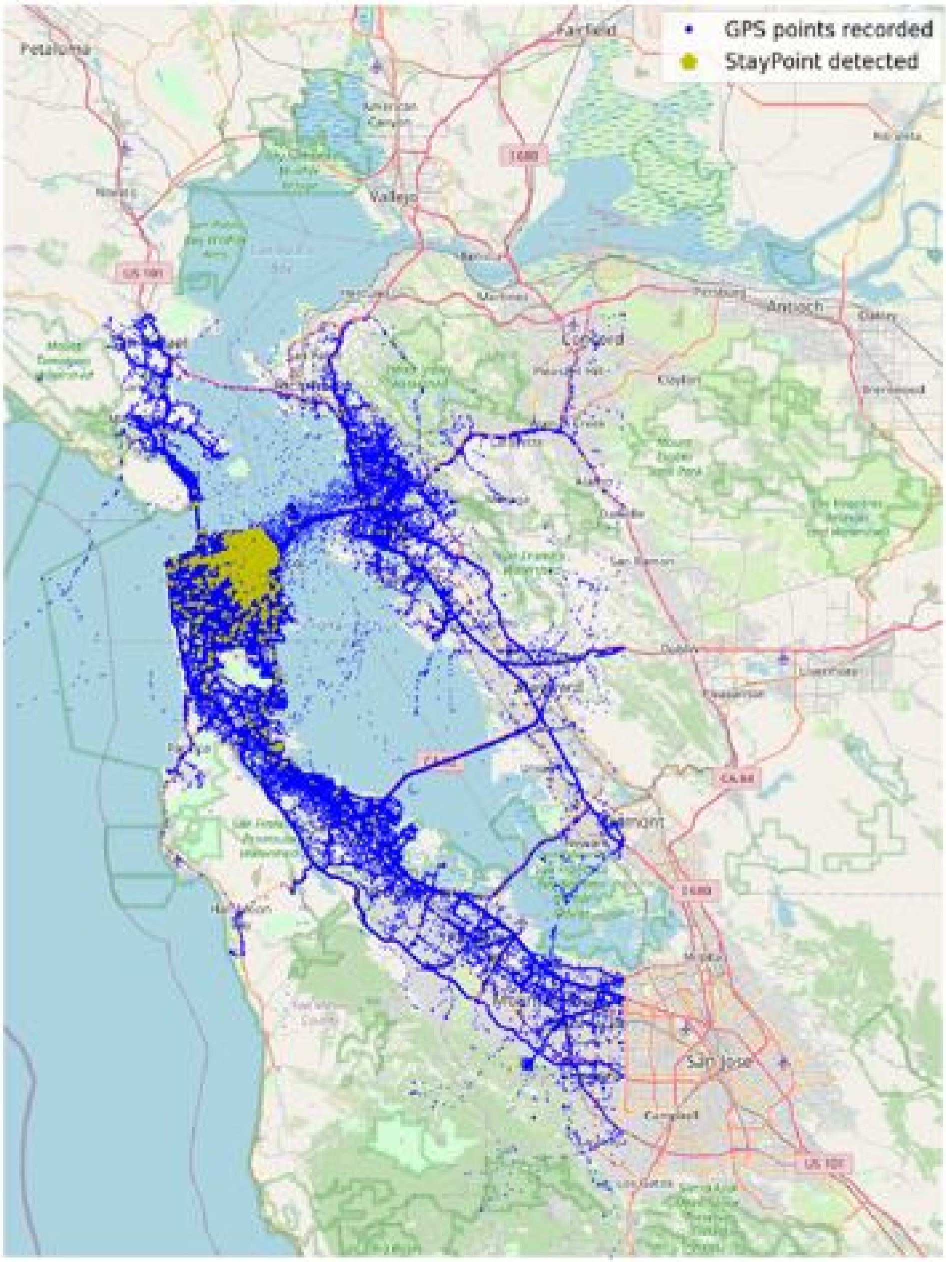


Figure 4: Blue points for trajectories and StayPoints obtained in yellow.

and shown as red dots. Some areas consisting of nearby cells have many more *SPs* than others, hence red dots are more dense in some areas than others, as shown in the said figure.

In order to determine whether a cell A is a frequent destination, the *Support* for each cell was calculated. The *Support* is the ratio between the number of trajectories that contain the cell and the total number of trajectories. If this ratio exceeds a certain threshold, i.e. if cell A

Figure 5: A zoomed in map of an area in Figure 4, showing flows in magenta and the nearby StayPoints in green.

is crossed by a certain number of different trajectories (10% value was chosen for Minimum Support, i.e. 0.1), then the cell (containing one or more *SPs*) will be a *frequently visited cell*.

Our experiments on the above said taxi dataset have shown that there are 43 cells visited by a number of users greater than or equal to 52. I.e. we can say that in the dataset there are 43 frequently visited SPs cells. This means that there has been a probable meeting in that cell, as users have remained stationary in the same time slot in the same cell. Data are updated in real time through the agents running on smartphones as an app, therefore the Minimum Support is fixed, however the number of frequent cells in output and the position of these frequently visited cells will vary over time.

By lowering the Minimum Support, i.e. the threshold of the minimum amount of people sharing the same cell, the number of cells considered as having a sufficient amount of people will increase and then the number of cells considered overcrowded will increase. In order to compute association rules only between the frequently visited cells in the dataset, we considered

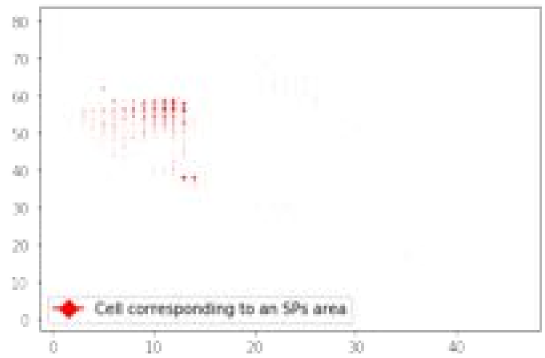


Figure 6: Grid formed by squared cells of 1 km per side, each red dot represent a cell having at least one SP.

the *Confidence* of the Market Basket Analysis for our approach.

Given two cells called A and B we have that:

$$Support(A \Rightarrow B) = \frac{Frequency(A, B)}{N}$$

the Support of the association rule $(A \Rightarrow B)$ denotes the percentage of trajectories containing A which contain also B, where N is the total number of trajectories.

$$Confidence(A \Rightarrow B) = \frac{Support(A \Rightarrow B)}{Support(A)}$$

Hence, confidence is an estimation of conditioned probability, which can be expressed as follows:

$$Confidence(A \Rightarrow B) = \frac{p(A \cap B)}{p(A)} = P(B|A).$$

In Market Basket Analysis, *Confidence* is the probability of purchasing item B, said *consequent*, given the purchase of object A, said *antecedent*, within the same transaction. The higher the *Confidence*, the greater the reliability of the $(A \Rightarrow B)$ rule (more details can be found in [26]). In our context, the value computed as the *Confidence*$(A \Rightarrow B)$ gives the probability that a user is in a *SP* in cell B moving there together with at least 10% of the total number of users, if he has already been in cell A and dwelling in one of its *SPs*.

Going forward along this procedure, we compute *Confidence*$(A, B \Rightarrow C)$ and after that *Confidence*$(A, B, C \Rightarrow D)$, in order to determine a common path that crosses several cells having highly visited *SPs*. We compute

$$Confidence(A, B \Rightarrow C) = \frac{Frequency(A, B, C)}{Frequency(A, B)}$$

and so on.

Therefore, the results obtained are useful to predict the number of gatherings on some place. Moreover, given that there is knowledge about an infected person on some area, our results can be used to predict whether a user can be potentially infected (as his trajectory is estimated), and predict who else he will infect (i.e. people whose trajectories are expected to pass through the same areas).

The Confidence limit is due to the fact that it does not consider the Support of the item on the right side of the rule and therefore does not provide a correct evaluation in case the groups of items are not stochastically independent.

A measure that takes this eventuality into account is *Lift*$(A \Rightarrow B)$, defined as:

$$Lift(A \Rightarrow B) = \frac{Confidence(A \Rightarrow B)}{Support(B)} = \frac{p(A \cap B)}{p(A) * p(B)}$$

Lift$(A \Rightarrow B)$ takes into account the importance (the Frequency) of B. Using such an amount, then we can say

- if $Lift > 1$ the events are positively correlated;
- if $Lift \leq 1$ the events are negatively correlated or independent.

Therefore, Lift indicates how the occurrence of one event raises the occurrences of the other. At this point, the setting would be a Minimum Confidence (0.6 i.e. 60%) to skim the results and obtain only the association rules that had a higher Confidence and also a Support higher than the Minimum Support (0.1 chosen), such a setting is named *Strong rules*. Finally, these rules were checked with the Lift, the last column of Table 1. Then, for the Association Rule ($[2587] \Rightarrow [2588]$) in row 6 the events of movement from cell A to cell B are negatively correlated.

The left panel in Figure 7 shows the plot of every Strong Rule obtained as a point, as a value for its Support and Confidence (the latter according to the Support). For association rules with higher support the Confidence, that is the probability of moving to the frequently visited cell B, decreases. The Lift and the Confidence of the Strong Rules obtained are directly proportional, as we can see in the right panel of Figure 7. The Pearson correlation coefficient between them is 0.9999999999999999 and this implies an exact linear relationship.

Moreover, the results tell us that the probability of transitioning from one cell with *SPs* to another is high even in correspondence with the indicated flows and that different highly visited cells having *SPs* belong to different flows. For each cell we have checked which taxis passed there and which passed at a later time on other flows passing through other frequently visited cells.

5. Conclusions

Having an educated guess on the amount of people that will gather in some place before planning a trip can be very useful to avoid overcrowded places and to keep with the current regulations. We have proposed an approach for predicting the probability of people moving to some destinations when it is known that a certain amount of people is in some other place. We use an app that senses the position of people and sends to a server such data. Then, such an amount is useful, together with previous statistics, to estimating the amount of people in

Table 1
A set of Cells and the related Support (Sup), Confidence (Conf), and Lift, when cell B is 2588, $Support(B) = 0.671128$, and having Strong Association Rules: Minimum Confidence 60% , Minimum Support 10%

$(A \Rightarrow B)$	A	$Sup(A)$	$Sup(A \Rightarrow B)$	$Conf(A \Rightarrow B)$	$Lift(A \Rightarrow B)$
1	[1715]	0.395793	0.284895	0.719807	1.072533
2	[1716]	0.313576	0.223709	0.713415	1.063008
3	[2405]	0.151052	0.108987	0.721519	1.075084
4	[2541]	0.137667	0.099426	0.722222	1.076132
5	[2543]	0.242830	0.175908	0.724409	1.079391
6	[2587]	0.435946	0.292543	0.671053	0.999888
7	[2633]	0.145315	0.103250	0.710526	1.058704
8	[2634]	0.281071	0.206501	0.734694	1.094715
9	[2635]	0.235182	0.177820	0.756098	1.126607
10	[2679]	0.202677	0.147228	0.726415	1.082379
11	[2680]	0.242830	0.175908	0.724409	1.079391
12	[1715, 1716]	0.149140	0.112811	0.756410	1.127073
13	[2587, 1715]	0.156788	0.112811	0.719512	1.072094
14	[2634, 2587]	0.141491	0.101338	0.716216	1.067183

another place at a later time. The experiments that we have performed on previously gathered geographical locations have shown the viability and reliability of our approach.

The more people use the app the more the approach would give a correct estimate. To make the approach more robust, it could be extended in order to include data available online from other services that give indications on queues, road traffic, gatherings.

Future work will consider the geometry of stations, museums, etc. of some popular destinations to compute the average distance of people given the estimated size of crowds. Moreover, how alerts are spread will consider both the people already in some place and the people moving towards it.

Acknowledgments

The authors acknowledge the support provided by means of a PO FSE 2014-2020 grant founded by Regione Siciliana, and by project TEAMS–TEchniques to support the Analysis of big data in Medicine, energy and Structures–Piano di incentivi per la ricerca di Ateneo 2020/2022.

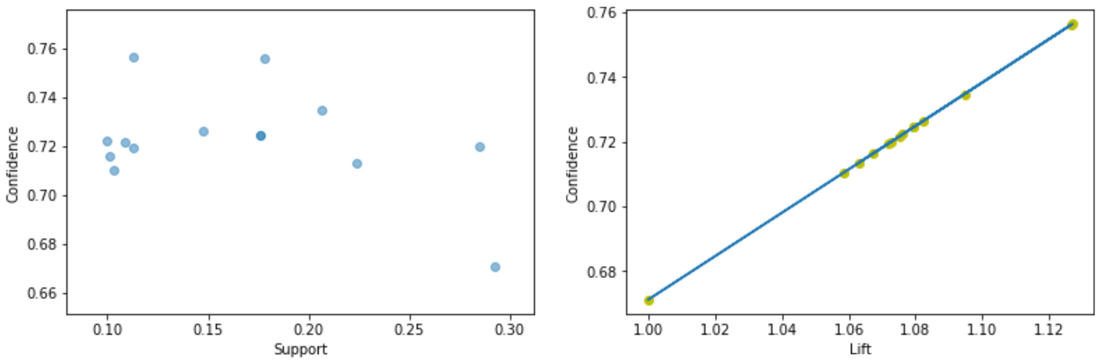


Figure 7: The left panel shows the Support vs Confidence for Strong Rules obtained by our analysis and given as rows in Table 1; the points show a pair of cells (A and B), or a triple of cells (for the last three rows of Table 1). The right panel shows Lift vs Confidence in this test, as for the points shown in the left panel.

References

[1] P. Castro, D. Zhang, S. Li, Urban traffic modelling and prediction using large scale taxi gps traces, in: Proceedings of of International Conference on Pervasive Computing, 2012, pp. 57–72. doi:10.1007/978-3-642-31205-2_4.

[2] J. Lee, J. Han, X. Li, A unifying framework of mining trajectory patterns of various temporal tightness, IEEE Transactions on Knowledge and Data Engineering 27 (2015) 1478–1490.

[3] Z. Wang, M. Lu, X. Yuan, J. Zhang, H. van de Wetering, Visual traffic jam analysis based on trajectory data, IEEE Transactions on Visualization and Computer Graphics 19 (2013) 2159–2168.

[4] N. Bicocchi, M. Mamei, Investigating ride sharing opportunities through mobility data analysis, Pervasive Mobile Computing 14 (2014) 83–94.

[5] R. Trasarti, et al., Exploring real mobility data with m-atlas, in: Proceedings of Machine Learning and Knowledge Discovery in Databases, Springer, 2010, pp. 624–627.

[6] J. Cranshaw, R. Schwartz, J. Hong, N. Sadeh, The livehoods project: Utilizing social media to understand the dynamics of a city, in: Proceedings of AAAI Conference on Weblogs and Social Media, 2012.

[7] C. Berzi, A. Gorrini, G. Vizzari, Mining the social media data for a bottom-up evaluation of walkability, in: Proceedings of International Conference on Traffic and Granular Flow, Springer, 2017, pp. 167–175.

[8] B. P. L. Lau, M. S. Hasala, V. S. Kadaba, B. Thirunavukarasu, C. Yuen, B. Yuen, R. Nayak, Extracting point of interest and classifying environment for low sampling crowd sensing smartphone sensor data, in: Proceedings of IEEE Pervasive Computing and Communications, 2017.

[9] C. Cavallaro, G. Verga, E. Tramontana, O. Muscato, Eliciting cities points of interest from people movements and suggesting effective itineraries, Intelligenza Artificiale (2020). doi:10.3233/IA-190040.

[10] K. P. Sycara, Multiagent systems, AI Magazine 19 (1998) 79. URL: https://www.aaai.org/ojs/index.php/aimagazine/article/view/1370. doi:10.1609/aimag.v19i2.1370.

[11] W. Shen, D. Norrie, Facilitators, mediators or autonomous agents, in: Proceedings of International Workshop on CSCW in Design, 1997, pp. 119–124.

[12] Q. Liu, L. Gao, P. Lou, Resource management based on multi-agent technology for cloud manufacturing, in: Proceedings of Electronics, Communications and Control (ICECC), IEEE, 2011, pp. 2821–2824.

[13] J. Z. Hernández, S. Ossowski, A. Garcıa-Serrano, Multiagent architectures for intelligent traffic management systems, Transportation Research Part C: Emerging Tech. 10 (2002) 473–506.

[14] E. Tramontana, Minimising changes when refactoring applications to run multiple threads, in: Proceedings of IEEE Asia-Pacific Software Engineering Conference (APSEC), Nara, Japan, 2018, pp. 713–714.

[15] G. Verga, A. Fornaia, S. Calcagno, E. Tramontana, Yet another way to unknowingly gather people coordinates and its countermeasures, in: Proceedings of Internet and Distributed Computing Systems (IDCS), Springer LNCS 11874, Naples, Italy, 2019, pp. 130–139.

[16] A. Noulas, S. Scellato, N. Lathia, C. Mascolo, Mining user mobility features for next place prediction in location-based services, in: Proceedings of IEEE International Conference on Data Mining, 2012, pp. 1038–1043.

[17] C. Cavallaro, J. Vitrià, Corridor detection from large gps trajectories datasets, Applied Sciences 10 (2020) 5003. doi:10.3390/app10145003.

[18] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of International Conference on Very Large Data Bases, Morgan Kaufmann, Los Altos, CA, 1994, pp. 478–499.

[19] E. Tramontana, G. Verga, Demo: Get spatio-temporal flows from gps data, in: Proceedings of IEEE International Conference on Smart Computing (SMARTCOMP), Taormina, Italy, 2018, pp. 282–284.

[20] L. Crociani, G. Vizzari, A. Gorrini, S. Bandini, Identification and Characterization of Lanes in Pedestrian Flows Through a Clustering Approach, volume 11298, Springer Verlag, 2018, pp. 71–82. doi:10.1007/978-3-030-03840-3_6.

[21] F. de Nijs, G. Theocharous, N. Vlassis, M. M. de Weerdt, M. T. J. Spaan, Capacity-aware sequential recommendations, in: Proceedings of International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2018, p. 416–424.

[22] M. Wooldridge, N. R. Jennings, Intelligent agents and multi-agent systems I, Applied artificial intelligence 9 (1995).

[23] P. Melville, V. Sindhwani, Recommender systems., Encyclopedia of machine learning 1 (2010) 829–838.

[24] M. Piorkowski, N. Sarafijanovic-Djukic, M. Grossglauser, CRAWDAD dataset epfl/mobility (v. 2009-02-24), 2009. doi:10.15783/C7J010.

[25] C. Cavallaro, G. Verga, E. Tramontana, O. Muscato, Multi-agent architecture for point of interest detection and recommendation., in: Proceedings of Workshop from Objects to Agents (WOA), Parma, Italy, 2019, pp. 98–104.

[26] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.