Matteo Baldoni
Federico Chesani
Paola Mello
Marco Montali
(Eds.)

PAI2013
Popularize Artificial Intelligence

AI*IA National Workshop
"Popularize Artificial Intelligence"
Held in conjunction with AI*IA 2013
Turin, Italy, December 5, 2013
Proceedings

PAI 2013 Home Page:
http://aixia2013.i-learn.unito.it/course/view.php?id=4

Copyright

Sponsoring Institutions

Associazione Italiana per l'Intelligenza Artificiale

Editors' addresses:
University of Turin
DI - Dipartimento di Informatica
Corso Svizzera, 185 10149 Torino, ITALY
baldoni@di.unito.it

University of Bologna
DISI - Dipartimento di Informatica - Scienza e Ingegneria
Viale Risorgimento, 2
40136 Bologna, Italy
federico.chesani@unibo.it
paola.mello@unibo.it

Free University of Bozen-Bolzano
Piazza Domenicani, 3
39100 Bolzano, Italy
montali@inf.unibz.it

Preface

The *2nd Workshop on Popularize Artificial Intelligence (PAI 2013)* follows the successful experience of the 1st edition, held in Rome 2012 to celebrate the 100th anniversary of Alan Turing's birth. It is organized as part of the *XIII Conference of the Italian Association for Artificial Intelligence (AI*IA)*, to celebrate another important event, namely the *25th anniversary of AI*IA*.

In the same spirit of the first edition, PAI 2013 aims at divulging the practical uses of Artificial Intelligence among researchers, practitioners, teachers and students. 13 contributions were submitted, and accepted after a reviewing process that produced from 2 to 3 reviews per paper. Papers have been grouped into three main categories: student experiences inside AI courses (8 contributions), research and academic experiences (4 contributions), and industrial experiences (1 contribution). They cover a wide range of AI techniques, from robotics and clustering to declarative problem solving and logic-based approaches, as wide as the range of application areas, from RoboCup to (video)games, ambient assisted living, healthcare, geology, mobile technologies and vision.

In accordance to the content of the papers and their reviews, the Program Committee and the Workshop Organisers awarded a *Best Paper Award* to:

> *AngryHEX: an Artificial Player for Angry Birds Based on Declarative Knowledge Bases*, by Francesco Calimeri, Michael Fink, Stefano Germano, Giovambattista Ianni, Christoph Redl, and Anton Wimmer.

The Organising Committee warmly thanks the authors and the members of the Program Committee for their scientific contribution, as well as the organizers of the XIII Conference of AI*IA and AI*IA itself for the provided support.

December 1, 2013

Matteo Baldoni
Federico Chesani
Paola Mello
Marco Montali

Organizing Committee

Matteo Baldoni, Univ. of Turin
Federico Chesani, Univ. of Bologna
Paola Mello, Univ. of Bologna
Marco Montali, Free Univ. of Bozen

Program Committee

Francesco Amigoni
Giuliano Armano
Cristina Baroglio
Andrea Bonarini
Emanuele Bottazzi
Francesco Calimeri
Luigia Carlucci Aiello
Federica Cena
Stefania Costantini
Nicola Di Mauro
Agostino Dovier
Aldo Franco Dragoni
Stefano Ferilli
Giorgio Fumera
Nicola Gatti
Marco Gavanelli
Rosella Gennari
Giuseppina Gini
Marco Gori
Nicola Guarino
Evelina Lamma
Vittorio Maniezzo
Angelo Marcelli
Alberto Martelli
Emanuele Menegatti
Alessio Micheli
Michela Milano
Daniele Nardi
Andrea Omicini
Agostino Poggi
Fabrizio Riguzzi
Andrea Roli
Gianfranco Rossi
Marco Schaerf
Giovanni Semeraro

Contents

RoboCup@Sapienza

Daniele Nardi, Luca Iocchi, and Luigia Carlucci Aiello

Dept. of Computer, Control, and Management Engineering,
Sapienza University of Rome, via Ariosto 25, 00185, Rome, Italy
{nardi,iocchi,aiello}@dis.uniroma1.it

1 RoboCup

RoboCup was started in 1997 by a group of Artificial Intelligence and Robotics researchers with the following grand challenge, to be achieved by 2050 [9]: "*to build a team of robot soccer players, which can beat a human World Cup champion team.*"

Since then, RoboCup Federation[1] established itself as a structured organization and as one of the major events in the research field of Artificial Intelligence and Robotics. A quote from the website of the RoboCup Federation, best describes its objective.

> *It is our intention to use RoboCup as a vehicle to promote robotics and AI research, by offering a publicly appealing, but formidable challenge. One of the effective ways to promote engineering research, apart from specific application developments, is to set a significant long term goal. When the accomplishment of such a goal has significant social impact, it is called grand challenge project. Building a robot that plays soccer by itself does not generate significant social and economic impact, but the accomplishment will certainly be considered as a major achievement of the field. We call this kind of project as a landmark project. RoboCup is a landmark project as well as a standard problem.*

Currently, the major scientific international events include a section dedicated to Robotic Soccer aiming at the grand challenge. A number of regional events have also been established as well, some reaching a size almost comparable to the main event. RoboCup has a broad international participation, with teams from more than forty countries worldwide.

The competitions inspired to soccer have different formats and address different research challenges. 2D and 3D soccer simulation leagues are performed in a simulated environment, where each simulated robot has limited and noisy perception of the environment; participant teams are required to develop software for each single robotic agent and for their coordination. The research focus in these simulation leagues is on multi-agent systems, coordination, learning, and game strategy. The small size-league is characterized by small wheeled robots (18 cm in diameter), playing in two teams of up to 6 robots in a field of 6 m x 4 m, using an orange golf ball. In this league, an external vision system allows for accurate perception and the main challenges are fast motion, planning complex actions and learning opponent behaviors. In the mid-size league, robot size is about 50 cm x 50 cm and two teams of 5 robots each play on

[1] www.robocup.org

a 18 m x 12 m field using a standard yellow or orange size 5 soccer ball. Here local perception requires sensor fusion as well as distributed approaches to robot coordination. Standard platform league is currently played with the Aldebaran NAO humanoid robots (formerly with the Sony Aibo), with two teams of 5 robots playing in a 9 m x 6 m field using an orange street hockey ball. In this league, in addition to the issues of the mid-size league to be addressed on a more challenging platform, the interaction between perception and action plays a critical role. Finally, the humanoid robot leagues (kid, teen and adult sizes), playing in a field similar to the one used in the Standard platform league, encompass all of the above mentioned issues, on top of a dedicated hardware development.

The second main section of RoboCup competitions targets systems with a more direct impact on our society. RoboCup Rescue is focussed on robots supporting human rescuers in the aftermath of disasters (e.g. Fukushima nuclear accidents or L'Aquila earthquake). This section includes both 2D and 3D simulated leagues as well as a main robot league, where robots perform several rescue tasks in arenas defined by the National Institute of Standards and Technology (NIST). RoboCup@Home addresses the development of robots that interact with humans helping them in home environments. The competition is performed in an arena reproducing an apartment with typical layouts and furniture, that is not known before-hand by the participating teams. Moreover, some test are executed in the public space of the main venues hosting the event, as well as in real restaurants, shopping malls nearby. Finally, more recently, RoboCup@Work has been set up as a league aiming at a new generation of industrial robots. In this league mobile manipulators are used for executing tasks related to manufacturing, automation, and general logistics.

The third section of RoboCup is dedicated to competitions among juniors. RoboCup Junior is an educational initiative for students up to the age of 19, providing a new and exciting way to understand science and technology through hands-on experiences with electronics, hardware and software.

In this paper, we overview the achievements of the group at Sapienza first in terms of participation and results in the competitions and then in terms of research contributions. We conclude by discussing the impact of RoboCup@Sapienza and by providing a retrospective analysis.

2 Participation and results

The Italian community joined RoboCup since its first edition, and our research group at Sapienza University of Rome, in 1998. Since then, we participated in RoboCup competitions, first within the ART national Italian team[2] and from 2001 as the SPQR team[3]. Figure 1 shows the evolution over the years of the robotic platforms used for the soccer competitions, moving from wheeled robots, to four-legged and finally to small humanoid robots; while Figure 2 shows rescue and @Home robots.

Moreover, Daniele Nardi is currently President of the RoboCup Federation (since 2011), and he has been Vice-President (2008-2011), and Trustee (since 2003). Luca Ioc-

[2] www.dis.uniroma1.it/ART

[3] spqr.dis.uniroma1.it

chi is Trustee (since 2013), formerly Executive for the RoboCup@Home league. They also contributed in the organization of scientific events, (e.g., co-chairs of RoboCup Symposia, workshops in international conferences, special issues in AI and robotics journals), as well as in the organization of regional competitions (such as Mediterranean Open 2010 and 2011, RomeCup, etc.).

Fig. 1. Soccer robots used in RoboCup: a) ART - Middle-Size (1998-2000), b) SPQR - Sony AIBO ERS-210 (2001-2003) c) SPQR - Sony AIBO ERS-7 (2004-2007), d) SPQR - NAO Humanoid (since 2008).

RoboCup was brought to the attention of the Italian Artificial Intelligence community by Luigia Carlucci Aiello and by Enrico Pagello, who organized a well attended workshop in the Fall 1997. After that a joint project ART (Azzurra Robot Team) (Figure 1 a)) [11, 12] was sponsored by Consorzio Padova Ricerche and supported by the Italian National Research Council. Daniele Nardi was the CT (Technical Coordinator) of the project. The first kicks to the ball by ART robots started in the Spring 1998, and the first participation in RoboCup was in 1998, Paris. In 1999, at JCAI Stockholm, ART obtained the second place in the mid-size league. The project ended in 2000 after winning a European championship in the Netherlands, and participating in RoboCup 2000 in Melbourne.

In 2000, the first SPQR (Soccer Players Quadruped Robots) team from Sapienza, entered the Sony Aibo league (Figure 1 b)), obtaining the fourth place. Since then, SPQR participated in the league till 2007, with different generations of AIBO robots (Figure 1 c)), reaching the quarter finals and developing several technical contributions, including the pass that won the “Best Demo” Award at AAMAS 2008 [13].

Fig. 2. Robots used in RoboCup Rescue and @Home: a) SPQR - Rescue robot (2003-2004), b) RoboCare - @Home robot (2006).

In RoboCup 2003, Padova, SPQR joined the new RoboCup Rescue Competition. SPQR then participated both in the Real Robot Rescue League (Figure 2 a)) and in the Virtual Robot Simulation League, building on the results achieved through a collaboration with the Italian Firemen Dept. In the Virtual Robot Simulation League SPQR team obtained the 3rd place in 2007 and won the Technical Challenge for the Human Robot Interface in 2009. Moreover, during RoboCup 2009, members of the SPQR team made the first demonstration in RoboCup Rescue with quadrotors.

In RoboCup 2006, Bremen, we obtained the third place in the RoboCup@Home league, based on the platform developed within RoboCare[4] (Figure 2 b)), the first Italian Project addressing the use of robots in a home environment to help the elderly people.

Since Robocup 2008, Suzhou, China, SPQR participated in the NAO humanoid league (Figure 1 d)), also teaming up with Univ. Santiago, Chile (2010). Worth mentioning is the first goal ever scored in the humanoid competitions with a back-kick. This year, SPQR won the Iranian Open, and obtained the third place in the German Open. In the last RoboCup in Eindhoven, SPQR was eliminated at the end of the second phase, because of the goal difference, after winning four games in a raw and loosing only the last one!

3 Research

Our participation in RoboCup has been motivated in the first place by the research challenges put forward by the competitions. Our experience in mid-size league focussed on two topics: localization and cooperation. Initially, we looked at specialized methods

[4] robocare.itsc.cnr.it

for localization that are applicable in RoboCup by looking at the lines of the soccer field [7]. After this initial work, localization and SLAM (Simoultaneous Localization And Mapping) have become one of the key research lines in our group. Cooperation among robots is needed in robot soccer and achieving it in a national team poses outstanding technical and scientific challenges. ART has been the first example of national team, and the key feature for success was the ability of the robots to cooperate through a simple, but effective approach, based on dynamic task assignment [4, 8], that afterwards has been a minimal prerequisite for all robot soccer teams. Cooperation and coordination became another research line in our group. We have then developed multi-robot systems for a variety of applications, including exploration and surveillance.

One of the most important abilities for soccer robots is understanding the situation through on-board sensors, in particular vision. An important challenge is thus real-time image processing and understanding with limited computational power. In this context, we developed an efficient method for color segmentation [6], that provides increased robustness to variable illumination conditions. Moreover, analysis of RGB-D data of cameras looking at the soccer field has been used to estimate the ground truth pose of the robots and the ball, useful for many evaluation tasks [14].

The results and competences acquired through the mid-size robots, have been exploited in two directions. First of all, the approach to teamwork for robot soccer has been the basis for our subsequent participation in the standard platform league. The design of an autonomous mobile robot has also been transferred into the realization of autonomous rescue robots, capable of exploring indoor environments, dangerous to enter for humans, searching for victims. Further developments on the wheeled rescue robots led to new strategies for navigation and exploration [3] as well as to novel approaches to the design of the interface for the remote operator controlling multiple robots [15].

Once we started working with legged robots, we had to deal with locomotion. While this is not a major research direction in our group, we were able to establish fruitful collaborations [19, 16], to apply learning techniques [5] and to develop new features in a 3D realistic simulator [17] (Best Paper Award at RoboCup Symp. 2006).

Given our background in knowledge representation and reasoning, our long term goal is to use symbolic representations and, specifically, devise plans for robot actions. After few years focussed on the development of basic skills, we were able to support the development of our systems with an explicit representation of actions and plans. To this end we developed a formalism, called PNP (Petri Net Plans) [18], that allows for several advanced features, such as sensing actions, parallel execution, interrupts and communication among agents. Using this formalism, we provided an explicit model of the pass [13], and showed how plans can be refined through learning [10].

Last, years of experience in the design of software for robotic systems led us to deliver our development environment OpenRDK [2], based on a blackboard architecture that supports data exchange among the modules as well as the ability to inspect data and processes and thus to build tools that suitably support debugging.

4 Impact

The presence and impact of RoboCup at Sapienza was far beyond our initial expectations: more than 200 students contributed over the years, with several master theses, projects and course work. At least half of them had the chance to participate in an International event. Our activity in RoboCup substantially contributed to the creation in 2009 of the Master Course (Laurea Magistrale) in Artificial Intelligence and Robotics, which is still one of the few curricula in Italy with a significant AI component.

We started RoboCup Camps, to share within the community the result of winning teams, thus fostering progress in the field. We organized RoboCup Camps, for midsize (Padova 2000), for four legged robots (Paris 2001), and for Rescue Robots (Rome 2004-2007). The success of this kind of hands-on schools extended outside RoboCup.

Robot competitions are sometimes viewed as pure educational activities of limited interest for research. However, the European Community has recently recognized the role of competitions by integrating its research programs on robotics with specific initiatives, supporting the benchmarking of robotic systems through competitions. Our research group is a member of the RoCKIn[5] Coordination Action, started January 2013.

While RoboCup aims at a grand challenge with no direct application to real life problems, several offsprings of RoboCup have become success stories, such as Kiva Systems, Aldebaran Robotics and the Quince Robot in Fukushima. Our best success story at Sapienza is the ARGOS system [1], currently deployed in the Grand Canal in Venice to track boats, that was built on the expertise acquired in tracking robot soccer players and the ball on a green carpet.

Finally, our RoboCup activity has attracted the interest of the media and of several initiatives aiming at promoting science and technology. We have been invited in several television programs in the main national Italian channels. Moreover, we have been organizing demonstrations at museums, exhibits and social spaces bringing the research in Artificial Intelligence to the attention of the general public. In particular, we have contributed in the organization of RomeCup (since 2009), mostly focused on RoboCup Junior and of Mediterranean Open (2010-2011), with international competitions of humanoid soccer robots.

5 Retrospective

In conclusion, we certainly did not expect all the above when we started more than fifteen years ago: RoboCup has become a well-recognized and well-established approach to research and education in Artificial Intelligence and Robotics and it has been a very successful driver for the research and academic development of our group.

Obviously, nothing comes without problems, and for those readers that might now be considering undertaking a similar project, here are potential difficulties they may have to face.

The first issue is financing: it is not easy to acquire the money needed to support the project. In fact, funding agencies (with few exceptions among which the grant we

[5] http://rockinrobotchallenge.eu/

obtained by Italian National Research Council), do not provide schemes to support projects that directly target competitions, unless they are sponsoring the competitions themselves (as in the case of the ongoing DARPA Challenge, or, more recently, the European Community). Consequently, we had to collect the budget to buy the robots from our university and from sponsorships, as detailed in the acknowledgements. However, any residual money from other projects has been absorbed by our RoboCup activities.

A second key issue is the cost in terms of human resources: besides hunting for funding, managing teams of students, driving them towards successful implementations, while keeping a focus on research goals are time consuming tasks. In particular, teamwork has been a real challenge and sometimes a source of difficult relationship among the team mates; the motivations that lead to enter a competition are often strong, and can give rise to stressful situations.

Another challenge is the maintenance of both hardware and software. While this is a well-known issue for projects that target robotic prototypes, robots that play soccer break more often than ordinary robots. Moreover, the software running on the platforms is difficult to maintain, not only because of the frequent releases of components that are needed for the competition, but also because the software is developed by students that leave after a big final rush of implementation, that is usually not released in a form that supports re-use by others.

It is sometimes argued that it is difficult to keep the right balance between engineering and research, when designing and implementing systems for a competition. This is the subject of an ongoing debate in the research community: our contribution to it can be easily inferred from the results presented in this paper. Finding the right balance is a challenge, but there is a significant pay-off both in terms of research achievements and in terms of the contribution to the student's skills and capabilities. Consequently, we keep going on.

Acknowledgements

We warmly acknowledge Sapienza University, in particular our Department and our Faculty, for continuous support to student participation in the competitions. In addition, our RoboCup activities have been supported by a number of other institutions that we gratefully acknowledge: Consorzio Padova Ricerche, Italian National Research Council, Netikos, AI*IA, Epistematica, Fondazione Antonio Ruberti, Space Software Italia, Zucchetti and Algoritmica.

References

1. Bloisi, D.D., Iocchi, L.: ARGOS - a video surveillance system for boat trafic monitoring in Venice. International Journal of Pattern Recognition and Artificial Intelligence 23(7), 1477–1502 (2009)
2. Calisi, D., Censi, A., Iocchi, L., D., N.: Design choices for modular and flexible robotic software development: the OpenRDK viewpoint. Journal of Software Engineering for Robotics (JOSER) 3(1), 13–27 (March 2012), http://www.joser.org

3. Calisi, D., Farinelli, A., Iocchi, L., Nardi, D.: Multi-objective exploration and search for autonomous rescue robots. Journal of Field Robotics, Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems 24, 763–777 (August - September 2007)
4. Castelpietra, C., Iocchi, L., Nardi, D., Piaggio, M., Scalzo, A., Sgorbissa, A.: Communication and coordination among heterogeneous mid-size players: ART99. In: Proceedings of Fourth International Workshop on RoboCup. pp. 149–158 (2000)
5. Cherubini, A., Giannone, F., Iocchi, L., Nardi, D., Palamara, P.F.: Policy gradient learning for quadruped soccer robots. Robotics and Autonomous Systems 58(7), 872–878 (2010), iSSN: 0921-8890
6. Iocchi, L.: Robust color segmentation through adaptive color distribution transformation. In: RoboCup 2006: Robot Soccer World Cup X. pp. 287–295. LNAI 4434, Springer (2006)
7. Iocchi, L., Nardi, D.: Hough localization for mobile robots in polygonal environments. Robotics and Autonomous Systems 40, 43–58 (2002)
8. Iocchi, L., Nardi, D., Piaggio, M., Sgorbissa, A.: Distributed coordination in heterogeneous multi-robot systems. Autonomous Robots 15(2), 155–168 (2003)
9. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: Robocup: A challenge problem for ai. AI Magazine 18(1), 73–85 (1997)
10. Leonetti, M., Iocchi, L.: LearnPNP: A tool for learning agent behaviors. In: RoboCup 2010: Robot Soccer World Cup XIV (LNCS 6556). pp. 418–429 (2011)
11. Nardi, D., Clemente, G., Pagello, E.: ART: Azzurra Robot Team. In: RoboCup 1998: Robot Soccer World Cup II (1998)
12. Nardi, D., et al.: ART-99: Azzurra Robot Team. In: RoboCup 1999: Robot Soccer World Cup III. pp. 695–698 (1999)
13. Palamara, P., Ziparo, V., Iocchi, L., Nardi, D., Lima, P., Costelha, H.: A robotic soccer passing task using Petri Net Plans (demo paper). In: Padgham, P.M., Parsons (eds.) Proceedings of 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008). pp. 1711–1712. IFAAMAS Press, Estoril, Portugal (May 2008)
14. Pennisi, A., Bloisi, D.D., Iocchi, L., Nardi, D.: Ground truth acquisition of humanoid soccer robot behaviour. In: Proceedings of the 17th Annual Robocup International Symposium. pp. 1–8 (2013)
15. Valero, A., Randelli, G., Saracini, C., Botta, F., Nardi, D.: Give me the control, I can see the robot! In: Proceedings of the IEEE International. Workshop on Safety, Security, and Rescue Robotics (SSRR 2009). pp. 1–6 (2009)
16. Xue, F., Chen, X., Liu, J., Nardi, D.: Real Time Biped Walking Gait Pattern Generator for a Real Robot, vol. 7416, pp. 210–221. Springer Berlin Heidelberg (2012)
17. Zaratti, M., Fratarcangeli, M., Iocchi, L.: A 3D simulator of multiple legged robots based on USARSim. In: RoboCup 2006: Robot Soccer World Cup X. pp. 13–24. LNAI 4434, Springer (2006)
18. Ziparo, V., Iocchi, L., Lima, P., Nardi, D., Palamara, P.: Petri Net Plans - A framework for collaboration and coordination in multi-robot systems. Autonomous Agents and Multi-Agent Systems 23(3), 344–383 (2011)
19. Zonfrilli, F., Oriolo, G., Nardi, D.: A biped locomotion strategy for the quadruped robot Sony ERS-210. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 2768–2774. Washington, DC, USA (2002)

LPAD-based fall risk assessment

Luca Cattelani[1], Pierpaolo Palumbo[1], Federico Chesani[2], Luca Palmerini[1], and Lorenzo Chiari[1]

[1] DEI - University of Bologna, Italy
{luca.cattelani, pierpaolo.palumbo, luca.palmerini, lorenzo.chiari}@unibo.it
[2] DISI - University of Bologna, Italy
federico.chesani@unibo.it

Abstract. About 30% of persons over 65 are subject to at least one fall during a year. A number of published studies identify statistical relations between risk factors and the probability of fall in terms of *odds ratios*. In this paper we present a tool based on the representation of risk factors as odds ratios. Such representation is exploited to automatically build a computational logic probabilistic program, that in turn computes the fall risk of the subject given the presence/absence of risk factors in his/her status.

Keywords: Odds Ratio, Logic Programming with Annotated Disjunctions (LPAD), Fall Risk Factor, Risk Assessment

1 Introduction

Epidemiological studies have shown in the past that almost 30% of people aged 65 or more are subject to an unintentional fall each year [12]. Falls consequences range from health- and psychological-related aspects (such as hip fracture and loss of self-confidence or personal autonomy), to financial burdens supported by the relatives as well as by the social health and welfare systems. A number of publicly- and privately-financed initiatives and projects are dealing with the many aspects related to the falls, such as fall risk assessment, fall risk prevention, falls detection, fall treatment etc. In particular, fall risk assessment/evaluation consists in determining the probability of a subject to experience a fall within a certain time window. Several risk assessment tools exist (e.g., see [9,7,8,5], or the review in [4]), each tool based on a different approach and/or different assumptions. However, as discussed in [11], only few tools have been tested and applied in different settings (e.g., community, home-support, acute-care settings), while the majority of them has been "tuned" for specific settings or for specific population sub-groups.

Usually, fall risk assessment tools focus on the presence or absence of *fall risk factors* in the subject under evaluation. A huge literature is available on risk factors, defined as "aspects of personal behaviour or lifestyle, environmental exposure, or inborn or inherited characteristic, which, on the basis of epidemiological evidence, are known to be associated with falls". For an example of a

systematic review, see [6]. The majority of these contributions follow a classic epidemiological approach, and compute as results the odds ratio, w.r.t. the fall event, of the two cohorts experiencing (and not, respectively) the risk factor. The odds ratio is a measure of effect size, describing the strength of association between two values. It can be defined in terms of group-wise odds: the odds ratio is the ratio of the odds of an event occurring in one group to the odds of it occurring in another group.

Within the European project Farseeing[3] we are investigating new models for assessing the fall risk of a subject. The approach we introduce in this work aims to exploit the existing literature, and in particular the statistical results, to directly compute the probability of a fall within a year as a consequence of a patient exhibiting one or more risk factors. To this end, we introduced a light classification of risk factors (a minimal ontology), and for each factor we take into consideration the Odds Ratios (OR) published in the literature. Starting from the ORs, we generate a Logic Program with Annotated Disjunction (LPAD, [13]). The program receives as input the characteristics of a subject, in terms of the list of known risk factors affecting her/him, and computes the overall fall probability by combining each contribution following the *Distribution Semantics* [10].

2 Architecture

The overall architecture of our tool is shown in Figure 1. The information about the risk factors is stored as an ontology expressed in the OWL language: from such knowledge base, a first component computes probabilities (from odds ratios) and generates three LPAD rule sets containing higher and lower bounds (from the confidence intervals) and average odds ratios. Then, these LPADs and a subject profile are fed to a Prolog engine extended to support also LPAD clauses: the subject profile specifies which are the risk factors the subject is exposed to. The Prolog engine computes the higher, lower and average estimates of risk probability, that are returned to the user.

The Ontology. Risk factors are represented through a simple ontology, containing a list of all relevant risk factors and an odds ratio for each of them. The majority of risk factors cited in the scientific literature can be classified into three different *types*, depending on how they contribute to the fall risk:

a *Dichotomic risk factors* are the most common: either the risk factor is present or not; its contribution to the fall risk is fixed, and depends only on the presence/absence of the risk factor. Typical examples of dichotomic risk factors are Parkinson or diabetes.
b *Scalar risk factors* too are either present or not in the subject's profile. However, these risk factors are observed with a certain degree: usually with more than two possible "levels", starting from zero (absence of the factor) to

[3] `http://farseeingresearch.eu/`

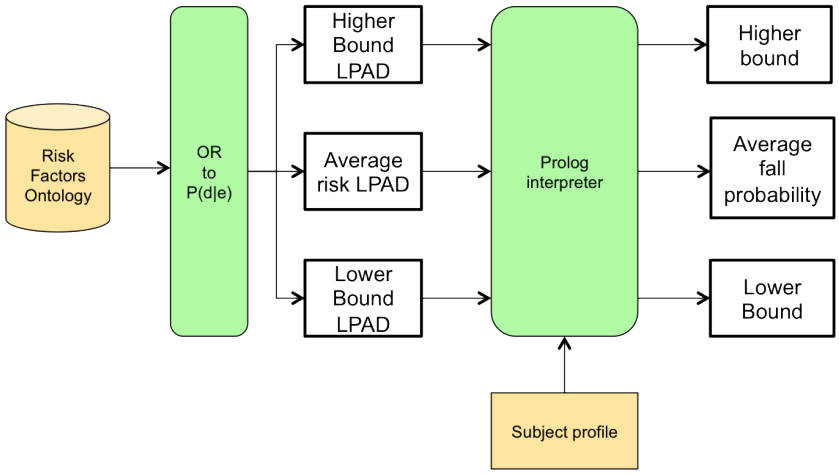


Fig. 1. Architectural overview

n (maximum magnitude of the risk factor). These risk factors contribute to the fall risk depending on their "level": a typical case is a linear contribution based on such level. Examples of this kind of factor are the age of the subject or the number of medications.

c *Synergy risk factors* exist when more than one of risk factor of other type are present contemporaneously. Synergy risk factors capture the well known nature of sets of risk factors to be synergistic and produce an increment to the risk that is bigger than the one produced by the factors when considered independently.

The ontology stores also data about the confidence intervals of the odds ratio (typically at 95%). Having confidence intervals data permits to run three risk assessing algorithms: one using the lower extremes, one using the upper extremes and another with the average values; this may give the user a feeling of the uncertainty in quantitative knowledge about risk factors.

Moreover, the ontology contains also the *estimators*, i.e. the concepts that provide indication of a subject being exposed to a risk factor. Indeed, certain risk factors can be directly characterised: e.g., either a subject suffers the Parkinson disease or not, depending on a medical diagnosis. However, there is a number of risk factors that are determined on the basis of many different assessments and criteria: e.g. , the visual impairment is a known risk factor, whose presence in a subject is determined on different medical exams (often alternative exams), such as the visual acuity on a three meter distance, the visual stereognosis, or the contrast sensitivity of the subject.

Generated LPAD The generated LPADs are sets of rules that, depending on the risk factor type, are differently defined. Given the list of estimators `L` for a specific subject, the rules are Prolog clauses of the form:

```
fall(L) : Pi :- member(Ri, L).
```

where `Pi` is the probability associated to risk factor `Ri`. Estimators can be easily taken into account by introducing further conditions in the bodies of the rules. For example, in the case of the visual impairment, we model the different estimators with the following rule:

```
fall(L) : 0.06 :- visionImpairment(L).
visionImpairment(L) :- member(visualAcuity3M(X),L), X =< 5.
visionImpairment(L) :- member(visualStereognosis(X),L), X =< 3.
visionImpairment(L) :- member(contrastSensitivity(X),L), X =< 16.
```

Subject profile The subject profile is a simple Prolog list of the estimators that characterise the subject. It has the form: `[age(71), 'Parkinson', 'diabetes']`, indicating that for example the subject has an age of 71, and she/he suffers diabetes and Parkinson diseases.

Moreover, the current version of our tool supports three different alternatives:

- if the subject suffers a risk factor, such risk factor is listed in the profile and it is labelled with a term `true`, and the risk probability associated to the factor directly contributes to the overall fall risk;
- if the subject does not suffer the risk factor, then the factor is not present in the list
- if the subject is unsure about suffering a risk factor, then the factor is present in the list with a label "unknown": in that case specific LPAD rules are used, and the distribution probability of the specific risk factor over the population is used to compute the overall fall risk.

3 Conclusions

In this paper we have introduced a tool for assessing the fall risk depending on a specific subject profile. Our architecture is still in a prototypical stage, and many aspects still need to be researched and developed. At the moment of writing, we are evaluating the quality of our tool by using a prototype based on the statistical findings in [6], and the InChianti data set available within the Farseeing Project. Although first results show that our approach performs similarly to other existing approaches, the validation is far to be completed, and definitely more investigation is required.

The current implementation is based on the findings in the scientific literature. We plan as future work to apply learning algorithms, like Expectation-Maximization, for parameters and structures of LPADs [3,2,1], in particular on the datasets available within the Farseeing Project: for example, the possibility of using the same LPAD structure but with parameters learned on specific situations is a promising way to customize the tool and gain better results for specific cases.

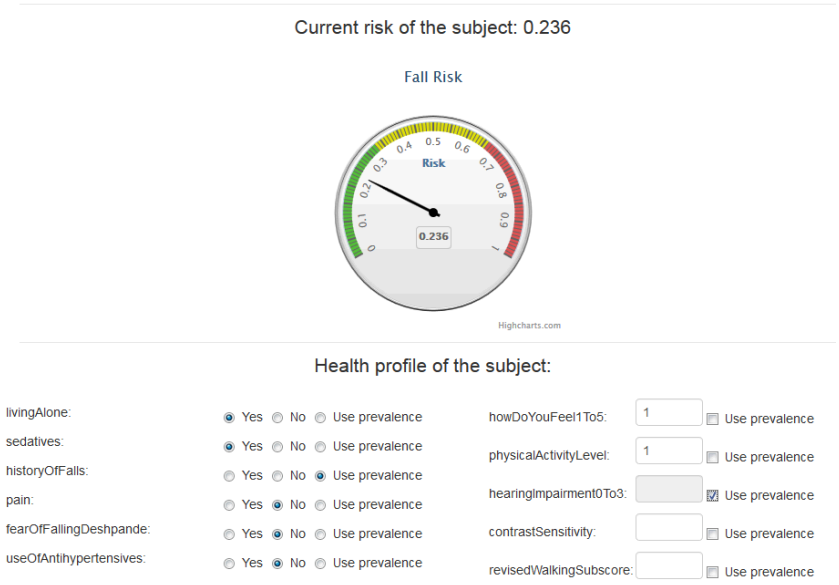


Fig. 2. Web-based interface to the Fall Risk Assessment Tool - FRAT

Finally, we are implementing a web-based application (see Figure 2), that allows the users to define the subject profile by means of a sort of questionnaire, and directly computes the fall risk.

Acknowledgments This work is supported by the FARSEEING project, cofunded by the European Commission, Seventh Framework Programme, CooperationICT, Grant Agreement no. 288940.

References

1. E. Bellodi. *Integration of Logic and Probability in Terminological and Inductive Reasoning*. PhD thesis, University of Ferrara, Italy, 2013.
2. E. Bellodi and F. Riguzzi. Learning the structure of probabilistic logic programs. In *Inductive Logic Programming 21st International Conference, ILP 2011, London, UK, July 31 - August 3, 2011. Revised Papers*, volume 7207 of *LNCS*, pages 61–75, Heidelberg, Germany, 2012. Springer.
3. E. Bellodi and F. Riguzzi. Expectation Maximization over binary decision diagrams for probabilistic logic programs. *Intelligent Data Analysis*, 17(2):343–363, 2013.
4. J. C. T. Close and S R Lord. Fall assessment in older people. *BMJ*, 343(sep14 1), September 2011.
5. G. Cuaya, A. Muñoz Meléndez, L. N. Carrera, E. F. Morales, I. Quiñones, A. I. Pérez, and A. Alessi. A dynamic Bayesian network for estimating the risk of falls from real gait data. *Medical & biological engineering & computing*, 51(1-2):29–37, February 2013.

6. S. Deandrea, E. Lucenteforte, F. Bravi, R. Foschi, C. La Vecchia, and E. Negri. Risk Factors for Falls in Community-dwelling Older People: A Systematic Review and Meta-analysis. *Epidemiology*, 21(5):658–668, 2010.
7. K. Delbaere, J. C. T. Close, J. Heim, P. S. Sachdev, H. Brodaty, M. J. Slavin, N. A. Kochan, and S. R. Lord. A multifactorial approach to understanding fall risk in older people. *Journal of the American Geriatrics Society*, 58(9):1679–85, September 2010.
8. D. Giansanti, G. Maccioni, S. Cesinaro, F. Benvenuti, and V. Macellari. Assessment of fall-risk by means of a neural network based on parameters assessed by a wearable device during posturography. *Medical engineering & physics*, 30(3):367–72, April 2008.
9. M. Marschollek, M. Gövercin, S. Rust, M. Gietzelt, M. Schulze, K. Wolf, and E. Steinhagen-Thiessen. Mining geriatric assessment data for in-patient fall prediction models and high-risk subgroups. *BMC medical informatics and decision making*, 12(1):19, January 2012.
10. T. Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP*, pages 715–729, 1995.
11. V. Scott, K. Votova, A. Scanlan, and J. Close. Multifactorial and functional mobility assessment tools for fall risk among older adults in community, home-support, long-term and acute care settings. *Age Ageing*, 36(2):130–139, Mar 2007.
12. A.M Tromp, S.M.F Pluijm, J.H Smit, D.J.H Deeg, L.M Bouter, and P Lips. Fall-risk screening test: A prospective study on predictors for falls in community-dwelling elderly. *Journal of Clinical Epidemiology*, 54(8):837 – 844, 2001.
13. J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In B. Demoen and V. Lifschitz, editors, *ICLP*, volume 3132 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2004.

VEGA-QSAR: AI inside a platform for predictive toxicology

Emilio Benfenati[1], Alberto Manganaro[1] and Giuseppina Gini[2]

[1]IRCCS- Istituto di Ricerche Farmacologiche Mario Negri, Milano, Italy
{benfenati, manganaro}@marionegri.it
[2]DEIB, Politecnico di Milano, Italy
giuseppina.gini@polimi.it

Abstract. Computer simulation and predictive models are widely used in engineering, much less considered in life sciences. We present an initiative aimed to establish a dialogue within the community of scientists, regulators, industry representatives, offering a platform which combines the predictive capability of computer models, with some explanation tools, which may be convincing and helpful for human users to derive a conclusion. The resulting system covers a large set of toxicological endpoints.

1 Introduction

Predictive toxicology is using models to predict biological endpoints, in particular toxicity, without making real experiments. The concept of "Structure-Activity Relationship" (SAR) is that the biological activity of a chemical can be related to its molecular structure. When quantified, this relationship is known as "QSAR". A QSAR model makes use of existing experimental toxicity data for a series of chemicals to build a model that relates experimentally observed toxicity with molecular descriptors in order to predict the toxicity of further chemicals. The term predictive toxicology [1] has been introduced by the AI community to indicate this approach to toxicology.

A series of regulations require producing information about the safety of the chemical substances, such as the European legislation REACH. This regulation states that for each chemical circulating in Europe a complete dossier on physico-chemical, environmental and toxicological properties has to be compiled. In order to prevent an over-usage of animal testing, REACH foresees the use of alternative methods, including predictive programs.

Life sciences are heavily impacted by the development of methods for data collection and analysis; they are moving from an analytical approach to a modelling approach. An important step in this direction is played by the changing mind from purely statistics usage of data to the data mining and machine learning view of the

recent years. Good models should (1) explain patterns in data; (2) correctly predict the results of new experiments or observations; and (3) be consistent with other ideas (models, beliefs). Of course the (3) requirement is the most critical one.

Today some QSAR models have proved to offer a valuable alternative to the classical *in vivo* methods [2]. Nevertheless, most of the QSAR models are not trusted by their targeted users, for several reasons [3], including the misunderstanding of the technology. We decided to cope with those issues through an open platform dedicated to the stakeholders potentially interested in using QSAR models.

This paper presents the developed platform. The user needs, captured during several workshops, interviews and exercises carried on through four recent European projects (DEMETRA, CAESAR, ORCHESTRA and ANTARES) have strongly guided the platform development. According to the user's requirement to keep as confidential the chemical structures, our solution is implemented both as a web-based application and as down-loadable software.

2 Using the VEGA-QSAR platform

Several institutes contributed to the development of the platform, called VEGA-QSAR, including regulators and public bodies in Europe and USA. VEGA freely offers tens of models for properties such as persistence, logP, bioconcentration factor (BCF), carcinogenicity, mutagenicity, skin sensitization.

The initial nucleus of VEGA models derives from the CAESAR models[1]. Other models have been added to simulate the models developed by the partners; this is the case of models developed by EPA (US Environmental Protection Agency) and ISS (Istituto Superiore di Sanità) for instance. All the models have been published in scientific literature before incorporating into VEGA. Moreover all the models have been successfully benchmarked against the few commercially available systems.

The steps of the workflow are clearly indicated in the GUI: insert the list of the molecules identifiers, choose where to send the prediction output, ask prediction, and get results. The input can be given in different standard formats used in the chemical domain, including SMILES and SDF files [4]. To avoid the well-known problems about the non-unique representations VEGA transforms all the chemical structure into a unified internal string format.

Figure 1 show, as an example, the output screen of the BCF model [5] with the prediction and the most similar compounds with their experimental and predicted values. BCF is a dose value, however for regulation classes are assigned according to thresholds. Since the uncertainty of the prediction can be calculated, it is graphically shown for each molecule as a worst-case analysis, as in Figure 2.

[1] http://journal.chemistrycentral.com/supplements/4/S1

Furthermore, the model provides other pieces of information useful to the evaluator, such as a plot showing the experimental values of the training set (Figure 3), to check for possible unusual behaviour of the target compound.

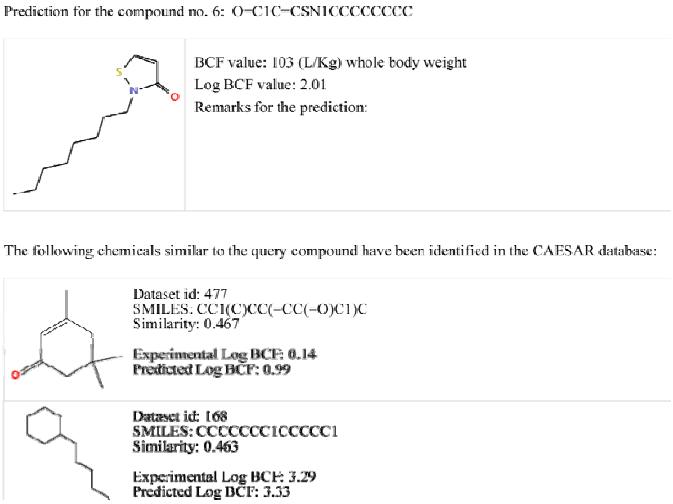


Figure 1 – Prediction of the bioconcentration factor (logBCF) for the compound and the most similar structures available in the dataset, with experimental and predicted values.

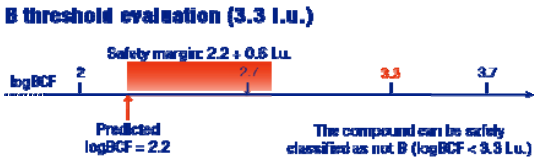


Figure 2 – The BCF model suggests the classification in the classes defined in REACH**.**

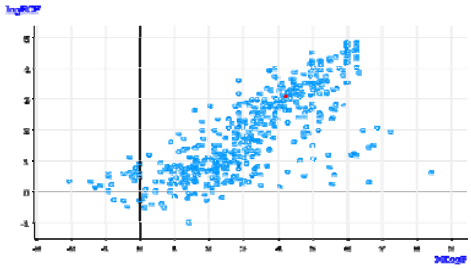


Figure 3 – The predicted logBCF value (red dot) inside the experimental values of the dataset.

The overall reliability of the prediction is measured by combining statistical values, elements of case based reasoning, and possibly presence of active substructures; the possible reasons of concern are underlined. All those considerations are weighted and summed up in an index (in 0 – 1) that is called Applicability Domain Index (ADI). Moreover the user can apply different models to the same endpoint, and VEGA suggests the best integration, as illustrated in Section 5.

3. AI inside VEGA

VEGA is mainly the result of the data driven and knowledge discovery approaches. The models implemented arise from different methods. At least three families are represented: (1) rule based expert systems, usually codified as the presence of chemical substructures called structural alerts (SA), defined by experts, as in Toxtree [7]; (2) data miners that extracts relevant fragments from the analysis of their correlation with the endpoint, as in SARpy [6]; (3) regression models that use molecular descriptors and non linear methods - ANN, SVM - as the mentioned BCF model; (4) ensemble methods as random forests as in Developmental toxicity; (5) hybrid models that mix the above methods as in CAESAR-mutagenicity.

The toxicity domain poses significant challenges to the AI methods. The first is the lack of available knowledge about reasons and mechanisms of toxicity for many of the endpoints that make it impossible to apply deductive approaches. The purely symbolic approach can be used for “mechanistic interpretation”, which is a vague indication about the toxicological pathway associated to that chemical on the basis of the presence of a specific chemical subgroup. Moreover, even in case some SAs are known, their presence is only a sufficient condition for toxicity; so the absence of SA in a molecule does not guarantee its safety.

This is the reason why we have developed new methods as in SARpy [6], with the aim of discovering both new SA and ”neutral substructures” (NS) of the molecule that may reinforce the classification as safe for molecules not containing SA but instead containing some NS.

QSAR systems make more use of probabilistic AI than symbolic AI, with the consequent problems of difficulty in understanding the results. There is generally a trade-off between prediction quality and interpretation quality of a model. Interpretable models are desired to make expert decisions; however those models suffer because the generalizations necessary to get them may be flawed by lack of enough data. To avoid the risk of excess generalization, often QSARs are simple linear regression in the small population of a chemical class. Those models have no predictive value outside this small population.

In VEGA models are generally not intended to provide transparency in se, but high accuracy on new data that the model has not used in training; since transpar-

ency is needed, this is obtained adding extra visualization and explanation features to the models, as presented in the previous subsection.

What is needed is a way to predict new chemicals and to deal with real substances that generally are mixtures of quite complex molecules, as in dyes and fuel, and that can be better modelled as large SAs and NSs.

4 Chemical space analysis tool

The identification of chemical features and their role in the prediction offer important modulating information about models. The idea is to find, for each model, a set of chemical features (i.e. molecular fragments or functional groups) that can identify a chemical class, and that show a statistically relevant correlation with the reliability of the prediction. If a set of chemical features are found only in compounds that yield accurate predictions, or vice versa non-accurate, this finding can provide also a more chemical-oriented explanation of the model's behaviour.

Two sets of fragments have been considered and implemented in VEGA and freely available[2]: Functional Groups (FC) that account for 154 chemical groups, and Atom-Cantered Fragments (ACF), for 115 fragments, each one corresponding to a type of atom with different connectivity. These fragments have been studied by experts in the fields of molecular descriptors [8], and have precise chemical meaning.

The software to analyse the chemical space checks for the presence of the above mentioned FG and ACF, then reports, for each of these chemical features, the total number of matches, the number of matches in each class, and its percentage. For each model, it t is possible to check the percentages of correct predictions for those fragments.

For example, for BCF all the compounds used in the model have been labelled as: correct prediction (the possible error is [-1, 1] log units), under-estimated (the predicted value is lower of at least 1 log units with respect to the experimental value), over-estimated (the predicted value is higher of more than 1 log units). We discovered that BCF does not always provide reliable predictions when more than one halogen atom is in the molecule. Moreover multiple halogens, and in particular multiple Fluorine atoms, lead to under estimated predictions.

Considering the CAESAR mutagenicity model, we discovered that the presence of hydroxylamines in a molecule leads to unreliable predictions; in particular in case of multiple hydroxylamines. Furthermore, this chemical feature is mostly related to false negatives, so it could be used as an alert. Instead halogenated compounds mostly give false positive predictions, and this can generate a warning.

The analysis of the molecule in the chemical space of each used model gives extra information to the stakeholder.

[2] http://kode-solutions.net/en/software_ist_chemfeat.php

5 Integrated models

During the ANTARES project we collected large data sets for some toxicity endpoints and benchmarked the available models. Since more models are implemented in VEGA for the same endpoints, we have devised different integration strategies according to the characteristics of models.

1 - *Trust the best model*. If one strong model clearly outperforms the others, the integration suggests:

- use the best model; check for possible mechanistic interpretation;
- if no prediction from the best model, then trust the others according to their applicability domain index value.

This strategy applies to ready biodegradability, where VEGA-SARpy outperforms the others, as in the ROC space in Figure 4. However BIOWIN will provide good results for chemicals out of the applicability domain of SARpy.

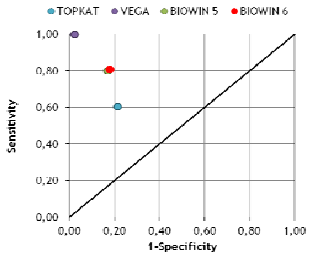


Figure 4 - ROC for molecules in test set for some ready biodegradability models

2 – *Take the best model and refine with chemical and SAs information***.** If the best model has a low accuracy, use it and check it for the chemical class and possibly for SAs.

This applies to carcinogenicity, where CAESAR model outperforms the others, as in the ROC of Figure 5, but is not very strong. The integrated strategy is.

- Take the prediction of CAESAR and check with Toxtree for possible SAs; if the results agree, accept, otherwise refine considering chemical classes

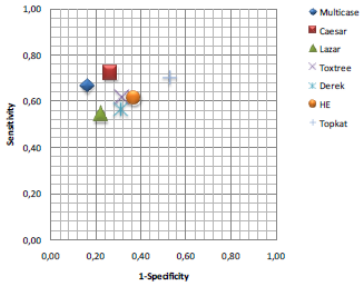


Figure 5 – ROC points for some carcinogenicity models.

3 – *Build a decision tree*. When two regression models have good performance and use different methods, it is wise to use them together.

This applies to BCF, to integrate the two VEGA models CAESAR and Meylan. The steps are:

- Compare the two values predicted by CESAR and Meylan.
- If their difference is less than 1 log unit, then exit with the highest value;
  else consider the 2 ADI indexes;
  - if none of them is greater than 07, then discharge the prediction and exit,
  else chose the predicted value with highest ADI index.

This integrated model [9], considering only the molecules in the new ADI, has R^2 =0.92, sensitivity of 85%, RMSE of 0.44 log units, with a considerable improvement from both the starting models

4 – *Make a mechanistic integration*. For good models it is possible to find a mechanistic integration, now considering the available SA-based models, in future automatically constructing them, on demand, using SARpy.

We see how this applies to mutagenicity. We run SARpy and CAESAR models. In case there is a consensus and the ADI is > 0.8, prediction is strongly supported. Otherwise, the SAs given by SARpy are to be considered. In this integration we obtain both high reliability and mechanistic interpretation.

5 – *Make statistic integration***.** In this case various models are ensembled through a gating network.

We see it again for mutagenicity that having a large dataset of more than 6000 molecules is a good candidate to statistical integration. We used a self-organizing tool of KnowledgeMiner Software[3], based on GMDH neural networks. For ensemble development the dataset was randomly half subdivided into training and testing datasets. For each compound the input vector contains the experimental class label, the prediction of the SARpy model and its applicability domain index, the predictions of the CAESAR model and its applicability domain index. The overall accuracy of the ensemble model is 88.6%, with sensitivity of 89.7%, and specificity of 87.5% that increases the classification accuracy of up to 10% relative to the individual models.

6 Final remarks and conclusions

In the development of the VEGA platform we analysed the needs of the users, and the barriers to the use of computers. We made AI approaches working in the language of the user, in order to improve understanding and acceptance of the technology [10].

[3] http://www.knowledgeminer.com

Stakeholders are continuously involved, and their feed-back used to improve the platform. Recently, an exercise [11] with tens of experts demonstrated that human experts identified reasons of possible concern evaluating the results of some QSAR models. Comparing VEGA with other platforms, the users appreciated the details and supporting information provided by VEGA.

Another important aspect is that models are inserted into VEGA-QSAR after an independent scientific evaluation, which guarantees not only their validity but also their ability to emulate the in-vivo tests results. Please keep in mind that the reproducibility of in-vivo tests is never 100%; for difficult endpoints as carcinogenicity a value less than 70% is reported, for mutagenicity 85% is reported. Predictors that have a similar accuracy have application potency.

VEGA at http://www.vega-qsar.eu contains the web application and the downloadable software. So far more than one thousand users downloaded it.

Acknowledgments

We acknowledge support of the EU Life+contracts CALEIDOS and PROSIL.

References

1. Gini G, Katritzky A. (Eds.) (1999) Predictive Toxicology of Chemicals: Experiences and Impact of Artificial Intelligence Tools. AAAI Spring Symposium on Predictive Toxicology, AAAI Press, Menlo Park, California.
2. Benfenati E., Crètien J.R., Gini G., Piclin N., Pintore M., Roncaglioni A. (2007) Validation of the models. In Quantitative Structure-Activity Relationships (QSAR) for Pesticide Regulatory Purposes, Elsevier, 185-200.
3. Benfenati E. et al. (2011) The acceptance of in silico models for REACH: Requirements, barriers, and perspectives. Chemistry Central Journal, Vol 5:58, 1-11.
4. Weininger D. (1988) SMILES a chemical language and information system. 1. Introduction to methodology and encoding rules. J. Chemical Information and Computer Science 28, 31–36.
5. Lombardo A., Roncaglioni A., Boriani E, Milan C., Benfenati E. (2010) Assessment and validation of the CAESAR predictive model for bioconcentration factor (BCF) in fish. Chemistry Central Journal, Jul 29; 4 Suppl 1:S1.
6. Gini G. et al. (2013) Automatic knowledge extraction from chemical structures: the case of mutagenicity prediction. SAR and QSAR in Environmental Research, Vol 24, N 5, 365-383.
7. Toxtree (2012) software download: http://sourceforge.net/projects/toxtree/
8. Todeschini R., Consonni V.(2009) Molecular Descriptors for Chemoinformatics. Wiley-VCH.
9. Gissi A. et al (2013) Integration of QSAR models for bioconcentration suitable for REACH. Science of the Total Environment 456-457, 325–332.
10. Gini G. (2013) How Far Chemistry and Toxicology Are Computational Science?. in Amigoni and Schiaffonati (eds): Methods and experimental techniques in computer engineering. Springer, 15-33, ISBN 978-3-319-00272-9 .
11. Benfenati E. et al. (2013) Using toxicological evidence from QSAR models in practice. Altex, Vol. 30, 19-40.

AngryHEX: an Artificial Player for Angry Birds Based on Declarative Knowledge Bases

Francesco Calimeri[1], Michael Fink[2], Stefano Germano[1], Giovambattista Ianni[1], Christoph Redl[2], and Anton Wimmer[2]

[1] Dipartimento di Matematica e Informatica, Università della Calabria, Italy
{calimeri,ianni}@mat.unical.it, stefanogermano0@gmail.com
[2] Institut für Informationssysteme, Technische Universität Wien
{fink,redl}@kr.tuwien.ac.at, anton.wimmer@tuwien.ac.at

1 Introduction

This work presents a joint project by the University of Calabria (UniCal) and the Vienna University of Technology (TU Vienna), aiming at developing an Intelligent Agent participating in the 2013 Angry Birds Artificial Intelligence Competition [1]. Angry Birds is a very popular video game where the main goal is to shoot at some pigs by means of birds of different characteristics from a slingshot. The game field (which is static until the player moves) features some structures that shelter pigs. Structures can be very complicated and can involve a number of different object categories with different properties, like wood, ice, stone, etc. The game scenario evolves largely complying with physics laws on a bi-dimensional plane; thus, it is possible, in principle, to infer how a structure will change if hit at a certain position by a certain bird.

The Angry Birds AI Competitions [1] are designed to test the abilities of Angry Birds artificial agents, playing on a variety of levels, on the Google Chrome version of the game. The competition runs on a client/server architecture, where the server runs an instance of the game for each participating agent. Each agent runs on a client computer, and communicates with the server according with a given protocol that allows agents to fetch screenshots of their own game screen at any time. An artificial player can also obtain the current high scores for each level, and can prompt the server for executing a shot, which will in turn be performed in the corresponding game screen. The long term goal of the Competition is to foster the building of AI agents that can play any new level better than the best human players. In order to successfully solve this challenge, participants are solicited to combine different areas of AI such as computer vision, knowledge representation and reasoning, planning, heuristic search, and machine learning. Successfully integrating methods from these areas is indeed one of the great challenges of AI.

2 ASP and HEX Programs

Our agent, called AngryHEX, models its internal knowledge of the game by means of an Answer Set Programming (ASP) [2,4] knowledge base. ASP became

widely used in AI and is recognized as a powerful tool for knowledge representation and reasoning (KRR), especially for its high expressiveness and the ability to deal also with incomplete knowledge [2]. The fully declarative nature of ASP allows one to encode a large variety of problems by means of simple and elegant logic programs. The semantics of ASP associates a knowledge base with none, one, or many "answer sets", each usually in one-to-one correspondence to the solutions of the problem at hand. Among the different extensions of ASP, we used HEX programs [3], as they are particularly well-suited when external knowledge and/or external sources of computation must be integrated within the same knowledge base.

3 The AngryHEX agent

We started the development of AngryHEX from the Base Framework provided by the organizers and enriching it with new functionalities, particularly the AI. The actual reasoning is carried out by computing the answer sets of an HEX-program P_{AI} that models the knowledge of the game. P_{AI} is enriched with scene information, and its answers sets encode the targets of choice.

3.1 Base Framework

The Base Framework consists of many parts (see fig. 1). The *Angry Birds Extension* works on top of the Google Chrome browser, and allows to interact with the game by offering functionalities such as capturing the game window or executing actions (e.g., moving the mouse, clicking, zooming). The *Vision Module* segments images, recognizes the minimum bounding rectangles of essential objects, such as birds of various colors and types, pigs, the slingshot, or bricks made of several materials. The *Trajectory Module* estimates the trajectory that a bird would follow given a particular release point set at a given distance and orientation with respect to the slingshot. The *AI Agent* stub is supposed to include the artificial intelligence programmed by participants. The *Game Server* interacts with the *Angry Birds Extension* via the *Proxy* module, which can handle commands like `CLICK` (left click of the mouse), `DRAG` (drag the cursor from one place to another), `MOUSEWHEEL` (scroll the mouse wheel), and `SCREENSHOT` (capture the current game window). The *Server/Client Communication Port* receives messages from agents and sends back feedbacks after the server executed the actions asked by the messages. There are many categories of messages (Configuration messages, Query messages, In-Game action messages and Level selection messages).

3.2 Our extensions to the Base Framework

We extended the Base Framework adding an Executor that communicates with the DLVHEX solver; the Executor encodes the information about the current scene into logic assertions, runs DLVHEX on the composed HEX-program and

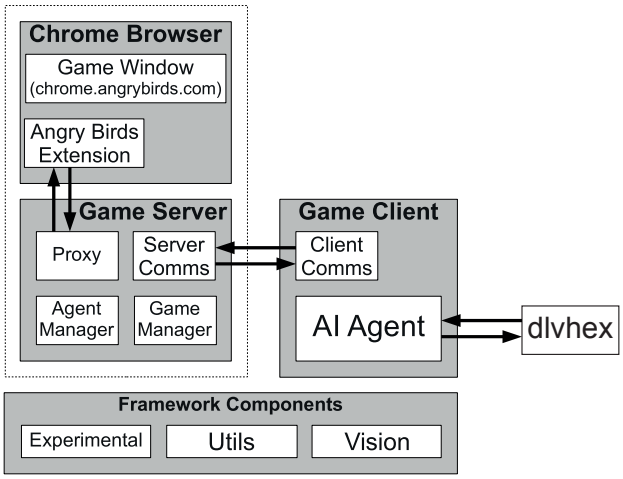


Fig. 1: Base Architecture

parses the output, thus identifying the next target. We also improved the Vision Module, adding the possibility of recognizing the orientation of the blocks; indeed, the Base Framework does not correctly recognize blocks that are not parallel to the coordinate axes. The Trajectory Module has been extended with the possibility of aiming a trajectory at the left and top side of objects; ideed, the original module within the Base Framework aims only at the centroid of an object, even if it is "covered" by other objects, and thus not directly shootable. We implemented also other minor adjustments, fixes and improvements.

3.3 Artificial Intelligence

We developed two different Intelligence layers for AngryHEX: the **Tactic** layer, which plans shots, and decides how to complete a level; and the **Strategy** layer, which decides the order of levels to play and possible multiple attempts to the same level. The **Tactic** layer is declaratively implemented using the DLVHEX solver, which computes optimal shots based on information about the current scene and on knowledge modeled within the HEX program P_{AI}. In particular, the **Tactic** layer behaves according to the following scheme:

- Input: Scene information encoded as a set of logic assertions S (position, size and orientation of pigs, ice, wood and stone blocks, slingshot, etc. as obtained by the Vision Module); a knowledge base P_{AI} encoding knowledge about the gameplay;
- Output: "Answer sets" of $P_{AI} \cup S$, that contain a dedicated atom describing the target to hit, and some information about the required shot;
- Physics simulation results and other information are accessed within P_{AI} via the so-called External Atoms constructs. External atoms allow to access information which is external to the knowledge bases, like, e.g.,

- asking if an object is "stable" (prone to an easy fall);
- asking if object B will fall if A falls;
- asking which objects intersect with the trajectory of a bird after hitting a given object;
- computing distances between objects;
- asking if an object O is "shootable" (i.e., there exist a trajectory where O is the first intersecting object);
- finding the best trajectory for a White Bird (the white birds have a peculiar behavior and require a special treatment);
- asking if an object can be "pushed" by another.

The knowledge modelled by P_{AI} is the following:

- Define each shootable target as an object which has a direct and unobstructed path from the slingshot;
- For each shootable target T, compute a measure of the estimated damage that can occur on all other objects if T is hit. The specific behavior of each bird type is taken into account (e.g. yellow birds are very effective on wood, etc.), and the estimated damage is properly weighted by a probability value;
- Targets are ranked by their priority, e.g., maximizing the estimated damage to pigs first; estimated damage to other objects is taken into account only for targets which tie in the estimated damage for pigs.

As an example of a logic rule used in AngryHEX, the following "computes" the likelihood of damage when an object pushes an adjacent one:

$$\begin{aligned} pushDamage(Obj_B, P_A, P) \leftarrow\ & pushDamage(Obj_A, \_, P_A), \\ & P_A > 0, \&canpush[ngobject](Obj_A, Obj_B), \\ & pushability(Obj_B, P_B), P = P_A * P_B/100. \end{aligned}$$

here the $\&canpush$ predicate out-sources to a C++ library the geometric calculations required to assess whether an object is within range of another (i.e. Obj_A can make Obj_B fall by domino effect); on the other hand, the *pushability* predicate is defined using empirical knowledge of the game and defines how much an object can be "pushed" in terms of its shape and material (e.g. long rods are very pushable, etc.).

We have developed also other rules that take into account the direct damage and the fall damage of each object. These depend on the probability of destroying a particular object type, and on the importance of the fall of an object, given its material type. Static object damage values are combined in order to form causal chains which terminate using an energy loss estimate. Energy losses take into account the residual energy available when the effects of a shot propagate from an object to another. For instance the following assertions:

$$damageProbability(blue, wood, 10).\ damageProbability(yellow, wood, 100).$$
$$damageProbability(blue, ice, 100).\quad damageProbability(yellow, ice, 10).$$

encode the damage probability of an object depending on the object material and on the type of bird hitting the object at hand. Such values were encoded

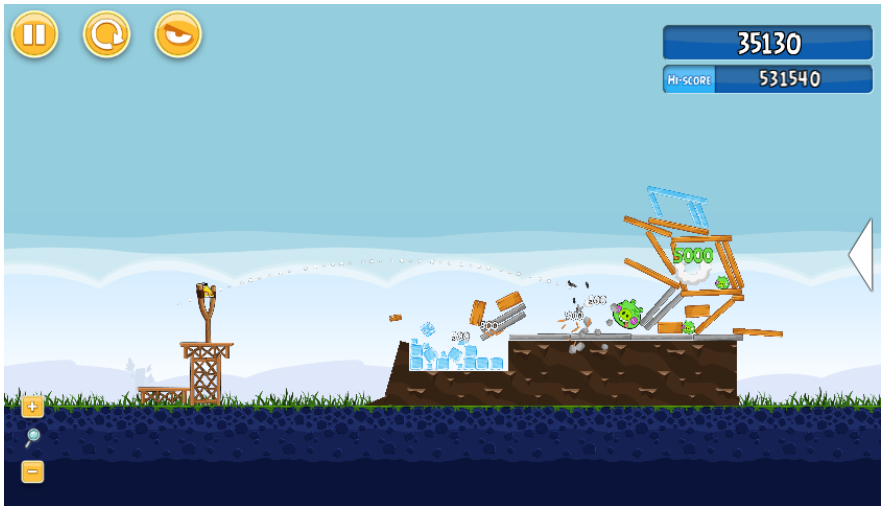


Fig. 2: A successful shot featuring an exploding black bird.

empirically based on specific knowledge of the game (e.g. blue birds are very effective on ice, etc.).

The **Strategy** layer decides, at the end of each level, which level to play next. This layer is implemented in Java according to the following scheme:

1. First, each available level is played once;
2. Then, levels where the agent score differs most from the current best score are selected (up to a limited number of attempts);
3. Next, levels where AngryHEX achieved a score higher than the current best scores and that have the minimum difference from the best score, are selected (up to another limited number of attempts);
4. Finally, the **Strategy** layer tries to play a random level (actually, this never occurred during the available timeframe in the competition rounds).

For each level, the **Strategy** layer keeps tracks of previously achieved scores and previously selected initial target objects. Previously selected targets are excluded, in order to force a different tactic per each attempt on the same level.

4 Computational Performance Considerations

Since the tactic layer is repeatedly prompted for deciding the chosen target on the current scene, its efficiency is crucial for achieving good time performance of the overall agent. The core of the tactic layer is an evaluation carried over the logical knowledge base P_{AI} coupled with a set of assertion S describing the current scene. Thus, both the size of S and P_{AI} affect in principle the performance of the tactic layer. Another performance factor concerns the time performance and the number of calls to external libraries.

Concerning the size of S, this is directly proportional to the number of objects in the current scene: there is one fact per each object in the scene, plus a constant number of facts (three) encoding respectively, the slingshot position, the current bird type, and the scale factor of the scene. P_{AI} is instead a fixed logic program comprising about 3 hundred statements, made of both rules and facts encoding fixed knowledge of the domain. For what calls to external libraries are concerned, these were optimized for reducing the number of possibly redundant calls[3]. External calls proved to be a key-point for what the impact on efficiency is concerned. Indeed, by means of external calls, we were allowed to out-source to imperatively coded modules a number of tasks in which ASP and logic programming in general are not well-tailored at, like simulation of future scenarios according to laws of physics, approximation of ballistic trajectories, etc.

With the help of an hybrid architecture featuring access to external sources of knowledge, we found that the capabilities of current ASP solvers fit very well the setting of the Angry Birds game and its AI Competition: first, Angry Birds can be seen as a game in between a real-time game and a turn-based one, allowing a fairly large time window for deciding the next shot[4]; and second, the size of the search space of the tactic layer is linearly proportional to the number of objects in the current scene. Note that the above setting can be lifted to a variety of other contexts in which *a)* there is an acceptable time window available for reasoning, and *b)* the set of actions that can be taken by the agent at hand is fairly small, do not underlies an exponentially larger search space, and no look-ahead of arbitrary depth on future scenarios is required (or convenient) to be performed. This generalized context covers e.g. automated room cleaning, semi-interactive turn-based games, etc.

Concerning actual performance, we could not yet perform accurate benchmarking, due to the unavailability to the public of the levels used in the Angry birds AI Competition. However, our testing on the publicly known first 21 levels of the "Poached Eggs" set[5], which reflect the competition setting in qualitative terms (type of materials, shape of objects and type of birds available), allowed us to draw some conclusion. First, the reasoning time was a negligible fraction with respect to the overall time of each shot. Most of the shot time is indeed taken by the animation and simulation of the shot itself (within tens of seconds), while reasoning takes generally less than 1-2 seconds, with some outlier case of no more than 5 seconds; second, given the low reasoning times, we could not appreciate any correlation between the reasoning time and the number of objects available[6], although we expect such correlation should appear clearer in more complex levels.

[3] Also, the burden of repeated identical calls to external sources has been mitigated by caching.

[4] In the Angry Birds AI Competition, depending on the round at hand, each shot was allowed about 10-20 seconds.

[5] "Poached Eggs" is the first level set available in the downloadable version of the game.

[6] In the "Poached Eggs" level set, the number of objects ranges from 8 to 65, including "ground" objects describing terrain areas. Levels proposed to participants in the

5 Conclusion

AngryHEX performed very well in the 2013 Competition, reaching semifinals. Notably, AngryHEX kept the first place out of 20 participants, in both the two qualification rounds. Benchmarking performed by the Competition Organizer after the Competition also shows AngryHEX ranking with the best score in the first 21 levels of the "Poached Eggs" level set. The levels used in the semifinal and final stage are unfortunately not publicly available; thus, we could not assess the quality of performance of AngryHEX on them, nor understand the reasons of the lower performance. Nevertheless, we conjecture that levels used in finals and semi-finals are very rich in the number of objects, thus triggering some scalability issues in the architecture of our agent.

We identified many points in which AngryHEX can be improved. In the future, we aim at introducing the planning of multiple shots based on the order of birds that must be shot; we think that it might be useful also to better study the interaction between objects. Some other possible improvements concern performance, and the declarative implementation of the **Strategy** layer.

References

1. Angry Birds AI Competition. `http://aibirds.org`
2. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
3. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer Set Programming. In: International Joint Conference on Artificial Intelligence (IJCAI) 2005. pp. 90–96. Edinburgh, UK (Aug 2005)
4. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)

AI Competition qualification rounds were only slightly more complex with levels reaching up to 100 objects.

Automated Landslide Monitoring through a Low-Cost Stereo Vision System

Mauro Antonello[1], Fabio Gabrieli[2], Simonetta Cola[2], and Emanuele Menegatti[1]

[1] Department of Information Engineering (DEI), via Gradenigo 6/B, Padova
[2] Dip. di Ingegneria Civile Edile ed Ambientale (ICEA), via Ognissanti 39, Padova

Abstract. In this paper we introduce an inexpensive yet efficient photogrammetry system that takes advantage of state of art computer vision techniques to monitor large natural environments. Specifically, our system provides a precise evaluation of the terrain flow in wide landslides through optical flow applied to 2D image sequences and a back-projection of the resulting motion gradients to a 3D model of the landslide. Providing such a wide 3D model is one of the key issues and is addressed relying to a wide baseline stereo vision system. To initialize the stereo vision system, we propose an effective multiview calibration process.

Keywords: Photogrammetry, Stereo Vision, Multiview Calibration

1 Introduction

In last few years, the significance of environmental monitoring for natural hazards prevention and mitigation has been constantly growing. Responsiveness requirements along with an increased amount of data coming from ambient sensors led to the necessity of automated systems able to detect critical situations and alert authorities.

In this work we address the problem of landslide monitoring, that means detecting the flow of landslide material, a motion that is very limited: usually only few meters over several weeks. The slipping of the landslide material is often monitored analyzing image sequences exploiting optical flow techniques. One of the main limitations of such process is that the direction and intensity of the flow are the projection in the camera plane of the real world flows. Therefore, in order to obtain a correct estimation of the material motion, flow gradients must be back-projected to the landslide 3D model. Unfortunately, this 3D model is hard to obtain due to the wideness of the monitored area.

Several works presented in the literature rely on expensive laser-scanner or aerial photogrammetry systems. Differently, our work propose an innovative stereo vision system that requires only two cameras, which ultimately makes it a low-cost alternative for large environmental 3D reconstruction. Stereo vision is one of the most widely used techniques for outdoor 3D reconstruction [1], especially in low cost solutions. Stereo vision only requires two cameras and can retrieve the distance of an area observed by both cameras from its slightly different appearance in the two images (see fig. 1). In our case, the large distance of

monitored areas and the presence of adverse natural conditions, like heavy wind or bad weather, make the calibration of a stereo vision system an hard task. In order to maintain a good reconstruction quality at farthest monitored areas, the distance between monitoring cameras (baseline, see sec. 2) needs to be a lot higher than what is commonly the case when calibrating a stereo camera pair. Indeed, the common way to calibrate a stereo vision system and find its extrinsic patameters (i.e. cameras mutual position) has been proposed by [2] and makes use of a small pattern with known geometry; however, this pattern needs to be observed by both camera during the calibration process making this technique infeasible when cameras are too far from each other. In our work we deal this issue through a robust multiview calibration system which lesser constraints in terms of maximum baseline allow us to extend the distance between cameras without loss of precision in the calibration process.

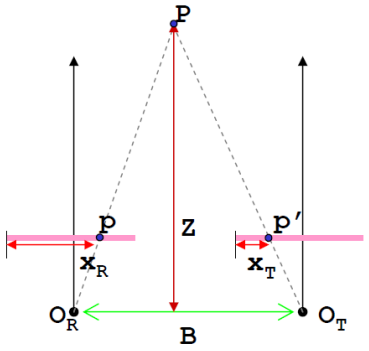


Fig. 1. The distance Z of a point P is retrieved from the disparity $d = X_R - X_T$. The distance the left and right optical centers (O_T and O_R) is called baseline B.

2 System Overview

The main issue connected to the reconstruction of wide areas is the sensitivity of the stereo matching with respect to the distance of the target area. In stereo vision, the error in the detected distance ϵ_z is related to the quantization error in digital images [3] and is computed by means of the following equation [4]:

$$\epsilon_z = \epsilon_d \frac{z^2}{bf}, \tag{1}$$

where b [m] is the baseline between left and right cameras, f [px] is the focal length, z [m] is the target distance and ϵ_d [px] is the quantization in the disparity map (usually one pixel).

Equation 1 shows that there is a quadratic dependency of the depth error to the distance between camera and observed region. Since farthest areas of the

landslide are located at more than 700 m from the cameras, a wide baseline is needed in order to keep low the depth error. In table 2 estimated depth errors with respect to target distance and baseline are reported for our camera installation (18M px and 30 mm focal length).

distance [m]	**baseline** [m] 10	12	14	16	18	20
200	0,59	0,49	0,42	0,37	0,33	0,30
400	2,37	1,97	1,69	1,48	1,31	1,18
600	**5,32**	**4,43**	3,80	3,33	2,96	2,66
800	**9,46**	**7,88**	**6,76**	**5,91**	**5,26**	**4,73**

Table 1. Estimated depth errors with respect to the target distance and cameras baseline.

2.1 Extrinsic Multiview Calibration

To correctly process a couple of stereo images and obtain observed area distances, the roto-translation between the two stereo cameras (extrinsic parameters) needs to be known. The calibration process is usually performed observing a pattern with a known geometry [2] but this simple method is not applicable in our case: the wide baseline and the terrain conformation do not allow the observation of the calibration pattern from both cameras.

We obtained a good estimation of the extrinsic parameters exploiting a multiview calibration [5]. This technique makes use of a large set of images of the same area taken from several viewpoints. Visual correspondences between all possible couples of images are matched in order to impose constraints on the roto-translation between each viewpoint couple; this way it is possible to perform calibration exploiting the features available in the framed scene without the need of dedicated patterns. In our work we collected a large set of images of the landslide, taken from a number of different viewpoints; we then added such images to those acquired by the stereo camera pair. Exploiting the multi-view calibration it is possible to obtain the mutual position between all couples of views, including the extrinsic calibration of the stereo couple.

2.2 Landslide 3D Reconstruction

Once obtained the extrinsic calibration of the stereo system, images taken from the two cameras are rectified (see fig. 2) and processed by a stereo matching algorithm called Semi Global Block Matching [6] in order to produce a disparity map. From the disparity map we retrieve the distance of each observed point and create a dense 3D point cloud representing the landslide reconstruction (see fig. 3).

c 

Fig. 2. Images processed by the stereo matching algorithm are first rectified. After rectification all correspondent points are located on the same row so the matching algorithm can search left-right correspondences in the same row.

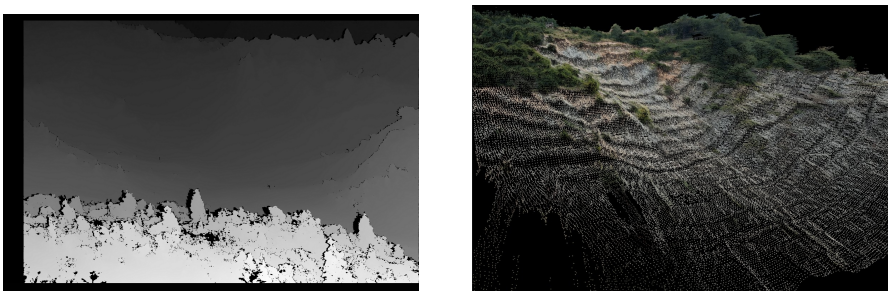


Fig. 3. Left-right disparity map and 3D point cloud reconstruction of the landslide. For image clarity the point cloud has been down-sampled so that only one point every 10cm is kept.

3 Results

The 3D reconstruction of the landslide allows us to precisely evaluate sliding of the ground. We detect particle flows in the image sequence using Normalized Cross-Correlation and then back-projecting the 2D flow onto the 3D landslide model, obtaining the motion flow of the rocks. The monitoring system proposed in this paper is completely autonomous and scalable and it is designed to issue an alert when the sliding effect exceeds a given threshold.

As a future work, we will employ a continuous camera calibration algorithm [7] to prevent the system to lose its calibration. This way it will be possible to obtain good performance over time, since the capability of the multiview stereo calibration of providing good estimation of extrinsic parameters is strongly dependent on the mutual position of the cameras.

References

1. Strecha, C., Von Hansen, W., Van Gool, L., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. (2008) 1–8
2. Zhang, Z.: A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on **22**(11) (2000) 1330–1334
3. Chang, C., Chatterjee, S.: Quantization error analysis in stereo vision. In: Signals, Systems and Computers, 1992. 1992 Conference Record of The Twenty-Sixth Asilomar Conference on. (1992) 1037–1041 vol.2
4. Gallup, D., Frahm, J.M., Mordohai, P., Pollefeys, M.: Variable baseline/resolution stereo. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. (2008) 1–8
5. Hiep, V.H., Keriven, R., Labatut, P., Pons, J.P.: Towards high-resolution large-scale multi-view stereo. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. (2009) 1430–1437
6. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. Pattern Analysis and Machine Intelligence, IEEE Transactions on **30**(2) (2008) 328–341
7. Dang, T., Hoffmann, C., Stiller, C.: Continuous stereo self-calibration by camera parameter tracking. Image Processing, IEEE Transactions on **18**(7) (2009) 1536–1550

"IAgo Vs Othello": An artificial intelligence agent playing Reversi

Jacopo Festa, Stanislao Davino

festajacopo@gmail.com, woods88@hotmail.it

Abstract. This paper presents the application of fundamental Artificial Intelligence algorithms for the realization of programs capable of effectively playing the game of Reversi. This application has been developed as a final project for the course of Artificial Intelligence, held in the academic year 2012/2013 at the Engineering Faculty of the University of Salerno, Italy. Several agents have been developed and tested, either in agent vs. agent and agent vs. human games. The application is freely available for download at the link specified in [8].

Keywords. Reversi, Minimax, Alpha-Beta Pruning, Heuristic Functions, Java, Game

1 Introduction

Our goal is to create an application that uses artificial players for the game of Reversi. This application is named "IAgo Vs Othello" (a wordplay that refers to the famous Shakespeare's tragedy of "Othello"), and is written in Java language.

It allows a human player to challenge any of the implemented artificial intelligences or to spectate a challenge between artificial players, or instead allows two humans to challenge each other. Our work focused on "classic" search algorithms like Minimax and Alpha-Beta Pruning and envisages different strategies in order to compute the next move in a reasonable time for a human opponent.

Different strategies have been implemented, which can be combined with any algorithm. Then we focused on testing and evaluating the performances of our agents, with different algorithms, on varying search depth, the used optimizations, and the strategies.

In Section 2 the theory on which the used algorithms are based upon is introduced, and the heuristic functions used are described too; Section 3 is about the application and the experiments done, and eventually, in Section 4 we will discuss the obtained results and draw some examples of future developments.

2 The Agent

An agent is an entity capable to evaluate the state of the game board and to select a move from the set of the available ones. An agent uses a specific algorithm to explore the available moves and their consequences, and a strategy to evaluate them.

2.1 The algorithm

The algorithms implemented in this work are variations around a basic algorithm, called Minimax. This algorithm comes from Von Neumann's Game Theory [1, chapter 5] [2]; it works mostly like a human would think: starting from his currently available moves, it selects one of them and evaluates its consequences, i.e. how the game state will evolve as a result to that particular move.

In this game state "projection" the algorithm knows that, as the players alternate between each other (unless a special situation occurs), it is the opponent player's turn, so it selects an opponent's move and evaluates its consequences; after this, it is the player's turn again, and a new move is selected.

The algorithm goes on until it reaches a game over state. It keeps track of the moves sequence that brought to that state and explores other possibilities, by going one step backward and selecting other moves. This is done systematically until all the game over states have been evaluated, and the best of these is elected as objective (i.e. a state that means victory for the agent, or if no victory is reachable, a state that means draw).

As the agent knows the sequence that leads to the objective state, it now simply selects the first move that brings the game towards it.

There are two main issues with this approach. Firstly, the chosen sequence is composed by both the player's and the opponent's moves; the agent can make a guess about which move the opponent will choose, but this is only a mere prediction. If the opponent chooses a different move, the game will evolve differently.

This is not a concern at all, because this agent is an optimum player (a player who always tries to maximize its final points) and works with the assumption that the opponent is an optimum player too, struggling to minimize the agent's score, so the agent will select as the opponent's moves those that lead towards a minimization of its final points. If the opponent doesn't choose optimum moves, then the player will get a higher score than expected because, if there was a different set of moves that could further minimize the agent's score, the opponent would have chosen it, so other non-optimum moves can only lead to a smaller score for the opponent, and a greater score for the agent.

The second issue is a more problematic one. Suppose, for the sake of simplicity, that at each step there are 10 available moves, that lead to 10 different game states. For each of these states, there are other 10 moves as well; considering all

the possible 2-moves sequences, we have a total of 100 different states. All the possible 3-moves sequences lead to a total of 1,000 different states, and so on. The number of states to be evaluated grows exponentially the deeper the algorithm goes. For Reversi, with a total of max 60 turns, the number of states to be evaluated has been estimated to be approximately 10^{58} [3].

Exploring so many states cannot be done at all with currently available computers: computing time required to evaluate a single move would be more than the age of the universe itself! Our objective is to have agents capable to compute a move in a reasonable time for a human opponent, i.e. a few seconds.

In order to do this, beside Minimax, we have implemented another well known algorithm, called Alpha-Beta Pruning [1, chapter 5]; it works like Minimax, except for the fact that it stops the evaluation of a possible move when it knows that this move is always worse than another one already evaluated. Alpha-Beta Pruning is a good optimization of Minimax because achieves the same results using less time and memory, as less moves and less states are evaluated.

Even using Alpha-Beta Pruning, though, computing time is still too much heavy. So, the search is cut when a certain maximum depth is reached. With this variation, *terminal states* (i.e. states evaluated at the max depth) generally aren't game over states, and the agent does not know if they are good or bad ones. To evaluate these states we must give the agents some knowledge that can be used to determine what is the right thing to do in a particular game situation. Different knowledge (called "*heuristics*") lead to different strategies and different playing styles.

2.2 Heuristics

2.2.1 Heuristic on Score (HS)

The simplest heuristic function for the game of Reversi calculates the score of the player (i.e. the number of disks of his color currently on the game's table). Intuitively, may seem like a good strategy, because in the end, victory goes to the player with the highest number of disks of his color. Actually, the score during the game may greatly vary from one turn to the other in such a way that having more disks in a certain game state does not necessarily represents a real advantage. This heuristic does not have good performance, but is useful to compare the behavior of other and more complex heuristic relative to a "beginner" player behavior.

2.2.2 Heuristic on Mobility (HM)

With this heuristic, we do not consider the actual score of the player, instead we calculate some parameters:

- *Mobility of the player:* the number of available moves;
- *Potential mobility of the player:* the number of empty slots next to at least one disk belonging to the opponent;
- *Potential mobility of the opponent:* the number of empty slots next to at least one player's disk;

It's important to note that if a player cannot play legal moves (in other words his mobility is equal to zero), he is obliged to pass the turn, that is a serious disadvantage for him and a big advantage for the opponent, who probably will win the game. This way to estimate the utility is much better than HS, but is anyway far from the optimum because, in late game, mobility tends to decrease for both the players.

2.2.3 Heuristic on Mobility and Corners (HMC)

One fundamental strategy in Reversi is to focus on capturing the corners. These represent very important positions. A disk is *stable* if it can't be turned to the opponent's color; any disk placed in a corner is, by definition, stable and makes stable every adjacent disk along the edges. For this reason, in addition to mobility, a good heuristic function has to consider the corners and the positions near them. Taking the corners is more important than keeping an high mobility, and so we appoint an heavy weight to the corners in the evaluation. We also assign a negative score to those positions that give to the opponent access to the corners as well, because those are undesirable positions.

2.2.4 Heuristic on Mobility, Corners and Edges (HMCE)

Corner positions are fundamental, but positions along the edges are important too; this heuristic function tries to take into account this fact by considering the number of disks the player has on each edge, and subtracting the number of opponent's disks to this value. With this heuristic, the agent tries to play more on the edges (in addition to the corners) and tries to stop the opponent from taking them.

2.2.5 Heuristic on Mobility, Corners, Edges and Stability (HMCES)

HMCE confers a positive score to a state of the game in which the player has more disks along the edges than the opponent (including corners), and a negative score otherwise. However, the strategic meaning of the disks along the edges is more important if those disks are also stable, i.e. they can't be captured by the opponent. An edge disk is stable when:

- It is placed in one of the four corners;
- It is placed in a row or in a column of disks of the same color that ends at least in one of the four corners.

We chose to evaluate only the stability at the edges positions, as these positions are easier to compute; central disks' stability, after all, depends from the stability at the edges (due to the recursive nature of the stability definition), and it is less important.

2.2.6 Heuristic on Mobility, Corners, Edges and Stability, Time-variant (HMCEST)

During a Reversi match, several strategies can be used depending on which phase the game is in. A Reversi match requires 60 turns (unless someone succeeds to win earlier). For this reason we decide to evaluate the scores on mobility and stability depending on the current game turn, assigning more or less importance to both strategies depending on the current phase the game is in. This heuristic works by computing separately the mobility-related score and the stability-related one; then the final score is given by multiplying these scores by some weights and sum them with the corner score (which is not weighted). Let w_m and w_s be the mobility and stability weights, respectively, and T the current turn number; weights are determined as follows:

- $w_m = 1.5$ and $w_s = 0.5$ if $T \leq 20$;
- $w_m = 1.5$ and $w_s = 0.5$ if $20 \leq T \leq 40$;
- $w_m = 1.5$ and $w_s = 0.5$ if $T \geq 40$.

3 Playing with "IAgo Vs Othello"

The application has been written in Java 1.6; we chose this programming language in order to have a cross-platform, self-contained application that can be instantly downloaded and launched, and because of the libraries that come with Java, which allow to make easily complex applications and good user interfaces.

"IAgo Vs Othello" is available for free download at [8], and it is released under a GPL3 license.

All the agents have been written from scratch, based on pseudocode available in [1, chapter 5]; there are, however, some variations with respect to the original version, because the algorithm has to manage the situation in which there isn't any available move; according to Reversi rules the player has to pass the turn, so the node currently evaluated must create a new child node without any variations on the game state, where now is the opponent's turn. If also the opponent has no available moves, then this child node represents a game over state.

The implemented algorithms are:

- Minimax;
- Alpha-Beta Pruning;
- H-Minimax (Minimax with support for heuristic functions);
- H-Alpha-Beta Pruning;
- Randomizer (simply selects the move randomly);
- Greedy (selects the move that flips the maximum number of opponent's disk).

Minimax and Alpha-Beta Pruning, without heuristics, cannot be used on a 8x8 table, so they haven't been tested in our experiments; they can choose a move in a reasonable time on a 4x4 table though. Minimax and Alpha-Beta-Pruning (and their heuristic versions too) have a time complexity that goes like $O(b^d)$ and $O(b^{d/2})$, respectively, according to the theoretical ones, where *b* is the *branching factor* (mean number of moves available in any game state), and *d* is the maximum depth. Computing times have been also experimentally verified.

The application allows a user to play against an artificial agent, or against another human; it also allows to spectate a match between two artificial agents and see the outcome of it.

There are two different interfaces available to start a new game; the complex one allows to generate one or two personalized agents by selecting an algorithm and some options, like the heuristic function and the max depth; the basic one presents the user with five different predefined agents, which correspond to different levels of difficulty.

To evaluate the agent performance, we let the agents we have developed play against each other and collect information from the matches, like the number of victories, defeats, draws, mean root branching factor for each turn, mean time required to compute the next move for each turn, etc., for each agent. We use also agents that use the same strategy but they evaluate the moves at different depths: these maximum depths are: 1 (only the currently available moves are evaluated), 3, 5 and 7. All the agents in the shown experiment use the same algorithm (H-Alpha-Beta Pruning) in order to keep computing time in a reasonable limit. The goal of this experiment was to measure how depth affects both computing time and game performance, and to evaluate the quality of the tested strategies. Results are shown in Table 3.1.

This experiment has shown that:

- As max depth increases, we can see a better playing ability, which is demonstrated by the raw victories/defeats ratio;
- As max depth increases, we can see mean computing time for each move strongly increases, in according to their exponential time complexity;
- At the maximum tested depth (= 7), we can see that HMC is the heuristic that behaves better. This can be explained by the fact that, as depth increase, a more complex heuristic may cause an agent to be distracted by more, simultaneous, and less important objectives, while a simpler one can focus towards more important ones; this may suggest that, as an objective, stability is somewhat less

important than mobility and the edges capturing; maybe an even better agent can be created by properly weighting these scores, like we tried to do with the HMCEST heuristic.

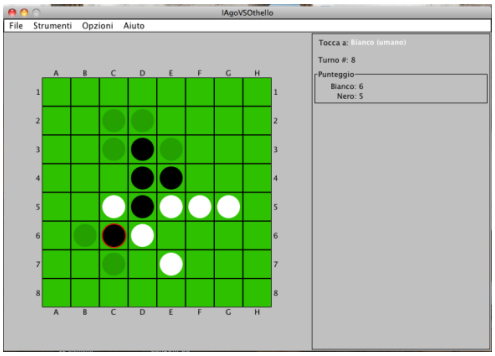


Fig. 3.1. Game situation in a Reversi game. Legal moves for the current player (the white one in this example) are highlighted with dark green disks.

Agent	Victories	Defeats	Mean computing time (microseconds)
H-ABPruning-D1-HMC	44	295	116
H-ABPruning-D3-HMC	115	223	3498
H-ABPruning-D5-HMC	169	171	61762
H-ABPruning-D7-HMC	329	11	1079018
H-ABPruning-D1-HMCE	99	240	124
H-ABPruning-D3-HMCE	176	163	3614
H-ABPruning-D5-HMCE	250	90	66650
H-ABPruning-D7-HMCE	300	40	1152387
H-ABPruning-D1-HMCES	99	241	134
H-ABPruning-D3-HMCES	190	150	3837
H-ABPruning-D5-HMCES	260	80	69279
H-ABPruning-D7-HMCES	300	40	1223536
H-ABPruning-D1-HMCEST	65	264	127
H-ABPruning-D3-HMCEST	139	191	3774
H-ABPruning-D5-HMCEST	210	130	72598
H-ABPruning-D7-HMCEST	260	80	1089245

Table 3.1. Victories, defeats, and mean computing time for each tested agent.

A series of games against human players have been done too, using agents corresponding to five different difficulty levels; so far, human players have performed well against *Beginner*, *Easy* and *Medium*. Some have been able to defeat *Hard*, but no one could defeat *Very Hard*. We plan to arrange a Reversi contest with more human players to play against the artificial agents, in a strictly con-

trolled environment, in which we can collect data such as victories/defeats ratios, computing time, etc. for both human players and the artificial agents. As there are many available implementations of agents playing Reversi, like Logistello [6] and Cassio [7], we also plan to try our agents against those programs.

4 Conclusions and Future Development

Despite the relative simplicity of the implemented algorithms, these reveal some kind of intelligent behavior; some behaviors we seen with HMC and more advanced heuristic include:

- The agent forces the opponent to select bad moves, from which the agent itself can benefit;
- The agent tries to drive the game flow towards situations in which it can take over a corner; but sometimes it just doesn't take control of the corner immediately, but, knowing that the agent can take it in a second time because the position isn't threated by the opponent, the agent prefers to focus on taking other strategic positions and then takes the corner later;
- Some moves seem to be stupid at first sight (like moving in the center of the table instead of the edge), but, a few turns later, they reveal to be part of a complex scheme that the agent choose for its strategy.

The algorithms implemented and presented in this paper are, however, "basic" ones; our application could be extended by implementing and testing new algorithms based on search trees exploration (*Beam Search*, for example), or to approach the problem differently by implementing an agent capable of learning to play through accumulated game experience.

5 References

[1] Stuart *Russell*, Peter *Norvig, Artificial Intelligence: A Modern Approach (3rd edition), Prentice Hall, ISBN-13: 978-0136042594;*
[2] John von Neumann, *On the Theory of Games of Strategy*, "Contributions to the Theory of Games", n. IV, 13-42, 1959;
[3] Victor Allis, *Searching for Solutions in Games and Artificial Intelligence*, Ph.D. Thesis, University of Limburg, Maastricht, 1994;
[4] Brian Rose, *Othello: A Minute to Learn... A Lifetime to Master*, 2005: http://othellogateway.com/rose/book.pdf;
[5] Reversi Wikipedia page: http://en.wikipedia.org/wiki/Reversi;
[6] Logistello Homepage: https://skatgame.net/mburo/log.html;
[7] Cassio Homepage: http://cassio.free.fr/;
[8] IAgo VS Othello download link: http://nclab.diiie.unisa.it/iago_vs_othello/IAGO_VS_OTHELLO.jar

CME: A Tool for Designing Business Models based on Commitment Patterns

Stefano Lanza, Simone Vallana, Cristina Baroglio

Università degli Studi di Torino, Dipartimento di Informatica
stefano.lanza@studenti.unito.it, simone.vallana@studenti.unito.it,
cristina.baroglio@unito.it

Abstract. Commitment-based patterns are an emerging technology, originally developed in the area of Multi-Agent Systems, for representing business models. In order to be used effectively, a similar approach calls for the development of graphical tools, supporting the designer in the task of knowledge representation. This paper presents CME, an Eclipse plugin developed at this very aim. CME supports the application of the Comma methodology for the design of business models, based on commitment-based patterns. It also allows the creation of new commitment-based patterns and pattern libraries.[1]

1 Introduction and Motivation

Patterns, as introduced by Telang in [6], are frequently used schemas of engagement which specify the roles that are involved in an interaction and the commitments in which they are involved. They can be used as building blocks in the construction of complex interactions models. By relying on commitments [4], patterns allow capturing, in a natural way, both the business intents, and the responsibilities each role has. At the same time, they support a declarative representation which does not prescribe specific courses of action but lets agents free to decide which actions to take as long as they respect their commitments. This approach, therefore, respects the autonomy of agents, advised in the Multi-Agent systems research area.

Commitments (and therefore patterns) reside at a *social level*: agents share a common interpretation of actions which are relevant to the interaction (e.g. message utterances). Each action "counts as" a social meaning. So, for instance, in a commercial transaction there is a mutual agreement that in case a client takes an object he or she has a commitment to pay for it; on the other hand, if the client gives money for an object the merchant has a commitment to give (or to send) the object at issue to the client. This is an example of a pattern. Notice that at this level there is no specification of who should act first, this depends on the circumstances and on the agents themselves. No procedure is encoded. This level of representation allows agents to adapt more easily to contingent conditions,

[1] This work is the third year degree thesis of Stefano Lanza and Simone Vallana, whose advisor is Cristina Baroglio.

to different environments, and to take advantage of occasions because it leaves them free to decide their actions.

Another advantage is that responsibilities are clearly stated inside commitments. Each commitment, in fact, has a "debtor" agent, which is the agent who is responsible for the achievement of the commitment consequent condition. This responsibility is accepted by the agents and, for this reason, in case a commitment is violated, it is possible to take action against the debtor agent. As such, this approach is very different than more traditional approaches for specifying interaction, e.g. UML interaction diagrams and UML collaboration diagrams, which basically rely on capturing the flow of messages.

Commitment-based patterns supply the building blocks by which it is possible to compose (complex) business models. To implement this vision, it is necessary to define guidelines that the designer can follow in order to build a business model. In [5], Telang and Singh propose *Comma*, a methodology for the design of business models, based on commitment-based patterns. The methodology supports the analysis of cross-organizational scenarios, the identification of the involved roles, of their tasks, and of their interaction, based on predefined libraries of patterns of business relationships. An example of cross-organizational scenario is the invitation to tender, where bidders are called to make offers that are then evaluated by the organization which issued the call. In this context, for instance, when issuing the call, the organization, which is acting as an initiator of the interaction, commits to evaluate the proposals by the bidders, who, in turn, commit to execute what requested in the call if their proposal is accepted. Another example is outsourcing, where an organization outsources part of its functions towards some clients to another organization.

The representation of patterns is inherently graphical but to the best of our knowledge there is a *lack of tools*, that are specifically thought for supporting the designer in the construction of patterns as well as in their use. This work tries to fill the gap by proposing the *Commitment Model Editor* (CME for short), an Eclipse plug-in, which allows trained designers to build commitment-based pattern libraries as well as to build pattern-based business models. Patterns and business models are represented internally in a homogeneous way, allowing business processes to be easily turned into new patterns if needed. The paper is organized as follows. Section 2 reports background knowledge on commitments and patterns. Section 3 explains the main characteristics of CME with the help of a running example. Final remarks end the paper.

2 Commitments and Commitment-based Patterns

Commitments [4] are represented with the notation $C(x, y, r, p)$, capturing that the agent x (*debtor*) commits to agent y (*creditor*) to bring about the consequent condition p when the antecedent condition r holds. When r equals *true*, the short notation $C(x, y, p)$ is used, and the commitment is said to be *active*. Commitments have a *regulative* nature: agents are expected to behave so as to achieve the consequent conditions of the active commitments of which they are

the debtors. In that case, commitments are *satisfied*. The business partners share a *social state* that contains commitments. These can be manipulated by means of the standard operations *create*, *cancel*, *release*, *assign*, *delegate*, *discharge*. Cancel and delegate are performed by the debtor, release and assign by the creditor. Discharge is performed when the consequent condition is achieved.

Commitment-based patterns are recipes for capturing recurrent business scenarios [6] – Telang and Singh in their works discuss various patterns extracted from RosettaNet [1]. A pattern consists of a set of *roles* and a set of *commitments*. Graphically, roles and commitments can be represented as the nodes of a directed graph. Arcs are either directed from a role to a commitment or from a commitment to a role. In the former case, the role is the debtor of the commitment, in the latter it is the creditor. So, for instance, in Figure 2, *Tender Issuer* is the creditor of commitment $C0$, while *Subcontractor* is the debtor. Patterns can be grouped into *pattern libraries*, which are then made available to the designers.

The *Comma methodology* encompasses the following steps:

1. Given a scenario description, extract subscenarios, each matching a pattern from the Comma pattern library;
2. Identify the roles for each subscenario, based on business functions (e.g., Shipper) that remove any ambiguity;
3. For each subscenario, identify business tasks (e.g., payment) that a role executes. Typically tasks are specified as actions executed by the participants;
4. Introduce into the business model a pattern corresponding to each subscenario. Rename the pattern characters with the roles from Step 2, and introduce the tasks from Step 3 as the antecedents and consequents of the appropriate commitments.
5. For each pattern, introduce its *message sequence chart* (MSC) into the operational model. Rename the roles and messages in the MSCs to align them with those determined in Steps 2 and 3.

CME was developed to support the first four steps of Comma. A default library of patterns was developed which includes part of the RosettaNet patterns by Telang and Singh.

3 CME Implementation and Example

CME is a graphical tool for the design of commitment-based business processes. Even tough it was developed as an Eclipse plugin, it is a stand-alone application system which does not require to have Eclipse installed in order to be used. The pivotal aspect, on which the tool is centered, is the possibility of *creating* and *using commitment-based patterns*. Indeed, the designer can both create pattern libraries, that can be used by third parties through the same tool, and create business models, by using available patterns. We describe the system and its features with a the help of examples.

Pattern Libraries. CME allows creating and using *pattern libraries*. Many libraries can be used at once. Users can exploit CME wizards to configure the paths at which the desired libraries can be found in the file system. A tab on the right of the interface allows exploring the patterns inside the selected libraries. Pattern libraries are written in XML. The format respects the structure given by the grammar:

LIBRARY $\longrightarrow$ *name author description* PATTERN_LIST
PATTERN_LIST $\longrightarrow$ PATTERN PATTERN_LIST | ε
PATTERN $\longrightarrow$ *name description* R_LIST C_LIST
R_LIST $\longrightarrow$ ROLE R_LIST |ε
C_LIST $\longrightarrow$ COMMITMENT C_LIST | ε
ROLE $\longrightarrow$ id *name description*
COMMITMENT $\longrightarrow$ id *name description state debtor creditor antecedent consequent*

As mentioned, the default pattern library used by CME contains patterns from RosettaNet and developed by Telang and Singh [1, 6]. In order to test the system, we developed a little pattern library containing patterns for the most common procedures for tenders which are applied by the Italian public administration.

Patterns. Patterns are given through the sets of roles and of commitments they include. Edges are implicitly defined by the debtor/creditor specifications. Commitments have two identifiers: one is numerical, used internally by the system, the other can be assigned by the designer and is human-oriented. The graphs are drawn by exploiting available libraries for graph representation.

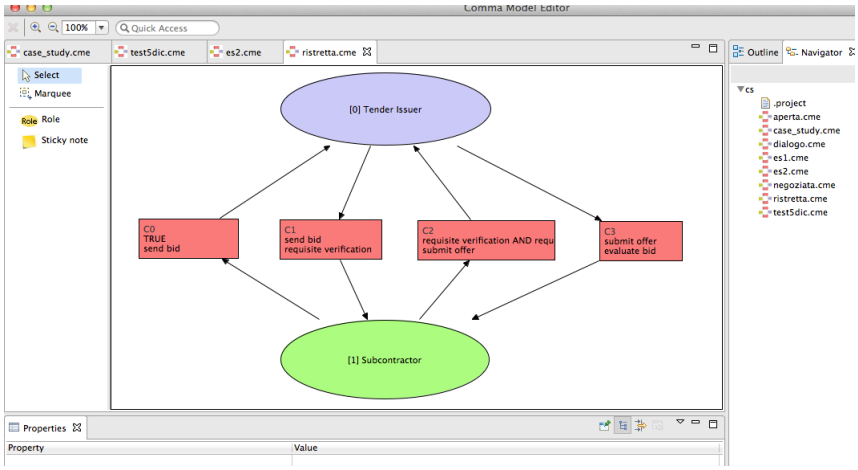


Fig. 1. Pattern for restricted procedure: ovals are roles, rectangles commitments.

As an example of pattern, Figure 1 reports the *restricted procedure* for tenders [2, 3] as it is represented inside CME. This standard procedure foresees

that a *Tender Issuer* (contracting agent) calls for bids from some *Subcontractors*. Each subcontractor who answers the call is evaluated and also its proposal is evaluated. The ovals in the picture are the roles of the pattern; rectangles represent commitments. Antecedent and consequent conditions are written on separate lines. CME does not require conditions to be written by following a specific grammar. In other terms, it is not commitment-language specific.

Interface. When CME is run, an interface containing three frames is shown: The central frame contains a GEF editor that allows the graphical specification of patterns (or business models). The user can directly insert roles and sticky notes, commitments are inserted with the help of a wizard. The lower frame lists and allows the modification of all the properties of the currently selected Node object. The right frame contains views: these allow the management of pattern libraries and supply both a navigator and an outline view of the edited model. Another characteristic of the system is that it is possible to work on many business models at the same time: each one will have an own tab within the interface.

When a business model consists of many pattern instances, it is not easy to grasp the structure of the plain graph because this may include a large number of roles, of commitments, and of edges. In order to help the designer, we developed also a "zoom-out" functionality which shows the business model as a graph containing two kinds of nodes: the defined roles and the instantiated patterns. If a role is used inside a pattern, then an edge connects the corresponding role node to the pattern instance node.

Business models vs. Patterns. In general, a business model can compose several patterns and the identity of such patterns is kept in the internal representation adopted by CME. On the contrary, a pattern is a plain graph of roles and of commitments. It is, therefore, easy to convert a business model into a pattern because it is sufficient to discard part of the information stored within the business model. CME supplies a proper *export* operation which allows to convert a business model into a pattern.

3.1 Example of application of Comma with CME

We now show a little example of application of Comma, underlining how the steps are helped by the CME tool. Let us suppose, that the Municipality of Torino needs to call for bids for the management of the cleansing services of its offices, and that it will assign the task of evaluating the requisites of the bidding companies to an external Audit Authority. Let us repeat the Comma steps and see what the designer does in this case. The result is reported in Figure 2.

1. *Identify subscenarios*: there are two subscenarios:
   (a) the interaction of the municipality with bidders,
   (b) the interaction of the municipality with the Audit Authority.
   The patterns to be applied are respectively:

   (a) the pattern for restricted procedures (from the pattern library that we developed), which foresees two roles: *Tender Issuer* and *Subcontractor*;
   (b) the pattern for commercial transaction (from the default pattern library, containing patterns from RosettaNet), which foresees the roles: *Partner1* and *Partner2*.
2. *Identify roles*: there are three roles (Public Administration, Audit Authority and Cleaning Service), one of them (Public Administration) glues the two applied patterns and matches two different pattern roles (Tender Issuer and Partner1). Therefore, we rename the roles *Tender Issuer* and *Partner1* by *Public Administration*, the role *Subcontractor* by *Cleaning Service*, and the role *Partner2* by *Audit Authority*. Such roles can be created in CME directly by the user without the need of relying on specific wizards.
3. *Identify actions per role*: here the designer needs to get into the depths of how evaluation and selection are made. We will not list all actions exhaustively but, as examples, in Figure 2, the Cleaning Service should "send bids" and "submit orders"; instead, "requisite verification" is one of the Audit Authority's actions;
4. The business model will contain a single instance of each of the identified patterns, to which a proper renaming was applied. To this aim, it is possible to use the uploaded libraries and select the desired patterns. A guided procedure allows the user to perform the necessary renaming, thus connecting patterns to roles.

The resulting business model binds the agents which will play it, e.g., the Municipality of Torino as a Public Administration. It is worthwhile to comment the commercial transaction pattern: here, there seems to be a loop (if you provide the service I pay, if you pay I provide the service). However, recall that this kind of specification is not a procedure. The two peers can autonomously decide when to act. If the Audit Authority acts first, $C4$ is satisfied, while $C3$ becomes detached, and the Municipality is bound to pay. Similarly, if the Municipality pays first.

4 Possible usage and final remarks

In this paper, we introduced CME, a graphical editor for commitment-based patterns and pattern libraries. Commitment-based patterns, and therefore also CME, are particularly useful in the design of cross-organizational business models, where the interaction of the parties cannot rely on allowing direct, mutual access to their information systems, but, rather, calls for the specification of the boundaries and the expectations of the interaction at a higher and normative level.

It is, indeed, a current trend for companies to focus their internal resources on the core business delegating to third parties other functions. Similarly, Public Administrations more and more often need to interact with other authorities and organizations. When such organizations maintain relations with plenty of other actors, it is easy that some of the commitments foreseen by the contracts are

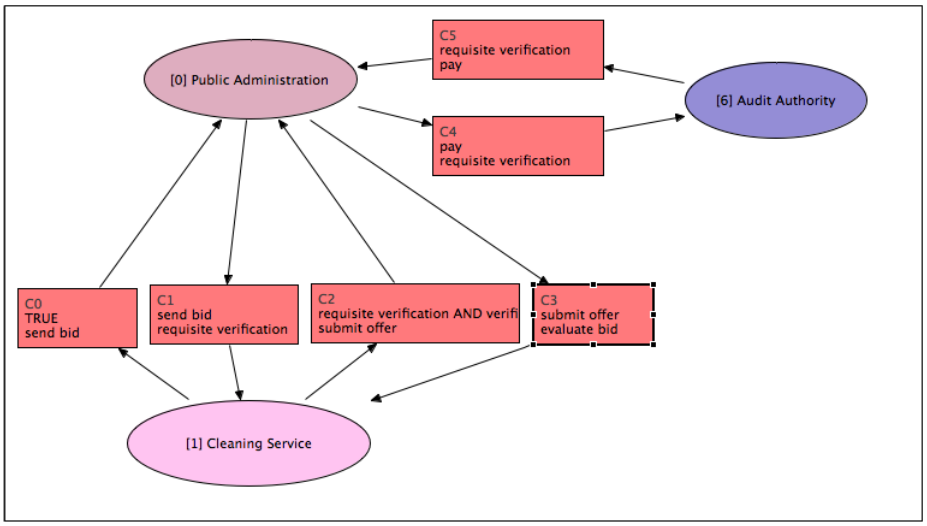


Fig. 2. Business model combining two patterns.

violated. In such cases, a software like CME can be very useful. The tool, in fact, allows specifying in a clear way the many business relationships that are built and that, if integrated with the information systems of the involved realities, would also allow the monitoring of responsibilities and of the satisfaction (or violation) of commitments. There would, thus, be a significant impact in terms of bureaucratic simplification and reduction of costs.

References

1. RosettaNet. http://www.rosettanet.org/.
2. Gazzetta ufficiale n. L 313, pp. 0003–0035, November 2009.
3. ITACA. Guida operativa per l'utilizzo del criterio di aggiudicazione dell'offerta economicamente più vantaggiosa negli appalti di lavori pubblici di sola esecuzione, 2013.
4. Munindar P. Singh. An Ontology for Commitments in Multiagent Systems. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
5. Pankaj R. Telang and Munindar P. Singh. Comma: a commitment-based business modeling methodology and its empirical evaluation. In *AAMAS*, pages 1073–1080. IFAAMAS, 2012.
6. Pankaj R. Telang and Munindar P. Singh. Specifying and Verifying Cross-Organizational Business Models: An Agent-Oriented Approach. *IEEE T. Services Computing*, 5(3):305–318, 2012.

Smart usage of Mobile Phones Sensors within an Event Calculus Engine

Student experiences inside AI courses

Valerio Mazza, Michele Solimando

Dipartimento di Informatica – Scienza e Ingegneria
Università di Bologna
Viale Risorgimento, 2, 40136 Bologna, Italy

valerio.mazza@studio.unibo.it

michele.solimando@studio.unibo.it

Abstract. In the following paper we show how we define and integrate an Event Calculus rule system, written by us with xtext Dsl on an android application to work with the mobile phone sensors, in order to smartly use them.

Keywords: Event Calculus, Xtext, DSL Language, Drools, Android.

1 Introduction and Motivation

Nowadays computer systems are so complex, in a way that even expert programmers, have difficulties to verify whether they operate correctly. Moreover it is also very difficult to introduce new features without weaken its internal logic.

In literature there are a few proposal of logical formalisms which can be used to mitigate these drawbacks. One of them, for instance, is the Event Calculus (EC). A possible approach, in fact, consists in using a EC machinery that operates in the deductive mode to monitor the application and verify that it is compliant with what it is expected. Unfortunately, these frameworks are difficult to master either to build advanced applications.

The goal of this Paper, is to present a framework based on EC with a twofold purpose. On one side we have developed in Xtext[1] an Integrated Development Environment (IDE) to allow a domain expert to express a problem in terms of EC concepts. We chose Xtext/Xtend because is a statically typed functional and object-orient language. It automatically compiles to readable Java code. We will describe this process properly in the next chapters. On the other side we have defined a procedure that automatically converts the given problem into an Android application[2] which uses a Drools[3] implementation of Event Calculus.

Consequently thanks to such a integrated system, a user with any level of understanding, can leverage the complexity of defining an EC problem and automatically build an Android application around it. In particular we show a case study in medical rehabilitation that demonstrates the potential of this approach.

This paper is organized as follows: in the follow section we present the framework with the tools and the components of interest that we used. The third section shows the potential of the system developed, taking as example a medical study case. The last section illustrates the objective we have reached, our system limits and future developments.

2 Framework

We briefly describe the tools and components of our framework in order to understand them.

2.1 Event Calculus

It is a logical formalism introduced by Kowalski and Sergot in 1986. There are many different version with varying degrees of expressiveness and efficiency. In this work we use the multivalued reactive variant (most efficient), Drools based, already presented in [1]. This particular version allows you to use any real value, instead of only Boolean value, to represent the magnitudes of the domain.

This formalism owes his fortune to the logical correctness and simplicity. In fact, it is based on two concepts only: the Event that is the notification of an action occurred in the considered domain in a specific time instant; and the Fluent that is single measurable aspects of the domain that can change over time. The EC formalism is completed by other axioms to report the relation between Event and Fluent. For more details see [1].

[1] `http://www.eclipse.org/Xtext`
[2] `http://developer.android.com/index.html`
[3] `http://www.jboss.org/drools`

2.2 Drools and ECA-rules

Production rules are one the most common way to represent knowledge in many areas of Artificial Intelligence (AI). Drool is a Production Rule System that use the Modus Ponens (MP) as rule of inference. In this type of reasoning formally if one proposition P implies a second proposition Q and P is true, it concludes that Q is also true. Rules in PRSs are called instead productions and they primarily express some behaviour that transforms the available information about the domain in new information.

Drools consists of many components: a Production Memory that contains all the productions; a Working Memory that contains information about the current state of the system, the knowledge and the assertions; an Agenda that contains all the activations of all productions. The first action taken by PRS is the pattern matching, that is to build a network of constraints. The algorithm used is RETE algorithm. This algorithm filters the domain knowledge to identify the set of data that satisfies the preconditions of some productions. The active productions are passed to the Agenda. Last task of the PRS is the execution of the actions.

The RETE algorithm can handle the ECA-rules (Event Condition Action) [2]. The ECA-rules consists of three parts: an Event that triggers the invocation of the rule and, if the Condition is satisfied, provoke the Action, a set of updates to the system.

2.3 Domain Specific Language and Xtext

A General Purpose Languages (GPLs) are designed to address a wide variety of application domains, while Domain Specific Languages (DSLs) do not aim to solve every problem in every domain, but focus on a specific restricted one in order to solve problems inside it in a better way than a GPL language would do.

Our approach exploits Xtext capabilities to define a DSL language for EC which can be used by Xtext itself to guide the automatic conversion of any problem into a working application built with a GPL, namely Drools. In addition, the Xtext framework conveniently provides a mechanism to build an IDE for the given DSL language with no additional efforts. Therefore the resulting tool allows to simplify the expression of EC concepts while taking care of the translation of them toward the chosen DSL.

Our language is essentially intuitive because it is similar to the way in which the humans represent knowledge and is readable because the syntax of the rules appears like the spoken language understandable by all.

2.4 Android

Android is a Linux-based operating system designed for mobile devices such as smartphones and tablet computers.

While creating a standard android application, building up a logic to use the phone sensors (and not just them) can be tricky, and not easily maintainable. When you start having several different patterns of behavior to observe, the complexity of the soft-

ware increase sensibly. With our Event based language it takes less to create a fully working logic, and it separates the two part of the application, making it cleaner and less complex.

3 Case Study

The designed system shows its potential in our case study. We used our DSL to assist doctors in tasks that involved physical effort, such as rehabilitation in physiotherapy. Our goal is to present how different type of users can take advantage of our development environment to simplify their everyday jobs, both doctors and patients.

As we all know, in the last decades, life expectation is getting longer, causing bad outcomes such as medical complications in the older age segment. Specifically, the problems due to reduced joint mobility are particularly insidious because they seriously affect the quality of elderly life style.

We have formulated an EC problem to provide suggestions and incitements to the patient, that are based on the performance during the exercises assigned from a physiotherapist. The suggestions are notified through a simple Android application. The innovation is, thanks to the powerful DSL implemented and the intrinsic logic of the EC, that the physiotherapist will be able to sketch the important characteristic of the particular exercise. He will also have the possibility to create a new one, or modify a given one, without the action of an IT engineer.

Let us see how the tools we presented before can help us to reach our goal.

3.1 Workflow

In the figure 1 we show the entire workflow of the system. In the figure there are two types of users: the physical therapist, expert user of the system, and the patient, passive user.

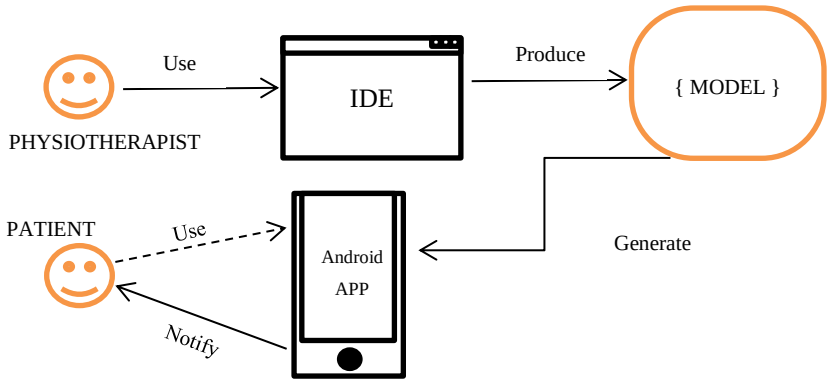


Fig. 1. Workflow

In the first place the Physiotherapist has to express an EC problem throughout the editor built in the Xtext generated IDE as shown in figure 2. In this case the user of

our system is the physiotherapist, who will have in hand a series of Events which map the smartphone sensor to the EC world.

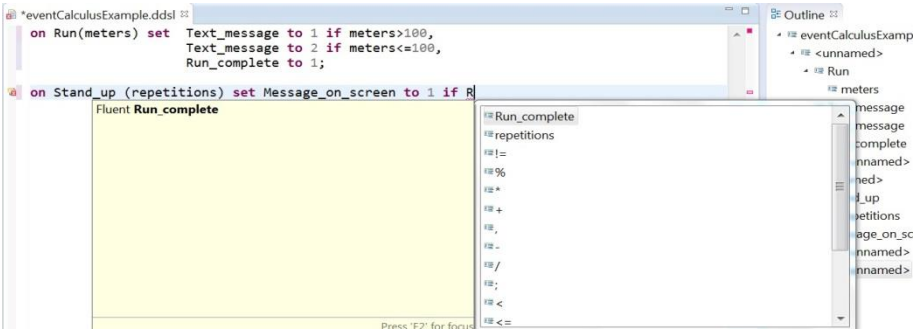


Fig. 2. IDE Editor where the Physiotherapist inserts his Event Calculus rules.

Consequently the physiotherapist will know exactly when a given Event that represents, for istance the accelerometer behavior, is thrown, the Event will map a specific pattern of movement, rather than standing or sitting. The system establishes a correspondence between fluents and interface elements, giving the possibility to the physiotherapist to decide how the application shows its notification of the thrown events.

When the Physiotherapist is satisfied with the model (an example is presented in next paragraph), he may request its conversion into an Android application. The conversion is performed through additional buttons introduced in the IDE, as shown in figure 3.

The application will monitor the behavior of the patient acting accordingly as indicated by the logic defined by the physiotherapist.

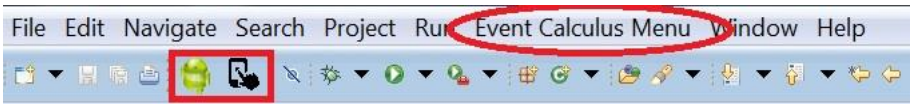


Fig. 3. Custom IDE menus and buttons

In the end, the Physiotherapist, by connecting the patient smartphone to his PC, will install the App directly into it.

3.2 Example

In the following listing we present a basic EC problem consisting of simply movement exercises.

```
on Run(meters) set
      Text_message to “SUPER” if meters>100,
      Text_message to “GO-GO” if meters<=100,
      Run_complete to 1;

on Stand_up(repetitions) set
      Message_on_screen to “Exercise Completed” if
            ((Run_complete == 1) AND (repetitions>10)),
      Message_on_screen to “Restart Exercise” if
            Run_complete == 0;

on SitDown(time) set
      Message_on_screen in (now+1000) to “Are you OK?” if
            time > 10000,
      Button_on_screen1 to “YES, I’m OK” if
            Message_on_screen == “Are you OK?”,
      Button_on_screen2 to “NO, I'm sick” if
            Message_on_screen == “Are you OK?”;

on Press_button_on_screen2 set Call_Doctor to “URGENT”;
```

Listing 1. Rehabilitation exercises synthesized into a EC problem.

With these few rules, we want to show that it is possible to provide information to the EC mechanism by passing parameters along with the event notification. Also note that, for each fluent, is possible to explicit an expression that, when evaluated, provides the value to be attributed to the fluent.

The notifications are represented by strings that may have a numeric identifier. For ease of exposition in this example we used the strings instead of numeric value.

4 Conclusion and future development

The goal of realize an easy to approach language was met. The Editor as well is simple but efficient in the implementation of the rules created.

Our intention is to make something useful for the case of study we have. A limitation of the application is surely the narrowed freedom to use the smartphone features.

In this regard, it could extend the mapping between events and sensors to make the monitoring of patient activity more complete.

Furthermore after careful field testing, followed closely by medical professionals, we could build databases that inform us on unwanted behaviors that lead to disagreeable consequences, for a timely prevention.

From the theoretical point of view, the powerful logic EC engine can be expanded with the use of the ECE-rules (Event Condition Expectation). The ECE-rules are evolution of the well-known ECA-rules and they can express exceptions and anomalies of the system.

Another aspect that would increase the potential of our application, is the possibility of having a fuzzy logic rather than exact, in order to express also approximate reasoning (rather than exact), also very useful in our case study.

References

[1] S. Bragaglia, Monitoring Complex Processes to verify System Conformance, 2013.

[2] Schmidt, K.-U. a. Stuhmer, R. a. Stojanovic and Ljil-jana, Blending complex event processing with the RETE algorithm, 2008.

[3] L. Bettini, Implementing Domain Specific Languages with Xtext and Xtend, Packt Publishing, 2013.

[4] S. Russel and N. P., Intelligenza Artificiale: un approccio moderno, Prentice Hall, 2010.

Emerging Stable Configurations in Cellular Automata

Mattia Vinci and Roberto Micalizio

Dipartimento di Informatica
Università di Torino
corso Svizzera 185, 10149 Torino
mattia.vinci.1@gmail.com,micalizio@di.unito.it

Abstract. This paper considers how simple and limited entities, such as Cellular Automata (CA), can organize themselves giving rise to stable and complex behaviors. We begin by introducing Cellular Automata, simple discrete computational models useful to describe the evolution of a system following simple rules. Then we propose a possible architecture to implement a class of CA onto a Unix system. In particular, we propose a distributed solution where a number of cooperating processes cooperate in order to evolve the current configuration of the Universe.

Keywords: Cellular Automation, Self-stabilization, Conway's Game of Life

1 Introduction

Distributed and multiagent systems are nowadays common in many aspects of our daily life. From financial agents selling and buying stocks, to swarms of UAVs flying in formation, there are many scenarios in which heterogeneous agents interact with one another in order to accomplish their own goals. In principle, one can think of these systems, or societies, as composed by a number of single entities, so that the behavior of the society as a whole can be represented just in terms of the behaviors of its entities. Unfortunately, such a simplistic view does not take into account that complex behaviors can emerge within an agent society even though these behaviors have not been designed. In other words, an agent society can exhibit an organized behavior even though no (external) agent organizes it. A simple, yet effective, way to study emerging behaviors consists in the adoption of Cellular Automata (CA). CA are simple computational models represented as matrices, in which each cell is a simple Finite-State Machine, often with just two possible states: *alive* and *dead*. It is worth noting that the initial conditions of a CA system satisfy the ones defining a *chaotic system*:

a. sensitive to initial conditions
b. topologically mixing
c. having dense periodic orbits

where point *a.* means that an arbitrarily small perturbation of the current trajectory may lead to significantly different future behaviour - what is usually known as *the Butterfly Effect*; being *topologically mixing* means evolving over time so that any given region or open set of its phase space will eventually overlap with any other given region. Point *c.* states that every point in the space is approached arbitrarily closely by periodic orbits.
Although these premises would suggest that a CA could evolve in a chaotic way, it has been demonstrated in many practical scenarios that the simple rules governing the states of the cells do have an impact on the system as a whole, and in particular, can give rise to a simple form of self-organization. In this paper, we start by giving a description of the functioning of a Cellular Automata and in particular of the "Game of Life", and then present a possible architecture for implementing a CA system. Ideally, since each cell in the system is autonomous (i.e., it is a simple agent), it should be implemented as an independent process, or thread. For scalability reasons, however, we propose an architecture where a number of cooperating processes, named *Evolving Processes*, coordinate with one another in order to evolve consistently the entire system.

2 Background

2.1 Cellular Automata

A *Cellular Automata (CA)* system is a computing model in which the environment - usually referred to as *Universe* - is a grid of cells extending in a finite number of dimensions. Each cell can be in one of a finite number of states.
At each step of the universe evolution, generally called a *generation*, each cell evolves into a state that is a function of the previous cell's neighborhood states. The set of neighbouring cells can be defined in several ways, the most common being the *Von Neumann set* (the set of perpendicularly adjacent cells) and the *Moore set* (all adjacent cells).
The evolution rules can be expressed in several ways, the most common being the *Wolfram* code (by Wolfram & Packard, 1985) and *Mcell* (Mirek Wjtowicz, 1999). The Wolfram code is a decimal number whose binary represantation is a byte that describes, for each possible alive neighbouring set, the state the cell should be in.
The MCell representation encodes the same information in a string of the type Sx/By, in which x is a list of numbers of alive neighbours required for the Cell to survive, and y is the list of numbers of alive neighbours required for a dead cell to become alive.
The evolution of a CA can follow different path; based on their behaviours in time, CA are usually classified, following Stephen Wolfram [1], in four main categories:

- *Uniformity* automata in which patterns generally stabilize into homogenity
- *Repetition* automata in which patterns evolve into mostly stable or oscillating structures

- *Random* automata in which patterns evolve in a seemingly chaotic fashion
- *Complexity* automata in which patterns become extremely complex and may last for a long time, with stable local structures

The interests on CA is due to the fact that they can be set to model real-life situations. For instance, the emerging of fractal structures in many real-world shapes (e.g., trees, shells, and crystals) is easily emulated by many simple CA. These models have also been used to simulate more complex situations, such as social and economic processes [2], the evolution of bacteria colonies [3], urban processes [4] or to provide simpler models of common differential equations of physics, such as the heat and wave equations and the fluid equations. Moreover, using cells of different shape allows to construct even more structures; using hexagonal CA with the rule "alive cells die if only one cell in its neighborhood is dead" is possible to obtain the *von Koch snowflake* [5] and other crystalline structures.

2.2 Conway's Game of Life

In the 1970s the British mathematician Conway, looking for rules that lead to interesting behaviors, introduced the rule *S23/B3*.
This rule states that any cell, if alive, can survive only if it has exactly 2 or 3 alive neighbors; otherwise it dies of "loneliness" or "overcrowding". A dead cell can come alive if it has exactly 3 alive neighbors.
This rule presents a behavior that at first sight can look similar to evolving patterns found in real biological world, and because of this is widely known as *the Game of Life*.
The most interesting feature of this rule is its natural tendency to self-organization: given a random initial condition, the Game may evolve forming stable small structures that can be:

- *still life* if they reach a stable still state
- *oscillators* if they reach a set of repeating states
- *spaceships*, structures that move themselves around the grid

It has also been demonstrated that *Life*-like CA can implement logic gates, realize arithmetic tasks, and simulate Finite State Machines. Indeed, Wolfram [1] has demonstrated that a cellular automata system, when behaving under a specific rule, is Turing-complete.

3 Architecture and Implementation

We realized a Cellular Automata system as a part of a laboratory project on Unix operating system. It must therefore be noticed that the main purpose of the project was not AI, but how to use the Inter-Process Communication facilities offered by Unix.

Of course, it is not hard to see that concurrent processes can be seen as agents. In addition, processes using semaphores and shared memories, can be seen as

agents competing for accessing the resources, but at the same time, these agents could cooperate in order to reach a common goal. After these simple premises in mind, we implemented the CA system by means of four types of processes (i.e., agents):

- Memory Manager
- Evolving Agents
- Universe Displayer
- User Menu

The Memory Manager is the main agent of our implementation, it allocates in a segment of shared memory a matrix of `short` representing the Universe: an alive cell has value 1, a dead one has value 0; the default size of the universe is 130x43 cells.
The synchronization of all the agents, as well as the consistency of the displayed configuration, are assured by a pool of semaphores, allocated by the Manager, that also guarantees mutually exclusive access to the shared memory. The Manager also allocates three queues to allow the communication among the agents.
The default number of *Evolving Agents* is 5. Each of these agents asks the Memory Manager the permission to access the shared memory; if granted, the Manager inserts the process information into a list of currently active agents, and sends back a message containing the identifiers of the shared memory and of the semaphores together with the portion of the area assigned to that agent.
The evolution step requires that each Evolving Process copies its portion of shared memory into a local array. The local array is used to represent the current state of universe, in a way that the Evolving Agent can apply the evolution rules reading from its local array and modifying accordingly its segment of the shared matrix. When all the Evolving Agents have terminated an evolution step, the shared matrix will contain the new configuration. Counter Semaphores are used in order to synchronize the processes so that no one starts a new evolution step if some other process has not completed its previous one.
The *Displayer*, is now activated to show the current state of the universe in a terminal window using the *ncurses* library. To simplify the interface, we just use a matrix of characters putting a character '$\diamond$' where the corresponding cell is alive, and leaving the spot blank otherwise. In this case too, a semaphore is used to guarantee that a new evolution step is started only after the display of the current state has been completed.
The interaction with the user is allowed by means of an interactive menu (*User Menu* process), from which the user can select the type of automata and the rules to simulate.
The user can choose to randomly initialize the universe and evolving it following one of the hard-wired rules:

- Game of Life (S23 / B3)
- 34 Life (S34 / B34)
- High Life (S23 / B36)
- Day and Night (S34678 / B3678)

– Sierpinski Triangle (1D Xor based rule)

or she can select to load both initial pattern and evolution rule from a file.

The user can also interact with the *Universe* using the mouse to turn on and off single cells in order to see how slightly different initial conditions can give rise to very different evolutions, and compare them to see if and how they stabilize, and how long (how many evolution steps) it took.
At any time the user can interrupt the evolution loop using system signals. In addition, if activated, a mechanism to detect short periodic loops re-initializes the universe when overall patterns are closely repeating.

The entire code of this project is shared through GitHub. [10]

4 Characterizing Stable Configurations

Although the developed system is still in its early stage, it is important to note that it could be a useful tools to spread some basic ideas of Artificial Intelligence. As we have already noticed, many real-world systems, especially biological ecosystems, present stable or ordered configurations that emerge autonomously without the presence of a coordinator.

Since stable configurations are recurrent in CA systems too, our purpose is to characterize initial configurations and evolution rules. More precisely, our idea is to start from a stable configuration, and allow the user to perturb this initial configuration by adding or deleting alive cells. This change obviously creates an instable situation, that will eventually evolve in a new stable configuration. Such a kind of interaction can be useful for a user to understand which initial configurations evolve more rapidly into new stable configurations and with which rules. Moreover, even a non-expert user can easily understand how her/his changes impact on the whole Universe. The prototype could therefore be used as an educational tool for teaching young students the well-known "butterfly effect". In fact, it could be used as a game where the user has to change the world by adding/removing a specific number of alive cells, but the impacts of her/his changes should be as limited as possible. In other words, the new stable configuration obtained after the changes should be as similar as possible to the original one. To assess the similarity between two configurations different measures could be devised. For instance, the number of alive cells of the new stable configuration should be the same, or close, to that of the original one. Also the number of iterations required for evolving to the new stable configuration could be an interesting parameter to be studied. For example, a stable configuration C_1 could be preferred to another configuration C_2, if C_1 has been reached with a minor number of evolutions than C_2.

5 Conclusions

We have used Cellular Automation as a mean to observe evolution of seemingly chaotic systems into organized complex structures.

We presented a possible architecture to implement such automata onto a Unix system, using the operating system IPC structures (message queues, semaphores, signals, shared memory), various cooperating synchronized processes, and continuously displaying the evolution state into a terminal window.
Repeatedly observing the state of the system after a certain amount of time we noticed how the initial chaos evolved itself into a complex stable structure, as if the global equilibrium were achieved through the application of strictly local rules. This makes us conclude that it is possible for initially chaotic configurations to self stabilize and give rise to more complex entities by following local simple evolution rules.

This work represents just a first step in the study of CA. As a future work, we are interested to study how CA combined with Genetic Algorithms can simulate sophisticated real-world domains.

References

1. Wolfram, Stephen. A new kind of science. Vol. 5. Champaign: Wolfram media, 2002.
2. Bagnoli, Franci, Rechtman *Chaos in a simple cellular automaton model of a uniform society*, LNCS 3305, pp. 513-522, 2004.
3. Krawczyk1, Dzwinel, David A.Yuen, *Nonlinear Development of Bacterial Colony Modeled with Cellular Automata and Agent Objects*, International Journal of Modern Physics C 14.10 (2003): 1385-1404.
4. Santé, Inés and García, Andrés M and Miranda, David and Crecente, Rafael, "Cellular automata models for the simulation of real-world urban processes: A review and analysis." Landscape and Urban Planning 96.2 (2010): 108-122.
5. Gravner, Janko, and David Griffeath. "Modeling snow crystal growth I: Rigorous results for Packard's digital snowflakes." Experimental Mathematics 15.4 (2006): 421-444.
6. Rendell Attic *http://rendell-attic.org*
7. JSLife patterns *http://entropymine.com/jason/life*
8. Mirek Cellebration *http://www.mirekw.com/*
9. SUCS http://sucs.org/ pwb/report/
10. Code: https://github.com/sowdust/Cell_Auto.git

di4g: Uno strumento di *clustering* per l'analisi integrata di dati geologici

Alice Piva[1], Giacomo Gamberoni[1], Denis Ferraretti[1], Evelina Lamma[2]

[1] intelliWARE snc, via J.F.Kennedy 15, 44122 Ferrara, Italia
{denis, giacomo}@i-ware.it, alice88.piva@gmail.com
[2] Dipartimento di Ingegneria, Università di Ferrara, via Saragat 1, 44122 Ferrara, Italia
{evelina.lamma}@unife.it

1 Introduzione

di4g (data integrator for geology) è uno strumento sviluppato per l'analisi di dati geologici, in particolare per la geologia degli idrocarburi. Questa disciplina si occupa di cercare e valutare gli elementi fondamentali nella formazione di un giacimento di idrocarburi in un bacino sedimentario. Tipicamente, attraverso sonde calate nei pozzi esplorativi, si ottengono numerosi dati eterogenei, costituiti da *log* elettrici e rappresentati sotto forma di curve e *log* immagini (detti FMI) significative della conformazione delle pareti dei pozzi. Da queste immagini si possono ricavare informazioni riguardanti la tessitura delle rocce, il tipo di porosità, la presenza di fratture (rappresentate da sinusoidi). L'esperto geologo analizza questi *log* visivamente, per identificare le varie caratteristiche presenti all'interno delle immagini. Questa è però un'analisi complessa e soggettiva nell'interpretazione che richiede, inoltre, un elevato tempo di esecuzione. Per questo, è stato sviluppato I^2AM (Intelligent Image Analysis and Mapping), un software per l'interpretazione semiautomatica delle immagini provenienti dai pozzi petroliferi (1) e per individuare le caratteristiche visuali presenti (6, 8). Poiché tutti i dati disponibili sono eterogenei e non tutti considerati da I^2AM, per consentire l'allineamento e la fusione di diversi *dataset* è stato sviluppato di4g. di4g fonde la tabella delle caratteristiche prodotta da I^2AM con i *log* elettrici disponibili e consente anche di eseguire l'analisi integrata di dati provenienti da pozzi diversi.
Per fornire una prima classificazione sui dati in ingresso, di4g applica una tecnica di *clustering* individuando zone del pozzo simili e raggruppandole in *cluster* con un algoritmo di *clustering*. Ciò costituisce una prima ipotesi di divisione delle parti dei pozzi in zone simili, in ciascuna delle quali sono presenti diverse tipologie di roccia. Tramite l'utilizzo del software di4g il geologo può quindi eseguire un'interpretazione molto più veloce e interattiva dei dati ricavati dalle esplorazioni, grazie anche alle rappresentazioni grafiche di uscita dello strumento aggiornate in tempo reale. Sono stati sperimentati diversi approcci al fine di individuare -e in alcune situazioni prevedere- le classificazioni per determinati campi di pozzi (4, 5, 7, 9, 10), ma l'approccio non supervisionato con il coinvolgimento diretto del geologo si è dimostrato il più efficace (2). di4g è stato sviluppato in *partnership* tra due aziende, una software e una di consulenza geologica, e un Dipartimento Universitario di Ingegneria. La versione

di di4g descritta è stata sviluppata nel corso di un tirocinio e tesi di laurea magistrale svolti da parte del primo autore.

2 L'approccio di di4g

L'obiettivo del progetto è stato quello di sviluppare uno strumento per l'analisi integrata di dati da esplorazioni petrolifere, orientato allo studio di diversi pozzi e di tipi di dati eterogenei (3). In genere, in fase di analisi di un terreno, in un campo esplorativo si eseguono più trivellazioni di pozzi. I dati devono essere analizzati e confrontati simultaneamente per identificare quali tra questi pozzi hanno le caratteristiche migliori per essere un buon giacimento.
Il flusso di lavoro di di4g è suddiviso in tre fasi principali (si veda Figura 1):

- nella prima fase avviene l'**importazione** dei file. L'applicazione consente di importare più file contenenti dati riferiti ad ogni pozzo e tramite un algoritmo di allineamento costruisce il *dataset* utilizzato nel successivo processo di *clustering*. Altra funzionalità fondamentale in questa fase è la possibilità di importare file contenenti dati riferiti a più pozzi; in questo modo è possibile analizzare contemporaneamente le caratteristiche di un intero campo esplorativo;
- nella seconda fase avviene il processo di **clustering**. In questa fase si normalizzano i dati, e la normalizzazione è solitamente applicata sull'intero *dataset*. Avendo inoltre la possibilità di importare dati riferiti a più pozzi, il sistema è stato reso in grado di calcolare anche una normalizzazione riferita a ogni singolo pozzo. In questa fase è stato anche ottimizzato il processo di *clustering*.
- nella terza fase avviene l'**esportazione** del *dataset* risultante dal processo di *clustering*. In questa fase è importante per il geologo poter costruire diversi tipi di grafici a partire dal *dataset* ottenuto dal *clustering*. I grafici aiutano il geologo nel processo di identificazione dei tipi di rocce del pozzo esplorato. Sono stati anche integrati direttamente nell'applicazione di4g moduli di visualizzazione dei grafici e una tabella contenente le statistiche per ogni singolo *cluster* creato.

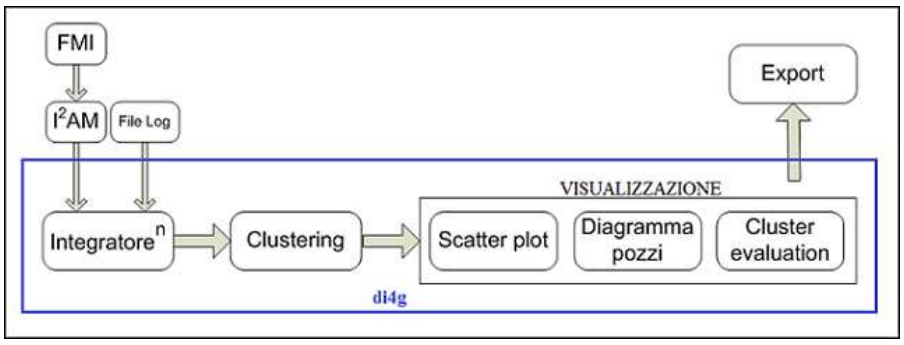


Fig. 1. Workflow di di4g

2.1 Importazione e integrazione dei dati

L'applicazione fornisce un'interfaccia che consente l'importazione di file provenienti da tecniche e strumenti diversi, per numerosi pozzi. Il file d'ingresso dovrà essere una tabella numerica in formato testo con tante righe quante sono le profondità analizzate e tante colonne (separate da un delimitatore o lunghezza fissa) quante sono le variabili prese in esame riguardanti ad esempio la densità e la porosità delle rocce; dovrà inoltre contenere gli identificativi dei diversi pozzi. Un'analisi globale dell'insieme di pozzi, fornisce una classificazione unificata e quindi una rappresentazione coerente delle tipologie di rocce (questo è particolarmente utile solo per pozzi geograficamente vicini).
Come detto in precedenza i dati possono essere generati utilizzando strumenti differenti e avere così anche risoluzioni differenti; il sistema fornisce all'utente la possibilità di importare molti formati di testo, potendo scegliere separatore, numero di righe d'intestazione, e altre caratteristiche.
Inoltre è necessario in questa prima fase eseguire un allineamento dei diversi file importati. L'allineamento avviene analizzando ogni profondità della matrice, ricavata dai dati del primo file, con lo scopo di trovare per ognuna di esse tutte le righe della seconda matrice la cui profondità è "vicina" a quella analizzata. Così facendo l'elenco delle profondità del primo file fa da "guida" per tutti i file. Durante questa fase ci si può inoltre trovare nella situazione in cui questa debba avvenire tra dati di un pozzo che però presentano profondità iniziali e finali che non coincidono: di4g effettua quindi una 'pulizia' delle profondità iniziali e finali della matrice da allineare e si ottiene in questo modo un allineamento molto più uniforme e coerente delle profondità iniziali e finali di ogni pozzo.

2.2 La fase di clustering

Nella seconda fase, di4g esegue il processo di *clustering*. Con il termine *clustering* si intende un insieme di tecniche volte a individuare e raggruppare gli elementi omogenei in un insieme di dati; in particolare, il *clustering* ha l'obiettivo di trovare in un *dataset* dei gruppi che siano il più differenti possibili gli uni dagli altri, ma allo stesso tempo con membri di uno stesso gruppo il più possibile simili tra loro.
Come tecnica di *clustering*, di4g utilizza il *clustering* gerarchico agglomerativo non supervisionato (11), cioè un approccio dal basso verso l'alto che parte inserendo ogni elemento in un *cluster* differente, iterativamente calcola la distanza tra i diversi *cluster* e fonde i *cluster* che si trovano a distanza minore, ossia quelli più simili; il procedimento è iterato fino ad ottenere un unico *cluster*. Lo strumento realizzato consente di scegliere la distanza da utilizzare per misurare la similarità tra due elementi (sono disponibili la distanza di Manhattan, Euclidea e Pearson); inoltre permette di scegliere fra tre diverse metriche (Merge Max, Merge Min, Merge Avg) da utilizzare per selezionare la coppia di *cluster* da fondere.
La normalizzazione è importante per rendere il risultato indipendente dalle unità di misura delle variabili, facendo in modo che tutte le variabili contribuiscano in ugual misura; in questo modo si esegue il calcolo delle distanze su variabili che sono con-

frontabili. Oltre a una classica normalizzazione sull'intero *dataset* è stata realizzata anche la possibilità di eseguire una normalizzazione su ogni singolo pozzo. La normalizzazione sull'intero *dataset* è consigliata quando si è in presenza di dati uniformi (ad esempio se sono stati usati gli stessi strumenti e parametri per la rilevazione); la normalizzazione su ogni singolo pozzo è invece la modalità appropriata nel caso in cui sono stati definiti parametri e tarature differenti degli strumenti utilizzati per l'estrazione dei dati riferiti ai diversi pozzi: così facendo i dati vengono allineati e portati nello stesso *range* correggendo le diverse tarature utilizzate.
In di4g il processo di *clustering* è realizzato tramite la costruzione di una matrice delle distanze in cui si cerca il minimo. Tale matrice può occupare uno spazio considerevole in memoria (ordine $O(N^2)$ dove N è il numero di foglie presenti nel *dataset*) se siamo in presenza di *dataset* di grandi dimensioni e la ricerca del minimo all'interno di tale matrice può diventare un processo oneroso. Per questo motivo di4g ottimizza la ricerca del minimo all'interno di questa matrice tramite alcuni vettori ausiliari. Una quota importante del lavoro si è concentrata nell'analizzare e ridurre i tempi del processo di *clustering*; in particolare l'ottimizzazione del popolamento di questi vettori ha ridotto drasticamente i tempi di esecuzione.
Il risultato del *clustering* gerarchico è un diagramma ad albero chiamato dendrogramma sul quale è possibile generare un taglio per ottenere una configurazione di *cluster*. La finestra di uscita prodotta da di4g consente di eseguire due diversi tipi di taglio: il classico taglio orizzontale e il taglio obliquo (rappresentato in Figura 2). Il taglio obliquo permette di tagliare l'albero ad altezze diverse, semplicemente selezionando i nodi che si desiderano "aprire". In questo modo è possibile espandere in profondità una sola parte dell'albero, e il geologo ha quindi la flessibilità di scegliere la configurazione di *cluster* per lui più rappresentativa.

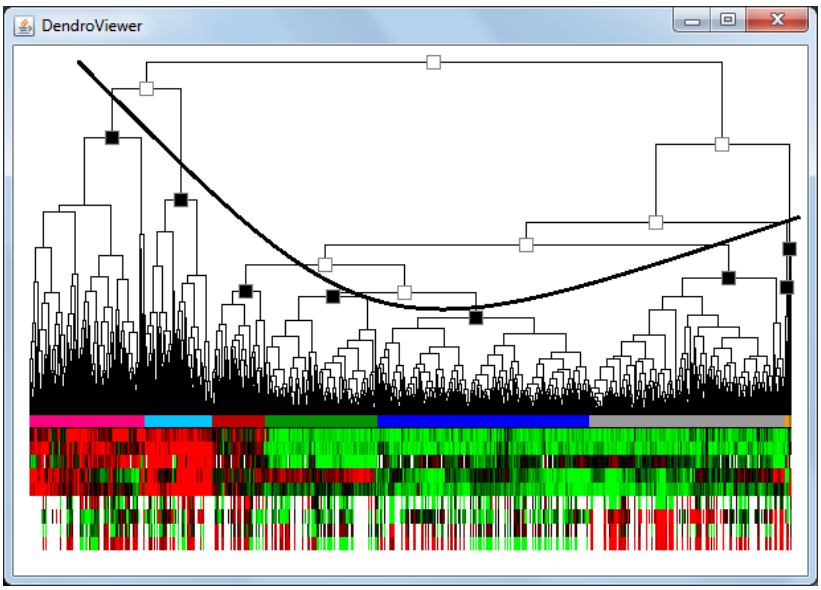


Fig. 2. Dendrogramma con esempio di taglio obliquo (nodi in nero)

2.3 Visualizzazione interattiva

La terza fase di di4g consente la visualizzazione di vari tipi di grafici interattivi aggiornabili simultaneamente nel momento in cui si cambia la configurazione di *cluster* agendo sul dendrogramma oppure se è avviata una nuova procedura di *clustering*.
Uno dei grafici visualizzabili è il diagramma pozzo: rappresentazione del pozzo in verticale con ogni profondità colorata secondo il *cluster* di appartenenza (Figura 3). Di questo diagramma si possono visualizzare diverse configurazioni: su un solo pozzo oppure su pozzi multipli, in questo caso possono essere rappresentati allineati a partire dall'alto oppure scalati secondo la loro profondità reale. Per il geologo il diagramma pozzo è fondamentale in quanto gli consente di individuare le caratteristiche del pozzo alle diverse profondità, in particolare in caso di diagramma su pozzi multipli, il geologo può effettuare un confronto simultaneo e individuare più rapidamente le varie caratteristiche dei diversi pozzi.

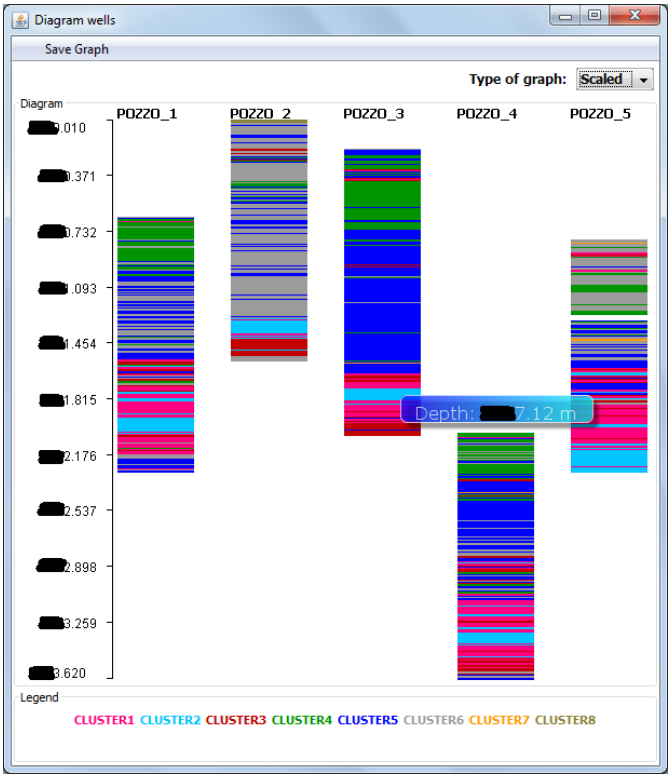


Fig. 3. Diagramma pozzo

Con il grafico di correlazione (Figura 4), di4g rappresenta in uno spazio cartesiano il grado di dipendenza tra due variabili (si possono considerare tutti i cluster contemporaneamente o uno specifico *cluster*).

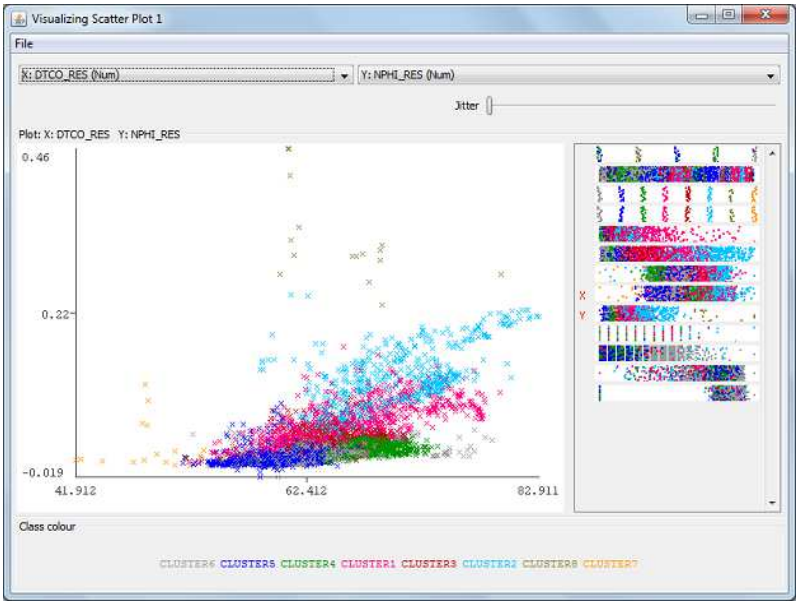


Fig. 4. Scatter plot

Un'ultima funzionalità permette di visualizzare una tabella di 'Cluster Evaluation' riassuntiva delle statistiche dei *cluster*; per ognuno di essi il geologo può analizzare ogni variabile presa in esame. Per semplificare ulteriormente l'interpretazione effettuata dal geologo dalla tabella di 'Cluster Evaluation' per ogni singola variabile è possibile visualizzare il box-plot che consente di rappresentare la distribuzione di una determinata variabile all'interno dei diversi *cluster* e confrontarle tra loro; è inoltre possibile visualizzare anche la distribuzione di una determinata variabile in un *cluster* all'interno dell'istogramma.
Come detto in precedenza tutti i grafici descritti si rigenerano in seguito al cambiamento del taglio sul dendrogramma, in questo modo, l'analisi svolta dal geologo può essere maggiormente interattiva e molto più veloce.

3 Conclusioni

di4g è uno strumento che permette l'importazione di file che contengono grandi quantità di dati e la loro integrazione con dati visuali, per eseguire poi il processo di *clustering*. Questo strumento presenta diversi vantaggi: in primo luogo può essere utilizzato anche da utenti non esperti grazie ad un'interfaccia semplice e intuitiva. Inoltre, avendo la possibilità di importare file contenenti dati riferiti a più pozzi, permette di analizzare le caratteristiche dei diversi pozzi contemporaneamente, effettuando un confronto simultaneo e riducendo così i tempi di analisi. Un ulteriore vantaggio è dato dalla possibilità di visualizzare grafici che consentono di semplificare e velocizzare il lavoro del geologo analista. Ogni tipo di grafico realizzato porta dei vantaggi in fase

di analisi, infatti, è possibile verificare i legami tra due variabili analizzate, eseguire un confronto istantaneo sulle distribuzioni dei diversi pozzi, verificare la distribuzione dei *cluster* lungo la profondità di ogni singolo pozzo. Il loro aggiornamento simultaneo consente al geologo di avere sott'occhio costantemente le modifiche derivanti da una nuova configurazione di *cluster*. Questo consente un'interpretazione molto più veloce di ogni singola zona del pozzo e di verificare così quale tra le diverse configurazioni rappresenta al meglio i dati importati. Infine di4g è nato in ambito geologico, ma a nostro parere può essere utilizzato in diversi ambiti, ovvero in tutti quelli in cui si ha la necessità di eseguire un'analisi su dei dati in ingresso.

Riferimenti

1. **Denis Ferraretti; Giacomo Gamberoni; Evelina Lamma.** I2AM: a Semi-Automatic System for Data Interpretation in Petroleum Geology. PAI 2012, CEUR WS Vol.860.
2. **Denis Ferraretti.** *Data Mining for Petroleum Geology*. Tesi di Dottorato di Ricerca in Scienze dell'Ingegneria. Università degli Studi di Ferrara, 2012.
3. **Alice Piva.** *Sviluppo e ottimizzazione di uno strumento di clustering per la geologia.* Tesi di Laurea Magistrale in Ingegneria Informatica e dell'Automazione. Università degli Studi di Ferrara, 2012.
4. **Denis Ferraretti; Giacomo Gamberoni; Evelina Lamma.** Unsupervised and supervised learning in cascade for petroleum geology. Expert Syst. Appl. 2012, Volume 39, Issue 10, pp. 9504-9514. http://dx.doi.org/10.1016/j.eswa.2012.02.104
5. **Denis Ferraretti; Evelina Lamma; Giacomo Gamberoni; Michele Febo.** Clustering and Classification Techniques for Blind Predictions of Reservoir Facies. AI*IA 2011, LNAI 6934, pp. 348-359, Springer.
6. **Denis Ferraretti; Luca Casarotti; Giacomo Gamberoni; Evelina Lamma.** Spot Detection in Images with Noisy Background. ICIAP 2011, LNCS 6978, pp. 575-584, Springer.
7. **Denis Ferraretti; Evelina Lamma; Giacomo Gamberoni; Michele Febo; Raffaele Di Cuia.** Integrating Clustering and Classification Techniques: A Case Study for Reservoir Facies Prediction. ISMIS 2011, Emerging Intelligent Technologies in Industry, Studies in Computationl Intelligence 369, pp. 21-34, Springer.
8. **Denis Ferraretti; Luca Tagliavini; Raffaele Di Cuia; Mariachiara Puviani; Evelina Lamma; Sergio Storari.** Use Of Artificial Intelligence Techniques To The Interpretation Of Subsurface Log Images. Intelligenza Artificiale 2010, AI*IA, pp. 27-35, IOS Press.
9. **Denis Ferraretti; Giacomo Gamberoni; Evelina Lamma.** Automatic Cluster Selection Using Index Driven Search Strategy. *AI*IA 2009: Emergent Perspectives in Artificial Intelligence.* LNAI 5883, pp 172-181,Springer.
10. **Denis Ferraretti; Giacomo Gamberoni; Evelina Lamma; Raffaele Di Cuia; Chiara Turolla.** An AI Tool for the Petroleum Industry Based on Image Analysis and Hierarchical Clustering. IDEAL 2009, LNCS 5788, pp. 276-283, Springer.
11. **Stephen C. Johnson.** Hierarchical clustering schemes. *Psychometrika.* 1967, Volume 32, Issue 3 , pp 241-254.

Answer Set Programming and Declarative Problem Solving in Game AIs

Davide Fuscà, Stefano Germano, Jessica Zangari,
Francesco Calimeri, and Simona Perri

Dipartimento di Matematica e Informatica, Università della Calabria, Italy
{ddfusca,stefanogermano0,jessica.zangari.90}@gmail.com,
{calimeri,perri}@mat.unical.it

Abstract. Designing and implementing AI in games is an interesting, yet complex task. This paper briefly presents some applications that make use of Answer Set Programming for such a task, and show some advantages of declarative programming frameworks against imperative (algorithmic) approaches while dealing with knowledge representation and reasoning: solid theoretical bases, no need for algorithm design or coding, explicit (and thus easily modifiable/upgradeable) knowledge representation, declarative specifications which are already executable, very fast prototyping, quick error detection, modularity.

Keywords: Declarative Programming, Answer Set Programming, Artificial Intelligence, Computational Logic, Knowledge Representation and Reasoning

1 Introduction

This work presents some Artificial Intelligence applications designed and implemented during the Course of Artificial Intelligence in the context of the Computer Science Bachelor Degree at University of Calabria, Italy[1]. The aim of each project was to study and reproduce the behavior of a skilled player of some "classic" games. In particular, the explicit knowledge and the reasoning modules have been implemented by means of Answer Set Programming (ASP) techniques, and the projects can be seen as a nice showcase of features, power and advantages coming from the use of ASP itself. In the following, after a brief introduction to ASP, we will illustrate the above mentioned projects, especially focusing on the approaches adopted for the implementation of the AIs.

2 ASP and Declarative Problem Solving

Answer Set Programming (ASP) [3,11] became widely used in AI and is recognized as a powerful tool for knowledge representation and reasoning (KRR),

[1] http://www.mat.unical.it/ComputerScience

especially for its high expressiveness and the ability to deal also with incomplete knowledge [3]. The fully declarative nature of ASP allows one to encode a large variety of problems by means of simple and elegant logic programs. The semantics of ASP associates a program with none, one, or many answer sets, each one corresponding one-to-one to the solutions of the problem at hand.

For instance, let us consider the well-known NP-complete *3-Colorability* problem: given a graph, decide whether there exists an assignment of one out of three colors to each node, such that adjacent nodes never have have the same color. Each instance can be represented by a set of facts F over predicates $node(X)$ and $arc(X, Y)$. The following program, in combination with F, computes all 3-Colorings (as *answer sets*) of the graph represented by F.

$$\begin{array}{ll} r_1: & color(X, red) \mid color(X, green) \mid color(X, blue) \leftarrow node(X). \\ r_2: & \leftarrow \ color(X_1, C), color(X_2, C), arc(X_1, X_2). \end{array}$$

Rule r_1 expresses that each node must be colored either red, green, or blue; due to the minimality of the answer sets semantics, a node cannot be assigned more than one color. The integrity constraint r_2 enforces that no pair of adjacent nodes (connected by an arc) is assigned the same color.

The paradigm adopted above is one of the most commonly used among ASP programmers, and is referred to as the "Guess&Check" methodology [8]. An extension to this methodology is the so called "Guess/Check/Optimize" [4].

In summary, an ASP program that matches GCO features 3 modules:

- **Guessing Part** defines the search space (by means of Disjunctive Rules)
- **Checking Part** (optional) checks solution admissibility (by means of Integrity Constraints)
- **Optimizing Part** (optional) specifies a preference criterion (by means of Weak Constraints)

In the latest years many efforts have been spent in order to obtain solid and efficient systems supporting ASP, and a number of modern systems are now available (see [5] for a pretty comprehensive list and a more detailed bibliography about ASP). In the present works we used DLV [12], a deductive database system supporting Disjunctive Datalog properly enriched with weak constraints (to express optimization problems), aggregates (to better deal with real data and applications), queries and other language extensions.

It is worth noting that many other logic formalisms out there can explicitly represent an agent's knowledge, and allow one to accomplish the AI jobs herein discussed; one of the most widely known is Prolog, that has already been employed before in AI games [1,6,7,10]. However, we wanted to specifically follow a fully declarative approach. Prolog, for instance, requires to know the resolution algorithm while writing a program, while in ASP the order of rules within a program, as well as the order of subgoals in a rule, is irrelevant. This paper does not aim at specifically discussing differences between ASP and other formalisms; for further details about such topics we refer the reader to the extensive existing literature.

3 Applications

3.1 General Architecture

The applications herein presented share the same basic architecture (see Figure 1), which can be seen as consisting of three layers: the *core*, the *ai* and the *gui*. The *core* layers connects the *ai* layer with the *gui* layer: it manages the game via the *gui* while providing the *ai* with proper information and getting in turn the (hopefully "right") actions to be performed. The ad-hoc *gui* layer allows a user to play the game against the machine in an intuitive way.

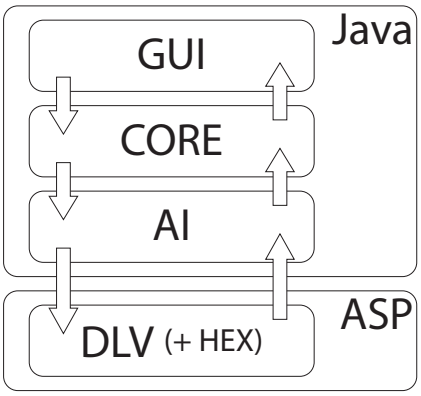


Fig. 1: General Architecture

Indeed, the architecture hides a more general framework for ease the development of applications in which AI is crucial. This is not only the case of games, but also the scenario of relevant practical problems in presence of incomplete or contradictory knowledge. The *ai* module is uncoupled from the *core* and the *gui*, that can be designed independently from the AI; in addition, the latter can be gradually improved (or easily interchangeable).

In the present setting, the *ai* module is based on ASP and uses the DLV system as the actual ASP solver.

The applications are implemented in Java, and in particular for the 2D visualization of the game we used the Java GUI widget toolkit Swing.

3.2 Connect Four

Game Description Connect Four is played by two opponents on a vertical 7×6 rectangular board. The players fill the board by dropping 1 disk in turn from the top to the bottom of the board: if a disk is dropped in a column, it falls down onto the lowest unoccupied position within it. The winner is the first player who gets four of her disks in a line, connected either horizontally, vertically, or diagonally. As common in board games, we will refer to the "White" player as the one who begins the game, and to the "Black" player as the other one.

Implementing the Artificial Intelligence Connect four was mathematically solved in 1988 by Victor Allis[2], showing if both players play perfectly, the White one is always able to win if she begins from the middle column, while starting from another column enables the second player to always force a drawn.

Three levels of AI have been implemented: the hardest one implements the rules of perfect play emerged from Allis' work, while the easiest relies on some classical heuristic strategies, and the intermediate mixes some basic Allis' rules with some common strategies. Furthermore, the White and Black strategies sensibly differ. Intuitively, the White player needs to *keep* the advantage deriving from the simple fact that she starts the game, while the other has heavier burden: she needs to *turn* the game on his advantage defending from the natural ongoing of the game in favour of the opponent and trying to exploit the possible "errors" made by the adversary. This has been straightforwardly accomplished by developing different ASP programs, each one based on the "Guess/Check/Optimize" technique:

- **WhiteAdvanced**, **BlackAdvanced**: highest level for both players;
- **WhiteIntermediate**, **BlackIntermediate**: intermediate level;
- **Easy**: easiest level; in this case we can observe the consequences of undifferentiated strategies.

A further ASP module, called **CheckVictory**, is used to check if one of the player is the winner each time that player makes a move. In order to provide the reader with an idea about the way ASP is employed here, we show next some ASP rules shared by all the modules; full ASP encodings are available online (see Section 4).

Guess : $selectedCell(R,C) \mid notSelectedCell(R,C) \leftarrow playableCell(R,C).$
Check 1 : $\leftarrow \neg\#count\{R,C : selectedCell(R,C)\} = 1.$
Check 2 : $\leftarrow \neg selectedCell(5,3), playableCell(5,3).$
Optimize 1 : $:\sim threat(A,R,C), \neg selectedCell(R,C),$
$playableCell(R,C), me(A).[1:5]$
Optimize 2 : $:\sim threat(A,R,C), \neg selectedCell(R,C),$
$playableCell(R,C), opponent(A).[1:4]$

The **Guess** rule generates all possible split of the "playable" cells (R,C standing for row and column, respectively) into two sets, the selected for the next move, and the rest. But the player can occupy only one cell at each turn: the **Check 1** rule enforces this. This is a basic player, with no particular "intelligent" behaviour.

The **Check 2** provides us with a first strategic glimpse: if cell (5,3) is still "available"[2], the player has to occupy it: this is known to give the player an advantage. The last two rules are part of the "optimize" task: the first is an

[2] A cell is "playable" if it is not occupied yet and stands in the lowest row, or if the cell below it in the same column is already occupied.

attack rule, while the second is a defence one. A "threat" for a player A is a cell which, if taken by player A, connects four of her disks: *threat*(A,R,C) means that the cell (R,C) is a threat for player A. If the player has a threat, she should occupy the threat cell as soon as possible: this is "pushed" by the **Optimize 1** rule. Similarly, rule **Optimize 2** "pushes" the player to occupy a cell if it is a threat for the opponent. The two optimization statements have different weights: in case the player can choose among an attack or a defence move, the rational behaviour is to perform an attack. Actually, rule **Optimize 1** has the greatest value among all the other "optimize" rules describing different strategies: if victory is just a move away, just make that move!

Intuitively, the various AI program differs especially in their optimization. The aim is to depict different scenarios by means of different strategies: this made the game very well suited to be analysed from an AI perspective. It is of clear interest to compare the different AI levels in a game between two artificial players, assessing the "perfect" against the heuristic-based strategies; in addition, we can assess the AIs against human players, who can provide a wide range of different strategies. A further positive aspect, is that by avoiding a brute-force search approach in favour of a *knowledge-based approach*, not only changes are easy to implement and assess, but different *styles* can also be easily described and actually implemented.

3.3 Reversi

Game Description Reversi is a strategy board game with 64 game pawns, each one featuring both a black and a white side. Differently from the typical assumption, Black plays first and places a disc with the black side up. At each move, there must exist at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and another black piece, with one or more contiguous white pawns between them. Once the piece is placed, black turns over all white discs lying on such straight line between the new piece and any anchoring black pawns. Similarly does the White. If a player does not have a legal move available at any time, she must pass, and her opponent plays again; if neither player has a legal move, the game ends. The winner is the player with the higher number of pawns of his own color on the board.

Implementing the Artificial Intelligence The intelligence is described by means of logic rules according to the GCO (Guess/Check/Optimize) technique. At each turn, some facts representing the board state are coupled with the logic program, which has been conceived so that answer sets represent the moves: strong constraints "drop" any answer set representing non-valid moves, while weak constraints "select" the best one.

We started from a basic game manual[3]. In order to create different levels of AIs we selected some strategies with different features, each one represented by

[3] available at `http://www.fngo.it/corsobase.asp` on the FNGO (Italian Federation of the Othello Game) website

a set of rules that can be added incrementally: each level uses all the strategies from of the previous, plus some more sophisticated. The different AIs can be summarized as follows:

- **None**: the simplest one; trivially chooses a valid move;
- **Basic**: tries to maximize the number of pawns eaten with each move;
- **Medium**: adds the strategy of "corners and stable pawns";
- **Hard**: adds the strategy of "border pawns and walls".

Five different ASP modules helps at representing such AIs. One module models the Guess/Check, one models the current state of the board (plus some other useful pieces of information), and we have one additional module for each strategy. The artificial player can change its behaviour by simply running the first two modules along with the one related to the desired strategy. We present next some ASP rules featured by the mentioned modules; full ASP encodings are available online (see Section 4).

Guess	$selectedCell(R, C) \mid notSelectedCell(R, C) \leftarrow validCell(R, C).$
Check	$\leftarrow \neg \#count(R, C : selectedCell(R, C)) = 1.$
Optimize	$:\sim notSelectedCell(R, C), cornerPawn(R, C).\ [1 : 15]$

Similarly to the previous case, the two rules (**Guess**) and (**Check**) select exactly one cell, among all legal moves. The rule (**Optimize**) is a Weak Constraint that expresses the fact that, if possible, the player should choose the corner cells at the corners (these are known to be "strategic positions" in the game); this is a nice example of how easy is to incorporate explicit knowledge of a domain, as it might be provided by an expert.

The Reversi Action Addon We have designed and implemented also an add-on (the *Reversi Action Addon*) for the *Action Plugin* of the DLVHEX solver [9], thus making the system able to play an online version of Reversi[4]. It employs the same logic rules herein described, with some adaptations for the sake of compatibility with the DLVHEX solver and the online game. The Add-on is completely autonomous, with no need of human intervention (apart from the boot). By means of Javascript and Perl scripts it logins into the website, recognizes the game status and makes its moves. It's also able to wait the opponent's move and to understand when the game is over.

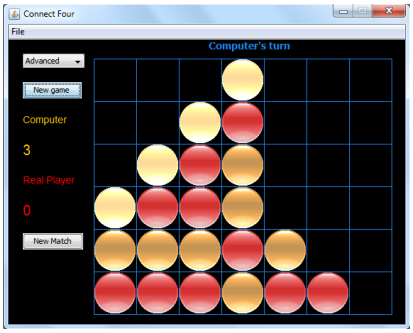
4 Conclusion

In this paper we presented some applications that make use of the capabilities of ASP in the design and implementation of the AI, which is, in these cases, the most complex (and interesting) part. We implemented some "classic" strategies,

[4] Online game available at `http://www.yourturnmyturn.com`

typically difficult to implement when dealing with the imperative programming, in a rather simple and intuitive way. Moreover, we had the chance to test the AI without the need for rebuilding the application each time we made an update, thus observing "on the fly" the impact of changes: this constitutes one of the most interesting features granted by the *explicit* knowledge representation. In addition, we developed different versions of the AIs, in order to show how easy is to refine the quality or to generate different strategies or "styles"; and these include also non-winning, *human-like* behaviors.

The games herein presented can be downloaded at `https://www.mat.unical.it/calimeri/files/AIgames/PAI2013/games.zip`; the packages contain the full ASP programs herein sketched.



(a) Connect 4 Screenshot

(b) Reversi Screenshot

References

1. CGLIB- A Constraint-based Graphics Library for B-Prolog. `http://probp.com/cg_examples.htm`
2. Allis, L.V.: A knowledge-based approach of connect-four. Vrije Universiteit, Subfaculteit Wiskunde en Informatica (1988)
3. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
4. Buccafurri, F., Leone, N., Rullo, P.: Strong and Weak Constraints in Disjunctive Datalog. In: Dix, J., Furbach, U., Nerode, A. (eds.) Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'97). Lecture Notes in AI (LNAI), vol. 1265, pp. 2–17. Springer Verlag, Dagstuhl, Germany (Jul 1997)
5. Calimeri, F., Ianni, G., Ricca, F.: The third open answer set programming competition. Theory and Practice of Logic Programming 1(1), 1–19 (2012)
6. Clark, K.L.: Negation as failure. In: Logic and data bases, pp. 293–322. Springer (1978)
7. Colmeraner, A., Kanoui, H., Pasero, R., Roussel, P.: Un systeme de communication homme-machine en francais. Luminy (1973)

8. Eiter, T., Faber, W., Leone, N., Pfeifer, G.: Declarative problem-solving using the dlv system. In: Logic-based artificial intelligence, pp. 79–103. Springer (2000)
9. Fink, M., Germano, S., Ianni, G., Redl, C., Schüller, P.: Acthex: Implementing hex programs with action atoms. In: 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), LNAI 8148. pp. 317–322 (2013)
10. Finnsson, H., Björnsson, Y.: Simulation-based approach to general game playing. In: AAAI. vol. 8, pp. 259–264 (2008)
11. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
12. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM Transactions on Computational Logic 7(3), 499–562 (Jul 2006)

Towards smart robots: rock-paper-scissors gaming versus human players

Gabriele Pozzato, Stefano Michieletto, and Emanuele Menegatti

Intelligent Autonomous Systems Lab (IAS-Lab)
Department of Information Engineering (DEI)
University of Padova
{pozzatog,stefano.michieletto,emg}@dei.unipd.it
http://robotics.dei.unipd.it

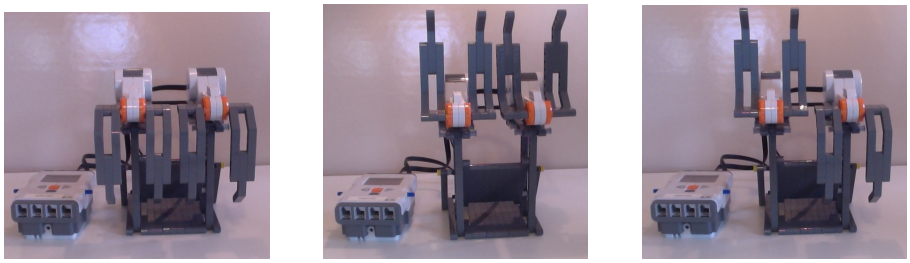
Abstract. In this project a human robot interaction system was developed in order to let people naturally play rock-paper-scissors games against a smart robotic opponent. The robot does not perform random choices, the system is able to analyze the previous rounds trying to forecast the next move. A Machine Learning algorithm based on Gaussian Mixture Model (GMM) allows us to increase the percentage of robot victories. This is a very important aspect in the natural interaction between human and robot, in fact, people do not like playing against "stupid" machines, while they are stimulated in confronting with a skilled opponent.

1 Introduction

This research aims to develop an application to let users playing with a smart robot in a series of rock-paper-scissors games. Users can interact naturally with the robot without any remote controls or keyboards: this application wants to be as user-friendly as possible. Humans can communicate with the robot via an RGB-D sensor using a default set of positions and gestures. The used robot is a NXT of the LEGO Mindstorms series (Fig. 1).

Vision system provided by the sensor and control of the NXT are coordinated using ROS (Robot Operative System). Analyzing depth images given by the sensor we are able to understand positions and gestures of the human being. The application development consists of three steps. At first, the system recognizes the user pose, at the pose we associated the number of games to play, a game is composed by three rounds. Once a round is started, a hand gesture recognition algorithm developed by LTTM (Multimedia Technology and Telecommunications Lab) looks at the move played by the human. At the same time, the system chooses the robot gesture, using an Artificial Intelligence (AI) algorithm we developed. The algorithm is able to infer human moves starting from its previous game experience.

The remainder of the paper is organized as follows: in Section 2 the robotics interface is presented. In Section 3 the learning system used to infer human strategies is described. Section 4 presents the tests performed on the AI algorithm and the related results. Finally, Section 5 contains conclusions.



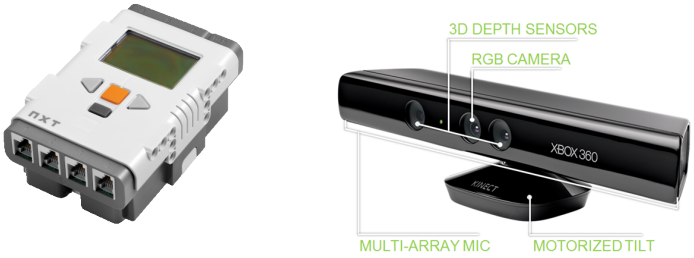
(a) (b) (c)

Fig. 1. Robot gestures: rock (a), paper (b), scissors (c)

2 The robotics interface

The robot used in this work is a NXT of the LEGO Mindstorms series. NXT is a low cost programmable robotics kit developed by LEGO. The core of this kit is an "intelligent" brick (Fig. 2a) where are located the CPU, a Flash memory and several I/O ports to control motors and sensors. The aim of this kit is providing a platform that can be simply used also by inexperienced users.

In our research, the central unit of the LEGO platform (NXT brick) is used to control two different motors simulating the three human gestures. The three configurations are showed in Figure 1.



(a) (b)

Fig. 2. The hardware used in this work: the NXT brick (a) and the Microsoft Kinect sensor (b)

For the understanding of gestures played by human users a RGB-D sensor is used. In specific the vision system is provided by the low cost Microsoft Kinect sensor (Fig. 2b). Communication and synchronization between sensor and robot are obtained using ROS (Robot Operative System). This open-source framework allows us controlling the used hardware via C++ and Python programming languages.

3 The learning system

As humans do not play random moves it is also fundamental the robot use different strategies in order to increase its probability to win a match. The AI algorithm performing the robot choice is based on Gaussian Mixture Model. Different works have already been developed [1] [2] in our lab by using GMM to solve complex problems. In this case we tested the GMM flexibility by solving a problem that sounds simply if we look at the possible options, but equally hard to solve by identifying a clear strategy even for humans being.

3.1 Gaussian Mixture Model

A GMM [3] θ is a parametric probability density function represented as a weighted sum of K Gaussian components given by the equation:

$$p(\boldsymbol{x} \mid \theta) = \sum_{k=1}^{K} \omega_k g(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k), \tag{1}$$

where $\boldsymbol{x}$ is a vector $\boldsymbol{x} = \{x_1, \ldots, x_d\}$ with size d and ω_k are the weights of the various components $g(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k)_{k=1}^{K}$. Each component is a d-variate Gaussian function:

$$g(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2} \mid \Sigma_k \mid^{1/2}} exp\Big\{ -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)' \Sigma_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k) \Big\} \tag{2}$$

with mean vector $\boldsymbol{\mu}_k$ and covariance matrix Σ_k. The weights ω_k satisfy the normalization property $\sum_{k=1}^{K} \omega_k = 1$.

The GMM is parametrized by mean vectors, covariance matrices and mixture weights from all the K components. These parameters are represented by the notation:

$$\boldsymbol{\theta} = \{\theta_k\}_{k=1}^{K} = \{\omega_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^{K} \tag{3}$$

Extremely important is the training stage and the maximum likelihood (ML) parameters estimation. Using the Expectation-Maximization algorithm we obtained the value of the parameters which maximize the likelihood. Furthermore the estimation of the optimum K is reached using the Bayesian Information Criterion (BIC) [4]. BIC selects the better model in a class of models with different number of parameters. In this way we determined the model with the minor number of parameters that guarantees a good data fitting. For further details see [5] and [6].

3.2 Rock-Paper-Scissors

A dataset X = $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ with N the number of instances is used to train the GMM. GMM features $\boldsymbol{x}_j|_{j=1}^{N}$ have dimension $d = 8$ and they are defined as

$$\boldsymbol{x}_j = \{(r_{j-3}, h_{j-3}), (r_{j-2}, h_{j-2}), (r_{j-1}, h_{j-1}), (r_j, h_j)\} \tag{4}$$

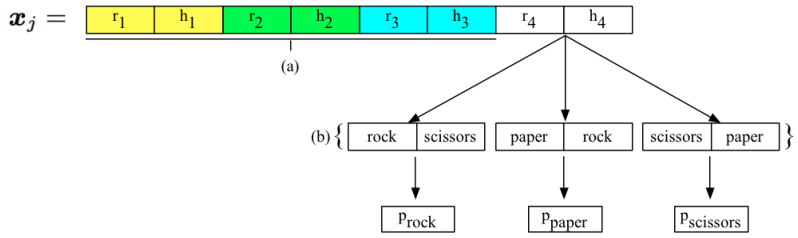


Fig. 3. (a) The three previous rounds used to determine r_4. (b) The three pairs used to complete the sequence of three previous rounds. From this sequences we obtain the probabilities of rock, paper and scissors.

where (r_j, h_j) is the j^{th} round played, with r_j as robot move and h_j human move. Both $r_j, h_j \in \{rock, paper, scissors\}$. Each instance in the dataset consists of four rounds, the j^{th} one and three previously played.

The idea is using a history of the three previous rounds to determine the gesture r_j the robot will play in the current game to contrast the h_j gesture played by the human. In the testing phase we complete the sequence of three previous games in Fig. 3(a) with each of the three "favorable" pairs in Fig. 3(b), in order to obtain three prior probabilities related to these configurations. The pairs are called "favorable" because they correspond to a robot victory.

Thus, after obtaining the three completed sequences called $\boldsymbol{x}_{rock}$, $\boldsymbol{x}_{paper}$ and $\boldsymbol{x}_{scissors}$ in relation to the first symbol of completion we obtain three priors from the GMM for rock (p_{rock}), paper (p_{paper}) and scissors ($p_{scissors}$):

$$p_i = P(\boldsymbol{x}_i) = \sum_{m=1}^{K} P(\theta_i = m \mid \boldsymbol{x}_i), \tag{5}$$

where θ_k denote the k^{th} weighted Gaussian component and $i = rock, paper, scissors$.

Using these updated probabilities we choose the robot gesture generating a random number n in the interval [0,1] and then partitioning the interval [0,1] in three disjoint subintervals. The final gesture will be:

- Rock, if $n \in [0, p_{rock})$,
- Paper, if $n \in [p_{rock}, p_{rock} + p_{paper})$,
- Scissors, if $n \in [p_{rock} + p_{paper}, 1]$.

After the round is played and we have obtained both robot and human gestures we are able to complete a new sequence $\boldsymbol{x}_j$ and we could use it in order to update the training set. GMM is updated every iteration t reloading the entire training set.
BIC is executed every 2^t iterations, limiting the value of K to 50.

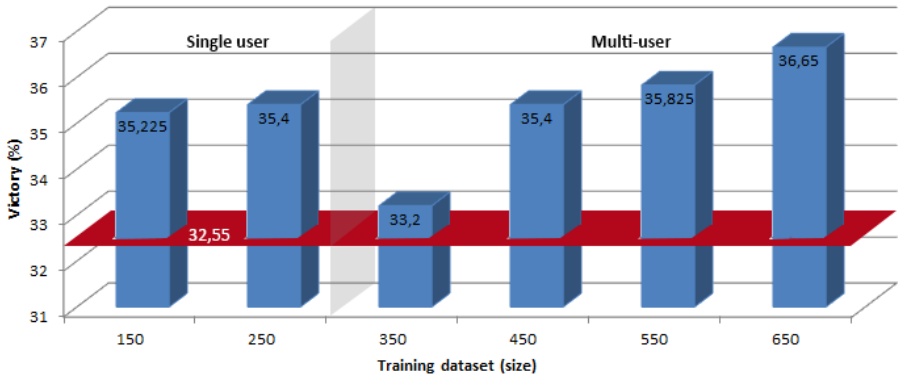


Fig. 4. The histogram shows the percentages of robot victories for training sets of different size. The red plane shows the percentage of victories obtained with the equiprobability policy in the first test.

4 Experimental results

We tested our approach comparing it with the standard random choice strategy. Tests are performed on 200 rounds and we averaged the results on 20 iterations.

Random strategy Rock, paper and scissors are equiprobable and robot gestures are chosen randomly. In this case we obtain a percentage of robot victories of 32.55% (red plane in Fig. 4).

Our GMM based strategy The strategy adopted is the one previously described in this paper. Probabilities were updated over time and the test was performed on different size training dataset: five different initial dataset were considered from 150 to 650 elements (Fig. 4). Datasets composed by 150 and 250 elements are referred to a single user, while in the ones composed by 350, 450, 550, and 650 elements we asked to several users to try out our system. Every user played in more than one session. A session was composed from ten to twenty consecutive games. In the single user cases it is easy to see a quick increment of the percentage of victories of even with relative small size datasets. With a dataset of 250 instances, the robot will win with a probability of the 35,4%. In the multi-user version more data were needed for GMM settling. When the model starts generalizing and managing an higher number of users the percentage grows to a maximum of 36.65% for the dataset of 650 instances.

Our choice to adopt a specific strategy using a GMM approach in order to predict next human move with respect using a random strategy is also supported by humans' percentages of use of the three gestures rock, paper and scissors (Fig. 5). We can notice that users do not play randomly but they adopt a precise strategy.

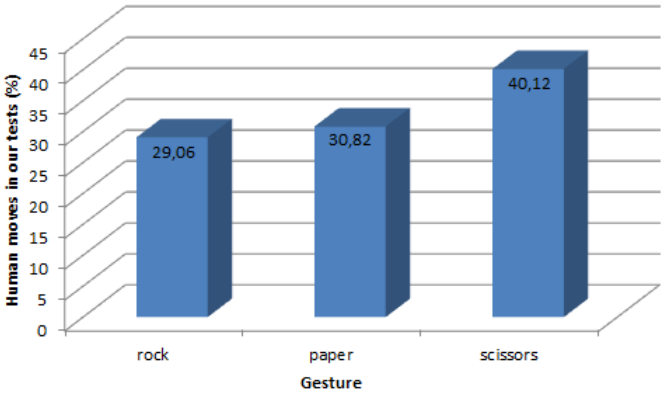


Fig. 5. The histogram shows humans' percentages of use of the three gestures.

5 Conclusions

In conclusion we have implemented an AI algorithm to allow human users playing naturally with an intelligent robot. Considering the maximum training dataset size (650 instances) the chosen approach has led to a robot winning percentage of 36.65% with respect the percentage of 32.55% obtained with a random choice strategy.

In this work we focused our attention to the victory percentages of the robot without taking care of how much human users are interested and involved in the game. In a similar way it is possible to fit with evidence and practice in game design, where the aim is to match the ability of the opponent to keep her/him involved in the game. In fact, we record both human and robot moves so we can easily choose the policy by using the history of previously played matches.

In the future we want to test the algorithm on larger datasets for a better estimation of winning percentage and implement an algorithm able to generate a specific model for every individual user. Finally an interesting improvement could be a comparison between the GMM trained strategy and a strategy choosing the gesture to play according to the human percentages of use reported in Fig. 5.

Acknowledgment

We would like to thank Fabio Dominio and dr. Pietro Zanuttigh of the University of Padova, members of the LTTM (Multimedia Technology and Telecommunications Lab) directed by Prof. G.M. Cortelazzo, for the hand gesture recognition library for the RGB-D sensor.

References

1. Michieletto, S., Chessa, N., Menegatti, E.: Learning how to approach industrial robot tasks from natural demonstrations. In: Proceedings of IEEE Workshop on Advanced Robotics and its Social Impacts ARSO2013, IEEE (2013)
2. Michieletto, S., Rizzi, A., Menegatti, E.: Robot learning by observing humans activities and modeling failures. In: IROS workshops: Cognitive Robotics Systems (CRS2013), IEEE (Nov 2013)
3. Reynolds, D.: Gaussian mixture models. Encyclopedia of Biometric Recognition (2008)
4. Konishi, S., Kitagawa, G.: Bayesian information criteria. In: Information Criteria and Statistical Modeling. Springer Series in Statistics. Springer New York (2008) 211–237
5. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc., New York, NY, USA (2001)
6. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. 2 edn. Pearson Education (2003)

Stabilize humanoid robot teleoperated by a RGB-D sensor

Andrea Bisson*, Andrea Busatto*, Stefano Michieletto, and Emanuele Menegatti

Intelligent Autonomous Systems Lab (IAS-Lab)
Department of Information Engineering (DEI)
University of Padova
{bissonan,busattoa,stefano.michieletto,emg}@dei.unipd.it
http://robotics.dei.unipd.it

Abstract. An easy way to let a robot execute complex actions is to let the robot copy human moves. Useful information are read by sensors and elaborated to convert them into robot movements.
This work focuses on keeping the robot balanced while it is performing an action: grasp an object laying on the ground in front of the robot. Experiments are performed with a human user moving in front of the sensor using a humanoid robot performing the same action, the Vstone Robovie-X.

1 Introduction

In this paper we describe a laboratory experience involving a humanoid robot an its behaviours. This work follows to some similar experiences [1] already developed in our lab. The proposed system elaborates online human movements in order to control the robot joints. The goal is to make robot picking up an object teleoperated by an human actor. The robot has to avoid unstable situations by automatically balancing the input movements could make it fall down. *Robot Stabilization* is the key step of the complete process (Fig. 1) used to compute suitable joint values: the algorithm elaborates a feedback signal to keep the robot balanced during the movement.

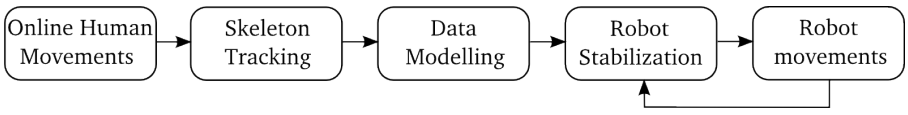


Fig. 1. Overview of the entire system proposed in this paper.

The robot used in this work is a humanoid: the Vstone Robovie-X. Robovie-X is a flexible and economic platform already used in robotics courses to make

* These authors contributed equally to this work.

students familiar with humanoids. Our lab also provided us a simulated version of this humanoid in order to prevent robot damages during the tests.

The remainder of the paper is organized as follows: in Section 2 the data acquisition system is explained, while the robot structure is illustrated in Section 3. The stability algorithms used, math and physics behind them and some specific tests performed are described in Section 4. Finally, Section 5 contains conclusions.

2 Data acquisition

In this paper, data are acquired from a low cost RGB-D sensor: a person can perform the desired actions in front of the sensor without any need of additional equipment. A skeleton tracking system is used to extract human joints positions and angles from the raw data provided by the sensor, namely a Microsoft Kinect. The skeleton information (Figure 2) is subsequently remapped to the robot model, so that the person acting in front of the camera could simply teleoperate the robot by his own body, similarly to the systems described in [2], [3], and [4].

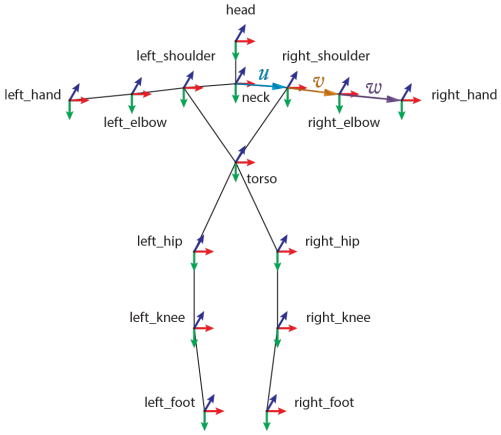


Fig. 2. Skeleton joints provided by the tracker.

3 Robot structure analysis

The robot used in this work is a Vstone Robovie-X (Fig. 3). It is a small humanoid robot with 17 DoF (Head 1, Arms 6, Legs 10). In this work the whole robot body has been used to get it stable. In particular the upper body is also involved in a grasping action so arms and shoulders can't be enforced to completely

keep the stability. The lower body is mainly controlled in order to maintain balance but it is also important to let the robot reach objects easily.

Fig. 3. The small humanoid used in this work: the Vstone Robovie-X.

4 Robot stability

In the following subsections are described the stability algorithms and optimizations for the robot movement.

4.1 Base strategy

As we said before, our primary goal is to make a robot pick up an object laying in front of it by imitating the human movements coming from a skeleton tracking system. The main challenge is to keep the robot stable while it is crouching and grasping the object.

A consolidated method to maintain the robot stability [5] is to keep the Center of Mass (CoM) projection point inside the contact area of the feet with the ground. The CoM projection point of the robot should be calculated in order to reach two different purposes: maintain the point inside the safe balanced area and keep the robot movements as similar as possible to the user motions.

At each instant, the CoM is equal to:

$$CoM_x = \frac{\sum_{k=1}^{N} m_k\, x_k}{M}; \quad CoM_y = \frac{\sum_{k=1}^{N} m_k\, y_k}{M}; \quad CoM_z = \frac{\sum_{k=1}^{N} m_k\, z_k}{M} \qquad (1)$$

where $N = 17$ is the number of joints, m_k is the k^{th} joint inertial mass, M is the mass of the robot, and x_k, y_k, z_k are the coordinates of the k^{th} joint with respect to the Torso joint.

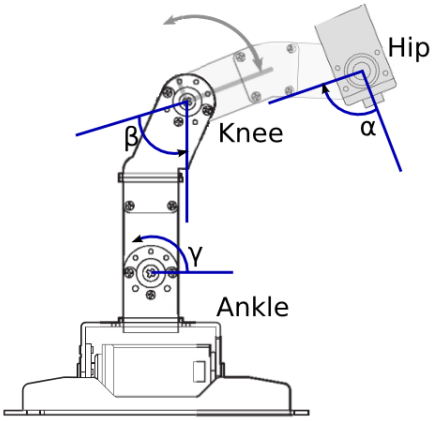


Fig. 4. Main joint angles involved in balance.

Thus, the ground projection of the CoM is given by (CoM_x, CoM_y), and it has to satisfy the constraints $-90cm < Com_x < 90cm$ and $-35.5cm < Com_y < 35.5cm$ that is the area covered by the robot foot in the initial standing position. The selected movement is quite simple, so several solutions are feasible to solve the problem. We imposed a strong relation between hip, knee an ankle joints in order to involve all the lower body joints and at the same time adapt to the human natural behaviour.

$$\gamma = \alpha = \frac{\beta}{2}\,, \qquad 0 \leq \beta \leq 157^o \tag{2}$$

where α is the pitch hip angle at instant t, β is the knee angle at instant t, and γ is the pitch ankle angle at instant t. Fig. 4 shows the three described angles in the robot model. The method was tested experimentally on both simulated and real robot.

4.2 Refinement

We refine the described technique applying different strategies in order to avoid some rough robot movements we noticed during the initial tests. Studying the dynamics of the task, we decided to limit roll movements (i.e. lateral movements) in joints not involved in reaching the goal, like hips and ankles.

Fast and sudden movements could threaten the robot stability while performing an activity. Ensuring the smoothness of all robot moves is essential in balancing purpose. Without any focused control, the input data can make the robot move jerkily. This problem is due to the fast human movements compared with the frame rate of the sensor and to the margin of error of the skeleton tracker computing joints positions.

The data acquired by the RGB-D sensor are filtered to remove the noise by calculating the mean value of the last three angle values of every joint in order to avoid rough robot movements and obtain a smoother motion.

The pose feedback equation is:

$$\hat{\xi}_t = \frac{\hat{\xi}_{t-2} + \hat{\xi}_{t-1} + \xi_t}{3} \tag{3}$$

where ξ_t is the raw value given by the skeleton tracker for a certain joint, $\hat{\xi}_t$ is the computed value at the instant t for the considered joint, $\hat{\xi}_{t-1}$ is the computed value at the instant $t-1$ for the considered joint, $\hat{\xi}_{t-2}$ is the computed value at the instant $t-2$ for the considered joint.

Moreover, the movements of the right and left side of the robot body are coordinated in order to make easier for the robot to grasp the object. In this way, we also increase the robot stability and its precision during the motion.

Finally, the proportions between lower body angles are computed according to the equation 2. These refined data are used as input to the algorithm checking if the CoM projection on the ground is inside the stability area.

Again, the whole system has been tested with many users and different objects[1] on both real and simulated environment. The applied refinements significantly improved the performance and the users easily reached the goal. It is worth to notice that a slight delay is introduced by USB connection between the system and the real robot, nevertheless no delay is present in the simulated model, that works at 30 fps. We also can make real and simulated robot work together, so humans can take advantage of the information provided by the virtual model.

5 Conclusions

In this paper, a robot behaviour was developed in order to maintain robot stability during a pinking up task performed by human teleoperation. The system keeps the robot stable using movements as similar as possible to the user ones. Our technique reach real-time performances, the robot is able to move smoothly according to user movements.

The system has been tested with different users and objects and it has also been exposed as a working demo to the "The Researchers Night"[2] in Padova.

The work described in this paper could be used jointly with a Robot Learning from Demonstration (RLfD) framework already developed in our lab [6]. RLfD is a programming paradigm that uses demonstrations in order to make a robot learn new tasks. The system can be applied to improve the stability of the robot while it is performing the learned activity. The idea is to extend our lab

[1] Few videos of some tests realized: `https://www.youtube.com/watch?v=LJyXT6gAyo8` `https://www.youtube.com/watch?v=AOIkVLn3Kng` `https://www.youtube.com/watch?v=GS9A4prXfpI`

[2] http://www.near-nottedeiricercatori.it/

experience by modeling the movements of the person in front of the sensor using a Gaussian Mixture Model (GMM). The GMM will provide us a probabilistic way to to classify a movement as safe or unsafe in order to prevent robot falls.

As future work we also will improve the system by increasing the set of supported actions and applying it to scenarios involving innovative robotic platforms like exoskeletons.

References

1. Michieletto, S., Ghidoni, S., Pagello, E., Moro, M., Menegatti, E.: Why teach robotics using ROS. Journal of Automation, Mobile Robotics & Intelligent Systems (in press)
2. Pollard, N.S., Hodgins, J.K., Riley, M.J., Atkeson, C.G.: Adapting human motion for the control of a humanoid robot. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. Volume 2., IEEE (2002) 1390–1397
3. Dariush, B., Gienger, M., Arumbakkam, A., Zhu, Y., Jian, B., Fujimura, K., Goerick, C.: Online transfer of human motion to humanoids. International Journal of Humanoid Robotics **6**(02) (2009) 265–289
4. Michieletto, S., Chessa, N., Menegatti, E.: Learning how to approach industrial robot tasks from natural demonstrations. In: Proceedings of IEEE Workshop on Advanced Robotics and its Social Impacts ARSO2013, IEEE (2013)
5. Vukobratović, M., Stepanenko, J.: On the stability of anthropomorphic systems. Mathematical Biosciences **15**(1) (1972) 1–37
6. Michieletto, S., Rizzi, A., Menegatti, E.: Robot learning by observing humans activities and modeling failures. In: IROS workshops: Cognitive Robotics Systems (CRS2013), IEEE (Nov 2013)

Author Index