IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling (IPS 2022), RCRA Incontri E Confronti Workshop (RiCeRcA 2022), and Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (SPIRIT 2022)

Riccardo De Benedictis, Nicola Gatti, Marco Maratea, Andrea Micheli,
Aniello Murano, Enrico Scala, Luciano Serafini, Ivan Serina, Alessandro Umbrico
and Mauro Vallati

Udine, November 28 – December 2, 2022.

IPS 2022 (10th Italian workshop on Planning and Scheduling), RiCeRcA 2022 (RCRA Incontri E Confronti), and SPIRIT 2022 (Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy) Workshops, Udine – Italy.

Editors' address:
Riccardo De Benedictis
Institute for Cognitive Science and Technologies (ISTC),
National Research Council (CNR), Italy.
riccardo.debenedictis@istc.cnr.it

Nicola Gatti
Dipartimento di Elettronica. Informazione e Bioingegneria,
Politecnico di Milano, Italy.
nicola.gatti@polimi.it

Marco Maratea
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi,
Università degli Studi di Genova, Italy.
marco@dibris.unige.it &
Dipartimento di Matematica ed Informatica,
Università della Calabria.
maratea@mat.unical.it

Andrea Micheli
Embedded Systems (ES) Department,
Bruno Kessler Foundation, Trento, Italy.
amicheli@fbk.eu

Aiello Murano
Dipartimento di Ingegneria elettrica e delle Tecnologie dell'Informazione,
Università degli Studi di Napoli Federico II, Italy.
aniello.murano@unina.it

Enrico Scala
Department of Information Engineering,
University of Brescia, Italy.
enrico.scala@unibs.it

Luciano Serafini
Fondazione Bruno Kessler, Trento, Italy.
serafini@fbk.eu

Ivan Serina
Department of Information Engineering,
University of Brescia, Italy.
ivan.serina@unibs.it

Alessandro Umbrico
Institute for Cognitive Science and Technologies (ISTC),
National Research Council (CNR), Italy.
alessandro.umbrico@istc.cnr.it

Mauro Vallati
University of Huddersfield, UK.
m.vallati@hud.ac.uk

IPS 2022 Programme Chairs

Riccardo De Benedictis	National Research Council (CNR), Italy.
Andrea Micheli	Bruno Kessler Foundation, Italy.
Enrico Scala	University of Brescia, Italy.
Ivan Serina	University of Brescia, Italy.
Alessandro Umbrico	National Research Council (CNR), Italy.

IPS 2022 Programme Committee

Marco Baioletti,	University of Perugia, Italy.
Gabriella Cortellessa,	ISTC-CNR, Rome, Italy.
Giuseppe Della Penna,	University of L'Aquila, Italy.
Simone Fratini,	ESA, Germany.
Hector Geffner,	Universitat Pompeu Fabra, Barcelona, Spain.
Enrico Giunchiglia,	University of Genova, Italy.
Lee McCluskey,	University of Huddersfield, UK.
Fabio Mercorio,	University of Milan Bicocca, Italy.
Roberto Micalizio,	University of Torino, Italy.
Angelo Oddi,	ISTC-CNR, Rome, Italy.
Andrea Orlandini,	ISTC-CNR, Rome, Italy.
Francesco Percassi,	University of Huddersfield, UK.
Nicola Policella,	ESA, Germany.
Riccardo Rasconi,	ISTC-CNR, Rome, Italy.
Ioannis Refanidis,	University of Macedonia, Greece.
Alessandro Saetti,	University of Brescia, Italy.
Gabriele Sartor,	University of Torino, Italy.
Andrea Schaerf,	University of Udine, Italy.
Mauro Vallati,	University of Huddersfield, UK.
Kristen Brent Venable,	Tulane University, New Orleans, USA.

RiCeRcA 2022 Programme Chairs

Marco Maratea	Università degli Studi di Genova, Italy.
Luciano Serafini	Fondazione Bruno Kessler, Italy.
Mauro Vallati	University of Huddersfield, UK.

RiCeRcA 2022 Programme Committee

Matteo Cardellini	Politecnico di Torino, Italy.
Marco Maratea	Università degli Studi di Genova, Italy.
Marco Mochi	Università degli Studi di Genova, Italy.
Mauro Vallati	University of Huddersfield, UK.

SPIRIT 2022 Programme Chairs

Nicola Gatti	Politecnico di Milano, Italy
Aiello Murano	Università degli Studi di Napoli Federico II, Italy

SPIRIT 2022 Programme Committee

Vittorio Bilò	Università del Salento, Italy
Davide Catta	Tèlècom Paris, France
Antonio Di Stasio	La Sapienza di Roma, Italy
Diodato Ferraioli	Università di Salerno, Italy
Angelo Ferrando	Università di Genova, Italy
Nicola Gatti	Politecnico di Milano, Italy
Vadim Malvone	Tèlècom Paris, France
Munyque Mittelmann	Università di Napoli Federico II, Italy
Aniello Murano	Università di Napoli Federico II, Italy
Silvia Stranieri	Università di Napoli Federico II, Italy

Contents

IPS 2022 papers not included here and published elsewhere

RiCeRcA 2022 papers not included here and published elsewhere

Preface

This volume contains the papers presented at IPS 2022, (10th Italian Workshop on Planning and Scheduling http://ips2022.istc.cnr.it), RiCeRcA 2022 (RCRA Incontri E Confronti https://ricerca2022.wordpress.com) and SPIRIT 2022 (Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy https://gatti.faculty.polimi.it/spirit22/) Workshops, held within the XXI Conference of the Italian Association for Artificial Intelligence (AI*IA 2022), from November 28 to December 2, 2022.

The aim of the IPS series of workshop is to bring together researchers interested in different aspects of planning and scheduling, and to introduce new researchers to the community. Although the primary target of IPS workshops is the Italian community of planning and scheduling, the aim is also to attract an international gathering, fostering contributions and participation from around the world. In particular, this year, 9 papers were accepted for presentation at the workshop, involving different authors from Italy and other European countries. Moreover, this year in conjunction with SPIRIT 2022, we also had an invited talk by Prof. Giuseppe De Giacomo, a leader in several aspects of AI Planning, from theory to their applications.

The IPS accepted papers mainly focus on Hybrid Systems [1], Symbol Grounding Problem in Neuro-Symbolic Planning [2], Timeline-based Planning [3], Planning with Qualitative Constraints [4], Epistemic Planning [5], online grounding in Unknown Environments [6], on the use of the Unified Planning Framework [7], on the use of Transformers for Automated Planning [8] and Real-Time Urban Traffic Routing [9].

The scope of the RiCeRcA workshop is, instead, fostering the cross-fertilisation of ideas stemming from different areas, proposing benchmarks for new challenging problems, comparing models and algorithms from an experimental viewpoint, and, in general, comparing different approaches with respect to efficiency, problem modelling, and ease of development. In particular, this year 9 papers were accepted for presentation at the workshop, involving different authors from Italy and other European countries.

The 2022 edition of RiCeRcA accepted papers considering a wide range of combinatorial problems. In particular, the workshop included works leveraging on ASP techniques to perform urban traffic optimisation and master surgical scheduling [10, 11]; approaches for the verification of neural networks [12, 13] and on the projectivity of Markov logic networks [14]; approaches for model learning [15] and supervisions [16], and exciting discussions around divagrafie [17] and the importance of declarative AI methods in videogames [18].

The scope of the SPIRIT workshop is gathering the scientific communities on artificial intelligence, machine learning, theoretical computer science, multi-agent systems, and microeconomics to promote their integration and contamination. Over the past fifteen years, researchers in artificial intelligence, machine learning, theoretical computer science, multi-agent systems, and microeconomics have joined forces to tackle problems involving incentives and computation. Interestingly, while microeconomics provides computer science with the basic models, computer science raises crucial questions related to computation and learning that suggest the study of new models. The result is a synergic integration of all these fields. Interestingly, the final goal is the provision of rigorous, theoretically-proved methods to deal with multiple strategic players. In the last years, these topics have been central in the Artificial and Machine Learning venues.

The SPIRIT 2022 accepted papers mainly focused algorithmic game theory, coalition formation, swap equilibria, price of anarchy and stability [19], influence maximization, stochastic probing, approximation algorithms, online learning [20], formal methods for multi-agent strategic reasoning [21, 22, 23, 24], including formal methods for the automated mechanism design and synthesis of strategies [25], formal aspects of attack graphs [26], model checking and runtime verification for multi-agent systems [27], and tools for the strategic reasoning [28, 27].

Riccardo De Benedictis, Nicola Gatti, Marco Maratea, Andrea Micheli, Aniello Murano, Enrico Scala, Luciano Serafini, Ivan Serina, Alessandro Umbrico, Mauro Vallati.

Workshops Organizers

References

[1] D. Aineto, E. Onaindia, M. Ramirez, E. Scala, I. Serina, Explaining the behaviour of hybrid systems with pddl+ planning (extended abstract), in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[2] A. Barbin, F. Cerutti, A. E. Gerevini, Addressing the symbol grounding problem with constraints in neuro-symbolic planning, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[3] R. D. Benedictis, G. Beraldo, A. Cesta, G. Cortellessa, Branching and pruning for timeline-based planning, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[4] L. Bonassi, E. Scala, A. E. Gerevini, Planning with pddl3 qualitative constraints for cost-optimal solutions through compilation, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[5] A. Burigana, F. Fabiano, The epistemic planning domain definition language, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[6] L. Lamanna, L. Serafini, A. Saetti, A. Gerevini, P. Traverso, Online grounding of symbolic planning domains in unknown environments, in: G. Kern-Isberner, G. Lakemeyer, T. Meyer (Eds.), Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel. July 31 - August 5, 2022, 2022.

[7] S. Morchedi, P. Jamakatel, J. J. Kiam, Exploring the unified planning framework for a more integrated and flexible fault-tolerant flight path planning system, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[8] L. Serina, M. Chiari, A. E. Gerevini, L. Putelli, I. Serina, A preliminary study on bert applied

to automated planning, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[9] M. Vallati, L. Chrpa, Effective real-time urban traffic routing: An automated planning approach, in: 7th International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2021, Heraklion, Greece, June 16-17, 2021, IEEE, 2021, pp. 1–6. doi:`10.1109/MT-ITS49943.2021.9529329`.

[10] M. Cardellini, An ASP framework for efficient urban traffic optimization, in: Y. Lierler, J. F. Morales, C. Dodaro, V. Dahl, M. Gebser, T. Tekle (Eds.), Proceedings 38th International Conference on Logic Programming, ICLP 2022 Technical Communications / Doctoral Consortium, Haifa, Israel, 31st July 2022 - 6th August 2022, volume 364 of *EPTCS*, 2022, pp. 217–227.

[11] L. Cadermatori, G. Galatà, C. L. Monaco, M. Maratea, M. Mochi, M. Schouten, An asp-based approach to master surgical scheduling, in: R. Calegari, G. Ciatto, A. Omicini (Eds.), Proceedings of the 37th Italian Conference on Computational Logic, Bologna, Italy, June 29 - July 1, 2022, volume 3204 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 313–328.

[12] R. Eramo, T. Fanni, D. Guidotti, L. Pandolfo, L. Pulina, K. Zedda, Verification of neural networks: Challenges and perspectives in the aidoart project, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[13] D. Guidotti, Verification of neural networks for safety and security-critical domains, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[14] S. Malhotra, L. Serafini, On projectivity in markov logic networks, CoRR abs/2204.04009 (2022). URL: https://doi.org/10.48550/arXiv.2204.04009. doi:`10.48550/arXiv.2204.04009`.

[15] T. Campari, L. Lamanna, P. Traverso, L. Serafini, L. Ballan, Online learning of reusable abstract models for object goal navigation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 14850–14859.

[16] G. Apriceno, A. Passerini, L. Serafini, A neuro-symbolic approach for real-world event recognition from weak supervision, in: A. Artikis, R. Posenato, S. Tonetta (Eds.), 29th International Symposium on Temporal Representation and Reasoning, TIME 2022, November 7-9, 2022, Virtual Conference, volume 247 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 12:1–12:19.

[17] L. Pandolfo, L. Cardone, L. Cutzu, B. Seligardi, G. Simi, Building the semantic portal of italian divagrafie, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[18] D. Angilica, G. Ianni, F. A. Lisi, L. Pulina, Ai and videogames: a “drosophila” for declarative methods, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[19] V. Bilò, G. Monaco, L. Moscardelli, Hedonic games with fixed-size coalitions, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[20] V. Auletta, D. Ferraioli, C. Vinci, On augmented stochastic submodular optimization: Adap-

tivity, multi-rounds, budgeted, and robustness, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[21] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713. URL: https://doi.org/10.1145/585265.585270. doi:`10.1145/585265.585270`.

[22] F. Belardinelli, A. Lomuscio, A. Murano, S. Rubin, Verification of multi-agent systems with public actions against strategy logic, Artif. Intell. 285 (2020) 103302. URL: https://doi.org/10.1016/j.artint.2020.103302. doi:`10.1016/j.artint.2020.103302`.

[23] F. Mogavero, A. Murano, G. Perelli, M. Y. Vardi, Reasoning about strategies: On the model-checking problem, ACM Trans. Comput. Log. 15 (2014) 34:1–34:47. URL: https://doi.org/10.1145/2631917. doi:`10.1145/2631917`.

[24] R. Berthon, B. Maubert, A. Murano, S. Rubin, M. Y. Vardi, Strategy logic with imperfect information, ACM Trans. Comput. Log. 22 (2021) 5:1–5:51. URL: https://doi.org/10.1145/3427955. doi:`10.1145/3427955`.

[25] M. Mittelmann, Logics for reasoning about auctions, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[26] D. Catta, J. Leneutre, V. Malvone, Towards a formal verification of attack graphs, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[27] A. Ferrando, V. Malvone, Give me a hand: How to use model checking for multi-agent systems to help runtime verification and vice versa, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[28] A. D. Stasio, Explicit and symbolic approaches for parity games, in: Proceedings of IPS 2022, R.i.C.e.R.c.A 2022 and SPIRIT22 Workshops, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

Explaining the Behaviour of Hybrid Systems with PDDL+ Planning (Extended Abstract)

Diego Aineto[1], Eva Onaindia[2], Miquel Ramirez[3], Enrico Scala[1] and Ivan Serina[1]

[1]*Università degli Studi di Brescia*
[2]*Universitat Politècnica de València*
[3]*University of Melbourne*

Keywords
Hybrid system, hybrid automata, model checking, automated planning, explanation

1. The Hybrid System Explanation Problem

A *hybrid system* (HS) is a dynamical system that exhibits a discrete and continuous behaviour, and captures the control of continuously evolving physical activities typical in automated manufacturing, chemical engineering and robotics systems. A *hybrid trajectory* is a particular execution of the HS that interleaves discrete and continuous behaviour. We aim to solve the problem of finding a hybrid trajectory that matches a sequence of observations of an HS. This problem, which we name the HS Explanation (HSE hereinafter) problem [1], is related to the Decoding [2] and Plan Recognition problems [3] studied in purely symbolic settings and enables the extension of such theories to hybrid settings [4, 5, 6, 7]. Figure 1 illustrates the HSE problem in a motion domain in planar space where observations are circular regions parameterized by a radius $r \in \{5, 10, 20, 40\}$. The black line represents the true trajectory while colored lines are explanations that match the observations for different values of r.

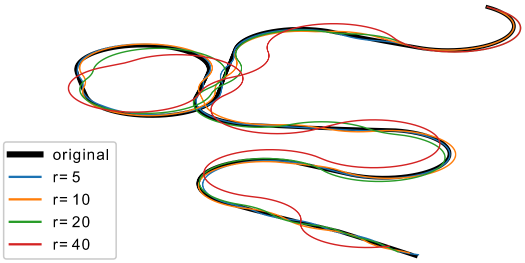


Figure 1: Original trajectory (black) and explanations (colored).

Our proposal is to leverage the formalism of *hybrid automata* (HA) [8] to formulate the

---

10th Italian Workshop on Planning and Scheduling (IPS-2022)

diego.ainetogarcia@unibs.it (D. Aineto); onaindia@upv.es (E. Onaindia); miquel.ramirez@unimelb.edu.au (M. Ramirez); enrico.scala@unibs.it (E. Scala); ivan.serina@unibs.it (I. Serina)

CEUR Workshop Proceedings (CEUR-WS.org)

	Thermostat				Platoon			Flight		
Size	HC	dR	HSE_P	HSE_P$_e$	dR	HSE_P	HSE_P$_e$	dR	HSE_P	HSE_P$_e$
2	10 (0.1)	10 (0.2)	10 (0.5)	10 (0.5)	0	10 (1.1)	10 (0.6)	10 (0.3)	10 (0.6)	10 (0.5)
4	10 (0.5)	10 (0.5)	10 (0.6)	10 (0.6)	0	10 (1.0)	10 (0.6)	1 (14)	10 (2.8)	10 (0.7)
6	10 (0.7)	8 (44)	10 (0.6)	10 (0.6)	0	10 (0.7)	10 (0.6)	0	10 (1.3)	10 (0.8)
8	10 (1.3)	3 (15)	10 (0.6)	10 (0.6)	0	10 (0.8)	10 (0.6)	0	10 (1.6)	10 (0.8)
10	10 (1.6)	0	10 (0.7)	10 (0.6)	0	10 (4.3)	10 (0.7)	0	10 (17)	10 (0.8)
20	10 (11)	0	10 (1.4)	10 (0.7)	0	10 (3.9)	10 (0.9)	0	7 (74)	10 (1.3)
30	10 (60)	0	10 (3.1)	10 (1.0)	0	8 (67)	10 (1.1)	0	1 (127)	10 (1.6)
40	10 (118)	0	10 (5.2)	10 (1.1)	0	1 (191)	10 (1.4)	0	0	10 (1.9)
50	5 (261)	0	10 (8.2)	10 (1.3)	0	0	10 (1.5)	0	0	10 (2.3)
60	0	0	10 (14)	10 (1.6)	0	0	10 (1.9)	0	0	10 (3.5)
70	0	0	10 (20)	10 (1.8)	0	0	10 (2.2)	0	0	10 (4.3)
80	0	0	10 (27)	10 (1.9)	0	0	10 (2.4)	0	0	10 (5.8)
90	0	0	10 (38)	10 (2.2)	0	0	10 (2.8)	0	0	10 (9.4)
100	0	0	10 (49)	10 (2.5)	0	0	10 (3.1)	0	0	10 (14)

Table 1
Coverage and run-time results. Each entry reports the number of solved problems, and the median time in seconds for each system. dR stands for dReach, HC for HyComp.

problem and to use the *planning* technology to solve it. We formulate the HSE problem through two HA: one automaton models the behaviour of the HS, and the other one tracks the run of the HS and checks when the produced trajectory matches the observations. The composition of both HA effectively restricts the trajectories of the HS to those that are consistent with the observations, that is, to the *language of explanations.* We then formalize the HSE problem as an optimization problem that looks over the language of explanations to find the most suitable one. In principle, this problem can be solved as a model checking (MC) problem but MC tools struggle to find concrete trajectories, assume limited knowledge of the systems dynamics, or only handle simple dynamics. To overcome this limitation, we further provide a formal mapping from HA to PDDL+ [9], which enables the use of off-the-shelf automated planners and powerful AI Planning heuristics that have in the past proven very effective for finding trajectories [10].

2. Empirical Analysis

We evaluate our approach on three domains with different types of dynamics: Thermostat (piece-wise constant), Platoon (linear) and Flight (nonlinear) and with problem sizes ranging from 2 to 100 observations. As baselines we use dReach [11] and HyComp[1] [12], two MC tools that are able to generate counter-examples (explanations in our setting). We used the ENHSP planner [13, 14, 15] for the 2 configurations of our planning approach: a basic translation (HSE_P) and an optimisation (HSE_P$_e$) supported by [1, Theorem 3.7]. Table 1 reports the number of problems solved (out of 10) and the median time over successful runs (shown in parentheses) using a time budget of 300s per problem. We observe that HyComp and dReach scale up poorly and were only able to solve the smaller instances. HSE_P shows better performance than the MC tools, but is still unable to solve the full suite of problems as computation times quickly ramp up for the linear and nonlinear domains. HSE_P$_e$, however, solves all the problems while keeping run-time low, hinting that it would be able to scale up to larger instances.

---

[1] No results for Platoon and Flight are shown for HyComp as it only supports the piece-wise constant dynamics.

References

[1] D. Aineto, E. Onaindia, M. Ramirez, E. Scala, I. Serina, Explaining the behaviour of hybrid systems with pddl+ planning, in: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, 2022, pp. 4567–4573.

[2] D. Aineto, S. Jimenez, E. Onaindia, Observation decoding with sensor models: Recognition tasks via classical planning, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 30, 2020, pp. 11–19.

[3] M. Ramírez, H. Geffner, Plan Recognition as Planning, in: International Joint conference on Artifical Intelligence, (IJCAI-09), AAAI Press, 2009, pp. 1778–1783.

[4] D. Aineto, S. Jiménez, E. Onaindia, Generalized temporal inference via planning, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, volume 18, 2021, pp. 22–31.

[5] D. Aineto, S. Jiménez, E. Onaindia, M. Ramírez, Model Recognition as Planning, in: International Conference on Automated Planning and Scheduling, (ICAPS-19), 2019, pp. 13–21.

[6] D. Aineto, S. Jiménez, E. Onaindia, Learning action models with minimal observability, Artificial Intelligence Journal 275 (2019) 104–137.

[7] D. Aineto, S. Jiménez, E. Onaindia, Learning STRIPS action models with classical planning, in: International Conference on Automated Planning and Scheduling, (ICAPS-18), 2018, pp. 399–407.

[8] T. A. Henzinger, The Theory of Hybrid Automata, in: Verification of Digital and Hybrid Systems, Springer Berlin Heidelberg, 2000, pp. 265–292.

[9] M. Fox, D. Long, Modelling mixed discrete-continuous domains for planning, J. Artif. Intell. Res. 27 (2006) 235–297.

[10] M. Wehrle, M. Helmert, The Causal Graph Revisited for Directed Model Checking, in: SAS, 2009.

[11] S. Kong, S. Gao, W. Chen, E. M. Clarke, dReach: δ-Reachability Analysis for Hybrid Systems, in: TACAS, 2015.

[12] A. Cimatti, A. Griggio, S. Mover, S. Tonetta, HyComp: An SMT-Based Model Checker for Hybrid Systems, in: TACAS, 2015.

[13] E. Scala, P. Haslum, S. Thiébaux, M. Ramirez, Subgoaling techniques for satisficing and optimal numeric planning, Journal of Artificial Intelligence Research 68 (2020) 691–752.

[14] E. Scala, A. Saetti, I. Serina, A. E. Gerevini, Search-guidance mechanisms for numeric planning through subgoaling relaxation, in: ICAPS, 2020.

[15] M. Ramírez, E. Scala, P. Haslum, S. Thiébaux, Numerical integration and dynamic discretization in heuristic search planning over hybrid domains, CoRR abs/1703.04232 (2017).

Addressing the Symbol Grounding Problem with Constraints in Neuro-Symbolic Planning

Aymeric Barbin[1,2,*], Federico Cerutti[2,3] and Alfonso Emilio Gerevini[2]

[1]*Sapienza Università di Roma, Italy*
[2]*Università degli Studi di Brescia, Italy*
[3]*Cardiff University, UK*

Abstract
In this paper, we address the Symbol Grounding Problem (SGP) in the context of neuro-symbolic planning, where the categorical vectors learned to represent high dimensional inputs suffer from instability, which poses a problem of efficiency during the planning phase. One way to alleviate the SGP is to enforce constraints — among the latent variables — by expressing them in the loss function during the learning process. Combining an existing tool for invariant search and ideas from Logic Tensor Networks (fuzzy logic), we propose to automatize the process of finding and enforcing relevant constraints. We apply our idea to LatPlan, a domain independent, image-based classical planner.

Keywords
Neuro-Symbolic Planning, Symbol Grounding Problem, Action Model Learning

1. Introduction

The core interest of Domain-Independent Planning [1] is developing general-purpose algorithms and systems that can solve planning problems independently of any specific knowledge of the latter. A planning problem can be specified using the Planning Domain Description Language (PDDL) [2] in terms of a symbolic description of the states and actions composing the domain, and initial state, and a goal.

When the states of the problem are only available as sub-symbolic data (e.g., images), generating a PDDL description necessarily needs to address the Symbol Grounding Problem (SGP) [3] which refers to the case where two different inputs (e.g., images) grounding the same symbol (e.g., digit "1") have different vector representations, revealing a lack of generalisation power.

LatPlan [4] is a neuro-symbolic architecture for planning proposed to address the bottleneck of PDDL construction from raw input data. It leverages Deep Learning to learn the PDDL description from a set of unlabeled pairs of transition images. More specifically, it uses variational autoencoders (VAE) [5] to learn and generate categorical vector representations of the images (states) and of the transitions between them (actions), which are then used to generate the

---

Italian Workshop on Planning and Scheduling (IPS-2022, 10th edition)
[*] Corresponding author.
aymeric.barbin@uniroma1.it (A. Barbin); federico.cerutti@unibs.it (F. Cerutti); alfonso.gerevini@unibs.it (A. E. Gerevini)

PDDL. LatPlan is also endowed with tools that alleviate the SGP of these representations, which can be assessed by measuring their variance on perturbed input.

Another way to address the SGP is to enforce constraints on the categorical vectors. If we consider a vector v as an ideally grounded symbol and its unstable version v', then v' contains some noise, i.e. variables with undesired values. If we know appropriate constraints to apply during training, the noisy representation corresponding to v' should converge to v.

In this paper, we propose to use an automatic tool to look for invariants in the PDDL generated by LatPlan and to test their incidence on the learning by expressing them as an auxiliary loss term. To do so, we borrow ideas from Logic Tensor Networks (LTN) [6], where this additional loss is created using the *t-norm fuzzy logic* [7]. In Section 2, we provide background on LatPlan, invariants, and fuzzy logic, then, in Section 3, we discuss our proposed solution to integrate the search for constraints into the training loop.

2. Background

2.1. LatPlan

LatPlan is a neuro-symbolic architecture that receives pairs of images representing transitions and learns categorical vector representations from them. Once trained, it can generate a PDDL representation of the states and actions of the problem, which can be input to a planner.

For our work, we are interested in improving the learning of two components of LatPlan, the State AutoEncoder network (SAE), which learns to represent images as categorical vectors, and the Action Model Acquisition network (AMA), which learns to represent actions as categorical vectors representing preconditions and effects. These two models are learned end-to-end from a dataset of pairs of images, each pair representing a valid transition occurring in the sub-symbolic world (e.g. the switching of a tile in the case of 8-puzzle).

The SAE learns a bi-directionnal mapping between sub-symbolic raw data x and propositional states $z \in \{0, 1\}^F$ (with F being the number of variables in the categorical vector). Concretely, it consists of the encoder and the decoder of a VAE that learns $\mathsf{DECODE}(\mathsf{ENCODE}(x)) = x$.

The AMA model consists of three networks: ACTION, APPLY and REGRESS. ACTION learns to associate two consecutive states (returned by the SAE) to an action (expressed as a one-hot vector). APPLY learns to predict the next categorical state from the previous one and an action. REGRESS is symmetric of APPLY and learns to predict backwards the previous state from the next one and from the action.

The SAE, APPLY and REGRESS networks all output binary categorical vectors of the same size as the SAE's output. Each element of these vectors is interpreted as a binary variable and is represented by a unary predicate in the PDDL files generated by LatPlan. This representation is negatively affected by the SGP, as it can break the identity assumption inherent to symbolic reasoning algorithms—in which a state must not change—and can cause disconnections during the search process, i.e. if two states exist that should represent the same state, an action might lead to one of them but not to the other, the latter would then be a dead end.

To address this issue, the authors of LatPlan propose two solutions. First, at test time, they replace the sampling of the categorical vector with an *argmax* layer, therefore removing stochasticity. Second, during training, they select a version of the prior distribution for the sampling (of the categorical vector) that favours sparsity of truthiness among the binary variables. This leads to more stable latent state vectors, and showed improved performances of the model in terms of next-state prediction accuracy (APPLY network) and planning performances. In our work, we are mainly interested in stabilizing the latent states vectors, generated by the SAE network; to assess it, we use the same metric as in LatPlan paper, i.e. the State Variance. To compute it, we use the state variance for noisy input, i.e., the variance of the latent vectors :

$$z^{i,0} = ENCODE(x^{i,0} + n)$$

where:

- $n \sim N(\mu = 0, \sigma = 0.3)$
- $x^{i,0}$: 1^{st} image of the i^{th} transition pair of the dataset

The State Variance is computed by iterating over 10 random vectors, then averaged over F bits in the latent space and over the dataset indexed by *i*. Formally:

$$\mathbb{E}_{f \in 0 \ldots F}\, \mathbb{E}_i \; Var_{j \in 0..10}[\mathsf{ENCODE}(x^{i,0}, n^j)_f] \tag{1}$$

2.2. Invariants and constraints in planning

In classical planning, an invariant is defined as a logical formula over variables of the domain which is true in any reachable state [8]. Invariants can be seen as hidden but logical properties of the domain, and can also be termed as *state constraints*. In LatPlan, we can enforce them in learning the SAE and AMA networks.

An important part of the research in classical planning focuses on discovering invariants in planning domains. Today, automatic tools [9] [10] already exist that can find a variety of invariants, such as predicate domain invariants (i.e., in the effect of an action), static invariants on predicates (ones that are unaffected by any operator), simple implicative invariants (e.g., if $z_1 \rightarrow z_2$), mutually exclusive invariant (e.g., $\neg z_1 \vee \neg z_2$), etc.

2.3. Logic Tensor Networks and Fuzzy logic

Extensive work on integrating logical constraints *during the training* of neural networks have been conducted in the last few years. Notably, one can express constraints among neural networks outputs taken as predicates thanks to *t-norm* operations and the fuzzy generalisation of First Order Logic (FOL) [11] [7].

For example, if we want to express the truth value of $\neg z_1 \vee \neg z_2$, we can compute it by its average over the dataset, i.e.,

$$\frac{1}{|Z|} \sum_{z_1, z_2 \in Z} (1 - P(z_1)) + (1 - P(z_2)) \tag{2}$$

where $P(z_1)$ is the (continuous) value of truth of one grounding of “z_1 *is true*”, and Z is the set of all the groundings. The inverse of this value can be directly appended as a penalty to the loss function, which eventually forces the network to re-adapt its weights to this new constraint.

3. Automatically finding and enforcing relevant invariant candidates

We first discuss how we intend to search for invariants of interest among LatPlan binary variables with the help of an automatic tool; then, we discuss how to enforce these invariants during training and we give details about integrating the search in the training loop.

3.1. Searching for invariants of interest

Automatic tools, like Fast Downward (FD) [10] and DISCOPLAN [9], can find invariants in the PDDL representation outputted by LatPlan using invariant synthesis . But, since LatPlan learns the PDDL by statistical inference, there is no guarantee that any of the invariants found in the PDDL maps to the ground truth invariants, i.e., to invariants of the (unknown) ground truth PDDL domain. For example if $z_1 \rightarrow z_2$ is returned by FD as an invariant, but z_1 is true only once in the whole dataset, it's possible that $z_1 \rightarrow z_2$ would be false if we had more data with z_1 being true. Thus, in our work we consider the invariants computed by an invariants generation tool (on LatPlan PDDL) as a set of *probable invariants* that we intend to test.

3.2. Applying the constraints

In Latplan, internal categorical representations are continuous vectors from which elements converge to binary values during training. At each batch, we can compute the truth value— thanks to fuzzy logic— of any constraint, expressed as a logical forumula, over these binary variables, for instance "z_1 *is true*." Then, by taking the inverse of this value and multiplying it by a normalizing factor, we obtain an additional loss that we can append to the total loss. This way, the network will adapt its weights through backpropagation to comply with the constraint.

3.3. Augmenting the training loop with a search over invariants

Our idea is to integrate an automatic invariant finder in the learning process of LatPlan. More precisely, we want to perform a search— similar to a hyperparameter search— on the invariants. Further details can be find in Appendix A. The implementation of this loop builds upon the t-norm like functions that already exist in Tensorflow [12] and the possibility to customize the training loop in Keras [13], LatPlan being coded with both of them.

4. Conclusions and Future Work

We discussed the idea of combining an automatic invariants finder with fuzzy logic to progress in solving the SGP that affects the latent representations of a neuro-symbolic architecture (LatPlan). More precisely, the invariants found by the automatic tool are *probable* invariants - since the PDDL they are issued from is learned by statistical inference - and we want to enforce them during training. If they bring a lower state variance, it is probable that they correspond to *ground true invariants.* The goal is to automatically find invariants that are responsible for more state stability. We are now implementing the search loop introduced in Section 3.

References

[1] D. E. Wilkins, Domain-independent planning representation and plan generation, Artificial Intelligence 22 (1984) 269–301.

[2] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, An introduction to the planning domain definition language, Synthesis Lectures on Artificial Intelligence and Machine Learning 13 (2019) 1–187.

[3] M. Taddeo, L. Floridi, Solving the symbol grounding problem: a critical review of fifteen years of research, Journal of Experimental & Theoretical Artificial Intelligence 17 (2005) 419–445.

[4] M. Asai, A. Fukunaga, Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary, in: Proceedings of the aaai conference on artificial intelligence, volume 32, 2018.

[5] D. P. Kingma, M. Welling, et al., An introduction to variational autoencoders, Foundations and Trends® in Machine Learning 12 (2019) 307–392.

[6] S. Badreddine, A. d. Garcez, L. Serafini, M. Spranger, Logic tensor networks, Artificial Intelligence 303 (2022) 103649.

[7] L. A. Zadeh, Fuzzy logic, Computer 21 (1988) 83–93.

[8] V. Alcázar, A. Torralba, A reminder about the importance of computing and exploiting invariants in planning, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 25, 2015, pp. 2–6.

[9] A. Gerevini, L. K. Schubert, Discovering state constraints in discoplan: Some new results, in: AAAI/IAAI, 2000, pp. 761–767.

[10] M. Helmert, Concise finite-domain representations for pddl planning tasks, Artificial Intelligence 173 (2009) 503–535.

[11] P. Hájek, Metamathematics of fuzzy logic, volume 4, Springer Science & Business Media, 2013.

[12] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., {TensorFlow}: a system for {Large-Scale} machine learning, in: 12th USENIX symposium on operating systems design and implementation (OSDI 16), 2016, pp. 265–283.

[13] F. Chollet, et al., Keras, https://keras.io, 2015.

A. The proposed augmented training loop

This augmented training loop, GreedyConstraintTraining, is illustrated in Algorithm 1, which uses the following auxiliary functions:

- LatPlan, which is a function that embeds the training process of LatPlan: it receives as input a training set, a set of propositional formulae to embed in the loss function, and outputs the learned functions and action descriptions as binary vectors;
- Vecs2Pddl transforms the action descriptions learned by LatPlan into PDDL format;
- Pddl2Inv is a function that returns a set of invariants of a planning problem in the PDDL format provided in the input. An example is the invariant finder of Fast Downward;

- MetricEval receives as input the learned functions and action descriptions of LatPlan and outputs a score value: the higher the score value, the better. In our case, it is the inverse of the State Variance given in Formula 1 (in this case only the SAE function of LatPlan is needed).
- PickAnInvariant is a heuristic that identifies the most promising invariant in a set; for instance an invariant with the less number of variables (because it has a higher probability to be a ground true invariant compared to one with more variables).

Algorithm 1 Our proposed GreedyConstraintTraining approach, which receives as input a training set $tSet$, and returns Ω, the invariants which improve on a chosen metric

```
$\Omega \leftarrow \emptyset$
$LPModel \leftarrow$ LatPlan$(tSet, \emptyset)$
$\Phi \leftarrow$ Pddl2Inv(Vecs2Pddl$(LPModel)$)
$score \leftarrow$ MetricEval$(LPModel)$
while $\Phi \neq \emptyset$ do
    $\phi \leftarrow$ PickAnInvariant$(\Phi)$
    $\Phi \leftarrow \Phi \setminus \{\phi\}$
    $LPModel' \leftarrow$ LatPlan$(tSet, \Omega \cup \{\phi\})$
    if MetricEval$(LPModel') > score$ then
        $\Omega \leftarrow \Omega \cup \{\phi\}$
        $\Phi \leftarrow \Phi \cup$ Pddl2Inv(Vecs2Pddl$(LPModel')$)
        $score \leftarrow$ MetricEval$(LPModel')$
    end if
end while
return $\Omega$
```

Branching and Pruning for Timeline-based Planning

Riccardo De Benedictis, Gloria Beraldo, Amedeo Cesta and Gabriella Cortellessa

Institute of Cognitive Sciences and Technologies (ISTC) - Via S. Martino della Battaglia 44, 00185 Roma (Italy) - National Research Council of Italy (CNR)

Abstract

One of the features that allowed classical planners to efficiently solve large problems is the ability their heuristics to prune large portions of the search space. These heuristics, however, by addressing classical approaches, do little to support the resolution of problems in which the temporal components are relevant and, even more so, they are not directly suitable for timeline-based approaches to automated planning. This paper takes advantage of some pruning techniques to increase the efficiency of timeline-based planning problems resolution. The elimination of some choices estimated as not very advantageous, in particular, allows the use of inference techniques to reduce the number of decisions to be made, reducing the risk of running into dead ends and, consequently, increasing the resolution efficiency of the solvers.

Keywords

Automated Planning, Timeline-based Planning, Heuristic search, Scheduling

1. Introduction

The introduction of domain independent *heuristics* within the Automated Planning [1] community immediately allowed solvers to efficiently solve large instances of complex problems. The different approaches that make up a solver's paraphernalia, range from the seminal h_{add} and h_{max} [2] to the more recent developments relying on *delete-relaxation*, like the h^{FF} heuristic [3] and the *causal graph* heuristics [4], on *landmarks*, like in [5, 6], on the *critical path*, like the h^m heuristic [7, 8] or, lastly, on *abstraction*, like in [9] or in [10, 11]. The salient aspects of such heuristics can be divided into the ability to direct the search process towards promising areas of the search space (*branching*) as well as the ability to avoid dead-ends that would require potentially expensive backtracking operations (*pruning*)[12].

While the above heuristics are significantly heterogeneous among them (although, often, they share some commonalities), they have in common the fact that they have been developed specifically for the resolution of a particular type of problem, characterized by a specific modeling language called PDDL [13], representing a natural evolution of the most long-lived STRIPS [14] formalism. Despite the PDDL, over the years, has been extended through different directions by introducing *durative-actions* and *numeric fluents* [15], *derived predicates* and *timed initial literals* [16], *continuous changes* [17], *state-trajectory constraints* and *preferences* [18] and *object-fluents*[1],

---

IPS-2022: 10th Italian Workshop on Planning and Scheduling, November 29, 2022, Udine, Italy

riccardo.debenedictis@istc.cnr.it (R. De Benedictis); gloria.beraldo@istc.cnr.it (G. Beraldo); amedeo.cesta@istc.cnr.it (A. Cesta); gabriella.cortellessa@istc.cnr.it (G. Cortellessa)
0000-0003-2344-4088 (R. De Benedictis); 0000-0001-8937-9739 (G. Beraldo); 0000-0002-0703-9122 (A. Cesta); 0000-0002-9835-1575 (G. Cortellessa)

[1] http://www.plg.inf.uc3m.es/ipc2011-deterministic/attachments/Resources/kovacs-pddl-3.1-2011.pdf

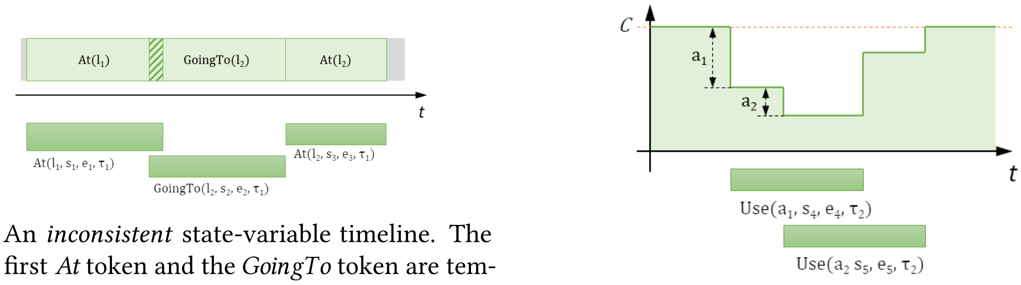
the development of heuristics for reasoning with these more expressive formal systems has remained relatively limited to a few cases (e.g., [19, 20]).

Timeline-based planning [21, 22] is an approach to automated planning which, by relying on partial-order planning [23], allows to generate plans which, during their execution, are, compared to the total order plans produced by classical planners, more easily adaptable. Analogously to the solvers reasoning upon the previous mentioned PDDL extensions, timeline-based planners have to cope with the high expressiveness of the formalisms which, despite making them particularly suited at addressing real-world applications, unavoidably leads to performance issues. In a recent work about timeline-based planning it has been shown that, thanks to the introduction of some domain-independent heuristics, the computation time could be effectively reduced [24]. The presented heuristics aim at directing the search process towards the most promising areas of the search space. As it will be shown in more detail in the following sections, however, the same data structures can also be used to partially avoid dead-ends and, consequently, potentially expensive backtracking operations.

2. Timeline-based planning

Timeline-based planning constitutes a form of deliberative reasoning which, in an integrated way, allows to carry out different forms of semantic and causal reasoning. Although this approach to planning has mostly been relegated to forms of causal reasoning in the space domain, many solvers have been proposed over the time like, for example, IxTeT [25], Europa [26], Aspen [27], the Trf [28, 29] on which the APSI framework [30] relies and, more recently, PLATINUm [31]. Some theoretical works on timeline-based planning like [32, 26] were mostly dedicated to identifying connections with classical planning a-la PDDL [15]. The work on IxTeT and Trf has tried to clarify some keys underlying principles but mostly succeeded in underscoring the role of time and resource reasoning [33, 34]. The planner CHIMP [35] follows a Meta-CSP approach having meta-Constraints which heavily resembles timelines. The Flexible Acting and Planning Environment (FAPE) [36, 37] tightly integrates structures similar to timelines with acting. The Action Notation Modeling Language (ANML) [38] is an interesting development which combines the Hierarchical Task Network (HTN) [39, 40, 41] decomposition methods with the expressiveness of the timeline representation. Finally, it is worth mentioning that the timeline-based approaches have been often associated to resource managing capabilities. By leveraging on constraint-based approaches, most of the above approaches like IxTeT [42, 34], [43], [44] or [45] integrate planning and scheduling capabilities. Finally, [46] proposes a recent formalization of timeline-based planning.

Given the mentioned link with the heuristics we will refer, in this paper, to the timeline-based planning formalization as defined in [24]. According to this formalization, specifically, the basic building block of timeline-based planning is the *token* which, intuitively, is used to represent the single unit of information. Through their introduction and their constraining during the planning process, in particular, tokens allow to represent the different components of the high-level plans. In its most general form, a token is formally described by an expression like $n(x_0, \dots, x_i)_\chi$. In particular, n is a *predicate* symbol, $x_0, \dots, x_i$ are its *parameters* (i.e., constants, numeric variables or object variables) and $\chi \in \{f, g\}$ is a constant representing the class of the


$At(l_1)$
$GoingTo(l_2)$
$At(l_2)$
t
$At(l_1, s_1, e_1, \tau_1)$
$At(l_2, s_3, e_3, \tau_1)$
$GoingTo(l_2, s_2, e_2, \tau_1)$
C
a_1
a_2
t
$Use(a_1, s_4, e_4, \tau_2)$
$Use(a_2\ s_5, e_5, \tau_2)$

(a) An *inconsistent* state-variable timeline. The first *At* token and the *GoingTo* token are temporally overlapping. The inconsistency can be removed, for example, by introducing a $e_1 \leq s_2$ constraint.

(b) A *consistent* reusable-resource timeline. The overlap of tokens is allowed as long as the simultaneous use of the resource is less than its capacity.

Figure 1: Different timelines extracted by their associated tokens.

token (i.e., either a *fact* or a *goal*).

The token's parameters are constituted, in general, by the variables of a *constraint network* $\mathcal{N}$ (refer to [47] for further details) and can be used, among other things, to represent temporal information such as the start or the end of some tasks. The semantics of the χ constant, on the contrary, is borrowed from Constraint Logic Programming (CLP) [48]. Specifically, while the facts are considered inherently true, the goals must be achieved as defined by a set of *rules*. Rules, in particular, are expressions of the form $n(x_0, \ldots, x_k) \leftarrow \mathbf{r}$ where $n(x_0, \ldots, x_k)$ is the *head* of the rule and $\mathbf{r}$ is the *body* of the rule. In particular, $\mathbf{r}$ represents the *requirement* for achieving any goal having the "form" of the head of the rule. Such requirements can be either a token, a *constraint* among tokens (possibly including the $x_0, \ldots, x_k$ variables), a *conjunction* of requirements or a *disjunction* of requirements. It is worth noting the recursive definition of requirement, which allows the definition of the body of a rule as any logical combination of tokens and constraints.

Similarly to CLP, through the application of the rules it is hence possible to establish and generate relationships among tokens. Compared to CLP, however, timelines introduce an added value: some tokens may be equipped with a special object variable τ that identifies the *timeline* affected by the token. Different tokens with the same value for the τ parameter, in particular, affect the same timeline and, depending on the nature of the timeline, might interact with each other. There can, indeed, be different types of timelines. In case of *state-variable* timelines (see Figure 1a), for example, different tokens on the same state-variable cannot temporally overlap. In case of *reusable-resource* timelines (see Figure 1b), on the contrary, tokens represent resource usages and can, hence, overlap as long as the concurrent uses remain below the resource's capacity.

Given the ingredients mentioned above we can now formally introduce the addressed planning problem. A *timeline-based planning problem*, specifically, is a triple $\mathcal{P} = (\mathbf{O}, \mathcal{R}, \mathbf{r})$, where $\mathbf{O}$ is a set of typed objects, needed for instantiating the initial domains of the constraint network variables and, consequently, the tokens' parameters, $\mathcal{R}$ is a set of rules and $\mathbf{r}$ is a requirement. Intuitively, a solution to such a problem should be described by a set of tokens whose parameters assume values so as to guarantee the satisfaction of all the constraints imposed by the problem's requirement, by the application of the rules, as well as by the cumulative constraints imposed by the timelines. Unfortunately, the previous definition, although intuitive, is not easily translatable into a reasoning process which guarantees its achievement starting from the definition of the planning problem. For this reason, just like common partial-order planners, timeline-based planners often rely on the concepts of *flaw* and *resolver*. The planner, in particular, internally maintains a data structure, called *token network*, which represents a partial plan $\pi = (\mathscr{T}, \mathscr{N})$, where $\mathscr{T}$ is a set of tokens whose parameters are constrained by the constraint network $\mathscr{N}$. During the resolution process, the reasoner incrementally refines the current token network π by identifying its flaws and by solving them through the application of resolvers, while maintaining consistent the constraints of $\mathscr{N}$.

There can be, in general, different types of flaws, each resolvable by applying the corresponding resolvers. The achievement of a goal, for example, can take place either through the application of a rule or through a *unification* with either a fact or another already achieved goal with the same predicate (i.e., the parameters of the current goal and the token with which is unifying are constrained to be pairwise equal). In case of disjunctions, introduced either in the initial problem or by the application of a rule, a disjunct must be chosen. The domain of all the variables that make up the token parameters must be reduced to a single allowed value. Finally, timelines must be consistent, possibly requiring the introduction of constraints which prevent not allowed overlaps. Thanks to the introduction of the flaw and resolver concepts, it is therefore possible to provide an implementable definition of solution. Specifically, a *solution* to a timeline-based planning problem is a flawless token network whose constraint network is consistent.

2.1. A Lifted Heuristic for Timeline-based Planning

Finding a solution to a timeline-based planning problem is far from simple. Choosing the *right* flaw and the *right* resolver, in particular, constitutes a crucial aspect for coping with the computational complexity and hence efficiently generating solutions. Taking a cue from classical planning heuristics, [24] describes how, by building a *causal graph* and by analyzing its topology, it is possible to estimate the costs for the resolution of the flaws and for the application of the resolvers. Flaws and resolvers, in particular, are seen as if they are, respectively, classical planning propositions and actions. Similarly to a proposition added by the positive effect of an action, in particular, the effect of applying a resolver is the resolution of a flaw. Additionally, just like the preconditions in a classical action, further flaws can be introduced in the case of the application of a rule or the choice of a disjunct in a disjunction. Starting from the initial facts, with a zero estimated resolution cost, the cost of applying a resolver can be estimated as an intrinsic cost of the resolver plus the maximum cost (h^{max} heuristic). The cost of resolving a flaw, on the other hand, is given by the minimum cost of its resolvers. Starting from the top-level goals present in the planning problem, initially estimated with infinite cost, a graph is constructed by proceeding backwards, considering all the possible resolvers for all the possible flaws. The estimated costs are updated every time a unification is found or in those cases in which the resolver does not introduce further flaws. Finally, the graph building procedure proceeds until a finite estimate cost for the top-level goals is reached.

Compared to other state-of-the-art timeline-based solvers, the above heuristics allow solving problems up to one order of magnitude faster [24]. The most interesting aspect for the current topic, however, concerns the management of the causal constraints in the causal graph. Similar to planning models based on satisfability [49], indeed, a set of propositional variables is assigned to flaws and to resolvers. For the sake of brevity we will use subscripts to indicate flaws (e.g., φ_0, φ_1, etc.), resolvers (e.g., ρ_0, ρ_1, etc.) as well as their associated propositional variables. Additionally, given a flaw φ, we refer to the set of its possible resolvers by means of $res(\varphi)$ and, by means of $cause(\varphi)$, to the set of resolvers (possibly empty, in case of the flaws of the problem's requirement) which are responsible for introducing it. Moreover, given a resolver ρ, we refer to the set of its preconditions (e.g., the set of tokens introduced by the application of a rule) by means of $precs(\rho)$ and to the flaw solved through its application by means of $eff(\rho)$. Using the above notation we can estimate the cost of a generic flaw φ as $G(\varphi) = min_{\rho \in res(\varphi)} G(\rho)$ (1) and the cost of a generic resolver ρ as $G(\rho) = c(\rho) + max_{\varphi \in precs(\rho)} G(\varphi)$ (2), in which $c(\rho)$ represents the intrinsic cost of the resolver ρ.

The introduction of such variables allows to constrain them so as to guarantee the satisfaction of the causal relations. Specifically, for each flaw φ_i, we guarantee that the preconditions of all the applied resolvers are satisfied ($\varphi_i = \bigwedge_{\rho_k \in cause(\varphi_i)} \rho_k$ (3)) and that at least one resolver is active whenever the flaw becomes active ($\varphi_i \Rightarrow \bigvee_{\rho_l \in res(\varphi_i)} \rho_l$ (4)). Additionally, we need a gimmick to link the presence of the tokens with the causality constraint. A further variable $\sigma \in \{inactive, active, unified\}$, in this regard, is associated to each token. A partial solution will hence consist solely of those tokens of the token network which are *active*. Moreover, in case such tokens are goals, the bodies of the associated rules must also be present within the solution. Later on, we refer to tokens by means of the σ variables (we will use subscripts to describe specific tokens, e.g., σ_0, σ_1, etc.) and to the flaws introduced by tokens by means of the $\varphi(\sigma)$ function.

The last aspect to consider concerns the update of such variables as a consequence of the activation of a rule application resolver and of a unification resolver. Specifically, each rule application resolver ρ_a binds the σ_a variable of the goal token, whose rule has been applied, to assume the *active* value (formally, $\rho_a = [\varphi(\sigma_a) = active]$). Finally, for each unification resolver ρ_u representing the unification of a token σ_u with a target token σ_t, the constraints $\rho_u = [\sigma_u = unified]$ and $\rho_u \Rightarrow [\sigma_t = active]$ guarantee the update of the σ variables while adding $\varphi(\sigma_t)$ to the preconditions of ρ_u guarantees the operation of the heuristic.

2.2. An explanatory example

In order to better understand how the heuristics and the causality constraints work, we introduce in this section a running example of an explanatory planning problem, whose objective is to plan a physical rehabilitation session for an hypothetical user. Figure 2 shows the causal graph which is generated for the problem, whose problem requirement is constituted by the sole goal σ_0. Estimated costs for flaws (boxes) and resolvers (circles) are on their upper right. The propositional variables that participate in the causal constraints are on their upper left. Solid (True) and dashed (Unassigned) contour lines are used to distinguish flaws' and resolvers' associated propositional variables' values. In the figure, in particular, the φ_0 variable, representing a flaw which is present in the problem requirement and therefore must necessarily be solved, assumes the True value.

It is worth noting that, in the example, the φ_0 flaw, for achieving the σ_0 goal, can only be solved through the ρ_0 resolver, which is hence directly applied (notice the solid line) as a consequence of the propagation of the causal constraints. Since $res(\varphi_0) = \{\rho_0\}$, indeed, the expression (4) translates into $\varphi_0 \Rightarrow \rho_0$. This, in turn, forces the σ_0 goal to assume the *active* value as a consequence of $\rho_0 = [\varphi(\sigma_0) = active]$. The ρ_0 resolver, furthermore, represents the application of a rule having a *PhysicalExercise*() in the head and, in the body, a conjunction of the two σ_1 and σ_2 goals. The application of this resolver, in particular, introduces the $\varphi_1 = \varphi(\sigma_1)$ and the $\varphi_2 = \varphi(\sigma_2)$ flaws, each of which must necessarily be resolved as a consequence of the $\varphi_1 = \rho_0$ and $\varphi_2 = \rho_0$ causal constraints, from the expression (3). These flaws, in turn, can be solved through the application of the ρ_1 and of the ρ_2 resolvers which introduce, respectively, the disjunctions represented by the φ_3 and φ_4 flaws.

Proceeding backwards, the propagation of the causal constraints no longer allows to infer what is present in the current partial plan (notice the dashed lines). The resolution of the φ_3 and φ_4 flaws, in particular, constitute two choices that the planner must make during the resolution process. The φ_3 flaw, for example, can be solved either by applying the *Disj*$_0$ disjunct, represented by the ρ_3 resolver, or by applying the *Disj*$_1$ disjunct, represented by the ρ_4 resolver. The graph construction process, however, which proceeds following a breadth-first approach, has identified, in the example, a possible solution for the φ_3 flaw by applying first the ρ_3 resolver and then the ρ_7 resolver (the latter corresponding, in this simple example, to a rule with an empty body). The heuristics' estimated costs propagation procedure, hence, makes the ρ_3 resolver, with an estimated cost of 2, much more attractive than the ρ_4 resolver, with an estimated cost of ∞. For a similar reason, the ρ_5 resolver will be preferred over the ρ_6 resolver, leading to a (possible) solution of the planning problem.

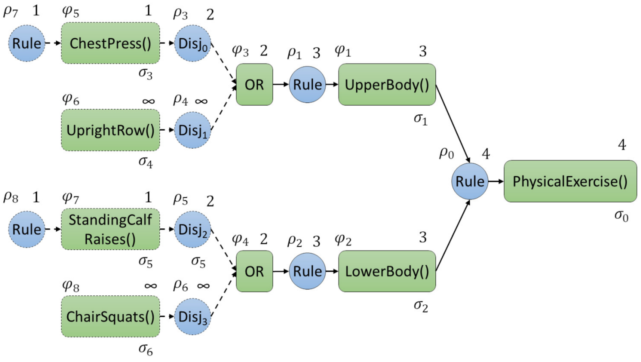


Figure 2: An example of causal graph for the planning of a physical rehabilitation session. Tokens' parameters are omitted to avoid burdening the notation.

It is worth noting that, for the sake of simplicity, the tokens' parameters are not represented in the example figure. All tokens, however, are endowed with numerical variables that represent the start and the end of the associated activities, appropriately constrained according to common sense. Upper and lower body exercises, for example, represented respectively by the σ_1 and by the σ_2 tokens, will take place as part of the more general physical exercise represented by the σ_0 token. The σ_3 and by the σ_5 tokens, additionally, are endowed with their τ variables which will avoid their temporal overlapping if they will assume the same value.

3. Pruning the causal graph

The construction of the causal graph is, inevitably, a partial task that cannot be carried out in an exhaustive manner. Consider, for example, the case of a not so smart robotic arm which, in order to move an object from one point to another, considers an infinite number of tasks in which it takes the object and puts it back where it initially was. The graph building procedure, in particular, starts by queuing the high-level flaws, defined in the planning problem, in an expansion queue. At each step, a flaw is extracted from the queue and expanded, possibly queuing further sub-flaws, proceeding backwards in a breadth-first manner until the high-level flaws do assume a finite estimated cost. It is worth noticing that, exception made for very simple cases, when the graph building procedure halts, the queue might still contain some flaws. These flaws, however, have an estimated infinite cost. Once given way to the resolution algorithm, in particular, these flaws would be avoided as much as possible by the search algorithm. In other words, for these flaws, the graph is not able to provide any indication for their resolution.

Can we, before the resolution process starts, remove them from the graph or, somehow, forbid their choice? As we will see, the answer is yes, but we can do even more. Some of the flaws in the graph expansion queue, indeed, might constitute the preconditions for the resolution of already expanded flaws which, within the causal graph, do already have a finite estimated cost. Such flaws, which we call *deferrable*, can easily be identified by traversing the causal graph following the direction of the arrows, until a flaw with a finite estimated cost is found (hence the flaw is classified as deferrable) or a sink node is reached (hence the flaw is classified as non-deferrable). Once removed from the graph's expansion queue, in particular, these flaws can immediately be re-queued for subsequent further processing.

When the graph expansion procedure halts, in particular, there will be, in the graph's expansion queue, a set of flaws, either deferrable or not expanded yet. Instead of discouraging the search algorithm from choosing such flaws, we can forcibly prevent it by imposing constraints on the corresponding causal variables (the causal variables are forced to false). It is worth noting that these flaws, having an infinite estimated cost, would not be chosen by the search anyway. The introduction of the constraints, however, causally propagates within the causal graph leading to some benefits in terms of performance.

Consider, for example, the graph presented in Figure 2. Before the expansion of the φ_5 flaw all the estimated costs are infinite. The expansion of the φ_5 flaw, however, introduces a resolver, ρ_7, that has no preconditions. The cost of such a resolver, in line with the h^{max} heuristic, can easily be estimated as the sole intrinsic cost of the resolver (Eq. 2). By applying the cost estimation formula, the cost update is propagated forward by assigning an estimated cost of 1 to the φ_5 flaw (Eq. 1), an estimated cost of 2 to the ρ_3 resolver, an estimated cost of 2 to the φ_3 flaw, an estimated cost of 3 to the ρ_1 resolver and an estimated cost of 3 to the φ_1 flaw. All the other flaws and resolvers maintain, at this point, an infinite estimated cost.

The φ_6 flaw is then removed from the graph's expansion queue and checked for deferrability. Since the φ_3 flaw has a finite estimated cost, the flow is recognized as deferrable and is re-queued into the graph expansion queue. It's now the turn of the φ_7 flaw, having a term similar to that of the φ_5 flaw. The cost update propagation, however, reaches, this time, the updating the cost of the high-level φ_0 flaw, determining the halting of the graph building procedure with the φ_6 and the φ_8 flaws in the graph expansion queue.

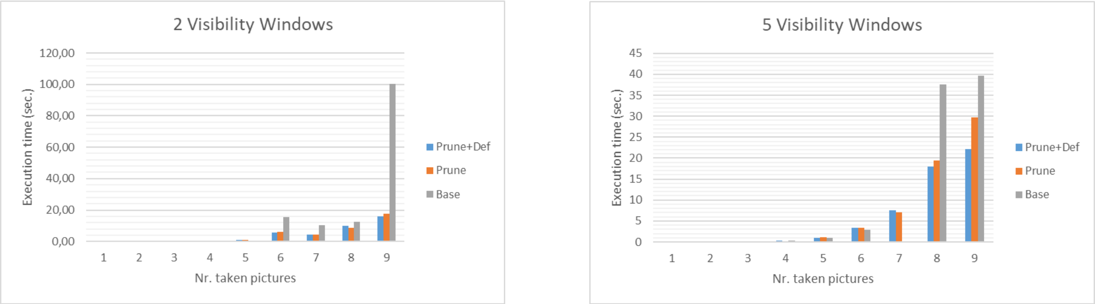


Figure 3: Different timelines extracted by their associated tokens.

The resulting graph, depicted in Figure 2, shows that the only two choices that the resolution algorithm should eventually make are the resolution of the φ_3 and of the φ_4 flaws. In solving such flaws, specifically, the heuristic-driven algorithm would choose the resolvers with the lowest estimated cost, respectively the ρ_3 and of the ρ_5 resolvers, strictly avoiding the choice of the ρ_4 and of the ρ_6 resolvers, whose estimated cost is infinite. If, however, the causal variables of the queued flaws (i.e., the φ_6 and the φ_8 flaws) are forced to false, the causal constraints (i.e., (4) and (3)) force the ρ_4 and of the ρ_6 resolvers at false false and, consequently, the ρ_3 and of the ρ_5 resolvers at true, together with the φ_5 and the φ_7 flaws and the ρ_7 and of the ρ_8 resolvers, solving the problem without the slightest need, at least from a causal point of view (a search may in fact still be required for scheduling the activities), to do any search.

4. Experimental results

In order to test the effectiveness of the proposed pruning techniques we have implemented both the procedures for pruning and for recognizing the deferrable flaws within the oRatio[2] planner, testing its performance on different instances of increasing complexity on the Goac domain. Specifically, the Goal Oriented Autonomous Controller (Goac) was an ESA initiative aimed at defining a new generation of software autonomous controllers to support increasing levels of autonomy for robotic task achievement. In particular, the domain, initially defined in [30] and more recently cited in [50], aims at controlling a rover to take a set of pictures, store them on board and dump the pictures when a given communication channel was available. The interesting aspect of this domain is that communication can only take place within specific visibility windows that take into account the astronomical motions of the planets/satellites which, in some cases, may stand between the transmitting and receiving stations. The presence of these visibility windows, in particular, requires an explicit modeling of temporal aspects in order to adequately plan the transmission of information and can hence easily be modeled through, and solved by, timeline-based planners. The problem is made more interesting by the presence of constraints which include the available resources (e.g., memory and battery) as well as by having a distance matrix, among the possible locations, which might be not completely connected.

Figure 3 shows the execution times of different configurations of the oRatio solver, allowing

---

[2] https://github.com/pstlab/oRatio

the comparison of the base solver, without pruning, pruning without recognizing the deferrable flaws and pruning with the recognition of the deferrable flaws. From the figure it is possible to see how, in general, the resolution times significantly benefit from the application of pruning on the proposed benchmarking problem, reducing by an order of magnitude the computation times in the case of the instance #9 of the problem with two visibility windows, or allowing resolution within the two minute timeout in the case of the instance #7 of the 5 visibility windows. Note how deferrable flaw recognition adds a little overhead in smaller instances, which still leads to a benefit in larger instances.

5. Conclusions

The efficiency of the planners' resolution processes is strongly influenced by the heuristics that, in some cases, guide the solution process while, in other cases, can prune the search space so as to favor the propagation of causal constraints and avoid possible dead-ends. The use of pruning techniques is not new in the case of classical planning, however, it is less explored in the case of partial-order planning and, even more so, in the case of high-expressive timeline-based planning. For this reason we have presented a pruning technique based on the construction of a lifted causal graph. The experimental results, although conducted on a single benchmarking problem, show significant benefits and are, therefore, encouraging. A more detailed experimentation on a greater number of benchmarking problems could highlight further benefits or weaknesses of the proposed approach and, therefore, will certainly be carried out in future work.

References

[1] M. Ghallab, D. Nau, P. Traverso, Automated Planning: Theory and Practice, Morgan Kaufmann Publishers Inc., 2004.

[2] B. Bonet, H. Geffner, Planning as Heuristic Search, Artificial Intelligence 129 (2001) 5–33.

[3] J. Hoffmann, B. Nebel, The FF Planning System: Fast Plan Generation Through Heuristic Search, Journal of Artificial Intelligence Research 14 (2001) 253–302.

[4] M. Helmert, The Fast Downward Planning System, Journal of Artificial Intelligence Research 26 (2006) 191–246.

[5] J. Hoffmann, J. Porteous, L. Sebastia, Ordered Landmarks in Planning, Journal of Artificial Intelligence Research 22 (2004) 215–278.

[6] J. Porteous, L. Sebastia, J. Hoffmann, On the Extraction, Ordering, and Usage of Landmarks in Planning, in: Sixth European Conference on Planning, 2014.

[7] P. Haslum, H. Geffner, Admissible Heuristics for Optimal Planning, in: Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge, CO, USA, April 14-17, 2000, AAAI Press, 2000, pp. 140–149.

[8] P. Haslum, B. Bonet, H. Geffner, New Admissible Heuristics for Domain-Independent Planning, in: AAAI, volume 5, 2005, pp. 9–13.

[9] S. Edelkamp, Planning with Pattern Databases, in: Sixth European Conference on Planning, 2014.

[10] M. Helmert, P. Haslum, J. Hoffmann, Flexible Abstraction Heuristics for Optimal Sequential Planning, in: ICAPS, 2007, pp. 176–183.

[11] M. Helmert, P. Haslum, J. Hoffmann, R. Nissim, Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces, Journal of the ACM (JACM) 61 (2014) 16.

[12] V. Vidal, H. Geffner, Branching and pruning: An optimal temporal pocl planner based on constraint programming., volume 170, 2004, pp. 570–577. doi:`10.1016/j.artint.2005.08.004`.

[13] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL—The Planning Domain Definition Language, 1998.

[14] R. Fikes, N. J. Nilsson, STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, in: IJCAI, 1971, pp. 608–620.

[15] M. Fox, D. Long, PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains, Journal of Artificial Intelligence Research 20 (2003) 61–124.

[16] S. Edelkamp, J. Hoffmann, PDDL2.2: The language for the Classical Part of the 4th International Planning Competition, Technical Report 195, Institut für Informatik, 2004.

[17] M. Fox, D. Long, Modelling Mixed Discrete-continuous Domains for Planning, Journal Of Artificial Intelligence Research 27 (2006) 235–297.

[18] A. E. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the fifth international planning competition: {PDDL3} and experimental evaluation of the planners, Artificial Intelligence 173 (2009) 619–668. URL: http://www.sciencedirect.com/science/article/pii/S0004370208001847. doi:`http://dx.doi.org/10.1016/j.artint.2008.10.012`, advances in Automated Plan Generation.

[19] W. Piotrowski, M. Fox, D. Long, D. Magazzeni, F. Mercorio, Heuristic Planning for PDDL+ Domains, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, AAAI Press, 2016, pp. 3213–3219.

[20] S. Franco, M. Vallati, A. Lindsay, T. L. McCluskey, Improving Planning Performance in PDDL+ Domains via Automated Predicate Reformulation, in: Computational Science – ICCS 2019, Springer International Publishing, 2019, pp. 491–498.

[21] N. Muscettola, S. Smith, A. Cesta, D. D'Aloisi, Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture, IEEE Control Systems 12 (1992).

[22] N. Muscettola, HSTS: Integrating Planning and Scheduling, in: Zweben, M. and Fox, M.S. (Ed.), Intelligent Scheduling, Morgan Kauffmann, 1994.

[23] D. S. Weld, An Introduction to Least Commitment Planning, AI Magazine 15 (1994) 27–61.

[24] R. De Benedictis, A. Cesta, Lifted Heuristics for Timeline-based Planning, in: ECAI-2020, 24th European Conference on Artificial Intelligence, Santiago de Compostela, Spain, 2020, pp. 498–2337.

[25] M. Ghallab, H. Laruelle, Representation and Control in IxTeT, a Temporal Planner, in: AIPS-94. Proceedings of the 2nd Int. Conf. on AI Planning and Scheduling, 1994, pp. 61–67.

[26] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, B. Smith, Planning in Interplanetary Space: Theory and Practice, in: AIPS-00. Proceedings of the Fifth Int. Conf. on AI Planning and Scheduling, 2000, pp. 177–186.

[27] S. Chien, D. Tran, G. Rabideau, S. Schaffer, D. Mandl, S. Frye, Timeline-Based Space Operations Scheduling with External Constraints, in: ICAPS-10. Proc. of the 20^{th} Int. Conf.

on Automated Planning and Scheduling, 2010, pp. 34–41.

[28] S. Fratini, F. Pecora, A. Cesta, Unifying Planning and Scheduling as Timelines in a Component-Based Perspective, Archives of Control Sciences 18 (2008) 231–271.

[29] A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, Developing an End-to-End Planning Application from a Timeline Representation Framework, in: IAAI-09. Proceedings of the 21^{st} Innovative Applications of Artificial Intelligence Conference, Pasadena, CA, USA, 2009, pp. 66–71.

[30] S. Fratini, A. Cesta, R. De Benedictis, A. Orlandini, R. Rasconi, APSI-based Deliberation in Goal Oriented Autonomous Controllers, ASTRA 11 (2011).

[31] A. Umbrico, A. Cesta, M. Cialdea Mayer, A. Orlandini, Platinum: A new framework for planning and acting, in: AI*IA 2017 Proceedings, 2017, pp. 498–512.

[32] J. Frank, A. K. Jónsson, Constraint-Based Attribute and Interval Planning, Constraints 8 (2003) 339–364.

[33] A. Cesta, A. Oddi, Gaining Efficiency and Flexibility in the Simple Temporal Problem, in: L. Chittaro, S. Goodwin, H. Hamilton, A. Montanari (Eds.), Proceedings of the Third International Workshop on Temporal Representation and Reasoning (TIME-96), IEEE Computer Society Press: Los Alamitos, CA, 1996, pp. 45–50.

[34] P. Laborie, Algorithms for propagating resource constraints in AI planning and scheduling: existing approaches and new results, Artificial Intelligence 143 (2003) 151–188.

[35] S. Stock, M. Mansouri, F. Pecora, J. Hertzberg, Hierarchical hybrid planning in a mobile service robot, in: KI 2015 Proceedings, 2015, pp. 309–315.

[36] A. Bit-Monnot, M. Ghallab, F. Ingrand, D. E. Smith, FAPE: a Constraint-based Planner for Generative and Hierarchical Temporal Planning, arXiv preprint arXiv:2010.13121 (2020).

[37] F. Dvorák, A. Bit-Monnot, F. Ingrand, M. Ghallab, Plan-Space Hierarchical Planning with the Action Notation Modeling Language, in: IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Limassol, Cyprus, 2014. URL: https://hal.archives-ouvertes.fr/hal-01138105.

[38] D. E. Smith, J. Frank, W. Cushing, The ANML language, in: ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), 2008.

[39] D. E. Wilkins, Practical planning: extending the classical AI planning paradigm / David E. Wilkins, Morgan Kaufmann Publishers San Mateo, Calif, 1988.

[40] D. S. Nau, T. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, F. Yaman, SHOP2: an HTN planning system, J. Artif. Intell. Res. 20 (2003) 379–404.

[41] L. Castillo, J. Fdez-Olivares, O. García-Pérez, F. Palao, Efficiently handling temporal knowledge in an htn planner, in: Proceedings of the Sixteenth International Conference on International Conference on Automated Planning and Scheduling, ICAPS'06, AAAI Press, 2006, pp. 63––72.

[42] P. Laborie, M. Ghallab, Planning with Sharable Resource Constraints, in: Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, IJCAI'95, Morgan Kaufmann Publishers Inc., 1995, pp. 1643–1649.

[43] A. Cesta, A. Oddi, S. F. Smith, A Constraint-Based Method for Project Scheduling with Time Windows, Journal of Heuristics 8 (2002) 109–136. URL: https://doi.org/10.1023/A:1013617802515. doi:`10.1023/A:1013617802515`.

[44] D. E. Smith, J. Frank, A. K. Jónsson, Bridging the Gap Between Planning and Scheduling, Knowledge Engineering Review (2000).

[45] G. Verfaillie, C. Pralet, M. Lemaître, How to model planning and scheduling problems using constraint networks on timelines, The Knowledge Engineering Review 25 (2010) 319–336.

[46] M. Cialdea Mayer, A. Orlandini, A. Umbrico, Planning and execution with flexible timelines: a formal account, Acta Informatica 53 (2016) 649–680. URL: http://dx.doi.org/10.1007/s00236-015-0252-z. doi:`10.1007/s00236-015-0252-z`.

[47] R. Dechter, Constraint Processing, Elsevier Morgan Kaufmann, 2003.

[48] K. R. Apt, M. G. Wallace, Constraint Logic Programming Using ECL^{i}PS^e, Cambridge University Press, New York, NY, USA, 2007.

[49] H. Kautz, B. Selman, Planning as Satisfiability, in: ECAI, volume 92, 1992, pp. 359–363.

[50] A. Coles, A. Coles, M. Martinez Munoz, O. Savas, J. Delfa, T. de la Rosa, Y. E-Martín, A. García Olaya, Efficiently Reasoning with Interval Constraints in Forward Search Planning, in: Proceedings of the Thirty Third AAAI Conference on Artificial Intelligence, AAAI Press, 2019.

Planning with PDDL3 Qualitative Constraints for Cost-Optimal Solutions Through Compilation

Luigi Bonassi[1], Enrico Scala[1] and Alfonso Emilio Gerevini[1]

[1]*Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy*

Abstract

We study the problem of finding cost-optimal solutions to planning problems that feature qualitative state trajectory constraints expressed in PDDL3. These constraints are properties that every plan must satisfy and can be seen as a fragment of LTL over finite traces. The state-of-the-art system for handling PDDL3 problems is a compilation-based approach. Such a compilation has been tested only using a satisficing planner, while the case where we require the planner to find optimal solutions is scarcely studied; with this paper, we want to fill this gap. We propose an experimental analysis that involves TCORE, the current state-of-the-art compilation approach to handle qualitative PDDL3 constraints, and two compilation approaches supporting arbitrary LTL formulas. We evaluate each system using two optimal planners, and we analyze the results using different metrics to explain the behavior of the considered approaches. Our analysis confirms the result previously obtained with a suboptimal planner; that is, in the optimal setting TCORE outperforms all other compilations over our benchmark domains.

Keywords

Automated Planning, PDDL3, Compilation, State-Trajectory Constraints

1. Introduction

The aim of this paper is to study the behavior of different state-of-the-art systems for handling trajectory constraints in the context of finding optimal solutions. In particular, we focus on planning problems that feature qualitative constraints defined by the PDDL3 language [1]. PDDL3 is a popular and standard planning formalism that provides primitives for the specification of such constraints through a subclass of LTL formulas [2]. State-of-the-art approaches deal with such problems either by directly modifying the search engines [3, 4, 5, 6] or by compiling temporal constraints away [7, 8, 9, 10, 11, 12, 13]. Compilation is a technique that works by reformulating a planning problem with temporal constraints into a new equivalent problem without them. Bonassi et al. [13] presents a novel compilation approach, named TCORE, for solving planning tasks with PDDL3 qualitative state-trajectory constraints. TCORE extends a planning task with atomic variables whose role is to maintain the truth of the temporal properties; then it exploits the concept of regression [14] to determine how actions would need to be modified to correctly evaluate the formula over time. Problems compiled through TCORE can be handled by any classical planner that supports conditional effects, making this approach highly modular. It has been shown that [13], when using a suboptimal planner, TCORE performs better than other state-of-the-art compilation approaches over the considered

---

10th Italian Workshop on Planning and Scheduling (IPS-2022)

l.bonassi005@unibs.it (L. Bonassi); enrico.scala@unibs.it (E. Scala); alfonso.gerevini@unibs.it (A. E. Gerevini)

benchmarks. Although the results obtained with a suboptimal planner provide a clear picture of the strengths and weaknesses of the tested compilations, the effectiveness of such approaches in the context of finding optimal solutions is still unknown. To understand how the optimal setting influences the performance of different compilations, we present additional experiments involving TCORE and other compilation approaches for handling arbitrary LTL formulas using optimal planners. The rest of the paper is structured as follows: Section 2 provides some background on planning problems with PDDL3 qualitative constraints; Section 3 summarizes the compilation schema of TCORE; Section 4 presents our experimental analysis; Section 5 gives the conclusions.

2. Background on Planning with PDDL3 constraints

We borrow standard notions and notations from propositional logic and the work by Gerevini et al. [1] on PDDL3; the reader is referred to this work for more details.

A classical planning problem is a tuple $\Pi = \langle F, A, I, G\rangle$ where F is a set of atoms, $I \subseteq F$ is the initial state, G is a propositional formula over F, and A is a set of actions. An action $a \in A$ is a pair $\langle Pre(a), Eff(a)\rangle$, where $Pre(a)$ is a formula over F expressing the preconditions of a, and $Eff(a)$ is a set of conditional effects, each of which is a pair $c \triangleright e$ where: c is a formula and e is a set of literals, both over F. With e^- and e^+ we indicate the partition of e featuring only negative and positive literals, respectively. A state s is a subset of F, with the meaning that if $p \in s$, then p is true in s, and if $p \notin s$, p is false in s. An action is applicable in s if $s \models Pre(a)$, and the application of an action a in s yields the state $s' = (s \setminus \bigcup_{\substack{c \triangleright e \in Eff(a) \\ \text{with } s \models c}} e^-) \cup \bigcup_{\substack{c \triangleright e \in Eff(a) \\ \text{with } s \models c}} e^+$. We indicate with $s' = s[a]$ the state resulting from applying action a in s, and assume conflicting effects (p and $\neg p$) are only yield by conditional effects having their conditions mutually exclusive in s.

A plan π for a problem $\Pi = \langle F, A, I, G\rangle$ is a sequence of actions $\langle a_0, a_1, ..., a_{n-1}\rangle$ in Π; plan π is valid for Π iff there exists a sequence of states (state trajectory) $\langle s_0, s_1, ..., s_n\rangle$ such that $s_0 = I$, $\forall\, i \in [0, \dots, n-1]$ we have that $s_i \models Pre(a_i)$ and $s_{i+1} = s_i[a_i]$, and $s_n \models G$. The cost of a plan $c(\pi)$, is given by the sum of the cost $c(a)$ of each action a in π. A plan π is said to be optimal iff no plan π' with $c(\pi') < c(\pi)$ exists. PDDL3 state-trajectory constraints are a class of temporal formulae over trajectory of states, and they involve necessary conditions that the state trajectory of a valid plan must satisfy. In this work, we consider planning tasks with constraints that in PDDL3 are called "qualitative" [1], because involving only non-numeric terms. In addition to the standard problem goals, a trajectory constraint can be of the following types: `always` ϕ (A_ϕ), which requires that every state traversed by the plan satisfies formula ϕ; `at-most-once` ϕ (AO_ϕ), which requires that formula ϕ is true in at most one continuous subsequence of traversed states; `sometime-before` ϕ ψ ($\mathsf{SB}_{\phi,\psi}$), which requires that if ϕ is true in a state traversed by the plan, then also ψ is true in a previously traversed state; `sometime` ϕ (ST_ϕ), which requires that there is at least one state traversed by the plan where ϕ is true; `sometime-after` ϕ ψ ($\mathsf{SA}_{\phi,\psi}$), which requires that if ϕ is true in a traversed state, then also ψ is true in that state or in a later traversed state.

A *PDDL3 planning problem* is a tuple $\langle \Pi, C\rangle$ where Π is a classical planning problem and C is a set of trajectory constraints; the valid plans of $\langle \Pi, C\rangle$ are all valid plans of Π whose state trajectories satisfy all constraints in C.

3. TCORE: Trajectory Constraints COmpilation via Regression

This section briefly describes how TCORE translates a PDDL3 problem into an equivalent planning problem without constraints. TCORE identifies two classes of constraints: *invariant trajectory constraints* (*ITCs*) and *landmark trajectory constraints* (*LTCs*). Intuitively, ITCs can be checked along any plan prefix and if they are violated, there is no way the planner can ever re-establish them; they are invariant conditions that must be maintained over the state trajectory of the plan. LTCs are constraints that require certain conditions true at *some* state over the state trajectory of the plan. TCORE works by extending the action preconditions and conditional effects in order to (i) block the violation of ITCs during planning, and (ii) keep track of the truth of relevant (w.r.t. trajectory constraints) formulae in the states generated by the plan prefix. This is achieved by making use of the effect regression operator R, and by introducing a set of *monitoring atoms.* The effect regression is a formula manipulation technique: given a propositional formula ϕ and an action a, $R(\phi, a)$ is a propositional formula such that, for any state s, $s \models R(\phi, a)$ iff $s[a] \models \phi$. The regression makes it possible to identify what actual influence the action has over the trajectory constraint of interest. Monitoring atoms serve the purpose of collecting relevant facts on the plan state trajectory and asserting their truth/falsity. The ITCs are: A_{ϕ}, AO_{ϕ}, $\mathsf{SB}_{\phi,\psi}$. For each AO_{ϕ} and $\mathsf{SB}_{\phi,\psi}$, TCORE adds the fresh predicates $seen_{\phi}$ and $seen_{\psi}$ to record whether ϕ and ψ have ever held. The LTCs are: ST_{ϕ} and $\mathsf{SA}_{\phi,\psi}$. For each LTC, the compilation adds a predicate $hold_c$. Such a predicate is meant to record whether the constraint is already satisfied or not according to the current plan prefix.

Algorithm 1 describes the full compilation. As a very first step, TCORE creates the necessary atoms (line 3) and sets up the initial state so as to reflect the current status of the trajectory constraints; in particular, the algorithm captures if a LTC is already achieved in I, or if a formula (ϕ or ψ) that is necessary for the evaluation of an ITC is already true in I. Then TCORE checks whether any ITC is already unsatisfied; if so, the problem is unsolvable.

After the initialization phase, the algorithm iterates over all actions and constraints to modify each original action model (preconditions and effects) by considering the interactions between the constraints and the action model. If the constraint is an ITC, the algorithm determines, by regression, a condition ρ such that, if ρ holds in the state where the action is applied, the execution of such an action will violate the constraint. For example, in the case of A_{ϕ}, ρ models whether the action makes formula ϕ false (line 10). The regressed condition ρ is negated and then conjoined with the precondition of the action. In this way, if the action will violate the constraint in a given state, such an action is deemed inapplicable by the planner. In the case of ITCs, conditional effects are added to keep track of whether relevant formulae have ever held in the state trajectory of the plan prefix. For instance, $\mathsf{SB}_{\phi,\psi}$ requires to deal with the truth of $seen_{\psi}$: if an action makes ψ true, then the action must make $seen_{\psi}$ true too. In this way, the compilation prevents applying an action a when it makes ϕ true and ψ has not held before in current plan state trajectory (lines 14–17).

For each LTC $c \in C$, the algorithm yields a formula ρ that is true only in those states where the action achieves the targeted formula expressed in c. Note here the slightly different treatment

Algorithm 1: TCORE

```
   Input   : A PDDL3 Planning Problem $\langle\langle F, A, I, G\rangle, C\rangle$
1  $A' = \{\}$
2  $G' = \top$
3  $F' = monitoringAtoms(C)$
4  $I' = \bigcup_{c:\mathsf{ST}_\phi \in C} \{hold_c \mid I \models \phi\} \cup \bigcup_{c:\mathsf{SA}_{\phi,\psi} \in C} \{hold_c \mid I \models \psi \vee \neg\phi\} \cup \bigcup_{\mathsf{SB}_{\phi,\psi} \in C} \{seen_\psi \mid I \models \psi\} \cup \bigcup_{\mathsf{AO}_\phi \in C} \{seen_\phi \mid I \models \phi\}$
5  if $\exists \mathsf{SB}_{\phi,\psi} \in C. I \models \phi \vee \exists \mathsf{A}_\phi \in C. I \models \neg\phi$ then
6      return Unsolvable Problem
7  foreach $a \in A$ do
8      $E = \{\}$
9      foreach ITC $c \in C$ do
10         if $c$ is $\mathsf{A}_\phi$ then $\rho = R(\neg\phi, a)$ ;
11         else if $c$ is $\mathsf{AO}_\phi$ then
12             $\rho = R(\phi, a) \wedge seen_\phi \wedge \neg\phi$
13             $E = E \cup \{R(\phi, a) \rhd \{seen_\phi\}\}$
14         else if $c$ is $\mathsf{SB}_{\phi,\psi}$ then
15             $\rho = R(\phi, a) \wedge \neg seen_\psi$
16             $E = E \cup \{R(\psi, a) \rhd \{seen_\psi\}\}$
17         $Pre(a) = Pre(a) \wedge \neg\rho$
18     foreach LTC $c \in C$ do
19         if $c$ is $\mathsf{ST}_\phi$ then $\rho = R(\phi, a)$ ;
20         else if $c$ is $\mathsf{SA}_{\phi,\psi}$ then
21             $E = E \cup \{R(\phi, a) \wedge \neg R(\psi, a) \rhd \{\neg hold_c\}\}$
22             $\rho = R(\psi, a)$
23         $E = E \cup \{\rho \rhd \{hold_c\}\}$
24     $Eff(a) = Eff(a) \cup E$
25     $A' = A' \cup \{\langle Pre(a), Eff(a)\rangle\}$
26 foreach LTC $c \in C$ do
27     $G' = G' \wedge hold_c$
28 return Classical Planning Problem $\langle F \cup F', A', I \cup I', G \wedge G'\rangle$
```

for the two types of LTCs. While ST_ϕ only requires ϕ to be true, for $\mathsf{SA}_{\phi,\psi}$ the compilation needs to signal the necessity of ψ only when ϕ becomes satisfied; this is done by introducing two conditional effects (lines 21 and 22) affecting the additional goal $hold_c$ of G' (line 27). Also observe that ϕ can become true multiple times, and each state satisfying ϕ needs to be followed by a state such that ψ is true again; this state can also be the same state in which ϕ holds, as prescribed by the semantics of PDDL3.

Note that Algorithm 1 can add irrelevant preconditions and conditional effects that can easily be omitted by looking at whether regression leaves a formula unaltered. E.g., for A_ϕ, if $\rho = R(\neg\phi, a) = \neg\phi$, there is no need to extend $Pre(a)$ with $\neg\rho = \phi$ at line 17. Such optimizations are implemented but omitted here for clarity and compactness.

As trajectory constraints are monitored along the entire plan, and regression through effects provides sufficient conditions for ensuring that no ITC is violated by an action and no LTC remains unsatisfied at the plan end, it is easy to see that the compiled problem always finds a solution that conforms with the trajectory constraints of the problem. Moreover, since the exploited regression establishes necessary conditions too, the existence of a solution in the compiled problem implies that the original problem is solvable.

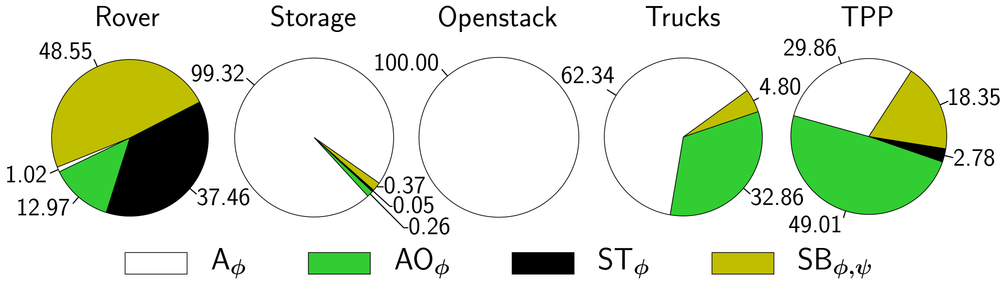


Figure 1: Constraints grouped by type across all domains.

4. Experimental Results

Our experimental analysis studies the behavior of TCORE and two state-of-the-art compilation approaches dealing with LTL constraints in the context of finding optimal solutions. Specifically, we considered the exponential compilation by Baier and McIlraith [8], in short EXP, and the polynomial compilation by Torres and Baier [11], in short POLY[1]. To evaluate the compilations, we considered two optimal planners: A^* with the h^{max} heuristic (implemented in FastDownward [15]) and SYM-K [16]. The former performs a classical A^* search guided by the admissible h^{max} heuristic, while the latter is a top-k planner[2]. Such planners are cost-optimal and support axioms, a feature that is necessary to solve problems compiled through EXP. In total, we tested six different compiler-planner combinations: the three compilations paired with $A^*(h^{max})$ ({TCORE, EXP, POLY}^{max} in short) and with SYM-K ({TCORE, EXP, POLY}^{symk} in short).

The performance of each system is evaluated in terms of the number of solved instances (coverage), the time spent to find a solution (computed as the compilation time plus planning time), the number of nodes expanded by $A^*(h^{max})$, and dimension of the compiled problems. All experiments were performed on a Xeon Gold 6140M 2.3 GHz, with time and memory limits of 1800s and 8GB, respectively. We tested the systems using the benchmark by Bonassi et al. [13]. The suite involves domains from the fifth IPC (https://lpg.unibs.it/ipc-5/), and has a total of 416 instances: 79 for `Trucks`, 90 for `Openstack`, 55 for `Storage`, 94 for `Rover`, and 98 for `TPP`. In such domains, each action a has cost $c(a) = 1$, i.e., the optimal plan is the shortest. Figure 1 gives a general overview on how each type of constraint is partitioned between the considered domains. Overall, the predominant constraint type is `always`, followed by `at-most-once`, `sometime-before` and `sometime`.

4.1. Results analysis

Problem dimensions. Table 1 presents the dimensions of the compiled instances in terms of average number of fluents and average number of effects. Such metrics are computed only

---

[1] POLY extends a planning problem with many additional synchronization actions that are necessary to update the automaton representing a LTL formula. In this experimental analysis, we set the cost of synchronization actions to zero, as this guarantees that optimal plans correspond to optimal plans of the original problem.

[2] The objective of top-k planning is to determine a set of k different plans with the lowest cost for a problem [16]. In our experimental analysis we used a symbolic bidirectional search obtained from the top-k planner with $k = 1$.

Domain	Fluents			Effects		
	TCORE	EXP	POLY	TCORE	EXP	POLY
`Openstack`	**251.5**	571.2	883.6	**3906.4**	452635.8	12815.8
`Rover`	**116.8**	346.5	614.1	**875.8**	155595.9	3238.0
`Storage`	**87.6**	184.9	315.2	**961.8**	18000.6	1666.7
`TPP`	**27.7**	98.7	268.3	**226.0**	3181.3	1014.7
`Trucks`	**374.6**	522.5	754.9	**8667.6**	497196.0	15225.9

Table 1
Average number of fluents and effects in the instances resulting from each compilation. For the sake of a fair comparison, the reported statics refer to the ground instances. For EXP, the number of fluents include the number of derived fluents. Averages are computed on instances compiled by all systems.

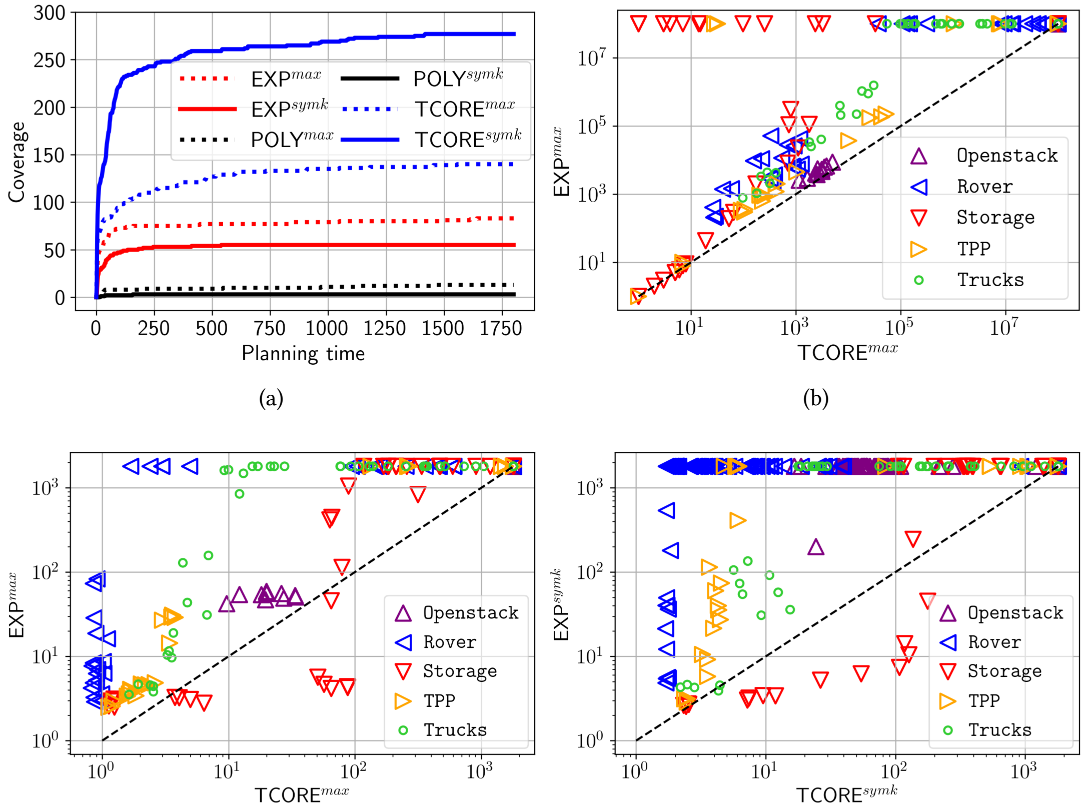
Domain	A^* with h^{max}			SYM-K		
	EXP	POLY	TCORE	EXP	POLY	TCORE
`Openstack` (90)	10	3	10	1	0	**88**
`Rover` (94)	19	0	34	12	0	**89**
`Storage` (55)	19	9	**33**	15	3	31
`TPP` (98)	18	1	24	14	0	**25**
`Trucks` (79)	17	0	39	13	0	**44**
Total (416)	83	13	140	55	3	**277**

Table 2
Coverage achieved by all systems across all domains. In parenthesis, the number of instances for a given domain.

for instances compiled by each system. EXP and POLY work on the lifted representation of a planning task, while TCORE has to instantiate the planning problem before performing the actual compilation. For the sake of a fair comparison, the averages reported in Table 1 were calculated by grounding the instances resulting from the two LTL systems using the FastDownward translator.

We can observe that the tasks compiled by TCORE contain on average less effects and less fluents compared to the instances compiled with EXP/POLY. This is due to the fact that EXP and POLY are capable of handling arbitrary nested LTL formulas (unsupported by PDDL3); to deal with such expressive power, these two compilations rely on automata theory to compile the LTL formulas away. Integrating the resulting automatons into the domain model requires many additional fluents and effects as shown by Table 1. On the other hand, TCORE is tailored at handling PDDL3 qualitative constraints and this allows for a more efficient compilation. Indeed, TCORE introduces up to one fluent for each trajectory constraint (zero in the case of an always) and takes advantage of the regression computation to minimize the number of additional effects.

Coverage. Table 2 shows an overall picture of the coverage achieved by each system across all domains. TCORE^{symk} achieves the highest coverage in four out of five domains, solving a total of 277 instances. Remarkable are the performances in `Rover` and `Openstack`; TCORE^{symk} solves 95% and 98% of the instances in these domains, respectively. The performances of TCORE



(a) (b)

(c) (d)

Figure 2: Coverage versus planning time (a). Comparison between TCORE^{max} and EXP^{max} in terms of expanded nodes (b). CPU-time comparison of TCORE with EXP when planning with $A^*(h^{max})$ (c) and with SYM-K (d).

with an optimal planner are close to those of TCORE with a satisficing planner: in the same instances, TCORE with LAMA [17] solves only 34 more instances [13]. This may indicate that, in many instances, the challenge is to effectively deal with the given set of constraints, rather than finding the optimal solution.

As expected, TCORE^{max} cannot solve many problems, achieving roughly half the coverage of TCORE^{symk}. We attribute this behavior to the fact that A^* with h^{max} is less sophisticated compared to SYM-K. Interestingly, though, A^* with h^{max} is more effective at solving instances compiled with EXP and POLY; with both compilations, such a system outperforms SYM-K in all domains. We believe that this is due to the fact that SYM-K does not scale well when problems have many atoms and effects, and this is the case for the instances compiled through EXP and POLY, as shown in Table 1.

Coverage-wise, Bonassi et al. [13] observed that with a suboptimal planner EXP outperforms TCORE in TPP. This is caused by the fact that TCORE fails to ground most instances in this domain, while EXP directly works on the first-order representation. Our experimental results show that the two optimal planners cannot exploit this advantage of EXP; this indicates that finding optimal solutions to problems compiled with EXP in `TPP` is a challenging task for the considered planners.

Analogously to the results presented by Bonassi et al. [13], EXP dominates POLY across all domains. One could expect that the polynomial compilation performs better than the exponential compilation. This is not the case, as the LTL formulas deriving from PDDL3 constrains do not lead to an exponential blow-up of the compilation performed by EXP.

CPU time analysis. Figure 2a reports how systems increase their coverage over time. TCORE^{symk} surpasses EXP^{symk} right from the start, and achieves 90% of its maximum coverage after 333 seconds. TCORE^{max} increases its coverage at a slower rate, achieving 90% of its maximum coverage in about 474 seconds. Differently, A^* with h^{max} is faster than SYM-K with EXP, as EXP^{max} dominates EXP^{symk}.

Figures 2c and 2d show a pairwise comparison of the runtimes of TCORE with the runtimes of EXP, while Figure 2b compares TCORE^{max} with EXP^{max} in terms of expanded nodes. Most of the PDDL3 instances are solved generally faster than the instances compiled with EXP. From Figure 2b, we can observe that TCORE^{max} expands less nodes than EXP^{max}, and this indicates that the h^{max} heuristic is more informed when tasks are compiled with TCORE; the EXP compilation introduces axioms, a feature that seems to be not fully supported by the heuristic. There is an exception for a small set of `Storage` problems; in such instances EXP performs more efficiently than TCORE regardless of the planner. We attribute this to the fact that TCORE has to spend time to ground the instances before performing the actual compilation, while EXP directly works on the first-order representation.

5. Conclusions

We have presented an experimental analysis that compares three different compilation-based systems to handle PDDL3 qualitative constraints with optimal planners. Results shows that TCORE remains the state-of-the-art approach to handle the considered class of problems. Recently, Bonassi et al. [18] has shown that, by expressing control knowledge in PDDL3, TCORE can be used as a tool to improve the coverage of a satisficing planner. In the future, we intend to test TCORE with new benchmarks featuring control knowledge in PDDL3 to improve the coverage of an optimal planner. Finally, we plan to extend TCORE for handling quantitative state-trajectory constraints and to study a compilation that works on the lifted representation of a planning task.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work has been partially supported by EU-H2020 projects AIPlan4EU (No. 101016442) and TAILOR (No. 952215), and by MUR PRIN-2020 project RIPER (No. 20203FFYLK).

References

[1] A. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners, Artif. Intell. 173 (2009) 619–668.

[2] A. Pnueli, The temporal logic of programs, in: FOCS, IEEE Computer Society, 1977, pp. 46–57.

[3] A. J. Coles, A. Coles, LPRPG-P: relaxed plan heuristics for planning with preferences, in: ICAPS, AAAI, 2011.

[4] J. Benton, A. J. Coles, A. Coles, Temporal planning with preferences and time-dependent continuous costs, in: ICAPS, AAAI, 2012.

[5] Y. Chen, B. W. Wah, C. Hsu, Temporal planning using subgoal partitioning and resolution in sgplan, J. Artif. Intell. Res. 26 (2006) 323–369.

[6] C. Hsu, B. W. Wah, R. Huang, Y. Chen, Constraint partitioning for solving planning problems with trajectory constraints and goal preferences, in: IJCAI, 2007, pp. 1924–1929.

[7] J. A. Baier, F. Bacchus, S. A. McIlraith, A heuristic search approach to planning with temporally extended preferences, Artif. Intell. 173 (2009) 593–618.

[8] J. A. Baier, S. A. McIlraith, Planning with first-order temporally extended goals using heuristic search, in: AAAI, AAAI Press, 2006, pp. 788–795.

[9] S. Edelkamp, S. Jabbar, M. Nazih, Large-scale optimal pddl3 planning with mips-xxl, 5th International Planning Competition Booklet (IPC-2006) (2006) 28–30.

[10] S. Edelkamp, On the compilation of plan constraints and preferences, in: ICAPS, AAAI, 2006, pp. 374–377.

[11] J. Torres, J. A. Baier, Polynomial-time reformulations of LTL temporally extended goals into final-state goals, in: IJCAI, AAAI Press, 2015, pp. 1696–1703.

[12] B. Wright, R. Mattmüller, B. Nebel, Compiling away soft trajectory constraints in planning, in: KR, AAAI Press, 2018, pp. 474–483.

[13] L. Bonassi, A. E. Gerevini, F. Percassi, E. Scala, On planning with qualitative state-trajectory constraints in PDDL3 by compiling them away, in: ICAPS, AAAI Press, 2021, pp. 46–50.

[14] J. Rintanen, Regression for classical and nondeterministic planning, in: ECAI, volume 178 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2008, pp. 568–572.

[15] M. Helmert, The fast downward planning system, J. Artif. Intell. Res. 26 (2006) 191–246.

[16] D. Speck, R. Mattmüller, B. Nebel, Symbolic top-k planning, in: AAAI, AAAI Press, 2020, pp. 9967–9974.

[17] S. Richter, M. Westphal, The LAMA planner: Guiding cost-based anytime planning with landmarks, J. Artif. Intell. Res. 39 (2010) 127–177.

[18] L. Bonassi, A. E. Gerevini, E. Scala, Planning with qualitative action-trajectory constraints in PDDL, in: IJCAI, ijcai.org, 2022, pp. 4606–4613.

The Epistemic Planning Domain Definition Language

Alessandro Burigana[1], Francesco Fabiano[2]

[1]Free University of Bozen-Bolzano, Bolzano, Italy
[2]University of Parma, Parma, Italy

Abstract
Epistemic planning is a branch of Artificial Intelligence that stems from the combination of automated planning and Dynamic Epistemic Logic (DEL). While DEL provides a very expressive framework—which comes at the cost of undecidability—to guarantee the feasibility of the planning process, various fragments have been studied in the literature, that rely on very specific syntax for representing domains. As a result, a comprehensive language that is able to capture the general DEL framework is currently not present in the literature. In this paper, we propose a new language called EPDDL (Epistemic Planning Domain Definition Language), through which we can capture the full DEL semantics, thus allowing for a general and unified syntax for representing epistemic planning domains.

1. Introduction

Epistemic planning [1] is an enrichment of automated planning, where the notions of *knowledge* and *belief* [2] are introduced. In this field, we take into account the perspective of one or more agents: *epistemic states* contain both factual information about the world and epistemic information concerning the knowledge/beliefs of agents. Similarly, *epistemic actions* describe how their knowledge/beliefs change. For instance, imagine that Anne received a letter from an university she applied for. She then starts reading it in front of Bob, who can not directly take a look at it. Once Anne has finished reading, her perspective will differ from Bob's: while she now *knows* whether she was admitted or not, Bob still considers both options to be possible. Thus, the epistemic action describing this situation has to consider the points of view of both agents.

Epistemic planning has been frequently built on the formalism of Dynamic Epistemic Logic (DEL) [3]. Since the general framework is undecidable, various epistemic planners implement fragments of DEL [4, 5, 6, 7, 8, 9]. One of the main issues is that, to represent planning domains, either solvers rely on languages tailored for a specific fragment [6, 10, 11], or they adopt no language at all, making the comparison of existing solvers more difficult.

In this paper, we introduce the *Epistemic Planning Domain Definition Language* (EPDDL). The language stems from the combination of the syntax of PDDL and the DEL semantics and it aims at overcoming the specificity of existing languages. Namely, EPDDL is able to handle the full DEL framework, which in turn supports an unified representation. The paper is organized as follows. In Section 2, we recall the semantics of DEL. In Section 3, we describe EPDDL and its main features. We conclude with a brief discussion and some future works.

IPS 2022: 10th Italian Workshop on Planning and Scheduling, November 28–December 2, 2022, Udine, Italy
burigana@inf.unibz.it (A. Burigana); francesco.fabiano@unipr.it (F. Fabiano)
0000-0002-9977-6735 (A. Burigana); 0000-0002-1161-0336 (F. Fabiano)

CEUR Workshop Proceedings (CEUR-WS.org)

2. Dynamic Epistemic Logic

Epistemic Models. Let $\mathcal{P}$ be a countable set of propositional atoms and $\mathcal{AG} = \{1, \dots, n\}$ be a finite set of agents. The language $\mathcal{L}^C_{\mathcal{P},\mathcal{AG}}$ of *multi-agent epistemic logic* is defined by the BNF: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Box_i\varphi \mid C_G\varphi$, where $p \in \mathcal{P}$, $i \in \mathcal{AG}$ and $G \subseteq \mathcal{AG}$. We read the formulae $\Box_i\varphi$ and $C_G\varphi$ as "agent i knows/believes that φ" and "group G has common knowledge/belief that φ", respectively. In the *possible world* semantics of DEL, a *Kripke model* [12] represents an *epistemic model.* Epistemic models contain both factual information on multiple possible worlds and epistemic information, *i.e.*, which worlds are considered possible by agents.

Definition 1 (Kripke Model). *A* Kripke model *is a triple $M = (W, R, V)$ where: 1) $W \neq \varnothing$ is the set of possible worlds, 2) $R : \mathcal{AG} \to 2^{W \times W}$ assigns to each agent i an accessibility relation $R(i)$ (abbreviated as R_i), 3) $V : \mathcal{P} \to 2^W$ assigns to each atom a set of worlds.*

An *epistemic state* is a pair (M, W_d), where $W_d \subseteq W$ is a non-empty set of *designated worlds* denoting the *real* state of affairs. Truth of formulae is defined as follows: i. $(M, w) \models p$ iff $w \in V(p)$; ii. $(M, w) \models \neg\varphi$ iff $(M, w) \not\models \varphi$; iii. $(M, w) \models \varphi \wedge \psi$ iff $(M, w) \models \varphi$ and $(M, w) \models \psi$; iv. $(M, w) \models \Box_i\varphi$ iff $\forall v$ if $(w, v) \in R_i$ then $(M, v) \models \varphi$; and v. $(M, w) \models C_G\varphi$ iff $\forall v$ if $(w, v) \in R^*_G$ then $(M, v) \models \varphi$ (where $R_G = \cup_{i \in G} R_i$ and R^*_G denotes its transitive closure). Finally, $(M, W_d) \models \varphi$ iff $(M, v) \models \varphi$, for all $v \in W_d$.

Event Models. In DEL, actions are modeled by *event models* [13, 14], that represent how new information changes the perspectives of agents. Here, *events* can be seen as *possible actions* and the accessibility relation Q_i describes which events are considered possible by agent i.

Definition 2 (Event Model). *An* event model *is a quadruple $\mathcal{E} = (E, Q, pre, post)$ where: 1) $E \neq \varnothing$ is the set of events, 2) $Q : \mathcal{AG} \to 2^{E \times E}$ assigns to each agent i an accessibility relation $Q(i)$ (abbreviated as Q_i), 3) $pre : E \to \mathcal{L}^C_{\mathcal{P},\mathcal{AG}}$ assigns to each event a* precondition*, 4) $post : E \to (\mathcal{P} \to \mathcal{L}^C_{\mathcal{P},\mathcal{AG}})$ assigns to each event a* postcondition *for each atom.*

Informally, preconditions capture the applicability of events, whereas postconditions determine whether an atom is true after the event is applied. A *(multi-)pointed event model* is a pair $(\mathcal{E}, E_d)$, where $E_d \subseteq E$ is a non-empty set of *designated events*, and it represents an *action* in DEL.

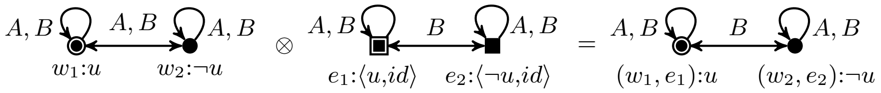
Product Update. The product update formalizes how actions bring about information change in epistemic states in DEL. An action $(\mathcal{E}, E_d)$ is *applicable* in (M, W_d) if and only if for each world $w \in W_d$ there exists an event $e \in E_d$ such that $(M, w) \models pre(e)$.

Definition 3 (Product Update). *Given an action $(\mathcal{E}, E_d)$ applicable in an epistemic state (M, W_d), where $M = (W, R, V)$ and $\mathcal{E} = (E, Q, pre, post)$, the* product update *of (M, W_d) with $(\mathcal{E}, E_d)$ is the multi-pointed Kripke model $(M, W_d) \otimes (\mathcal{E}, E_d) = ((W', R', V'), W'_d)$, where:*

1. $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$,
2. $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid (w, v) \in R_i$ *and* $(e, f) \in Q_i\}$,
3. $V'(p) = \{(w, e) \in W' \mid (M, w) \models post(e)(p)\}$,
4. $W'_d = \{(w, e) \in W' \mid w \in W_d$ *and* $e \in E_d\}$.

Example 1 (DEL). *Consider the above situation with Anne (A) and Bob (B). Before Anne reads the letter, both agents consider possible two options: either she was admitted in the university*

(u), or not. Thus, we need two worlds to represent this uncertainty (w_1 and w_2, respectively). As we are universal observers, we assume that we know that w_1 is the designated world (circled dot). This is modeled by the epistemic state on the left. When Anne reads the letter, Bob can not read its content. In the center of the picture, events e_1 (Anne learns that u) and e_2 (Anne learns that $\neg u$) are needed to represent Bob's uncertainty about the outcomes of the action (we use the notation $\langle pre, post\rangle$ for events). In this way, Anne learns that she was admitted, while Bob remains uncertain. Moreover, both Anne and Bob are reciprocally aware of their perspective. The epistemic state M' that results from this action is on the right. Notice that now Anne knows that she was admitted, i.e., *$(M', (w_1, e_1)) \models \Box_a u$, while Bob does not change his perspective.*



3. EPDDL

The development of the syntax of EPDDL was driven by three design principles: **(1)** to maintain the style of PDDL syntax, **(2)** to capture the *entire* DEL semantics (*i.e.*, to be able to represent *all* possible event models), and **(3)** to obtain an intuitive and usable language, even for researchers that are less familiar with epistemic planning and DEL. The main structure of EPDDL includes three main components: *problem* (specific part), *domain* and *action type library* (universal parts). Due to space limits, we do not provide the full BNF and we base our exposition on Example 1.

Problem. The problem defines "specific" aspects of a planning task. In EPDDL, in addition to objects, initial state and goal, we also list which *agents* are present in the particular instance. Goals in EPDDL are expressed by a generic formula $\varphi_g \in \mathcal{L}^C_{\mathcal{P},\mathcal{AG}}$. The syntax of propositional formulae is the same as in PDDL. Modal formulae such as $\Box_i\varphi$ and $C_G\varphi$ (with $|G| \geq 2$) are represented as `[i]`φ and `[G]`φ, respectively. Finally, initial states can be represented in two ways: either explicitly (specifying worlds, relations and valuation), or by means of a *finitary S5-theory* [15], *i.e.*, a *set of formulae* of a particular form that admits a finite number of (equivalent) finite models (computable in polynomial time). In our example, we use a finitary S5-theory to state that: 1) Anne was admitted to the university (line 4), and 2) Anne and Bob have common knowledge that they both do not know whether she was admitted (lines 5-6).

```
1 (define (problem p1)
2   (:domain example1)
3   (:agents Anne Bob)
4   (:init (u)
5         [Anne Bob](and (not [Anne](u)) (not [Anne](not (u))))
6         [Anne Bob](and (not [Bob](u))  (not [Bob](not (u)))))
7   (:goal [Anne](u)))
```

Domain and Action Type Library. In EPDDL, the universal aspects of a problem (types, predicates, actions) are jointly described by a *domain* and an *action type library*. Moreover, actions are defined on three separated levels of abstraction: *events*, *action types* and *actions*. Domains define types, predicates and actions, whereas action type libraries contain the description of events and action types. We now analyze more in depth these elements. Events constitute the atomic components, where preconditions and postconditions (if any) are defined. Events are then combined within action types to represent event models (Definition 2). Finally, each action must include its *type* in its specification. Actions and their type are described separately, since it is common for different actions to share the same underlying structure. This allows for a more concise and readable representation (design principle (3)).

Since a library is intended to describe *universal* aspects, action types should not refer to specific entities of a problem. To this end, we implemented two important design choices: 1) parametrized events and action types, and 2) *observability groups*. First, parameters allow to abstract from particular predicates and agents. Importantly, in EPDDL, apart from objects, parameters can also refer to *agents*, *formulae* and *postconditions*. This aspect is crucial to maintain the *universality* of action types: for instance, if we pass the preconditions of events as parameters, we can simply refer to them as variables within an action type. As a result, action type libraries can be used transversally across *different domains*. Second, observability groups generalize accessibility relations. Each group represents the perspective of one or more agents. For instance, when Anne reads the letter, she is *fully observant*, since she learns its content, while Bob is *partially observant*, since he remains uncertain about Anne's knowledge. Action types only refer to observability groups (lines 13-15). Then, in each action we assign each agent to a group (lines 29-33). We now show the EPDDL representation of the action of Example 1.

```
 1 (define (library lib)
 2 (:event e1
 3   :parameters (?sensed - predicate)
 4   :precondition (?sensed))
 5 (:event e2
 6   :parameters (?sensed - predicate)
 7   :precondition (not (?sensed)))
 8 (:action-type sensing
 9   :parameters (?p - predicate)
10   :observability-groups (Fully Partially)
11   :events     (e1 (?sensed :: ?p) )
12               (e2 (?sensed :: ?p) )
13   :relations  (Fully     (e1 e1) (e2 e2))
14               (Partially (e1 e1) (e2 e2)
15                          (e1 e2) (e2 e1))
16   :designated (e1)))
17 (define (domain example1)
18 (:action-type-libraries lib)
19 (:requirements :del :typing :equality
20                :universal-conditions)
21 (:predicates (u)
22              (has_letter ?ag - agent))
23 (:action read_letter
24   :parameters (?ag - agent)
25   :action-type (sensing (?p :: (u)) )
26   :precondition (has_letter ?ag)
27   :observability-conditions
28     (?ag Fully)
29     (forall (?ag2 - agent)
30         (if (not (= ?ag2 ?ag))
31             (Partially)
32         ))))
```

4. Discussion

In this paper, we have introduced a new language for describing epistemic planning domains, called EPDDL. Importantly, this language is able to capture the full DEL semantics. Intuitively, this follows from the fact that each component of an event model (events, relations, preconditions, postconditions) is captured by a corresponding syntactical element of EPDDL. Thus, we obtain a syntax through which we can represent the existing fragments of epistemic planning in an unified way. Due to space limits, we could not address all features of the language. As future work, we plan on finalizing the syntax of EPDDL and to implement a full-fledged parser (also including a *type checker*). Moreover, we intend to use EPDDL to create a public repository of epistemic planning domains to be used as benchmarks for epistemic planners.

References

[1] T. Bolander, M. B. Andersen, Epistemic Planning for Single and Multi-Agent Systems, J. Appl. Non Class. Logics 21 (2011) 9–34. URL: https://doi.org/10.3166/jancl.21.9-34. doi:`10.3166/jancl.21.9-34`.

[2] J. Hintikka, Knowledge and Belief: An Introduction to the Logic of the Two Notions, Contemporary Philosophy, Cornell University Press, 1962.

[3] H. P. van Ditmarsch, W. van der Hoek, B. P. Kooi, Dynamic Epistemic Logic, volume 337, Springer Netherlands, 2007. URL: https://doi.org/10.1007/978-1-4020-5839-4. doi:`10.1007/978-1-4020-5839-4`.

[4] A. Burigana, F. Fabiano, A. Dovier, E. Pontelli, Modelling Multi-Agent Epistemic Planning in ASP, Theory Pract. Log. Program. 20 (2020) 593–608. URL: https://doi.org/10.1017/S1471068420000289. doi:`10.1017/S1471068420000289`.

[5] F. Fabiano, A. Burigana, A. Dovier, E. Pontelli, EFP 2.0: A Multi-Agent Epistemic Solver with Multiple E-State Representations, in: J. C. Beck, O. Buffet, J. Hoffmann, E. Karpas, S. Sohrabi (Eds.), Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020, AAAI Press, 2020, pp. 101–109. URL: https://aaai.org/ojs/index.php/ICAPS/article/view/6650.

[6] X. Huang, B. Fang, H. Wan, Y. Liu, A general multi-agent epistemic planner based on higher-order belief change, in: C. Sierra (Ed.), Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, ijcai.org, 2017, pp. 1093–1101. URL: https://doi.org/10.24963/ijcai.2017/152. doi:`10.24963/ijcai.2017/152`.

[7] F. Kominis, H. Geffner, Beliefs In Multiagent Planning: From One Agent to Many, in: R. I. Brafman, C. Domshlak, P. Haslum, S. Zilberstein (Eds.), Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015, AAAI Press, 2015, pp. 147–155. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10617.

[8] T. Le, F. Fabiano, T. C. Son, E. Pontelli, EFP and PG-EFP: Epistemic Forward Search Planners in Multi-Agent Domains, in: M. de Weerdt, S. Koenig, G. Röger, M. T. J. Spaan (Eds.), Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018, AAAI Press, 2018, pp. 161–170.

[9] C. J. Muise, V. Belle, P. Felli, S. A. McIlraith, T. Miller, A. R. Pearce, L. Sonenberg, Planning Over Multi-Agent Epistemic States: A Classical Planning Approach, in: B. Bonet, S. Koenig (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, AAAI Press, 2015, pp. 3327–3334. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9974.

[10] C. Baral, G. Gelfond, E. Pontelli, T. C. Son, An Action Language for Multi-Agent Domains: Foundations, CoRR abs/1511.01960 (2015). URL: http://arxiv.org/abs/1511.01960.

[11] F. Fabiano, B. Srivastava, J. Lenchner, L. Horesh, F. Rossi, M. B. Ganapini, E-PDDL: A standardized way of defining epistemic planning problems, CoRR abs/2107.08739 (2021). URL: https://arxiv.org/abs/2107.08739. `arXiv:2107.08739`.

[12] S. A. Kripke, Semantical Considerations on Modal Logic, Acta Philosophica Fennica 16

(1963) 83–94.

[13] A. Baltag, L. S. Moss, S. Solecki, The Logic of Public Announcements and Common Knowledge and Private Suspicions, in: I. Gilboa (Ed.), Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98), Evanston, IL, USA, July 22-24, 1998, Morgan Kaufmann, 1998, pp. 43–56.

[14] H. van Ditmarsch, B. Kooi, Semantic Results for Ontic and Epistemic Change, Texts in Logic and Games 3, Amsterdam University Press, 2008, pp. 87–117.

[15] T. C. Son, E. Pontelli, C. Baral, G. Gelfond, Finitary s5-theories, in: E. Fermé, J. Leite (Eds.), Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings, volume 8761 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 239–252. URL: https://doi.org/10.1007/978-3-319-11558-0_17. doi:`10.1007/978-3-319-11558-0\_17`.

Exploring the Unified Planning Framework for a More Integrated and Flexible Fault-Tolerant Flight Path Planning System

Sondes Morchedi[1], Prakash Jamakatel[1] and Jane Jean Kiam[1,†]

[1]*Universität der Bundeswehr München, Institute for Flight Systems, Werner-Heisenberg-Weg 39, 85579 Neubiberg, Germany*

Abstract
This paper reports the ambitions of an ongoing work intended to exploit the Unified Planning Framework (UPF) to enhance and extend a flight path planning system we developed using an automated planner to compute flight trajectories in non-nominal situations. We identify i) multiple benefits to draw using UPF, specifically to overcome the shortcomings of our previous implementation and ii) possible extensions of the flight path planning system thanks to the functions the UPF offers.

Keywords
Planning, Fault-tolerant Flight Planning, Flight Path Planning, Unified Planning Framework, PDDL+

1. Introduction

For more sustainable mobility solutions, the next generation aircraft is expected to reduce emissions significantly during flights while providing a better overall aircraft performance and safety [1]. In line with this, future intelligent flight systems are required to adapt to varying conditions, to the system's dynamics, and to handle faults while keeping the pilot informed. A fault-tolerant flight path planning system is an essential part of an intelligent flight system. In addition to increasing fuel efficiency, it is also expected to reduce damage and casualties in emergency situations caused by unforeseen events, e.g. faulty mechanics or human errors [1].

In the previous work [4], an automated flight path planning system was developed to assist the pilot in a single-pilot ultralight aircraft. The system was implemented using a PDDL+ compatible automated planner [5], which uses a forward state-space search and heuristics capable of coping with non-linear mathematical operations. In this paper, we report an ongoing work aimed at extending the previous work, by exploring the benefits of the UPF [6]. We first provide a brief summary of the previous work, followed by an overview of the adapted system

---

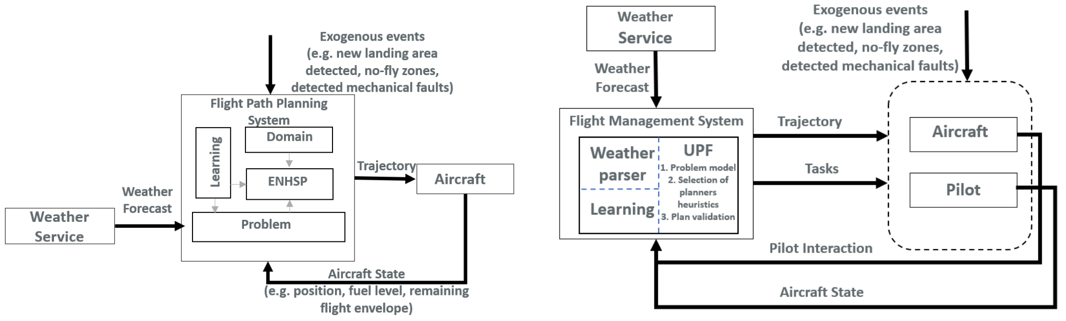
10th Italian Workshop on Planning and Scheduling (IPS-2022), November 28–02 December, 2022, Udine, IT

sondes.morchedi@unibw.de (S. Morchedi); prakash.jamakatel@unibw.de (P. Jamakatel); jane.kiam@unibw.de (J. J. Kiam)
0000-0002-2754-7095 (P. Jamakatel); 0000-0002-6830-9931 (J. J. Kiam)

[1] Especially in non-commercial aviation, the majority of the accidents can be credited to human errors [2], due mainly to the less rigorous demand set on the pilot for flight licence, i.e. lesser number of training hours and laxer medical requirements [3].

architecture and underline the expected benefits of exploiting the UPF. Subsequently, we discuss the ongoing progress and challenges faced, alongside with the future work considered.

2. Exploring the UPF for the Flight Path Planning System



(a) Automated fault-tolerant flight path planning with ENHSP.

(b) A more flexible flight path planning system with the UPF

Figure 1: Comparing the flight path planning system architectures.

The aim of the automated fault-tolerant flight path planning system is to determine a flight trajectory meant to guide an aircraft (e.g. a single-pilot ultralight in [4]) from its current position to a goal position placed within the vicinity of a safe landing area. This is helpful especially when the pilot is overwhelmed, typically when an emergency landing is required due to low-fuel level, or to a mechanical fault. Figure 1a illustrates the architecture adopted in the previous work [4], in which the automated flight-path planning system considers weather information known at planning time[2], the current state (i.e. position and fault detection state variables) of the aircraft, and the goal position, all encoded as part of the (PDDL+) problem instance. The aircraft's physical model is encoded as a problem domain adapted from [8]. The ENHSP planner solver [5] is exploited to compute an executable temporal plan $\pi =< a_0(t_0), a_1(t_1), \cdots, a_n(t_n) >$, with each a_* representing a set of control parameters, namely the turn rate and the climb rate, which will be used to predict the sequence of waypoints $< x_0(t_0), x_1(t_1), \cdots, x_n(t_n) >$, setting therefore a feasible reference trajectory for the autopilot to guide the airplane, in case the pilot is incapable of steering the aircraft. A learning module built on k-means clustering is part of the planning system to determine the optimal planning parameters for non-nominal situations (i.e. emergency landing with low fuel or maneuvering a faulty aircraft) using flight data generated from flight simulations, for tuning the ENHSP planner solver, as well as for selecting the ranges of the state variables in the planning problem, in order to simplify the search [9].

Nevertheless, there are several shortcomings worth mentioning in the previous works [4, 9]. To call the ENHSP planner solver, another bash instance is created within the flight path planning system, and run as an external process, limiting therefore the interaction with the planner solver. Furthermore, minimum changes in the problem instance, due to either a new

---

[2] The weather information used is from the National Oceanic and Atmospheric Administration (NOAA) [7].

goal condition, e.g. if a better landing area is detected, or to minimal changes in the weather information, the problem domain and instance have to be parsed and grounded entirely, which can be time-consuming for a more complex problem model. Therefore, we have identified the UPF as a more compact and efficient framework that can overcome these shortcomings, and offer more meaningful extensions to the fault-tolerant flight path planning system.

2.1. The Unified Planning Framework (UPF)

The UPF [6] is a collaborative effort under development within the AIPlan4EU project, with the purpose of providing flexibility in the planning problem definition (similar to *Tarski* [10]) and making a wide range of planning technologies accessible (similar to *Planutils* [11]), via their integration as Python libraries within a single framework.

2.2. Benefits of using UPF for the Flight Path Planning System

Based on the published information and objectives of the UPF in [6] and [12], we identified some features of the UPF that can be exploited to develop a more flexible and integrated flight path planning system, of which the architecture is as depicted in Figure 1b. The benefits drawn from the UPF-based flight path planning system architecture are described below.

Overcoming the shortcomings of the previous work: Without having to invoke an external bash process, it is more flexible to interact with the planning engines within the same Python framework, which is particularly useful when we consider a human-in-the-loop planning system, in which the pilot can modify a solution plan, for example by adding an intermediate waypoint. In this case, the `PlanValidator` can be called programmatically.

A more prominent improvement the UPF will bring forth is the coping with a dynamic environment, as it allows to change state variables programmatically without having to parse the entire problem instance completely, reducing thereby the overall computational time. With this flexible parsing, a partial planning problem containing typically weather data and airspace constraints can be pre-encoded, later completed by the pilot (with goals or additional constraints), who interacts with the flight path planning system via a cockpit user interface. Domain-specific validator can check the pilot's inputs for conflicts (with the constraints of the existing partial problem) before including them into the planning problem. Having this implemented, the flight path planning system will conform with the EASA Roadmap for the use of AI in a cockpit: the system is pilot-centered, with AI playing an assisting role [13].

Other potential extensions: To date, the only planner solver used for the (kinodynamic) flight path planning problem at hand is ENHSP. The main hindrance in using other planners is the lack of support for non-linear mathematical operations encoded in the problem domain. With the parser of UPF supporting the encoding of non-linear mathematical operations, it is likely that more planner solvers capable of solving this class of problem will be included in the UPF, leading to the possibility of creating multiple planning instances in parallel for different planners or for different heuristics of the same planner, emulating thereby "diverse planning" to obtain multiple solutions that can be presented to the pilot for assisting him/her in decision-making.

Since the current implementation relies only on one solver (ENHSP), plan repair, although can be considered, can only be done externally to the AI planning module, and in a "programmatic" and domain-specific manner, as in [8], where partial plans are "stitched" together by adding an artificial "bridge" to ensure continuity between them. With the UPF, even if the planner solver does not include plan repair capabilities, another planning engine can be used for repairing a plan, which is relevant to our use case, given the dynamic environment, e.g. a no-fly-zone is determined during flight due to detected moving objects nearby (such as drones, cranes, etc.).

The learning module in the previous work was based on an unsupervised learning method, with the aim to select the set of planning and control parameters to optimize the likelihood of a safe landing in non-nominal situations. This learning module can be extended to include inverse reinforcement learning to learn aircraft maneuvers from an experienced pilot in emergency landing, and use these "pre-learned" maneuvers as actions in the planning problem. Having planning libraries implemented in Python enables the exploitation of Python libraries such as scikit-learn [14] or OpenAI Gym [15] within the same Python-based system.

3. Discussion and Future Work

Having identified the benefits of the UPF for our current flight path planning system that was developed in Matlab in [4] and in C++ in [8], we are currently working towards porting the parser for weather data, the automated encoding of problem models, and the learning module into Python3. As soon as the integration of PDDL+ syntax and *global constraints*, which have been proven to be extremely handy in defining no-fly zones in a compact manner, is realized in the UPF, the architecture shown in Figure 1b can be fully implemented without much adaptation and its performance can be compared to the previous work. Subsequently, the improvements and extension described in Section 2.2 can also be included.

3.1. Future Work to Include Hierarchical Task Network Planning

As pointed out in the "Future Work" of [6], hierarchical structures will be added into the UPF, enabling thereby the solving of Hierarchical Task Network (HTN) planning problems. This will further benefit the extension of the flight path planning system to an even more versatile flight management system capable of assisting the pilot with tasks of higher abstraction levels[3] while simultaneously communicating a flight path plan to reach the desired goal location. With this extension, not only that (high-level) task and motion planning can be supported by the same framework and communicated to the pilot in a hierarchical manner (that is more comprehensible for human cognition), plan and goal recognition can also be performed in an automated fashion using HTN planning engines [17], resulting in a close-loop assisting system.

4. Acknowledgments

This work is done within the research group "Intelligent Control of Highly Over-Actuated Flight Systems" funded by Munich Aerospace and within the MOREALIS project funded by the German Federal Ministry of Economic Affairs and Climate Action.

---

[3] Pilots are in general instructed with tasks that are hierarchical in nature, and abstract away minute details, as domain-level knowledge is assumed [16].

5. Citations and Bibliographies

References

[1] European Union Aviation Safety Agency, Study on the societal acceptance of urban air mobility in europe, 2021.

[2] A. De Voogt, F. Chaves, E. Harden, M. Silvestre, P. Gamboa, Ultralight accidents in the us, uk, and portugal, Safety 4 (2018) 23.

[3] European Union Aviation Safety Agency, Light sport aircraft, 2022. Accessed: 2022-10-18.

[4] B. S. León, J. J. Kiam, A. Schulte, Model-based automated flight path planning for an ultralight aircraft (2020).

[5] E. Scala, P. Haslum, S. Thiébaux, M. Ramirez, Interval-based relaxation for general numeric planning, in: ECAI 2016, IOS Press, 2016, pp. 655–663.

[6] A. A. Micheli, A. Alexandre Bit-Monnot, al., Unified planning: A python library making planning technology accessible, in: International Conference on Automated Planning and Scheduling: System Demonstrations, 2022.

[7] G. Rutledge, J. Alpert, S. Ronald, B. Lawrence, The noaa operational model archive and distribution system (nomads) (2003).

[8] J. J. Kiam, E. Scala, M. R. Javega, A. Schulte, An ai-based planning framework for haps in a time-varying environment, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 30, 2020, pp. 412–420.

[9] B. S. León, J. J. Kiam, A. Schulte, A fault-tolerant automated flight path planning system for an ultralight aircraft, in: M. Baldoni, S. Bandini (Eds.), AIxIA 2020 – Advances in Artificial Intelligence, Springer International Publishing, Cham, 2021, pp. 175–190.

[10] M. Ramırez, G. Frances, Tarski, URL https://github. com/aig-upf/tarski. Accessed on (2021) 05–17.

[11] S. J. K. M. Muise Christian, Pommerening Florian, Planutils: Bringing planning to the masses, International Conference on Automated Planning and Scheduling: System Demonstrations (2022).

[12] A. Micheli, A. Arnold, A. Bit-Monnot, and other contributors, Unified-planning documentation, https://unified-planning.readthedocs.io/en/latest/, 2022. Accessed: 2022-10-21.

[13] European Union Aviation Safety Agency (EASA), Artifial intelligence roadmap: A human-centric approach to ai in aviation, 2020.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, arXiv preprint arXiv:1606.01540 (2016).

[16] Cessna Aircraft Company, Cessna 172 1974 skyhawk owner's manual: Pilot operating handbook (poh) / aircraft flight manual (afm), Independently Published, 1972.

[17] D. Höller, G. Behnke, P. Bercher, S. Biundo, Plan and goal recognition as htn planning, in: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2018, pp. 466–473.

A Preliminary Study on BERT applied to Automated Planning

Lorenzo Serina[1], Mattia Chiari[1], Alfonso E. Gerevini[1], Luca Putelli[1] and Ivan Serina[1]

[1]*Università degli Studi di Brescia, Brescia, Italy*

Abstract

Despite some similarities that have been pointed out in the literature, the parallelism between automated planning and natural language processing has not been fully analysed yet. However, the success of Transformer-based models and, more generally, deep learning techniques for NLP, could open interesting research lines also for automated planning. Therefore, in this work, we investigate whether these impressive results could be transferred to planning. In particular, we study how a BERT model trained on plans computed for three well-known planning domains is able to understand how a domain works, its actions and how they are related to each other. In order to do that, we designed a variation of the typical masked language modeling task which is used for the training of BERT, and two additional experiments into which, given a sequence of consecutive actions, the model has to predict what the agent did previously (*Previous Action Prediction*) and what it is going to do next (*Next Action Prediction*).

Keywords

Automated Planning and Natural Language Processing, Deep Learning, BERT

1. Introduction

In 2007, the work in [1] drew an interesting parallel between automated planning and natural language processing (NLP), presenting a new formulation for plan recognition (i.e. the task of inferring an agent's plan observing some of its actions) based on grammars. The authors claimed that the two disciplines could share some of the research results, but they stated that much of the recent work in both fields had gone unnoticed by the researchers in the other field. In the following years, the separation between these two fields has not diminished. For NLP, deep learning techniques such as word embedding [2], recurrent neural networks, the attention mechanism [3] and pre-trained Transformer-based architectures [4, 5] have revolutionised the field, reaching a completely new state of the art in many different tasks [6, 7]. Although automated planning is a key and current research field in artificial intelligence [8, 9, 10], deep learning had a limited impact on it. In fact, it is mostly used for predicting heuristics [11], processing sensor data in order to create a symbolic representation of a planning problem [12] and for goal recognition tasks [13, 14] or in specific applications [15].

In this work, we focus on one of the most recent deep learning architectures for NLP: BERT [5]. Processing a huge quantity of sentences and documents, BERT is able to learn how the language works and its capabilities can be exploited for text classification [16], sentiment

---

10th Italian Workshop on Planning and Scheduling (IPS-2022)

lorenzo.serina@unibs.it (L. Serina); m.chiari017@unibs.it (M. Chiari); alfonso.gerevini@unibs.it (A. E. Gerevini); luca.putelli1@unibs.it (L. Putelli); ivan.serina@unibs.it (I. Serina)

CEUR Workshop Proceedings (CEUR-WS.org)

analysis [17], question answering [18] and other NLP tasks. This is done by training the model to perform the so-called *masked language modeling* task. Basically, the input of the BERT model is an incomplete sentence, into which some words are replaced by a special marker, and the model has to reconstruct the complete sentence, predicting the missing words from context.

Inspired by the parallelism pointed out in [1], in this work we aim to transfer these techniques in the automated planning field. We claim that plans and actions can be seen similarly to sentences and words, and we have designed a *planning language modeling* task into which the model has to reconstruct an incomplete sequence of actions to train the model. With this technique, we train a single BERT model using actions from three well-known planning domains: LOGISTICS, SATELLITE and BLOCKSWORLD. Next, we evaluate the performance of the model in this task and in two additional experiments: *Next Action Prediction*, into which the model has to predict the action that follows a sequence of actions (which are the input of the model) extracted from a complete plan, and *Previous Action Prediction*, into which the predicted action is the one that precedes the input sequence. These tasks were made to demonstrate how well the model understands about how a planning domain works, its rules, its actions and their effects. Moreover, in this work, we present a detailed report of the main difficulties in the training process, the necessary choices the user has to make in order to train a similar model, the dimension of the datasets required to obtain good results, etc. Finally, we discuss the positive and negative aspects of applying these techniques to the planning domain.

2. Background on BERT

BERT (Bidirectional Encoder Representations from Transformer) [5] is a deep learning architecture based on Transformer [4] composed by several layers which progressively analyse a sequence of elements, named tokens, in order to understand how they relate to each other, their meaning and the overall properties of the entire sequence. Although BERT is a model typically used for Natural Language Processing, recently the same techniques have been applied in other contexts such as the analysis of programming languages [19], graphs [20] and images [21].

At first, each token is associated to an index. Next, an embedding layer calculates a vector of real numbers for each token, as in initial representation of its meaning. Considering a sequence of tokens S of length N, each token $t \in S$ is represented by a vector $x_t \in \mathbb{R}^d$. We denote as $X \in \mathbb{R}^{N \times d}$ the overall matrix of token embeddings.

After calculating this representation, these vectors are processed by several encoding layers. Each layer applies in parallel multiple self-attention mechanisms, called *heads*, into which the relation between each possible pair of tokens is computed. This mechanism produces a matrix $A_{i,j} \in \mathbb{R}^{N \times N}$, into which i is the number of the encoding layer and j is the head number. For each token $w \in S$, the vector $a_w \in A_{i,j}$ contains the attention weights that represent how much w is related to the other tokens in S. In order to calculate these weights, in each head the input representation of the token sequence $X \in \mathbb{R}^{N \times d}$ is projected into three new representations called key (K), query (Q) and value (V) with three weight matrices W_k, W_q and W_v:

$$K = X \times W_k, \ Q = X \times W_q, \ V = X \times W_v$$

Then, the self-attention weights are calculated using the softmax function on the scaled dot-product between Q and K. Finally, this mechanism computes a new vector representation of the input tokens (Z) by multiplying the attention weights for V.

$$A = softmax\left(\frac{Q \times K^{\intercal}}{\sqrt{d}}\right), \quad Z = A \times V$$

Given that in each encoding layer there are several heads (typically 8, 12, 16 or 24), in order to create a single representation provided by the *multi-head attention mechanism* the result of each head is concatenated and then passed to a feed-forward layer.

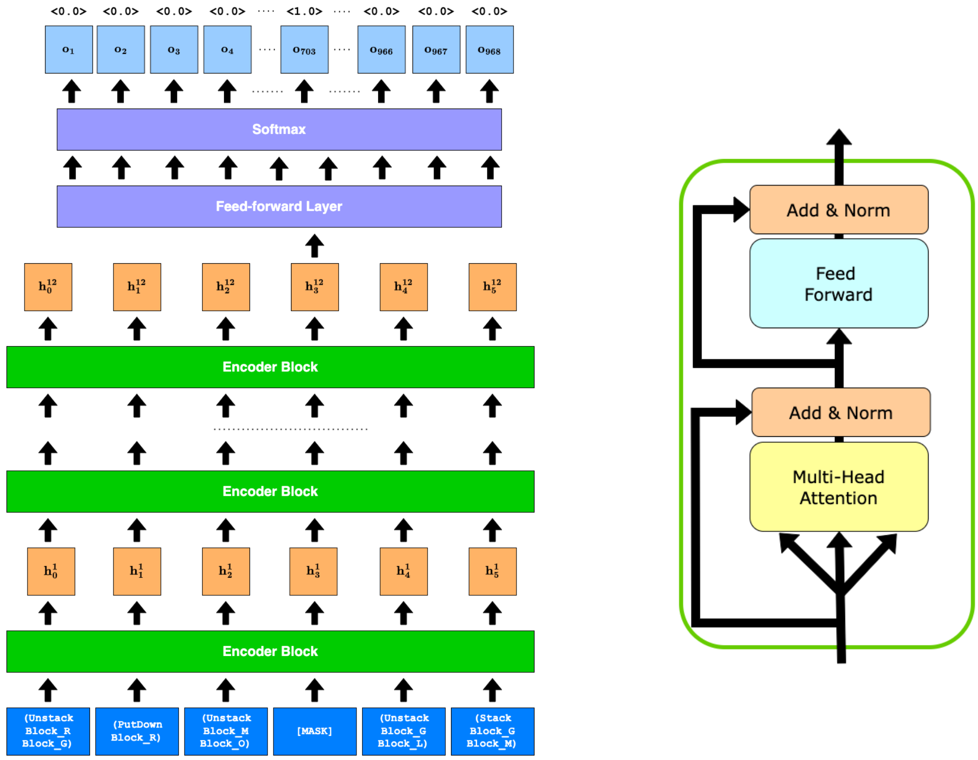
As described in [4], the multi-head attention mechanism is followed by a residual connection which adds the input representation with the one calculated by the multi-head attention. Next, the results is fed to a feed-forward layer and another residual connection. The output of the encoding layer is also the input of the next encoder. Typically, a BERT model is composed by 8, 12 or 16 encoding layers. A schematic representation of an encoding layer is shown in Figure 1b.

In the original model, designed for NLP tasks, the training was performed using 16GB of documents for a total of more than 3 billions of words [5]. However, more recent implementations exploit even larger datasets (for instance, RoBERTa [22] uses about 165GB of data). The training task is called *masked language modeling* and works as follows. Given a sequence of words (typically the maximum length is 512 tokens), a certain percentage of them is replaced with a special token called [MASK]. The encoding layers process this sequence and learns how to predict the masked words from the context formed by the remaining ones. In order to perform this operation, the output of the last encoding (i.e. the final representation of each token produced by the architecture) is connected to a fully-connected layer with softmax activation, into which each neuron represents a word from the vocabulary. Supposing that the overall corpus of documents contains $100K$ different words, this layer will be formed by $100K$ neurons.

3. Planning Language Modeling with BERT

In order to perform PLM, we approach the planning domains as a NLP task. In fact, our BERT model is trained using a slight variation of the typical language modeling task that we call *Planning Language Modeling*. In this technique, given a sequence of contiguous actions $\bar{p}$ composed of n actions a_i, with $i \in [1, n]$, we consider each action as a separated token, like words in a sentence. First, we divide the action sequence into separated tokens, using the WordPiece tokenizer [23]. Next, we substitute a certain percentage of actions with the special token [MASK] and give this incomplete sequence as input. The BERT model has the task of predicting the missing actions from the overall context of the action sequence. If the model is capable of performing this operation, we claim that it has the ability to understand how the planning domain works, the impact of the actions, when they are performed by an agent, and which effects they have. In order to perform the planning language modeling, the BERT model needs processing a large quantity of these training instances, progressively adjusting the training weights of the model with the backpropagation algorithm until it reaches a satisfying level of accuracy.

Our model architecture is very similar to the one proposed in [5]. We use a total of 12 encoding layers, each one of them with 12 heads, and we use 768 as the embedding size, i.e. actions are represented as vectors of 768 real numbers across the model.



(a) (b)

Figure 1: On the left (a) , schematic representation of a BERT model performing the PLM task. On the right (b), a simplified view of the structure of an encoder block.

Our choice is to develop a multi-domain model, considering Logistics, Blocksworld and Satellite. With a single model, following the current trend of multi-language models, we want to verify whether BERT could understand different domains at the same time. Next, we aim to study the training process for a model which was already trained on a different domain. The model was first trained on Logistics. It took 40 epochs to understand the domain and reach a good accuracy. Then we trained the model on Blocksworld and Satellite and it took less than 10 epochs to achieve good performance. Therefore, we verified that adding another domain to a pre-trained model does not require another, full training procedure. Instead, the model can easily adapt to the new domain as in fine-tuning operations, requiring only a few epochs to achieve a good accuracy on the new domain.

3.1. A Blocksworld Example

In this section, we present a simple example of how we structure our planning language modeling task. For simplicity's sake, let us assume that our BERT works only for this domain. We use a very simple plan instance in the well-known Blocksworld domain. In this domain, the agent has the goal of building one or more stacks of blocks, and only one block may be moved at a time. The agent can perform four types of actions: `Pick-Up` a block from the table, `Put-Down` a block on the table, both these operations take only one argument, `Stack` which puts a block on top of another one and `Unstack` which removes a block that is on another one, both have two arguments. We assume that our plans involve at most 22 blocks, therefore we have 22 possible Pick-Up action, 22 Put-Down actions, $22 * 21$ Stack actions and $22 * 21$ Unstack actions, for a total number of 968 actions.

Let's consider the plan formed by the following 6 actions: `(Unstack Block_R Block_G)`, `(PutDown Block_R)`, `(Unstack Block_M Block_O)`, `(Stack Block_M Block_R)`, `(Unstack Block_G Block_L)`, `(Stack Block_G Block_M)` with ids corresponding to indices 59, 462, 121, 703, 309 and 240. Next, we mask 1 action (17%) chosen randomly. Assuming that the special token [MASK] has index 0, the input of our BERT model will be the following sequence: [59, 462, 121, 0, 309, 240].

This sequence is then processed by the embedding layer and by all the encoding layers, producing a final representation of each element as a vector of 768 real numbers. Given that the fourth element of the sequence is masked, its embedding vector is passed to a feed-forward $f$layer composed of 968 neurons (one for each possible action of the domain) with the softmax activation function. Given that the original action had index 703, we want in output a vector formed by all zeroes except for a 1 at index 703. Therefore, the result produced by the feed-forward layer is compared with the desired output. This procedure is repeated for every training instance, evaluating the overall error made by the architecture and calculating a loss function, which will be used by the backpropagation algorithm to adjust the model weights.

In Figure 1a we show a simplified representation of the model architecture performing the planning language modeling task. The actions are encoded into vectors by several encoding blocks and, in the last step, the vectorial representation of [MASK] is passed to a feed-forward layer with softmax activation function that predicts a 1 at the index 703 and 0 for all the other indexes.

3.2. Training Technique

In order to properly train the model, there are some necessary choices that the user must make. Although typically BERT models treat a specific language, there are several multi-language models. Given that a planning domain has its own set of actions, predicates, and rules, we can consider it as a sort of an independent language. Therefore, the user should evaluate whether training a BERT model for each domain, perhaps specialising and improving its predictive capabilities, or building a multi-domain model, selecting which domains to consider and their quantity.

Next, an important constraint of the BERT model is that it must have a predefined vocabulary of actions. Therefore, during the training process the user has to define a predefined set of objects for each domain and generate all the possible actions that an agent could perform with them. Although the predefined vocabulary is sensitive to the object names, this operation only requires to set a maximum number of objects. In fact, a mapping algorithm can be easily implemented for translating new names into predefined ones.

As with all deep learning techniques, a crucial aspect for obtaining good performance is the number of training instances, i.e. the number of plans necessary to train the BERT model. The choices of how many domains to consider for training and how many objects to include have a

Domains	Objects	Vocab. size	Plans	Mean length	Max length	Augmented instances
LOGISTICS	52	15154	200k	29	382	360k
BLOCKSWORLD	22	968	180k	39	338	430k
SATELLITE	65	33225	140k	17	33	140k

Table 1
Characteristics of the dataset we used to train the BERT model in terms of overall number of objects considered, number of actions in the vocabulary, number of generated plans and their mean and maximum length. In the column Augmented instances we report the dimension of the dataset after the data augmentation process.

serious impact on the number of plans which have to be collected or generated for the training of the model, the amount of time required to this process, and its difficulty.

In our model, we have selected three well-known planning domains: LOGISTICS, BLOCKSWORLD and SATELLITE. In Table 1 we report the characteristics of our dataset. Please note that while in BLOCKSWORLD the only objects present in the domain are the blocks, in the other two domains there are several types of objects (for instance, LOGISTICS includes airplanes, airports, locations, cities, trucks and packages) and we report the overall sum of them for simplicity's sake.

In order to train the multi-domain model, the typical approach is simply considering a single training set composed by the plans for all the three domains. However, we have also experimentally verified that it is possible to add a domain to a pre-trained model extending its vocabulary and continuing its training with additional epochs using the plans generated from the new domain. In this phase, however, we have noticed that it is useful to include also a percentage (such as 25%) of plans belonging to the already processed domains.

The overall dimension of our data set is about 350 MB of data; therefore, it is definitively smaller than standard BERT architectures [5, 22].

3.3. Data Augmentation

Another technique that can be useful in training a BERT model is data augmentation. In this procedure, we divide the longest plans in several subsequences with random dimension and add them to the dataset. With this technique, the training set can contain not only entire plans but also smaller sequences of actions. In order to avoid including a large number of very similar instances, which could create overfitting issues to the model, we assure that plans which are too similar to the ones already present in the dataset are not added. In our context, for LOGISTICS and BLOCKSWORLD we divided the plans with more than 20 actions, randomly selecting a subsequence of the plan. The subsequence is added to the training set if the similarity calculated by the Ratcliff-Obershelp algorithm [24] with the other plans is lower than 0.4, otherwise it isn't. While this operation increases the predictive performance of the model for these two domains, in SATELLITE data augmentation did not improve the results. We claim that this is due to the smaller size of the SATELLITE plans, with respect to the other two domains. In Table 1 we report the dimension of the original dataset (with complete plans) and the augmented dataset, with the smaller subsequences generated by the data augmentation process.

4. Experiments and Results

In this section, we present the set of learning tasks we designed for our model. The model was trained on 2 NVIDIA V100, for 120 epochs, with a training time of almost 130 hours.

4.1. Designed Tasks and Evaluation Metrics

First of all, we test the capabilities of our model to perform the planning language modeling using a test set composed by plans which were not used during the training of the algorithm. In this experiment, we check if the predictions of the masked input actions are correct. Given that the softmax activation function of the output layers actually predicts a score between 0 and 1 for all the possible actions of our three domains, we evaluate the planning language masked in two configurations. In the first one (*Top-1* accuracy), we consider only the action predicted with the highest score. Given the very high number of possible classes in output (one for each action, i.e. more than 40 thousands in our multidomain model), we also use as an evaluation metric the *Top-5* accuracy [25] into which we consider the set of the 5 actions with the highest score, and, if the original action belongs to the set, we consider that the model correctly predicts the masked action. The planning language modeling is evaluated using the following percentages of masked actions: 10%, 15%, 25% and 50%.

We also design two additional tasks to test the capabilities of the model to handle sequences of actions and to infer some of the domain knowledge such as actions' preconditions and effects. For both of these tasks, we use the same plans used for the planning language modeling. In the first task, called *Next Action Prediction*, we ask our model to predict the action that follows an input action sequence. In the second, called *Previous Action Prediction*, we ask our model to predict the action that precedes an input action sequence. In detail, given an action sequence of length l, for *Next Action Prediction*, we pass the first $l - 1$ actions to the model and ask the model to predict the l-th action. Similarly, for *Previous Action Prediction* the model should predict the first action of the sequence while the remaining actions are passed in input. We evaluate *Next Action Prediction* and *Previous Action Prediction* using different input sequence lengths (i.e. 3, 5, 10, 20) and with the entire plan without the last and first action, respectively (Tot).

For the evaluation of these tasks, in our opinion, verifying whether the predicted action matches the one in the input plan is not enough. In fact, the input action sequence filled with the predicted action might be valid even though the predicted action does not match the one in the label. Moreover, we are not providing the model the initial state or the goal (see Section 5), therefore the model cannot always predict the best action based only on an incomplete set of actions. For this reason we use VAL [26] to check if, starting from the initial state of the problem, the action sequence, which also comprehends the action predicted by the model, can be executed or not, providing insight into how the model is able to learn the inner or working of a planning domain[1].

---

[1] Please note that VAL does not always validate our plan as the goal of the problem is often not reached; nonetheless, we use VAL errors to determine whether the sequence is valid or not

Task	10 %	15%	25%	50%
Top 1	0.710	0.671	0.574	0.333
Top 5	0.821	0.802	0.725	0.486

Table 2
Results for the Planning Language Modeling task in terms of *Top-1* and *Top-5* accuracy. On the columns, we report the percentage of masked actions in the test sequences.

Length	3	5	10	20	Tot
Top 1	0.234	0.291	0.200	0.333	0.596
Top 1 (V)	0.650	0.675	0.454	0.664	0.759
Top 5 (V)	0.885	0.906	0.755	0.928	0.929

Table 3
Results for the *Next Action Prediction* task. Length stands for the number of input actions (column tot means that we pass an entire plan to the model except for the last action). On the rows, we report the *Top-1* accuracy considering only the correct actions and *Top-1* and *Top-5* accuracy considering also the valid actions (V).

4.2. Experimental Results

Considering all three domains, in Table 2 we show the results, in terms of *Top-1* and *Top-5*, of the planning language modeling task, using 3000 plans, 1000 for each domain, as a test set. In this experiment, we masked different percentages of actions: 10%, 15%, 25% and 50%. As it should be expected, the accuracy decreases proportionally to the increasing of the percentage of masked actions (i.e. less input data). The overall performances are more than acceptable, in particular for the 50% in *Top-5*, where the model, analyzing just half of the plan, is able to correctly predict almost half of the actions, highlighting a deep comprehension of the context. With less masked actions, the accuracy is very high, with more than 80% in *Top-5* for 10% and 15% and good performance also in terms of *Top-1*.

The results for the *Next Action Prediction* task in terms of *Top-1*, *Top-5* are shown in Table 3. In each column, we consider a different length for the input sequence. For instance, if the length is 5 we have 5 consecutive actions in input and we want to predict the 6-th.

In each row we can notice the same trend: the accuracy increases with the length of the plan, except for the plans with length 10, where there is a drop of the performances. However, in terms of *Top-5* we have very good results even with small action sequences, underlining a deep understanding of the inner workings of the planning domain. Focusing only on the *Top-1* correct, the results are much lower. That could be due to the fact that our model does not receive any information about the agent's goal and therefore it can only predict a set of executable actions, into which it is possible to find the right one.

To better understand these results, we study the performance for each considered domain for the prediction of valid actions. In Figure 2, we analyze the accuracy in terms of *Top-1* and *Top-5* accuracy. On the left, we can see how *Top-1* accuracy increases proportionally to the length of the plans in the LOGISTICS domain, from 51% to 92%, while it's almost constant in SATELLITE

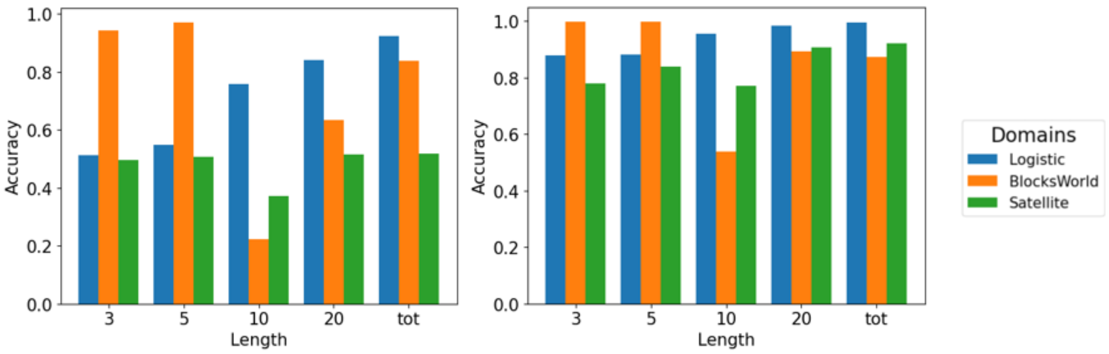


Figure 2: Results for the *Next Action Prediction* task in terms of *Top-1* (left) and *Top-5* (right) for each domain. In this experiment we considered the 5 most probable predictions of the model. On the x-axis we found the length of the action sequence and on the y-axis we found the accuracy of the predictions, computed as validated answers on total instances in the specific domain.

Length	3	5	10	20	Tot
Top 1	0.287	0.385	0.511	0.545	0.566
Top 1 (V)	0.423	0.510	0.594	0.600	0.602
Top 5 (V)	0.637	0.740	0.834	0.843	0.854

Table 4
Results for the *Previous Action Prediction* task. Length stands for the number of input actions (column tot means that we pass an entire plan to the model except for the first action). On the rows, we report the *Top-1* accuracy considering only the correct actions and *Top-1* and *Top-5* accuracy (V) considering also the valid actions.

domain. In the BLOCKSWORLD domain the results are excellent with shorter plans, with 95%, while they drop with length equals to 10, to 22%. Then they recover with longer plans, arriving to 84%. As it can be seen in Figure 2, the performance drop for length 10 regards almost only the BLOCKSWORLD domain. We will address this issue in Section 5. In terms of *Top-5* accuracy, on the right of Figure 2 we can see the same trend as the previous task on the performances, with good results in predictions, on LOGISTICS up to to 99% and on SATELLITE up to 92% .

In Table 4 we report the results for the *Previous Action Prediction* task using the same metrics and format used in the evaluation of the *Next Action Prediction* task. Similarly to the *Next Action Prediction* results, we can notice that, for each row, the accuracy increases as the input length increases. These results are very promising, even if they are slightly lower than the ones obtained in *Next Action Prediction.* The performances increase with the length of the plan and the collapse on length 10, seen in the previous task, is not present.

As we did previously, in Figure 3 we show the results for each domain in terms of *Top-1* and *Top-5* accuracy (considering also the valid actions) on the *Previous Action Prediction* task. In terms of *Top-1*, we can notice that the performances on the LOGISTICS and the BLOCKSWORLD domains remains quite constant, (with a maximum of 47% and 57% respectively), while on the SATELLITE domain they increase proportionally with the length of the plans, from 33% to 84%. In

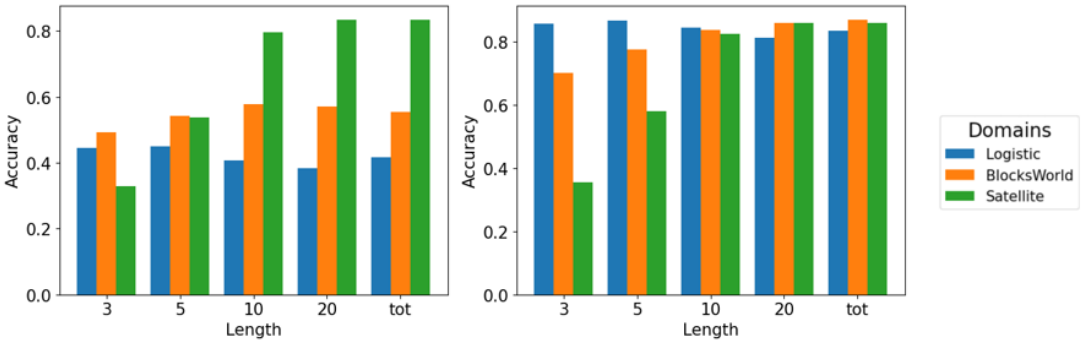


Figure 3: Results for the *Previous Action Prediction* task in *Top-1* (left) and *Top-5* (right) for each domain. In this experiment we considered the 5 most probable predictions of the model. On the x-axis we found the length of the action sequence and on the y-axis we found the accuracy of the predictions, computed as validated answers on total instances in the specific domain.

terms of *Top-5*, we have higher results for all three domains. Howevever, while the performance in the LOGISTICS domain remains almost the same for all lengths, we can notice a small increase in BLOCKSWORLD and a strong increase in SATELLITE. This domain in particular has some issues with actions sequences of length 3 (36%) but reaches 86% at length 10, showing that for this domain we need an adequate number of actions in order to have significant predictive results.

5. Discussion

The results presented in Section 4.2 show that a deep learning model such as BERT, trained on a set of action sequences from well-known planning domains, can effectively accomplish three predictive tasks: the planning language modeling, i.e. predicting actions from context, the prediction of the next action given sequences of different length and, in the same conditions, the prediction of the previous action.

In our opinion, this demonstrates that BERT is able to understand how a planning domain works and which rules are followed by the action sequences. Despite a great number of possible actions to predict (especially for the SATELLITE domain, which has more than 33k actions in our configuration), our results in terms of *Top-1* and *Top-5* accuracy show that the model is able to identify which are the most probable ones. Although we have relatively bad results in terms of *Top-1* for the *Next Action Prediction* and *Previous Action Prediction* tasks considering only if the action belonged to the original plan, the performance are much higher considering the valid actions. This can be also an indicator that our model does not simply memorize a lot of action sequences, but it is really capable to adapt its knowledge with new plans, avoiding overfitting issues.

A problem of this first implementation is that we do not provide any data on the initial state and the goal to our model. This could be the reason for the performance drop in BLOCKSWORLD with sequences of length 10 in the *Next Action Prediction* task. In fact, while the model seems capable of predicting the next action with smaller sequences, when the complexity increases it could be necessary to provide more data in order to direct its reasoning. However, it is interesting to note that for LOGISTICS this does not happen. This may be due to the fact that information about initial state could be indirectly contained in the action sequences. While we will conduct a further analysis of these results, considering also other domains, we think that including the initial state and the goal in the model could improve the results and avoid this performance drop.

While we have verified that a BERT model has this learning capability, we are also conscious of several limits. For instance, as we show in Table 1, the number of plans necessary to train such a model is very high. In a real-world application, this can be a huge problem because it would require a long time to collect the necessary data to exploit this kind of model. While our data augmentation technique shows interesting results, perhaps similar results could be obtained with simpler models, which could require less training data. Another related problem with this kind of architectures is the training time and the computational resources required. An interesting line of research could be apply knowledge distillation techniques and build smaller versions of our model [27].

Moreover, the original BERT for natural language processing [5] is a pre-trained model and adapted for other tasks such as document classification. To do that, the authors introduced a special token called [CLS] for representing the entire document. In our context, we could have the same token for obtaining a vector representation of the entire plan or action sequence and use it for several tasks such as heuristic prediction or plan and goal recognition. However, training [CLS] in BERT required another task, called Next Sentence Prediction, into which the model learns the ability to predict whether two sentences are consecutive in a document. Although this is a very intuitive operation in NLP, the same concept is not present in automated planning. Thus, there is the need to design a completely new task which could lead the model to learn an informative representation of a sequence of actions.

6. Related work

In the last few years, deep learning techniques started to surface across several contexts and tasks related with automated planning.

For instance, the work in [12] exploits variational autoencoders based on neural networks to create latent and symbolic representations of actions starting from pairs of images which describe a transition between two states. This representation can be further exploited for other purposes, such as Goal Recognition, which is defined as the task of recognising the goal that an agent is trying to achieve from observations about the agent's behaviour in the environment [28]. Using only image-based domains such as MNIST or 8-Puzzle, the work [13] first computes a latent representation using the tool proposed by [12] and then analyses the sequence of actions using an LSTM Neural Network [29] for predicting the goal.

Goal Recognition has seen several other applications of deep learning techniques. For instance, the work [15] presents an LSTM network to infer player goals from the observation of their low-level actions in the context of digital games. However, more general works such as [30] and [31] used the classical domains of automated planning and, starting from partial observations of states or actions represented symbolically, trained neural networks or other machine learning algorithms in order to predict the agent's goal.

Deep learning techniques are also used in many path planning applications with robots [32]. For example, in [33] a neural network is used to process data from robot's sensor in order to get an approximate direction. This approximation is then used to build a navigation system. Furthermore, the controller for an autonomous mobile robot presented in [34] implements a feed forward neural network which predicts the steering angle given the obstacles distances from the left, right and front directions.

Another problem which has been addressed by deep learning techniques is the learning of heuristic functions for classical planning. For instance, the authors of [11] trained a neural network to compute the distance between two states and used the output measure as a heuristic. However, one of the drawbacks of their approach is the need to train a neural network for each planning task. The work in [35] solves this issue for a subset of planning domains by learning a strong heuristic function for PDDL domains with an underlying grid structure. The authors implement a convolutional neural network structure that takes the current state and the goal state as input and predicts both the next action and the heuristic value.

To the best of our knowledge, ours is the first implementation of a Transformer-based architecture such as BERT in the context of automated planning. While most of the works show slight variations or improvements [22, 36], versions for some specific contexts [37] or jargons [38], the work in [19] shows how these techniques can be useful even for artificial languages such as programming languages, proposing a BERT model for evaluating the semantic similarity between two portions of code. Their model is trained using publicly available open source code repositories from GitHub. In [39], a similar model for code summarization and generation is proposed. An interesting study about the capabilities of such models is shown in [40].

7. Conclusions and Future work

In this work, we trained a BERT model in the context of automated planning. While this type of deep learning architecture has completely changed the state-of-the art for several NLP tasks [5, 22], at the best of our knowledge this is the first implementation of such technique in the planning context. We designed an adaption of the masked language modeling task, called *planning language modeling* into which, using plans generated from three well-known planning domains (Logistics, Satellite and Blocksworld), this multi-domain model receives in input an action sequence with a percentage of missing actions and learns how to identify them from context. We have also presented a detailed account of the operations necessary to train such a model, in terms of pre-processing choices, vocabulary definition, data augmentation, etc.

In our experimental evaluation, we designed two tasks in order to verify if the model has the capability of understanding the inner working a planning domain: *Next Action Prediction* and *Previous Action Prediction*, into which given a sequence of actions the model has to predict the action that immediately follows or precedes the sequence. Our models obtains very promising results and we show that it has a high accuracy in predicting action that can be actually executed by an agent, before or after an action sequence.

However, our research is still preliminary. As future work, we will further investigate the capabilities of our model, including new domains and more complex configurations, with more objects and longer plans. Moreover, we will study how this pre-trained model can be exploited and adapted for other tasks such as goal recognition or heuristic prediction.

References

[1] C. W. Geib, M. Steedman, On natural language processing and plan recognition, in: M. M. Veloso (Ed.), IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, 2007, pp. 1612–1617.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: C. J. C. Burges, L. Bottou, Z. Ghahramani, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, 2013, pp. 3111–3119.

[3] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, 2015.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017, pp. 5998–6008.

[5] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 4171–4186.

[6] L. Putelli, A. E. Gerevini, A. Lavelli, R. Maroldi, I. Serina, Attention-based explanation in a deep learning model for classifying radiology reports, in: A. Tucker, P. H. Abreu, J. S. Cardoso, P. P. Rodrigues, D. Riaño (Eds.), Artificial Intelligence in Medicine - 19th International Conference on Artificial Intelligence in Medicine, AIME 2021, Virtual Event, June 15-18, 2021, Proceedings, volume 12721 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 367–372. URL: https://doi.org/10.1007/978-3-030-77211-6_42. doi:`10.1007/978-3-030-77211-6\_42`.

[7] L. Putelli, A. Gerevini, A. Lavelli, I. Serina, Applying self-interaction attention for extracting drug-drug interactions, in: M. Alviano, G. Greco, F. Scarcello (Eds.), AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings, volume 11946 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 445–460. URL: https://doi.org/10.1007/978-3-030-35166-3_32. doi:`10.1007/978-3-030-35166-3\_32`.

[8] A. Gerevini, A. Saetti, I. Serina, P. Toninelli, Fast planning in domains with derived predicates: An approach based on rule-action graphs and local search, in: M. M. Veloso, S. Kambhampati (Eds.), Proceedings, The Twentieth National Conference on Artificial

Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA, AAAI Press / The MIT Press, 2005, pp. 1157–1162. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-183.php.

[9] A. Gerevini, I. Serina, Planning as propositional CSP: from walksat to local search techniques for action graphs, Constraints An Int. J. 8 (2003) 389–413. URL: https://doi.org/10.1023/A:1025846120461. doi:`10.1023/A:1025846120461`.

[10] L. Bonassi, A. E. Gerevini, E. Scala, Planning with qualitative action-trajectory constraints in PDDL, in: L. D. Raedt (Ed.), Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022, ijcai.org, 2022, pp. 4606–4613. URL: https://doi.org/10.24963/ijcai.2022/639. doi:`10.24963/ijcai.2022/639`.

[11] P. Ferber, M. Helmert, J. Hoffmann, Neural network heuristics for classical planning: A study of hyperparameter space, in: ECAI 2020, IOS Press, 2020, pp. 2346–2353.

[12] M. Asai, A. Fukunaga, Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI Press, 2018, pp. 6094–6101.

[13] L. Amado, J. P. Aires, R. F. Pereira, M. C. Magnaguagno, R. Granada, F. Meneguzzi, Lstm-based goal recognition in latent space, CoRR abs/1808.05249 (2018).

[14] M. Chiari, A. E. Gerevini, L. Putelli, F. Percassi, I. Serina, Goal recognition as a deep learning task: the grnet approach, 2022. URL: https://arxiv.org/abs/2210.02377. doi:`10.48550/ARXIV.2210.02377`.

[15] W. Min, B. W. Mott, J. P. Rowe, B. Liu, J. C. Lester, Player Goal Recognition in Open-World Digital Games with Long Short-Term Memory Networks, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, IJCAI/AAAI Press, 2016, pp. 2590–2596.

[16] L. Putelli, A. E. Gerevini, A. Lavelli, T. Mehmood, I. Serina, On the behaviour of bert's attention for the classification of medical reports, in: C. Musto, R. Guidotti, A. Monreale, G. Semeraro (Eds.), Proceedings of the 3rd Italian Workshop on Explainable Artificial Intelligence co-located with 21th International Conference of the Italian Association for Artificial Intelligence(AIxIA 2022), Udine, Italy, November 28 - December 3, 2022, volume 3277 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 16–30. URL: http://ceur-ws.org/Vol-3277/paper2.pdf.

[17] M. Hoang, O. A. Bihorac, J. Rouces, Aspect-based sentiment analysis using bert, in: Proceedings of the 22nd nordic conference on computational linguistics, 2019, pp. 187–196.

[18] C. Qu, L. Yang, M. Qiu, W. B. Croft, Y. Zhang, M. Iyyer, Bert with history answer embedding for conversational question answering, in: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, 2019, pp. 1133–1136.

[19] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, M. Zhou, Codebert: A pre-trained model for programming and natural languages, in: T. Cohn, Y. He, Y. Liu (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020, volume EMNLP 2020 of *Findings of ACL*, Association for Computational Linguistics, 2020, pp. 1536–1547.

[20] J. Zhang, H. Zhang, C. Xia, L. Sun, Graph-bert: Only attention is needed for learning graph

representations, CoRR abs/2001.05140 (2020).

[21] H. Bao, L. Dong, S. Piao, F. Wei, Beit: BERT pre-training of image transformers, in: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, OpenReview.net, 2022.

[22] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, CoRR abs/1907.11692 (2019).

[23] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google's neural machine translation system: Bridging the gap between human and machine translation, arXiv preprint arXiv:1609.08144 (2016).

[24] J. W. Ratcliff, D. E. Metzener, Pattern-matching-the gestalt approach, Dr Dobbs Journal 13 (1988) 46.

[25] A. Thilagar, R. M. Frongillo, J. Finocchiaro, E. Goodwill, Consistent polyhedral surrogates for top-k classification and variants, in: K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, S. Sabato (Eds.), International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, volume 162 of *Proceedings of Machine Learning Research*, PMLR, 2022, pp. 21329–21359.

[26] R. Howey, D. Long, M. Fox, Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl, in: 16th IEEE International Conference on Tools with Artificial Intelligence, IEEE, 2004, pp. 294–301.

[27] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, Q. Liu, Tinybert: Distilling BERT for natural language understanding, in: T. Cohn, Y. He, Y. Liu (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020, volume EMNLP 2020 of *Findings of ACL*, Association for Computational Linguistics, 2020, pp. 4163–4174.

[28] F. A. Van-Horenbeke, A. Peer, Activity, Plan, and Goal Recognition: A Review, Frontiers Robotics AI 8 (2021) 643010.

[29] S. Hochreiter, J. Schmidhuber, Long Short-term Memory, Neural computation 9 (1997) 1735–80.

[30] M. Maynard, T. Duhamel, F. Kabanza, Cost-Based Goal Recognition Meets Deep Learning, Proceedings of the AAAI 2019 Workshop on Plan, Activity, and Intent Recognition, PAIR 2019 (2019).

[31] D. Borrajo, S. Gopalakrishnan, V. K. Potluru, Goal recognition via model-based and model-free techniques, Proceedings of the 1st Workshop on Planning for Financial Services at the Thirtieth International Conference on Automated Planning and Scheduling, FinPlan 2020 (2020).

[32] A.-M. Zou, Z.-G. Hou, S.-Y. Fu, M. Tan, Neural networks for mobile robot navigation: a survey, in: International Symposium on Neural Networks, Springer, 2006, pp. 1218–1226.

[33] S. H. Dezfoulian, D. Wu, I. S. Ahmad, A generalized neural network approach to mobile robot navigation and obstacle avoidance, in: Intelligent autonomous systems 12, Springer, 2013, pp. 25–42.

[34] M. K. Singh, D. R. Parhi, Path optimisation of a mobile robot using an artificial neural network controller, International Journal of Systems Science 42 (2011) 107–120.

[35] L. Chrestien, T. Pevny, A. Komenda, S. Edelkamp, Heuristic search planning with deep

neural networks using imitation, attention and curriculum learning, arXiv preprint arXiv:2112.01918 (2021).

[36] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, Association for Computational Linguistics, 2019, pp. 3980–3990.

[37] Y. Peng, S. Yan, Z. Lu, Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets, in: Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019), 2019, pp. 58–65.

[38] M. Polignano, P. Basile, M. De Gemmis, G. Semeraro, V. Basile, Alberto: Italian bert language understanding model for nlp challenging tasks based on tweets, in: 6th Italian Conference on Computational Linguistics, CLiC-it 2019, volume 2481, CEUR, 2019, pp. 1–6.

[39] W. U. Ahmad, S. Chakraborty, B. Ray, K. Chang, Unified pre-training for program understanding and generation, in: K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, Y. Zhou (Eds.), Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, Association for Computational Linguistics, 2021, pp. 2655–2668.

[40] R. Sharma, F. Chen, F. H. Fard, D. Lo, An exploratory study on code attention in BERT, CoRR abs/2204.10200 (2022).

AI and videogames: a “drosophila” for declarative methods

Denise Angilica[1], Giovambattista Ianni[1], Francesca A. Lisi[2] and Luca Pulina[3]

[1] *University of Calabria*
[2] *University of Bari Aldo Moro*
[3] *University of Sassari*

Abstract

Videogames and research in artificial intelligence (AI) techniques share a fruitful past of reciprocal knowledge exchange. On the one hand, videogames offer unsolved challenges to the academy: these challenges are as hard as those offered by what we can call “serious” applications, yet they can be faced in a controlled and reproducible setting. This characteristic makes videogames a kind of advanced “drosophila” for the researcher in AI: they are the ideal controlled ground in which to invent, experiment, and test new AI paradigms, methodologies and techniques. It is not uncommon to resort to simulated game environments as lesser expensive, yet comparably complex, digital twins. On the other hand, the videogame industry (VI) itself is exemplary of typical demands that AI research cannot meet yet. The VI looks for reduced development costs, better integration with AI tools and real-time performance. Especially, knowledge transfer between developers and AI tools must be as fast and smooth as possible: this is one of the reasons why the machine learning (ML) revolution had so far a controversial reception in the VI, in that ML provides black-box AI modules which are not easily “tunable” and configurable at will, and have non-negligible design-time costs. In order to overcome the above limits, one can consider declarative knowledge representation techniques (DKR). However, some of the shortcomings of DKR methods are fairly challenging to be addressed: performance, ease of use, integration, mining of reusable deductive knowledge are not up to the par yet. All these limitations prevent the real adoption of declarative paradigms in many highly-demanding applicative settings of which the videogame development field is an exemplary generalized testbed. Narrowing the above gaps is nontrivial challenge. In this paper we identify some of the key issues which we deem important for the research community and outline our current research progress.

Keywords
Declarative logic, Answer Set Programming, Knowledge Representation, Games and Videogames, Stream Reasoning,

1. Context and motivation

Videogames and the AI research. Many examples of the usage of declarative languages in the industrial videogame realm exists, starting from the pioneer F.E.A.R. game [1], which

---

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

denise.angilica@unical.it (D. Angilica); giovambattista.ianni@unical.it (G. Ianni); francesca.lisi@uniba.it (F. A. Lisi); lpulina@uniss.it (L. Pulina)
0000-0002-4069-978X (D. Angilica); 0000-0003-0534-6425 (G. Ianni); 0000-0001-5414-5844 (F. A. Lisi); 0000-0003-0258-3222 (L. Pulina)

CEUR Workshop Proceedings (CEUR-WS.org)

used STRIPS-based planning [2]. Other remarkable examples are the games Halo [3] and Black & White [4]. If we look at videogames from the basic research perspective, it must be noted the longstanding interest in using (video-)games as a controllable and reproducible setting in which to face open issues: one might cite the GDL [5], VGDL [6] and Ludocore [7] languages adopted for declaratively describing General Game Playing [8]. The Planning Domain Definition Language (PDDL) found natural usage in the videogame realm [9, 10, 11]; among its sister languages, we will herein focus particularly on Answer Set Programming (ASP), the known declarative paradigm with a tradition in modeling planning problems, robotics, computational biology as well as other industrial applications [12]. ASP does not come last in its experimental usage in videogames: it has been used to various extents, e.g., for declaratively generating level maps [13] and artificial architectural buildings [14]; it has been used as an alternative for specifying general game playing [15], for defining artificial players [16] in the Angry Birds AI competition [17], and for modelling resource production in real-time strategy games [18], to cite a few.

ASP specifications are composed of set of rules, hard and soft constraints, by means of which it is fairly easy to express qualitative and quantitative statements. In general, a set of input values F (called *facts*), describing the current state of the world, are fed together with an ASP specification S to a *solver*. Solvers in turn produce sets of outputs $AS(S \cup F)$ called *answer sets*. Answer sets contain the result of a decision-making process in terms of logical assertions which, depending on the application domain at hand, might encode actions to be made, employee shifts to be scheduled, protein sequences, and so on.

When evaluating S and F a traditional solver performs two consecutive steps: grounding (also called instantiation) and solving (also called answer set search). Instantiating a program consists in generating, rule by rule, substitutions of variables with constants, thus obtaining an equivalent propositional program.

Despite their potential advantages, the widespread adoption of methods like ASP in games is still prevented by a number of shortcomings, which one roughly categorize in *i)* integration issues and *ii)* performance issues.

Concerning the first, ASP-based solutions have been coupled with industrial applications using many schemes which were proposed in the last decade. These differ on how the so-called “procedural” side and “declarative” side are coupled, and on which of the sides is at the center of the development picture [19, 20, 21, 22].

However, during a videogame, decision-making processes are continuously repeated within the so called *game loop*. The game loop is, in a way, a sense-think-act cycle in disguise, where the *think* step might be occupied by a logic-based reasoner. This context makes the integration of reasoners/ASP solvers non-obvious: how to cope with fast changing sensor readings? how to accommodate long running reasoning tasks? What if a reasoning task is no longer needed because of a sudden change in the game scene? How to reuse at least part of the wasted computational effort?

The ThinkEngine [23], developed by our team, allows programmers to deal with AI modules inside the popular Unity game and industrial development engine [24] and is a good starting point for getting closer to the above goals. As for performance, ASP solvers greatly improved in the last years [25]. Effort has been made on incremental, online and stream computing of ASP inputs [26, 27, 28]. Fast, repeated reasoning tasks required by real-time applications seem, however, out of reach unless manual tuning and optimizations are introduced on a per application basis [29]. The ASP-based player appearing in the Angry Birds AI Competition [16] and other benchmarks on games show that the performance gap can be reduced with more research [30].

In this paper, we overview the key issues that prevent declarative paradigms being adopted in the VI. In section 2 we briefly illustrate our current research progress, introducing the ThinkEngine Asset; in section 3 we outline some selected open issues in the context of AI and videogames that we deem relevant; in section 4 we discuss the general advantages of videogames and declarative paradigms being coupled together; in section 5 we overview how this line of research could bridge videogames to academic research and to society in general.

When evaluating S and F a traditional solver performs two consecutive steps: grounding (also called instantiation) and solving (also called answer set search). Instantiating a program consists in generating, rule by rule, substitutions of variables with constants, thus obtaining an equivalent propositional program.

2. The ThinkEngine Asset

Figure 1, adapted from the description of our ThinkEngine tool [23] shows our proposal of an integration scheme between a reasoner based on declarative specifications and a game engine. A number of so-called *brains* are able to interact with a videogame scene.

An AI developer can use the ThinkEngine by first identifying objects and/or parts of the game logic that are to be connected to brains; *sensor readers* are wired from the game scene to brains, and *actuators* from brains to the game scene. A brain consists of a simple high-level specification that describes decision criteria: one can add rules, constraints, soft constraints and other declarative statement types. A brain reasoning task can be then triggered on specific events or on a cyclic basis. *Reactive brains* produce immediate modifications on the game, while *Planning brains* can be used to synthesize complex execution plans or, in principles, even arbitrarily reprogram reactive behaviors of artificial players.

In the case of our ThinkEngine tool, reasoning modules are implemented using declarative specifications written using ASP; an ASP solver executes such specifications, then its outputs are converted into actions or plans to be executed on the game world. Thinkengine is implemented respecting a paradigm-agnostic stance: a wiring infrastructure for connecting other solvers based on other declarative paradigms (PDDL, SMT or others) is provided. In the specific setting of Thinkengine, most of the interoperability burden is implemented within the EmbASP library [31].

The tool is available as a plugin for the known game development engine Unity [24]. Unity owns 45% of the market share as a game development engine: it is not however limited to the development of ludic products as it is appreciated also for the development of simulations and industrial applications.

From the software development perspective, the presence of brains based on declarative tools implies several potential benefits, like:

- declarative tools can be used for defining, in a very short time, different aspects of game

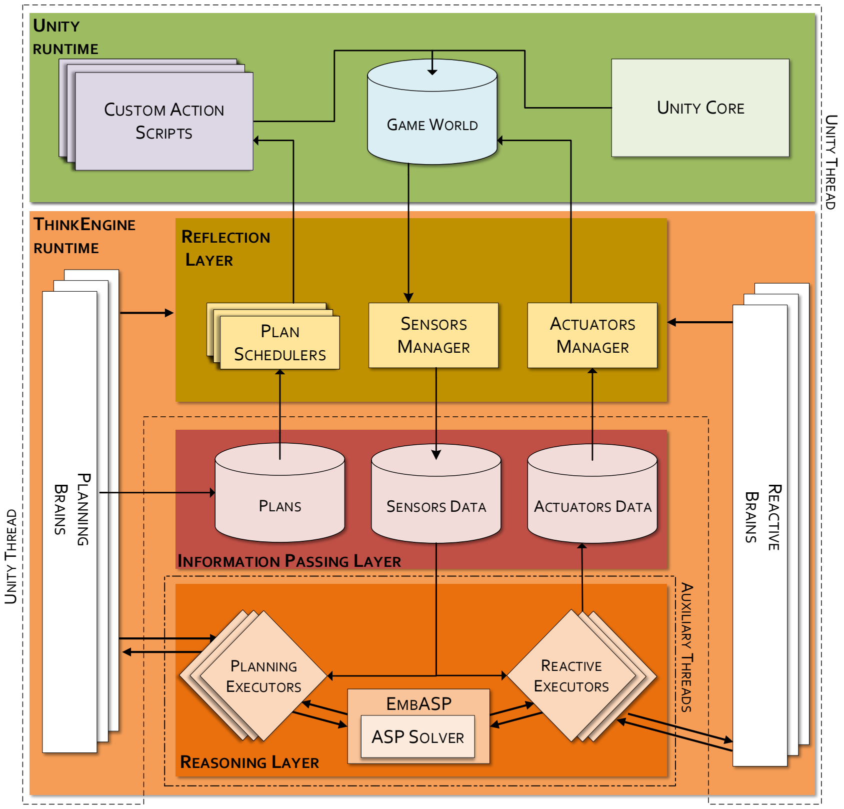


Figure 1: General run-time architecture of the ThinkEngine framework

AI, like: modelling of non-player characters (NPC) with spatio-temporal reasoning based AIs, automated game resource management, path planning, declarative (and not more "procedural") content generation, automated narrative generation based on qualitative descriptions, dynamic dialogue systems, etc.

- non programmers can participate in the implementation of decision making modules;
- the strong decoupling between declarative decision making and action execution makes much easier to separate and parallelize the development of both;
- the decoupling from the game engine itself allows standalone testing, isolated debugging and optimization.

However, the above advantages pair up with the known barriers to the adoption of declarative tools in demanding applications and in the videogame application settings in particular. Such barriers are mostly related to the actual declarativity of the formalism at hand, its evaluation efficiency, and the integration methodology with applications.

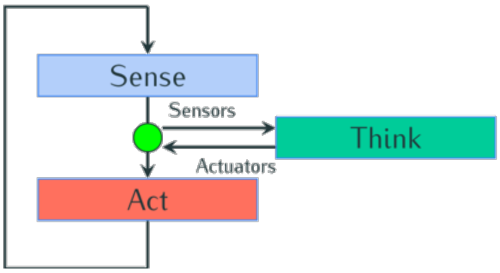


Figure 2: Hybrid Deliberative/Reactive (HDR) schema.

3. Some selected open issues

Integration. When one aims to integrate reasoning modules in industrial applications, and especially videogames, two main technical obstacles arise. One concerns the information flow and the coordination between reasoning tasks and other parts of the software; the second obstacle is related to the symbol grounding, i.e. the longstanding issue of abstracting raw data to a symbolic synthesis thereof (not to be confused with the close technical concept of "grounding" meant in the sense of "instantiation" of an ASP program).

Concerning the coordination of the reasoning tasks, recall that a videogame is typically executed by running the so called game-loop routine [32]. The game loop consists in a single-threaded repeated execution of update operations to the current game scene: within each step of this cycle, user input is processed, AI decision-making operations are made, and the game scene is modified accordingly. A physics simulation of the world and the general game logic is also taken into account when changing the game environment. The operations performed within the game loop strictly mimic the reactive sense-think-act cycle typical of agents and robotic systems [33], in which sensing, thinking and acting steps are repeatedly performed.

Similarly to real life applications, in videogames reactive AI modules coexist with non-reactive portions of the game logic: reactive modules are usually implemented using relatively fast techniques, such as behavior trees, finite state automata, rule sets or combinations thereof [34], and can run in the main game-loop. This type of module comes into play when quick "instinctive" decisions need to be taken by artificial characters: think, e.g., at programmed fight-or-flight responses when an artificial opponent is attacked.

On the other hand, strategic decision-making processes are likewise necessary. They are realized using long term, goal-oriented techniques, which often involve utility-based, planning, and simulation techniques. Longer running reasoning tasks are for instance required for common videogame AI modules such as path-planning, hierarchical task planning and resource management. As these require longer processing time, their execution within the default game-loop is troublesome. It is thus natural to generalize this reactive/non-reactive duality to the so-called hybrid deliberative/reactive paradigm [33] (HDR, Figure 2), in which fast "instinctive" behaviors are combined in a proper way with long term reasoning tasks such as plan generation.

This model has in turn been recently generalized both in economics and social science [35] and in the wider AI community [36], by proposing to model rational agents in terms of two interacting sides: *System 1*, the fast, instinctive, reactive, often non-symbolic subsystem; and *System 2*, a subsystem capable of better strategic decisions, yet requiring longer reasoning times and a proper symbolic preprocessing and abstraction layer. Both System 1 and System 2 decision-making activities can be deemed important and need to co-exist.

The interaction between both types of reasoning calls for the introduction of a scheme similar to the general HDR one, in which long-running reasoning tasks are made possible, but do not enforce heavy computational loads on the main game loop. The outcome of non-reactive tasks can be used to act directly on the game logic, or to re-program reactive behaviors of game characters. Although prototypical systems, like the ThinkEngine developed by our research group, offer a form of HDR scheme in which ASP reasoning tasks can be run asynchronously, many aspects are not properly systematized yet. Especially, there is no clear semantics on how deliberate modules can re-program reactive modules and execute plans. A good starting point in this respect can be hybrid ASP languages like ActHEX [21] that are meant to synthesize actions to be executed in an external environment.

Abstraction. Concerning the second integration issue, abstraction has been studied in ASP [37] especially aiming at reducing the number of symbols processed by solvers, thus reducing computing times yet preserving some form of semantic equivalence. An exploration of how fine-grained knowledge, such as raw spatial and temporal floating point data, can be systematically abstracted to a symbolic layer suitable for ASP reasoning has not been done yet. In this respect interval reasoning technique[38] look promising and could be beneficial on at least two aspects: performance-wise, abstraction can often make the difference in reaching the desired performance cutoff threshold. Abstraction is common practice in the videogame realm, such as when pixel-grained game maps are abstracted to a few strategic waypoints. As a second advantage, the availability of an automated abstraction layer can cut down design times, as this operation usually requires time-consuming manual design decisions.

Performance. Automated decision making in dynamic environments, where the world description is continuously and fastly changing, enforces strict reasoning time limits. Let us consider a decision-making reasoning task T, which can be either reactive or nonreactive. Specifically, if T is represented using ASP syntax, it is fed to an ASP solver composed of two separate computational steps: instantiation (also called "grounding", not to be confused with the related notion of symbol grounding), and resolution (also called "solving" in the proper sense). Both steps have their impact in performance, and especially grounding is crucial when big data flows and reactive tasks come into play. Rapid environment changes are typical of videogames, which are thus an ideal controlled environment for introducing *stop & restart techniques* capable of stopping no more valid reasoning tasks and resuming them under updated input data: this is needed, e.g., in case of events that trigger replanning. Also, no matter whether reactive or not, a time consuming "think" step may constitute a great performance bottleneck. The execution times of reasoning tasks can be greatly improved if *incremental techniques* are introduced. An incremental solver makes only quick differential updates to its internal state when some of the input data are changed. Videogames are exemplary also in this respect and could greatly benefit from AIs implemented using fast incremental solving techniques.

Whenever it is not desirable or possible to execute long, offline, reasoning tasks like in hybrid reactive/deliberative models, the introduction of incremental reasoning techniques becomes even more strategic. Much effort has been done recently in this direction: some ASP systems allow full incrementality, at the price of manually programming which and how parts of an ASP specifications should be (re-)evaluated [27], thus loosing some declarativity; other contributions enable a form of incrementality which is transparently performed "under-the-hood" from the perspective of the application designer, but are limited on the language expressivity [26]; the recent work of our group, based on the notion of "overgrounding" is also fully transparent to specification designers, yet it limits differential computations to the so called "grounding" stage only [30, 28, 39]. This choice limits performance improvements mostly to reactive tasks. None of the above work addressed so far the issue of computation restarts triggered by external input changes. We nonetheless foresee that our recently developed overgrounding techniques could be lifted to all the stages of the ASP solving pipeline, and can be generalized to introduce forms of computation restart techniques. New extensions should inherit the support of all the linguistic features of ASP, and keep the incremental evaluation out of the necessity of manual programming.

Whitebox AI procedures for content generation. Machine learning techniques are often used in the videogames industry for content generation of videogame levels that are aesthetically like human-authored examples (see, e.g., [40]). For this purpose, Generative Adversarial Networks (GANs) have been shown particularly effective on this task because they enable the creation of levels that are stylistically like human examples [41, 42]. However, it has been recently shown that GANs can often fail to produce videogame levels with playability criteria [43], making tiresome human intervention necessary to "repair" a machine-generated level. In order to automate this process, several techniques have been proposed, most of them focused on a generate-then-repair approach for first generating levels from models trained on human-authored examples and then repairing them with minimum cost edits to render them playable. Such techniques leverage on Mixed Integer Linear Programming (MILP) to encode the playability constraints for the repair procedure (see, e.g., [44]). It is appealing to investigate the effectiveness of symbolic methods, namely ASP, Pseudo-Boolean, and Satisfiability Modulo Theory (SMT), in combination with subsymbolic methods (GANs), to encode playability constraints for automated repair of generated content.

Declarative pattern mining. The detection of changes in the scenes or in the game characters is relevant e.g. for triggering NPC behavior changes, or for simply proposing different background music or elements. Pattern mining traditionally provides a bunch of techniques for the discovery of regularities in a data set, e.g. so-called frequent patterns are statistically significant regularities in a set of transactions. The classic problem of frequent pattern mining [45] concerns indeed the enumeration of all the patterns whose absolute support exceeds a user-defined minimum support threshold minSup.

A complementary problem is the discovery of statistically significant *differences* (or *contrast*) between two disjoint sets of transactions [46]. This requires the enumeration of all the patterns whose absolute support difference exceeds the user-defined minimum support threshold. The discovery of the pattern types such as contrast/difference patterns, as well as emerging patterns and change patterns, can be viewed as special cases of the same pattern mining problem, whose aim is to detect patterns relating two or more given datasets. These patterns have been shown to be a powerful method for constructing accurate classifiers, since they can describe emerging behavior with respect to a property of interest. One such feature is desirable also for the modeling of non-player characters.

An interesting direction of work is the investigation of some of these variants of the frequent pattern mining problem within a declarative framework, notably ASP. A driving feature of ASP solvers is indeed the possibility of enumerating all solutions, in the form of answer sets. This appears particularly suitable for pattern mining problems. Both contrast, emerging and change pattern mining fall into the broader category of constraint-based pattern mining [47], and can therefore be effectively encoded in declarative formalisms such as ASP, like in [48]. The problem specifications will naturally combine different constraints without having to devise new solving algorithms for specific mining tasks, thanks to the declarative nature of ASP. The challenges here are the identification of the most appropriate level of abstraction in the representation of a scene or a player, and the definition of change underlying the discovery process.

A second direction to pursue concerns the use of background knowledge related to the specific domain for the videogame in hand. Such prior domain knowledge might cover some aspects deemed of interest from either the methodological viewpoint or the applicative one. Particularly relevant in the videogame context is the spatial dimension. So, the background knowledge might come in the form of hierarchies of spatial objects, and patterns could then highlight spatial relations among the entities and offer descriptions of the scenes at multiple levels of granularity, as done in [49]. This direction of research would bridge the gap between the declarative approach and the object-based representations that are typical in the videogame context. Hybrid knowledge representation formalisms could be considered to the purpose, notably those formalisms which integrate ASP and DLs [50].

4. Impact for applicative research

The Videogame industry is an emblematic representative of a real-world application field where practical barriers prevent the wide adoption of AI methodologies, both symbolic/deductive and sub-symbolic/inductive. Elevated standards of efficiency and ease of use set a hard to reach cutoff threshold for the introduction of promising paradigms such as ASP in the above contexts. Today, ASP applications can be made performant, yet at the price of much "under-the-hood" optimization burden of which only ASP specialists are capable. This compromises ease of use and declarativity and is not welcome from the industrial perspective.

One might thus look at *redefined adherence to declarativity and a transparent efficiency*, confining back under-the-hood the technicalities of the ASP semantics yet ensuring efficient evaluation. Besides the widely accepted interest in explainability, a general goal for the AI community would be to regain the ability of *knowledge transfer* and *elaboration tolerance*: the first, not to be confused with transfer learning, is intended as achieving rapid transfer of knowledge from AI designers to AI modules. *Elaboration tolerance* [51] represents the capability of easily configuring and modifying the behavior of an AI module whenever new desiderata need to be added. It is thus important to offer whitebox AI components featuring “visible knobs” which can be used for instant tuning and configuration. This is in contrast with subsymbolic techniques that, although very useful in several respects, currently lack explainability, a fundamental requirement for achieving knowledge transfer and elaboration tolerance.

Besides these general goals, progress in videogames AI broadly overlaps with many applicative research fields. A punctual, yet incomplete list of possible technological applications of videogames AI includes:

Virtual environments and augmented reality. Serious games and virtual environment exploration tools are customarily built with game development tools, and share many requirements with videogames. In this respect declarative path-planning and declarative content generation are naturally appealing. Early research lines are moving in this direction [52].

Simulation and predictive maintenance. When realizing a simulation task in the automotive field, controlled content generation of images is of paramount importance in order to train ML algorithms. Concerning predictive maintenance, advances in integrating declarative AI in games could offer new digital twin-based solutions in order to test predictive maintenance techniques.

Robot guidance and robotic surgery. Among many industrial and research usages of the Unity engine it is worth mentioning robotic surgery [53], a field in which automated motion planning and spatial reasoning are seen with keen interest. In the same direction goes the research investigating the relationship between fine-grained motion planning and declarative methods [54].

5. Bridging videogames to research: a Drosophila for AI

There are countless research application fields where it is very expensive or difficult to let researchers access real data or physical resources and for which videogames represent the natural *digital twin*: traffic control, smart cities, autonomous driving, robotic surgery, digital forensics, are only a few examples. Basic KR research areas such as spatio-temporal reasoning, deontic and legal reasoning, qualitative physics, planning, to cite a few, can potentially have strong beneficial fall-out on applicative research like the above. This impact can be boosted if reproducible controlled environments were available and made of the same complexity of real contexts. Enabling easy-to-wire declarative modules in one of the most used game and simulation engines goes in this direction, also considering the role of ASP as a middle-ware for implementing many of the abovementioned KR formalisms.

An incomplete list of hot research topics can include:

Digital Forensics. Evidence Analysis, a crucial phase in Digital Forensics, ranges from the analysis of fragmented incomplete knowledge, to the reconstruction and aggregation of complex scenarios involving time, uncertainty, causality and alternate possibilities. The Scientific Investigation experts usually proceed by relying on their experience and intuition as no established methodology exists today. Games can be useful and powerful means for supporting digital forensics. They can provide digital twins where hypotheses, made using temporal reasoning and qualitative physics, can be evaluated in a comparative way. The automated discovery of regularities in crime scenes provides precious hints to investigators [55, 56, 57]: when empowered with reasoning modules, games show an added value with respect to other simulation environments. In this respect it is worth mentioning the COST Action "Digital forensics: evidence analysis via intelligent systems and practices" (DigForASP). DigForASP has set up an international network for exploring the potential of AI techniques (in particular, KR and Automated Reasoning) in the Digital Forensics field, and for creating synergies between these two fields.

Agent interaction, machine ethics. Among AI research areas possibly benefitting from the availability of a tool like ThinkEngine playing the role of a controlled environment for experimentation, one may consider using games for simulating the interaction between agents, e.g., in order to study their compliance with the ethical guidelines for a Trustworthy AI. In this context, a major obstacle to the operationalization and implementation of these requirements in AI systems is just the lack of datasets and benchmarks. Games might compensate for this lack, might help to get better insights into the dynamics of a corresponding real-world system, and can assess the practical challenges of building such a system. ASP, and more generally non-monotonic reasoning, is particularly suitable for dealing with ethical principles as testified by several proposals [58].

Stream reasoning. Besides the technical and scientific interest of connecting videogames to research, it is worth mentioning the very close connection with stream reasoning. The potential impact on applications like smart cities, power grid management, urban traffic control, where fast decision-making on big and dynamic data flows has immediate implications on the green economy, is fairly straightforward. All these applications fall under the stream reasoning umbrella where declarative techniques find a natural application. Stream reasoning also shares with videogames the need of performing AI reasoning under fast-paced input event flows.

Intrinsic value of videogames. The videogame market, the highest selling category in the entertainment industry [59], has seen a high increase in demand in the COVID-19 era, with new job openings and an even larger portfolio of games. Videogames, no matter if played with family, with friends, online in massive groups, or alone, were the place in which many of us found a form of sociality and consolation in the difficult times of COVID-19. One should recall that videogames are not just the place many scientists dream to leverage their ideas on, but also a way of living engaging stories, worlds and atmospheres. The realization of these latter necessitates innovative AI tools.

6. Acknowledgments

This article has been partially supported by the Italian MIUR Ministry and the Presidency of the Council of Ministers under the project "Declarative Reasoning over Streams" under the

“PRIN” 2017 call (Project 2017M9C25L_001), and includes work from COST Action 17124 “Digital forensics: evidence analysis via intelligent systems and practices (DigForASP)”, supported by COST (European Cooperation in Science and Technology).

References

[1] J. Orkin, Three states and a plan: the AI of F.E.A.R., in: Game Developers Conference, 2006.

[2] N. Nilsson, STRIPS planning systems, Artificial Intelligence: A New Synthesis (1998) 373–400.

[3] Halo, 2001. URL: https://www.xbox.com/en-US/games/halo.

[4] Black & White, 2001. URL: https://www.ea.com/games/black-and-white.

[5] M. R. Genesereth, N. Love, B. Pell, General game playing: Overview of the AAAI competition, AI Mag. 26 (2005) 62–72.

[6] T. Schaul, A video game description language for model-based or interactive learning, in: CIG, IEEE, 2013, pp. 1–8.

[7] A. M. Smith, M. J. Nelson, M. Mateas, LUDOCORE: A logical game engine for modeling videogames, in: CIG, IEEE, 2010, pp. 91–98.

[8] D. P. Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, General video game AI: competition, challenges and opportunities, in: AAAI, 2016, pp. 4335–4337.

[9] O. Bartheye, E. Jacopin, A real-time PDDL-based planning component for video games, in: AIIDE, The AAAI Press, 2009.

[10] J. Robertson, R. M. Young, The general mediation engine, Experimental AI in Games: Papers from the 2014 AIIDE Workshop. AAAI Technical Report WS-14-16 10 (2014) 65–66.

[11] J. Robertson, R. M. Young, Automated gameplay generation from declarative world representations, in: AIIDE, AAAI Press, 2015, pp. 72–78.

[12] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Mag. 37 (2016) 53–68.

[13] A. M. Smith, M. Mateas, Answer set programming for procedural content generation: A design space approach, IEEE Trans. Comput. Intell. AI Games 3 (2011) 187–200.

[14] L. van Aanholt, R. Bidarra, Declarative procedural generation of architecture with semantic architectural profiles, in: CoG, IEEE, 2020, pp. 351–358.

[15] M. Thielscher, Answer set programming for single-player games in general game playing, in: ICLP, volume 5649 of *LNCS*, Springer, 2009, pp. 327–341.

[16] F. Calimeri, M. Fink, S. Germano, A. Humenberger, G. Ianni, C. Redl, D. Stepanova, A. Tucci, A. Wimmer, Angry-hex: An artificial player for angry birds based on declarative knowledge bases, IEEE Trans. Comput. Intell. AI Games 8 (2016) 128–139.

[17] J. Renz, X. Ge, S. Gould, P. Zhang, The angry birds AI competition, AI Mag. 36 (2015) 85–87.

[18] M. Stanescu, M. Certický, Predicting opponent’s production in real-time strategy games with answer set programming, IEEE Trans. Comput. Intell. AI Games 8 (2016) 89–94.

[19] F. Calimeri, D. Fuscà, S. Germano, S. Perri, J. Zangari, Fostering the use of declarative

formalisms for real-world applications: The EmbASP framework, New Gener. Comput. 37 (2019) 29–65.

[20] O. Febbraro, N. Leone, G. Grasso, F. Ricca, JASP: A framework for integrating answer set programming with Java, in: KR, AAAI Press, 2012.

[21] M. Fink, S. Germano, G. Ianni, C. Redl, P. Schüller, Acthex: Implementing HEX programs with action atoms, in: LPNMR, volume 8148, Springer, 2013, pp. 317–322.

[22] J. Rath, C. Redl, Integrating answer set programming with object-oriented languages, in: PADL, 2017, pp. 50–67.

[23] D. Angilica, G. Ianni, F. Pacenza, Declarative AI design in unity using answer set programming, in: CoG, IEEE, 2022, pp. 417–424.

[24] Unity 3D game engine, Last accessed: Jan 2023. URL: https://unity3d.com/unity.

[25] M. Gebser, M. Maratea, F. Ricca, The seventh answer set programming competition: Design and results, Theory Pract. Log. Program. 20 (2020) 176–204.

[26] H. Beck, T. Eiter, C. Folie, Ticker: A system for incremental ASP-based stream reasoning, Theory Pract. Log. Program. 17 (2017) 744–763.

[27] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory Pract. Log. Program. 19 (2019) 27–82.

[28] G. Ianni, F. Pacenza, J. Zangari, Incremental maintenance of overgrounded logic programs with tailored simplifications, Theory Pract. Log. Program. 20 (2020) 719–734.

[29] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, Künstliche Intell. (2018) 165–176.

[30] F. Calimeri, G. Ianni, F. Pacenza, S. Perri, J. Zangari, Incremental answer set programming with overgrounding, Theory Pract. Log. Program. 19 (2019) 957–973.

[31] F. Calimeri, S. Germano, G. Ianni, F. Pacenza, S. Perri, J. Zangari, Integrating rule-based AI tools into mainstream game development, in: RuleML+RR, volume 11092, Springer, 2018, pp. 310–317.

[32] M. Joselli, et al., An adaptative game loop architecture with automatic distribution of tasks between CPU and GPU, Comput. Entertain. 7 (2009) 50:1–50:15.

[33] R. R. Murphy, Introduction to AI Robotics, MIT Press, 2000.

[34] I. Millington, Artificial Intelligence for Games, Third Edition, CRC Press, 2019.

[35] D. Kahneman, Thinking, fast and slow, Farrar, Straus and Giroux. New York, 2011.

[36] G. Booch, F. Fabiano, L. Horesh, K. Kate, J. Lenchner, N. Linck, A. Loreggia, K. Murugesan, N. Mattei, F. Rossi, B. Srivastava, Thinking fast and slow in AI, in: AAAI, AAAI Press, 2021, pp. 15042–15046.

[37] Z. G. Saribatur, T. Eiter, Omission-based abstraction for answer set programs, in: KR, AAAI Press, 2018, pp. 42–51.

[38] J. F. Allen, G. Ferguson, Actions and events in interval temporal logic, J. Log. Comput. 4 (1994) 531–579.

[39] F. Calimeri, G. Ianni, F. Pacenza, S. Perri, J. Zangari, ASP-based multi-shot reasoning via DLV2 with incremental grounding, in: PPDP, ACM, 2022, pp. 2:1–2:9.

[40] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, J. Togelius, Procedural content generation via machine learning (PCGML), IEEE Trans. Games 10 (2018) 257–270.

[41] E. Giacomello, P. L. Lanzi, D. Loiacono, DOOM level generation using generative adversarial

networks, in: GEM, IEEE, 2018, pp. 316–323.
[42] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. M. Smith, S. Risi, Evolving Mario levels in the latent space of a deep convolutional generative adversarial network, in: GECCO, ACM, 2018, pp. 221–228.
[43] R. R. Torrado, A. Khalifa, M. C. Green, N. Justesen, S. Risi, J. Togelius, Bootstrapping conditional GANs for video game level generation, in: CoG, IEEE, 2020, pp. 41–48.
[44] H. Zhang, M. C. Fontaine, A. K. Hoover, J. Togelius, B. Dilkina, S. Nikolaidis, Video game level repair via mixed integer linear programming, in: L. Lelis, D. Thue (Eds.), AIIDE, AAAI Press, 2020, pp. 151–158.
[45] C. C. Aggarwal, J. Han (Eds.), Frequent Pattern Mining, Springer, 2014.
[46] G. Dong, J. Bailey (Eds.), Contrast Data Mining: Concepts, Algorithms, and Applications, CRC Press, 2013.
[47] S. Nijssen, A. Zimmermann, Constraint-based pattern mining, in: C. C. Aggarwal, J. Han (Eds.), Frequent Pattern Mining, Springer, 2014, pp. 147–163.
[48] F. A. Lisi, G. Sterlicchio, A declarative approach to contrast pattern mining, in: AIxIA. To appear., 2023.
[49] F. A. Lisi, D. Malerba, Inducing multi-level association rules from multiple relations, Mach. Learn. 55 (2004) 175–210.
[50] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, H. Tompits, Combining answer set programming with description logics for the semantic web, Artif. Intell. 172 (2008) 1495–1539.
[51] J. McCarthy, Elaboration tolerance, https://stanford.io/3UIlwsM, 1998.
[52] A. Brännström, J. C. Nieves, A framework for developing interactive intelligent systems in unity, in: 10th EMAS workshop at AAMAS, 2022.
[53] A. Segato, M. D. Marzo, S. Zucchelli, S. Galvan, R. Secoli, E. D. Momi, Inverse reinforcement learning intra-operative path planning for steerable needle, IEEE Trans. Biomed. Eng. 69 (2022) 1995–2005.
[54] Y. Izmirlioglu, E. Erdem, Reasoning about cardinal directions between 3-dimensional extended objects using answer set programming, Theory Pract. Log. Program. 20 (2020) 942–957.
[55] F. A. Lisi, Combining knowledge representation and machine learning in forensics, Applications of AI to Forensics 2020 (AI2Forensics 2020) (2020) 1.
[56] F. A. Lisi, G. Sterlicchio, Declarative pattern mining in digital forensics: Preliminary results, in: CILC, volume 3204, CEUR-WS.org, 2022, pp. 232–246.
[57] F. A. Lisi, G. Sterlicchio, Mining sequences in phone recordings with answer set programming, in: P. Bruno, F. Calimeri, F. Cauteruccio, M. Maratea, G. Terracina, M. Vallati (Eds.), Joint Proceedings of (HYDRA 2022) and (RCRA 2022) at (LPNMR 2022), volume 3281 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 34–50. URL: http://ceur-ws.org/Vol-3281/paper4.pdf.
[58] A. Dyoub, S. Costantini, F. A. Lisi, Logic programming and machine ethics, in: ICLP, volume 325, 2020, pp. 6–17.
[59] Newzoo global games market report 2022, https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2022-free-version, Last accessed: Jan 2023.

Verification of Neural Networks: Challenges and Perspectives in the AIDOaRt Project

Romina Eramo[1], Tiziana Fanni[2], Dario Guidotti[3,*], Laura Pandolfo[3], Luca Pulina[3] and Katiuscia Zedda[2]

[1]*University of Teramo, Via R. Balzarini 1, Teramo, 64100, Italy*
[2]*Abinsula, Viale Umberto 64, Sassari, Italy*
[3]*University of Sassari, Piazza Università 21, Sassari, 07100, Italy*

Abstract

Neural networks are increasingly being used for dealing with complex real-world applications. Despite their success, there are still important open issues such as their limited application in safety and security-critical contexts, wherein assurance about networks' behavior must be provided. The development of reliable neural networks for safety-critical contexts is one of the topics investigated in the AIDOaRt Project, a 3 years long H2020-ECSEL European project focusing on Artificial Intelligence augmented automation supporting modeling, coding, testing, monitoring, and continuous development of Cyber-Physical Systems. In this paper, we present an interesting safety-critical use case – related to the automotive domain – from the AIDOaRt project. In addition, we outline the challenges we are facing in bridging the gap between the scalability of state-of-the-art verification methodologies and the complexity of the neural networks best suited for the task of interest.

Keywords

Trustworthy AI, Neural Networks, Formal Verification, Automotive

1. Introduction

Neural networks (NNs) are one of the most investigated and widely used techniques in Machine Learning (ML) and have found successful application in many different domains across computer science [1, 2]. However, despite their success, they still find limited application in safety and security-critical contexts, wherein assurance about networks' behavior must be provided. As an example, a specific concern about the reliability of NNs is their vulnerability to adversarial attacks [3], which are small variations of the inputs which cause unforeseeable changes in the behavior of the neural network. This kind of vulnerability is both a safety (e.g., rain droplets on the cameras causing a misclassification of a vehicle) and a security issue (e.g., minor modification applied to a vehicle or road sign by a malevolent actor [4]) as we show in Figure 1. Automated

---

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

*Corresponding author.

reramo@unite.it (R. Eramo); tiziana.fanni@abinsula.com (T. Fanni); dguidotti@uniss.it (D. Guidotti); lpandolfo@uniss.it (L. Pandolfo); lpulina@uniss.it (L. Pulina); katiuscia.zedda@abinsula.com (K. Zedda)
0000-0002-3572-5875 (R. Eramo); 0000-0002-4301-6497 (T. Fanni); 0000-0001-8284-5266 (D. Guidotti); 0000-0002-5785-5638 (L. Pandolfo); 0000-0003-0258-3222 (L. Pulina)

CEUR Workshop Proceedings (CEUR-WS.org)

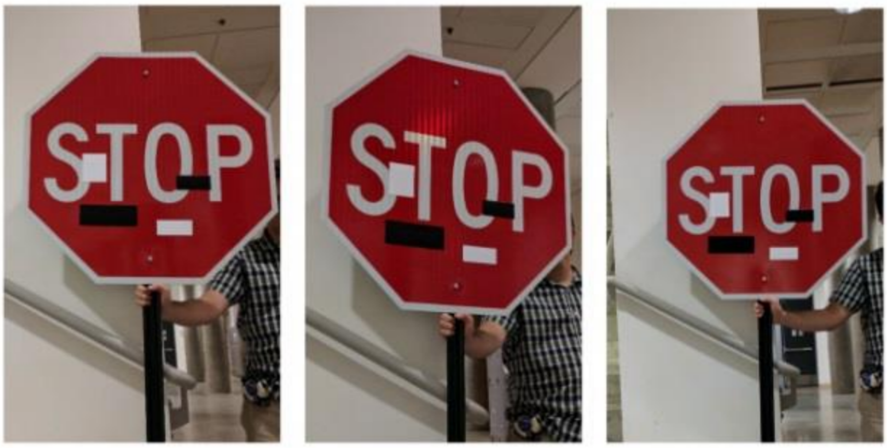


Figure 1: Example of a real-life adversarial attack from [4]: small graffiti on a stop signal can easily mislead a neural network to misclassify it as a speed limit.

formal verification — see, e.g., [5] for a survey — provides an effective answer to the problem of establishing the correctness of the behavior of a NN and opens the path to their adoption in safety and security-critical applications [6].

Indeed, the development of reliable NNs for safety-critical contexts is one of the topics investigated by the team of the University of Sassari in the AIDOaRt[1] Project [7, 8], a Key Digital Technologies Joint Undertaking (KTD JU) project started on April 2021, involving 32 organizations grouped in national clusters from 7 different countries. The overall idea of the project is to efficiently support the system engineering life-cycle, from requirements to testing and deployment, including software design, coding, and verification. In particular, AIDOaRt focuses on supporting the continuous development of Cyber-Physical Systems (CPSs) via Artificial Intelligence (AI)-augmented automation. The AIDOaRt framework will enable the observation and analysis of collected data from both runtime and design-time in order to provide dedicated AI-augmented solutions, that will then be validated in concrete industrial cases involving complex CPSs in domains such as railway, automotive, restaurants, etc.

The use of AI in general, and ML in particular, is a key aspect of the AIDOaRt project. Different AI-augmented capabilities will be provided, and to this intent, the AIDOaRt AI-augmented Toolkit will be designed and developed in the context of the project in order to support additional capabilities related to different CPS development tasks. The dissemination of such techniques in a regulated industry can rapidly enable systems to decide and act in a more and more automated manner, sometimes even without direct human control. As a consequence, this demands for a responsible approach to ensure a safe and beneficial use of AI technologies. This approach has to consider both the implications of (co)decision making by machines and related ethical issues. Within AIDOaRt, one of the main challenges is ensuring that systems are designed responsibly. The integrated core framework and AI-based solutions will be applied and validated in practice in the context of the different AIDOaRt project use cases. At the end of the project, an industrial

---

[1] AI-augmented automation supporting modelling, coding, testing, monitoring and continuous development in Cyber-Physical Systems

uptake of AIDOaRt results via developing real complex and large-scale CPSs is expected.

In this paper, we briefly present an interesting safety-critical use case from the automotive domain and the challenges we are facing in bridging the gap between the scalability of state-of-the-art verification methodologies and the complexity of the NNs best suited for the task of interest. Section 2 briefly presents the relevant background and state of the art. In Section 3 we present the Abinsula Use Case and, finally, in Section 4 we present the challenges we are facing and our ideas on how to overcome them.

2. Background

2.1. Neural Networks

Neural Networks are machine learning models composed of interconnected computing units called neurons. In a feed-forward network, the neurons are arranged in disjointed layers and each layer is connected only with the following ones forming a direct acyclic graph (DAG). Every layer of a NN performs specific computations on the inputs it receives from the previous layers or, for the first layer of the network, from the outside. The number of neurons in a layer or, more in general, in a network is directly correlated with the complexity of the computations carried out by it and it is often used to formally define such complexity. As an example, two of the most commonly used kind of layers are *linear* and *activation* layers: linear layers apply an affine transformation $f(x) = Ax + b$ to their input tensor x, whereas activation layers apply element-wise a specific (usually non-linear) function $f(x) = \sigma(x)$ to it. Another kind of frequently used layer is the convolutional one, which has been especially successful when used in architectures applied in computer vision tasks. The idea behind the convolutional layer is to analyze the input images using a set of perception filters (also known as feature maps) which can be learned from the data.

2.2. Verification

Formal verification aims to guarantee that NNs satisfy stated input-output relations and in the last decade several verification methodologies have been proposed for different specifications and architectures [9, 10, 11, 12, 5]. In general, verification methodologies can be divided into two categories: *complete* and *incomplete*. Complete verification algorithms [13, 14, 15, 16, 17] leverage technologies like Satisfiability Modulo Theories (SMT) and Mixed Integer Linear Programming (MILP) solvers or methods like branch and bound (BnB) to provide a definitive answer on the compliance of the NN to the property of interest, at the price of a greatly increased computational complexity. Conversely, incomplete verification algorithms [18, 19, 20, 21, 22] leverage methods like abstract interpretation and bound propagation to provide an answer subject to a degree of uncertainty: in particular, when an incomplete method is unable to certify that a network satisfies the property of interest it is in doubt if the property is violated or if the precision of the verification methodology was not good enough. Whereas when an incomplete algorithm produces a positive result regarding the satisfaction of the property, it is certain that the network behavior is compliant with it. This degree of uncertainty is in exchange for a greatly reduced computational complexity. It should be noted that, in recent times, many verification tools began to combine complete and incomplete algorithms to enhance the scalability of the firsts and the precision of the seconds.

3. Safety critical systems in the automotive domain using disruption technology

The technologies developed within the AIDOaRt project will be applied and evaluated on different 15 industrial use cases (UCs), including the one related to the automotive domain proposed by Abinsula Srl, a company based in Sassari (Italy) that provides innovative Information and Communication Technology (ICT) solutions worldwide [2].

Modern cars are connected systems that acquire inputs from the environment, thus they can be considered as CPSs. With the increment of sources of information and data, safety represents even more a critical objective and new challenges in the development process are arising. This is especially true where several stakeholders, such as hardware specialists, software developers and system designers have to work together with safety engineers to ensure a reliable and safe system. In this context, the emergence of standards, such as ISO 26262 and ISO 16505, has helped the automotive industry to focus on practice to address safety in a systematic and consistent way.

This opened the path to new research for guaranteeing safety in the automotive context. The use of AI and ML is on the rise in order to enhance the automated verification of systems applied in real safety-critical applications. However, if it is true that AI is now recognized as innovative technology, it is far from being applied in real safety-critical applications due to the lack of methodologies, for example for the predictability of the system in domains such as the automotive one.

The use case brought by Abinsula aims at enhancing the human interaction and driving experience, proposing an electronic rear-view mirror that gets data from a set of cameras and provides the rear image on a screen. Image processing and sensor fusion, more in general the AI application to sensors data processing, allow for increasing the informative content of the rear environment image providing alerts or suggestions for a safer and effective drive. The use case involves the usage of four cameras that capture 1920x1080 images. Data related to the camera include an up to 60 FPS stream and the camera ID. The exploitation of NNs in image processing is meant to replace what the human brain does in terms of recognition and processing while driving. Therefore, a fundamental step to implement a virtual rear mirror able to perform as the human brain, while guaranteeing safety, is the formal verification of the adopted NNs to ensure the predictability of the system.

The main expected improvements are related to the introduction of AI techniques both in the modeling and testing phase of the system development life-cycle. In particular, in the modeling and implementation phase, the use of AI techniques will be employed for the verification of Deep Neural Network (DNN) models/implementations.

---

[2] https://abinsula.com/

4. Challenges and Perspectives

Taking into account the scope of the Abinsula UC, we analyzed the state of the art regarding NNs applied in computer vision and we identified a first set of architectures that could be near to optimal for the applications of interest. In particular, we identified the YOLO [23] network architecture, which is one of the most popular learning models used for object detection. This architecture is able to process videos at 45 frames per second (fps), whereas a more optimized version manages to reach 150 fps. Moreover, it greatly outperforms the contemporary models leveraging classification and learns better generalizable representations of objects.

Regarding the verification tools, we considered the winner of the 2nd International Verification of Neural Networks Competition [24] (VNN-COMP'21) ALPHA-BETA-CROWN [25], which is a NN verifier based on an efficient bound propagation algorithm and a branch and bound methodology. It also leverages dedicated hardware (i.e., GPUs) in order to scale to relatively large convolutional networks and supports a wide range of architectures. Naturally, we also considered the runner-ups of the same competition, which present comparable performances to ALPHA-BETA-CROWN. However, through a first comparison between the benchmarks used during the VNN-COMP'21 (11 convolutional layers) and the YOLO architecture (109 convolutional layers in its last version), it immediately appears clear that the scalability of the current state-of-the-art verification tools is far from being enough to support our ideal architecture.

Therefore we identified different strategies to bridge the gap between our ideal architecture and the effective scalability of the state-of-the-art verification tools. In particular, we intend to investigate pruning and/or quantization as means to produce smaller network models which should enable verification without a significant loss in performances [26]. Furthermore, we believe that it could be possible to focus on subsets of the network architecture which are of particular interest for the task at hand and, at the same time, small enough to be feasible to verify [27]. Finally, we are evaluating how to enhance existing verification tools and methodologies to support network architectures similar to YOLO.

Acknowledgments

This research work has received funding through the AIDOaRt project from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, and Spain. The research of Luca Pulina has been partially funded by the University of Sassari, research fund FAR2020PULINAL.

References

[1] E. Giunchiglia, A. Nemchenko, M. van der Schaar, RNN-SURV: A deep recurrent model for survival analysis, in: V. Kurková, Y. Manolopoulos, B. Hammer, L. S. Iliadis, I. Maglogiannis (Eds.), Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III, volume 11141 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 23–32.

[2] Y. LeCun, Y. Bengio, G. E. Hinton, Deep learning, Nat. 521 (2015) 436–444.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: Y. Bengio, Y. LeCun (Eds.), 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.

[4] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, Robust physical-world attacks on deep learning visual classification, in: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 1625–1634.

[5] F. Leofante, N. Narodytska, L. Pulina, A. Tacchella, Automated verification of neural networks: Advances, challenges and perspectives, CoRR abs/1805.09938 (2018).

[6] S. Demarchi, D. Guidotti, A. Pitto, A. Tacchella, Formal verification of neural networks: A case study about adaptive cruise control, in: I. A. Hameed, A. Hasan, S. A. Alaliyat (Eds.), Proceedings of the 36th ECMS International Conference on Modelling and Simulation, ECMS 2022, Ålesund, Norway, May 30 - June 3, 2022, European Council for Modeling and Simulation, 2022, pp. 310–316.

[7] H. Bruneliere, V. Muttillo, R. Eramo, L. Berardinelli, A. Gomez, A. Bagnato, A. Sadovykh, A. Cicchetti, Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber–physical systems, Microprocessors and Microsystems 94 (2022) 104672.

[8] R. Eramo, V. Muttillo, L. Berardinelli, H. Bruneliere, A. Gomez, A. Bagnato, A. Sadovykh, A. Cicchetti, Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber-physical systems, in: 2021 24th Euromicro Conference on Digital System Design (DSD), IEEE, 2021, pp. 303–310.

[9] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, X. Yi, A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability, Comput. Sci. Rev. 37 (2020) 100270.

[10] D. Guidotti, Enhancing neural networks through formal verification, in: M. Alviano, G. Greco, M. Maratea, F. Scarcello (Eds.), Discussion and Doctoral Consortium papers of AI*IA 2019 - 18th International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, volume 2495 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 107–112.

[11] D. Guidotti, Verification and repair of neural networks, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 15714–15715.

[12] D. Guidotti, Safety analysis of deep neural networks, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 4887–4888.

[13] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljic, D. L. Dill, M. J. Kochenderfer, C. W. Barrett, The marabou framework for verification and analysis of deep neural networks, in: Computer Aided Verification -

31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I, volume 11561 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 443–452.

[14] P. Henriksen, A. R. Lomuscio, Efficient neural network verification via adaptive refinement and adversarial search, in: G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang (Eds.), ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020), volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 2513–2520.

[15] P. Henriksen, A. Lomuscio, DEEPSPLIT: an efficient splitting method for neural network verification via indirect effect analysis, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 2549–2555.

[16] R. Bunel, J. Lu, I. Turkaslan, P. H. S. Torr, P. Kohli, M. P. Kumar, Branch and bound for piecewise linear neural network verification, J. Mach. Learn. Res. 21 (2020) 42:1–42:39.

[17] A. D. Palma, R. Bunel, A. Desmaison, K. Dvijotham, P. Kohli, P. H. S. Torr, M. P. Kumar, Improved branch and bound for neural network verification via lagrangian decomposition, CoRR abs/2104.06718 (2021).

[18] D. Guidotti, L. Pulina, A. Tacchella, pynever: A framework for learning and verification of neural networks, in: Z. Hou, V. Ganesh (Eds.), Automated Technology for Verification and Analysis - 19th International Symposium, ATVA 2021, Gold Coast, QLD, Australia, October 18-22, 2021, Proceedings, volume 12971 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 357–363.

[19] L. Pulina, A. Tacchella, Never: a tool for artificial neural networks verification, Ann. Math. Artif. Intell. 62 (2011) 403–425.

[20] G. Singh, T. Gehr, M. Püschel, M. T. Vechev, An abstract domain for certifying neural networks, Proc. ACM Program. Lang. 3 (2019) 41:1–41:30.

[21] H. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, T. T. Johnson, NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems, in: S. K. Lahiri, C. Wang (Eds.), Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I, volume 12224 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 3–17.

[22] H. Zhang, T. Weng, P. Chen, C. Hsieh, L. Daniel, Efficient neural network robustness certification with general activation functions, in: S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018, pp. 4944–4953.

[23] J. Redmon, S. K. Divvala, R. B. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society, 2016, pp. 779–788.

[24] S. Bak, C. Liu, T. T. Johnson, The second international verification of neural networks competition (VNN-COMP 2021): Summary and results, CoRR abs/2109.00498 (2021).

[25] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C. Hsieh, J. Z. Kolter, Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification, in: M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 29909–29921.

[26] D. Guidotti, F. Leofante, L. Pulina, A. Tacchella, Verification of neural networks: Enhancing scalability through pruning, in: G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang (Eds.), ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020), volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 2505–2512.

[27] D. Guidotti, F. Leofante, L. Pulina, A. Tacchella, Verification and repair of neural networks: A progress report on convolutional models, in: M. Alviano, G. Greco, F. Scarcello (Eds.), AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings, volume 11946 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 405–417.

Verification of Neural Networks for Safety and Security-critical Domains

Dario Guidotti[1,*]

[1]University of Sassari, Piazza Università 21, Sassari, 07100, Italy

Abstract

In recent times, machine learning has gained incredible traction in the artificial intelligence community, and neural networks in particular have been leveraged in many successful applications originating from various domains. However, it is hard to provide any formal guarantee on the behavior of this kind of models, and therefore their reliability is still in doubt, especially concerning their deployment in safety and security-critical applications. In this work, we will present our contributions on the topic of formal verification, which recently emerged as a promising solution to address some of these problems. We will also present two novel use cases originating from real-world applications we are working on and the related challenges and perspectives.

Keywords

Trustworthy AI, Neural Networks, Formal Verification

1. Introduction

In the last few decades, artificial intelligence (AI) has become increasingly popular as it has been employed in many different applications with a great degree of success [1, 2]. Among those domains, safety and security-critical ones are often of particular interest both to the research and industry communities. As an example, the automotive domain has seen increasingly substantial investments in financial and time resources from both. However, industrial applications in this kind of domain require formal guarantees on the behavior of the algorithms and models used, since thorough regulations have been established by national and international authorities regarding the employment of AI in these areas. Unfortunately, the currently most popular AI technologies, that is, machine learning (ML) and neural networks, provide only statistical guarantees on their behavior which, while may be enough for many applications of interests, still fall short when human lives, or large amounts of money, are at stake. Furthermore, neural networks have been proven to be subject to reliability issues like adversarial attacks, which are small variations of the inputs which cause unforeseeable changes in their behavior. It is easy to see how this kind of vulnerability can be both a safety and a security issue with a couple of examples from the automotive domain: burned pixels in the cameras may cause the misclassification of a curve as a straight road [3], or small graffiti written by a malevolent actor on

---

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

[*] Corresponding author.

✉ dguidotti@uniss.it (D. Guidotti)
0000-0001-8284-5266 (D. Guidotti)

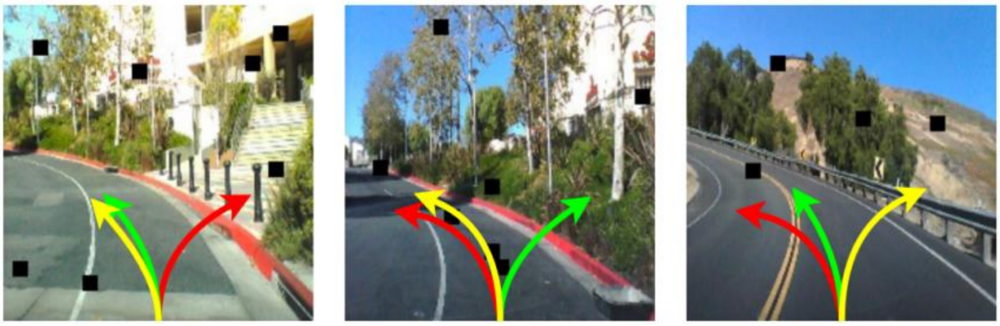


Figure 1: Example of an adversarial attack in the automotive domain from [3]: a few burned pixels in a camera can easily mislead a neural network to misclassify a left turn as a right one.

a stop sign may make the model recognize it as a speed limit [4]. As a consequence, part of the AI research community focused on developing methodologies to formally evaluate the correctness of the behavior of neural networks and, as witnessed in [5] automated formal verification seems to provide a path to their adoption in safety and security-critical applications [6]. In this paper, we will present some of our contributions to this last topic, their prospective applications, and the open challenges we are facing in the scope of some new use cases originating from realistic domains. Section 2 present the relevant background and state of the art for verification of neural networks. In Section 3 we briefly explain our contribution in the domain of verification of neural networks, whereas in Section 4 we present the new use cases which are guiding our recent research efforts. Finally, in Section 5 we summarize the challenges we are facing with the new use cases and our perspectives on how to overcome them.

2. Background

2.1. Neural Networks

Neural networks are machine learning models inspired by the structure of biological neural networks: more specifically they can be seen as directed graphs composed of interconnected computing units called neurons. In a feed-forward network, the neurons are arranged in disjointed layers and each layer is connected only with the following ones forming a direct acyclic graph (DAG). Every layer of a neural network performs specific computations on the inputs it receives from the previous layers or, for the first layer of the network, from the outside. The computational complexity of a layer is directly correlated with both the number of neurons in such layer and the specific operation carried out by them. It clearly follows that the same quantities can be used to estimate the complexity of a whole network and, indeed, they are often used to formally define such complexity.

Some examples of particularly important layers are the *linear* and *activation* layers, which are normally the building blocks used to assemble even the most basic neural networks. In particular, linear layers apply an affine transformation to their input tensor, whereas activation layers apply a specific (usually non-linear) function element-wise. Even with only these two kinds of layers, it is possible to build neural networks which can, in principle, approximate any continuous function for inputs within a specific range [7].

Finally, another significant layer is the convolutional one, which has been especially successful when used in architectures applied to computer vision tasks. The idea behind the convolutional layer is to analyze the input images using a set of perception filters (also known as feature maps) which can be learned from the data.

2.2. Verification

The aim of formal verification is to provide guarantees regarding the behavior of neural networks: more specifically verification methodologies try to prove if specific neural networks satisfy stated input-output relations. For this purpose, in the last decade, several verification methodologies have been proposed for different specifications and architectures [8, 9, 10, 11, 5].

A common distinction between verification methodologies is the one between *complete* and *incomplete* algorithms: complete verification algorithms [12, 13, 14, 15, 16] leverage techniques like Branch and Bound, Satisfiability Modulo Theories (SMT) and Mixed Integer Linear Programming (MILP) to provide a final answer on the compliance of the neural network to the property of interest, at the price of a greatly increased computational complexity. On the other hand, incomplete verification methodologies [17, 18, 19, 20, 21] are typically based on methods like abstract interpretation and bound propagation and, as consequence, they provide an answer subject to a degree of uncertainty. That is, when an incomplete method is unable to certify that a network satisfies the property of interest, it is still unsure if the property is truly violated or if the precision of the approximation used was not good enough. However, when an incomplete algorithm certifies that the network of interest satisfies a certain property, it is certain that the network behavior is compliant with it. While the use of over-approximate methods causes this kind of uncertainty it also produces algorithms whose computational complexity is greatly reduced.

While this categorization is still commonly used, it should be noted that, in recent times, many verification tools began to combine complete and incomplete algorithms to get, as much as possible, the best of both worlds.

3. Contributions

In [22] we investigated if pruning, a methodology developed by the learning community whose main application until now has been to reduce the dimension of neural networks so that they can be deployed on hardware with low memory resources, could be leveraged to produce a training pipeline providing networks easier to verify for the existing verification methodologies. In our experiments, we considered two different pruning methodologies: one based on the reduction of the number of connections between the neurons of a neural network, called weight pruning, and the other based on the removal of whole neurons, called neuron pruning. By leveraging these two pruning algorithms to reduce different networks, we managed to produce models with performances comparable to the original ones but, at the same time, much easier to verify for existing verification tools. In particular, the networks on which neuron pruning was applied were the easiest to verify: we believe this is because the elimination of whole neurons from the networks eliminates a corresponding number of non-linearities, which are, in general, the main culprit behind the high computational complexity of verification algorithms.

In [17] we developed the first version of our tool pyNeVer which provides capabilities for the training, pruning, and verification of neural networks. The main contribution of this work was a novel algorithm based on over-approximation for the verification of neural networks: in particular, the novelty of this algorithm is in the mechanism to dynamically control the coarseness of the over-approximation down to the single neuron level and a specific eager heuristic for choosing the neurons on which to apply the more precise abstraction. The resulting algorithm resulted to be comparable with the state of the art and even outperforms a similar one on some of our experimental benchmarks. Further experiments were done in [6] on models generated from an automotive case study.

Finally, in [23] we focused on trying to repair neural networks subject to adversarial examples so that they became more robust to particular adversarial examples. To do so we leveraged a MILP solver to find a configuration of the network parameters which made it resistant to the adversarials of interest. However, most of the networks were too complex to be directly modified with such a methodology. Therefore, we selected a specific subset of the network architecture and replaced it with a less complex model which then was repaired. In our experimental evaluation we confirmed that, while this methodology did not manage to make the model resistant to adversarial attack in general, it did make it more robust to specific adversarial examples.

4. Use Cases

Currently, we are working on two new use cases (UCs) from safety and security-critical domains. In the first one, the task is developing reliable neural controllers for self-piloting a drone during the course of different activities. The second one consists in developing reliable neural networks for object detection and recognition tasks in the automotive domain. This second use case is part of the effort to develop reliable neural networks for safety-critical contexts in the scope of the AIDOaRt Project, a 3 years long H2020-ECSEL European project focusing on Artificial Intelligence augmented automation supporting modeling, coding, testing, monitoring, and continuous development of Cyber-Physical Systems.

4.1. Drone Control

Since the scope of the project was to develop various neural controllers for different tasks, we focused on building a modular setup for the training of neural controllers in simulated environments using well-maintained and stable resources. In particular, we leveraged:

- Gym[1]: an open-source python library providing a standard API for communication between reinforcement learning algorithms and environments.
- Stable Baselıne3[2]: an open source training framework providing scripts for training and evaluating RL agents using standard state-of-the-art algorithms.

---

[1] https://github.com/openai/gym

[2] https://github.com/DLR-RM/stable-baselines3

Figure 2: The Bitcraze Crazyflie 2.1 drone considered in our first use case.

- PyBullet[3]: an open source physics simulator for robotics and reinforcement learning.
- gym-pybullet-drones[4]: an open source Gym-stile environment supporting the definition of various learning tasks on the control of one or more quadcopters.

Using these open-source resources we greatly simplified the complexity of our setup and we were able to directly train the network of interest in the environment corresponding to our case study with the chosen state-of-the-art RL algorithm. In Figure 2 we show the quadcopter model of choice, which was the default one proposed in gym-pybullet-drones and which we intend to use to test our neural controllers in real environments.

Beyond the scope of the project, we also intend to leverage the presented setup to produce novel benchmarks for the verification of neural networks. The motivation for doing so is that, while the verification community has been prolific in developing novel methodologies, very few general benchmarks have been proposed, among which the most popular is still the ACAS XU benchmark [24], released in 2017. Furthermore, drone control is a task relevant to modern applications and, at the same time, the neural networks used in this kind of control task are usually small enough for the existing verification methodologies to be successfully applied.

4.2. Automotive

One of the use cases on which the technologies developed within the AIDOaRt project will be evaluated is the one related to the automotive domain proposed by Abinsula Srl, a company based in Sassari (Italy) that provides innovative ICT solutions worldwide [5].

Given the wealth of interconnected sensors and software supports modern cars can be easily seen as cyber-physical systems. This implies that the various stakeholders in the development

---

[3] https://pybullet.org/

[4] https://github.com/utiasDSL/gym-pybullet-drones

[5] https://abinsula.com/

process of such systems have to work together to ensure that the corresponding product is safe and reliable. While the methodologies to guarantee the safety and reliability of the hardware and much of the software are quite well established, the same cannot be said for the components based on artificial intelligence and machine learning, whose formal certification is still an open challenge for both the industrial and research communities.

The use case brought by Abinsula aims at enhancing the human interaction and driving experience, proposing an electronic rear-view mirror that gets data from a set of cameras and provides the rear image on a screen. The application of AI and ML to sensor data processing allows for increasing the informative content of the rear environment image providing, for example, alerts or suggestions for a safer and more effective drive. The use case involves the usage of four cameras that capture 1920x1080 images in an up to 60 fps stream and also provide the relevant camera ID. The idea behind the usage of neural networks in this kind of image processing is to reduce the effort the human user needs to apply for recognizing and processing the relevant objects in the image they are seeing. As consequence, a fundamental step to implement a virtual rear mirror able to reliably assist the human user is the formal verification of the adopted neural networks to ensure the predictability of the system.

5. Challenges and Perspectives

Given the scope of the drone control UC we first analyzed the state of the art regarding the learning of neural controllers for the kind of tasks we had in mind and we selected a promising reinforcement learning algorithm, that is, the Soft Actor-Critic. The main advantage of this algorithm is that it uses two different neural networks: the actor one models the controller, whereas the critic is used to provide an estimation of how good the actor is. At high level this allows us to keep down the complexity of the controller (*i.e.*, the actor) that therefore will be easier to verify. Even so, clearly more complex tasks will require more complex architectures and therefore we will need to enhance both the scalability and the generality of pyNeVer. Furthermore, we are interested in developing methodologies which would allow to leverage the results obtained by an unsuccessful verification of the neural controllers to enhance their training process so that they became compliant with the property of interest.

Regarding the Abinsula UC, first we focused on the state of the art regarding neural networks applied in computer vision tasks and we identified an initial set of architectures that could be viable for the applications of interest. In particular, we identified the YOLO [25] network architecture, which is one of the most popular learning models used for object detection. This architecture is able to process videos at 45 frames per second (fps), whereas a more optimized version manages to reach 150 fps. Furthermore, it greatly outperforms the contemporary models leveraging classification and learns better generalizable representations of objects. We also surveyed the state of the art of the verification tools, and we focused on the winner of the 2nd International Verification of Neural Networks Competition [26] (VNN-COMP'21) alpha-beta-crown [27], which is a neural network verifier based on an efficient bound propagation algorithm and a branch and bound methodology. It also leverages dedicated hardware (i.e., GPUs) in order to scale to relatively large convolutional networks and supports a wide range of architectures. Of course, we also considered the runner-ups of the same competition, which present comparable performances to alpha-beta-crown. Nevertheless, from a first comparison between the benchmarks used during the VNN-COMP'21 (11 convolutional layers) and the YOLO architecture (109 convolutional layers in its last version), it would seem that the current state-of-the-art verification tools are far from reaching the scalability needed to support our preferred architectures. As consequence we focused on identifying various strategies to bridge this gap between our favored architectures and the effective scalability of verification tools. To do so we believe that it could be possible to focus our verification on subsets of the network architecture which are of particular interest for the task at hand and, at the same time, small enough to be feasible to verify, similarly to what was done in [23]. We also intend to investigate pruning and/or quantization as means to produce smaller network models which should enable verification without a significant loss in performances, as we managed to do with less complex models in [22]. Finally, as for the drone control UC, we are evaluating how to enhance existing verification tools and methodologies to support network architectures similar to YOLO.

Acknowledgments

This research work has received funding through the AIDOaRt project from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union's Horizon 2020 research and innovation program and Sweden, Austria, Czech Republic, Finland, France, Italy, and Spain.

The research on reliable drone control has been supported by Fondazione di Sardegna, project "Tecniche e strumenti per la verifica di reti neurali".

References

[1] E. Giunchiglia, A. Nemchenko, M. van der Schaar, RNN-SURV: A deep recurrent model for survival analysis, in: V. Kurková, Y. Manolopoulos, B. Hammer, L. S. Iliadis, I. Maglogiannis (Eds.), Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III, volume 11141 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 23–32.

[2] Y. LeCun, Y. Bengio, G. E. Hinton, Deep learning, Nat. 521 (2015) 436–444.

[3] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: automated whitebox testing of deep learning systems, Commun. ACM 62 (2019) 137–145.

[4] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, Robust physical-world attacks on deep learning visual classification, in: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 1625–1634.

[5] F. Leofante, N. Narodytska, L. Pulina, A. Tacchella, Automated verification of neural networks: Advances, challenges and perspectives, CoRR abs/1805.09938 (2018).

[6] S. Demarchi, D. Guidotti, A. Pitto, A. Tacchella, Formal verification of neural networks: A

case study about adaptive cruise control, in: I. A. Hameed, A. Hasan, S. A. Alaliyat (Eds.), Proceedings of the 36th ECMS International Conference on Modelling and Simulation, ECMS 2022, Ålesund, Norway, May 30 - June 3, 2022, European Council for Modeling and Simulation, 2022, pp. 310–316.

[7] K. Hornik, M. B. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[8] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, X. Yi, A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability, Comput. Sci. Rev. 37 (2020) 100270.

[9] D. Guidotti, Enhancing neural networks through formal verification, in: M. Alviano, G. Greco, M. Maratea, F. Scarcello (Eds.), Discussion and Doctoral Consortium papers of AI*IA 2019 - 18th International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, volume 2495 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 107–112.

[10] D. Guidotti, Verification and repair of neural networks, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 15714–15715.

[11] D. Guidotti, Safety analysis of deep neural networks, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 4887–4888.

[12] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljic, D. L. Dill, M. J. Kochenderfer, C. W. Barrett, The marabou framework for verification and analysis of deep neural networks, in: Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I, volume 11561 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 443–452.

[13] P. Henriksen, A. R. Lomuscio, Efficient neural network verification via adaptive refinement and adversarial search, in: G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang (Eds.), ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020), volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 2513–2520.

[14] P. Henriksen, A. Lomuscio, DEEPSPLIT: an efficient splitting method for neural network verification via indirect effect analysis, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 2549–2555.

[15] R. Bunel, J. Lu, I. Turkaslan, P. H. S. Torr, P. Kohli, M. P. Kumar, Branch and bound for piecewise linear neural network verification, J. Mach. Learn. Res. 21 (2020) 42:1–42:39.

[16] A. D. Palma, R. Bunel, A. Desmaison, K. Dvijotham, P. Kohli, P. H. S. Torr, M. P. Kumar, Improved branch and bound for neural network verification via lagrangian decomposition, CoRR abs/2104.06718 (2021).

[17] D. Guidotti, L. Pulina, A. Tacchella, pynever: A framework for learning and verification of neural networks, in: Z. Hou, V. Ganesh (Eds.), Automated Technology for Verification and Analysis - 19th International Symposium, ATVA 2021, Gold Coast, QLD, Australia, October 18-22, 2021, Proceedings, volume 12971 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 357–363.

[18] L. Pulina, A. Tacchella, Never: a tool for artificial neural networks verification, Ann. Math. Artif. Intell. 62 (2011) 403–425.

[19] G. Singh, T. Gehr, M. Püschel, M. T. Vechev, An abstract domain for certifying neural networks, Proc. ACM Program. Lang. 3 (2019) 41:1–41:30.

[20] H. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, T. T. Johnson, NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems, in: S. K. Lahiri, C. Wang (Eds.), Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I, volume 12224 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 3–17.

[21] H. Zhang, T. Weng, P. Chen, C. Hsieh, L. Daniel, Efficient neural network robustness certification with general activation functions, in: S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018, pp. 4944–4953.

[22] D. Guidotti, F. Leofante, L. Pulina, A. Tacchella, Verification of neural networks: Enhancing scalability through pruning, in: G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, J. Lang (Eds.), ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020), volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 2505–2512.

[23] D. Guidotti, F. Leofante, L. Pulina, A. Tacchella, Verification and repair of neural networks: A progress report on convolutional models, in: M. Alviano, G. Greco, F. Scarcello (Eds.), AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings, volume 11946 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 405–417.

[24] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer, Reluplex: An efficient SMT solver for verifying deep neural networks, in: R. Majumdar, V. Kuncak (Eds.), Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I, volume 10426 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 97–117.

[25] J. Redmon, S. K. Divvala, R. B. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society, 2016, pp. 779–788.

[26] S. Bak, C. Liu, T. T. Johnson, The second international verification of neural networks competition (VNN-COMP 2021): Summary and results, CoRR abs/2109.00498 (2021).

[27] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C. Hsieh, J. Z. Kolter, Beta-crown: Efficient

bound propagation with per-neuron split constraints for neural network robustness verification, in: M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 29909–29921.

Building the Semantic Portal of Italian Divagrafie

Laura Pandolfo*, Lucia Cardone, Luisa Cutzu, Beatrice Seligardi and Giulia Simi

DUMAS, University of Sassari, via Roma 151, Sassari, Italy

Abstract

In this paper, we present the preliminary research activities on building a semantic digital archive for publishing heterogeneous data about a literary corpus provided by Italian actresses, known as Divagrafie. This corpus represents a unique collection of cultural data for scholars in Film and Literature Studies through which they can analyze the phenomenology, characteristics, and historical evolution of the writings produced by Italian actresses. In this paper, we present the vision behind the development of the semantic digital archive and explore the potential applications and its expected impacts in the research community and society.

Keywords

Semantic Web, Digital Archive, Film and Literature Studies

1. Context and Motivation

The availability of cultural heritage data has rapidly boosted the emerging new research area in Digital Humanities (DH) [1] where an increasing number of scholars are dealing with new computational methods developed and applied to literary corpus for solving problems in humanities and social sciences. Recently, DH research has shifted the focus to providing the user with integrated tools for solving research problems in interactive ways [2]. One of the characteristics of this kind of systems is to use computational methods for solving humanities research questions by using large datasets and applying on them the so-called "distant reading" approach [3], namely a set of different techniques, such as novel statistical methods, sentiment analysis, topic modeling and network analysis. The main benefit of using the distant reading approach comes from the fact that it opens new horizons for computational literary studies, without the hard effort required by classical humanistic computer research, which often needs rigorous coding and document annotations. However, this approach has several critical issues, as identified by Ciotti [4, 5]. One shortcoming lies in the fact that most of the applied computational methods are independent from the context, while humanities and literary data need to be heavily contextualized. Another important critical point concerns the interpretation of text, which is usually an intentional process. Statistical computational methods are hardly able to detect the

---

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

*Corresponding author.

lpandolfo@uniss.it (L. Pandolfo); cardone@uniss.it (L. Cardone); lcutzu@uniss.it (L. Cutzu); bseligardi@uniss.it (B. Seligardi); gsimi@uniss.it (G. Simi)

0000-0002-5785-5638 (L. Pandolfo); 0000-0003-0770-6324 (L. Cardone); 0000-0002-9400-0866 (B. Seligardi); 0000-0002-2206-2945 (G. Simi)

CEUR Workshop Proceedings (CEUR-WS.org)

true interpretation since the meaning of a word is usually determined by the attribution of sense and meaning by the author and by the reader.

Semantic Web (SW) [6] technologies and Linked Data [7] can overcome these issues by enriching the distant reading approach of new methods able to capture the semantic nature of literary texts [8]. Therefore, cultural heritage has become an active area of application of SW, where cultural content and metadata are available openly for research and public use based on collections in museums, libraries, archives, and media organizations [2, 9]. In the last two decades, large amount of data has been aggregated in huge national and international portals, libraries and repositories such as Europeana [1] by forming a significant part of DBpedia [2] and Wikidata [3]. The concept of ontology [10] plays a central role, since it is commonly used as a sort of schema capturing knowledge about a specific domain via providing relevant concepts and relations between them. Currently,several examples of ontology-based digital archives and libraries in the humanities have been reported [11, 12, 13, 14, 15, 16].

This paper introduces our ongoing research work on developing an ontology-based archive named *WOmen Writing around the camera* (WOW) which will collect semantic data about relevant writings produced by Italian actresses ("Divagrafie") focusing on the dynamics of exchange between writing, acting performance and the construction of the star image. This research idea builds-on and extends the "Drawing a Map of Italian Actresses in Writing" (DaMA) funded by PRIN 2017 [4] which aims at investigating the extension, phenomenology, characteristics, and historical evolution of the writings produced by Italian actresses. The main goal is to build a semantic portal containing different resource materials related to Italian actresses' visual and self-representation history able to bring to light and interrogating together different kinds of documents (mainly photographs and texts) with a defined set of investigation methods with the help of DH tools, particularly those based on SW technologies and distant reading methods. The paper is organized as follows. Section 2 describes our current research progress and outlines the planned phases related to the development of the WOW semantic archive, while Section 3 concludes the paper by discussing the potential applications and expected impacts resulting from this line of research both for the academic point of view but also society in general.

2. The WOW Semantic Archive

The idea to develop the WOW Semantic Archive stems from the in-progress research project DaMA which aims at investigating the extension, phenomenology, characteristics, and historical evolution of the writings produced by Italian actresses, focusing in particular on the dynamics of exchange between writing, acting performance and construction of the star image. Defined as Divagrafie [17], such writings represent a multifaceted and interesting corpus of texts traditionally overlooked by the academic community. The important contribution to the understanding of stardom provided by this kind of self-narratives or fiction writings has been only partially acknowledged within the field of star studies; similarly, the field is also lacking

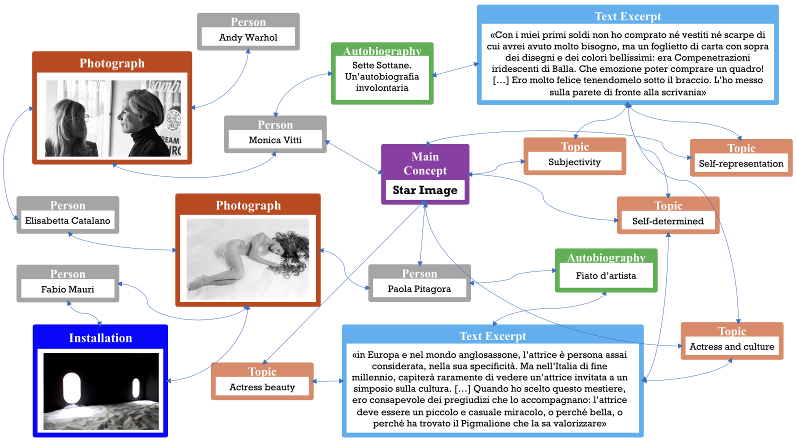
---

[1] https://www.europeana.eu/en
[2] https://www.dbpedia.org/
[3] https://www.wikidata.org/wiki/Wikidata:Main_Page
[4] https://www.damadivagrafie.org/

an in-depth analysis dedicated to the relationship between the writings of a specific actress and the construction of her own star image. This ongoing project is revealing a still unknown territory and has so far classified a wide number of texts, including 80 autobiographies written by 47 actresses. This literary corpus – which we are intended to focus on – represents a unique testing ground for a convergent methodology that applies DH methods, SW technologies and the most recent advances in the fields of Film and Media Studies, Literature and Gender Studies. These volumes have been analyzed so far by the DaMA team with a *close reading* approach focusing on paradigmatic examples and eliciting a variety of recurring topics [18, 19, 20], with a particular focus on the interconnections with the perspective of performing and actors' studies, stardom and celebrities studies and by using also the videographic analysis tools.

By following and expanding the multidisciplinary approach already explored by the DaMA research, our current and future research work focuses on increasing the actual corpus and applying on it SW standards and DH tools. Below the main phases of our methodological approach:

1. **Recognition of DaMA project's outcomes and integration of new materials**. All the materials collected and analyzed by the DAMA research project team will be subjected to a further and extended analysis in order to provide a first taxonomy describing the corpus. During this phase, new documents and resources will be integrated into the original corpus, in particular we we will focus on the retrieval of materials preserved in the Elisabetta Catalano archive (Rome), the Museo di Fotografia Contemporanea - MuFoCo archive (Cinisello Balsamo, MI), the Museo del Cinema archive (Turin), the Cinemazero archive, the Luisa Di Gaetano archive (Rome), Archivia - Casa Internazionale delle Donne (Rome). In addition, we intend to study cinema and cultural periodicals in order to verify the circulation and the incidence of the work of female photographers in the national press.
2. **Conceptualization and formalization**. The main goal of this activity is to formalize the acquired knowledge in terms of concepts and relations with respect to specific ontological schema and models detected in this phase. In this domain, the use of ontologies provides a range of benefits for the users, e.g., in searching and browsing by concept rather than string-based only. For example, the photographs that Elisabetta Catalano has taken of a variety of actors and actresses, writers and directors can be put in relationships with all the excerpts in the Italian actresses' autobiographies in which the relations with the cultural field strongly emerge as a symptom of overcoming the stereotype of actresses as a pure "not thinking" beautiful and fashioned body. As we can see in Figure 1, Elisabetta Catalano's photograph of Monica Vitti talking to Andy Warhol can be intertwined with passages in her autobiographies related to art as a means of self-expression and with other several entities and concepts in the corpus.
3. **Semantic annotations and analysis of the corpus**. In this phase, ontological schema and models will be used to guide the semantic annotations of all the corpus materials (texts and images). During this phase, the corpus will be enriched with metadata describing, for example, references that link the content to specific concepts. Moreover, a variety of literary analysis based on distant reading approaches will be applied to the corpus materials. In particular, we will explore typical computational linguistics methods based



«Con i miei primi soldi non ho comprato né vestiti né scarpe di cui avrei avuto molto bisogno, ma un foglietto di carta con sopra dei disegni e dei colori bellissimi: era Compenetrazioni iridescenti di Balla. Che emozione poter comprare un quadro! […] Ero molto felice tenendomelo sotto il braccio. L'ho messo sulla parete di fronte alla scrivania»

«in Europa e nel mondo anglosassone, l'attrice è persona assai considerata, nella sua specificità. Ma nell'Italia di fine millennio, capiterà raramente di vedere un'attrice invitata a un simposio sulla cultura. […] Quando ho scelto questo mestiere, ero consapevole dei pregiudizi che lo accompagnano: l'attrice deve essere un piccolo e casuale miracolo, o perché bella, o perché ha trovato il Pigmalione che la sa valorizzare»

Figure 1: Example of entities and relationships in the examined corpus.

on Natural Language Processing techniques in order to identify recursive narrative patterns as well as similar characteristics of the texts which will allow the recognition of an attantial typology specifically linked to Divagrafie.

4. **Development of the semantic archive**. The main goal of this last phase will be the development of the WOW semantic archive. It will provide the researchers with a set of powerful and efficient tools that can be used to query, analyze and study the data in order to make possible intertwine and connect different data. The interdisciplinary approach will be made visible and easily traceable by a digital user interface able to link different objects and produce new sets of knowledge. This will be particularly useful for scholars and researchers of different fields, such as History of Cinema, History of Photography, Visual Studies, Literature, Contemporary History and so on. The semantic archive will be the hub through which not only find materials to be linked to other studies and researches, but also to find new ways to analyze and study them with innovative approaches. In this phase, we intend to investigate some automatic techniques for ontology population, such as those presented in [21, 22]. All the features of the semantic archive will be integrated into an easy-to-use graphical interface which will provide the visualization of the data in various formats, such as graphs, interactive maps, timelines, facets, etc.

3. Potential Applications and Expected Impacts

The development of the WOW Semantic Archive could have significant potential applications within the national and international research communities involved in the process. In fact, the archive will make accessible a wide corpus of materials related to Italian Actresses which crosses a wide range of studies also in the international academic communities: Film Studies, Photography Studies, Women's Studies and Literature Studies, and so on. To our knowledge, this is the first time that such a corpus is being created following the unambiguous, rigorous, consistent and well-documented practices provided by the DH approach. Moreover, this is the first time that computational methods and SW technologies will be applied to this kind of document resources. The data integration feature provided by SW technologies will allow the connection between data contained in the WOW Semantic Archive to data included to other international archives, such as the European Film Gateway [5], by increasing the international impact.

The semantic archive will be useful also for upper-secondary teachers and students, curators of film-related events, film/photography/literature enthusiasts, and in general for whoever needs to retrieve the data and the researches included in it. In addition to the ambition to fill a gap in research and reflection on the area of study, namely the analysis of the relationship between women, photography and cinema from a feminist perspective, which in many Western countries has already been addressed, this work has another ambition, more complex and therefore even more important and stimulating: this research work aims at contributing to the debate and to the cultural framework of Italy, in which gender issues, women's emancipation, feminist heritage, women's art are still not sufficiently considered as a fundamental part of the cultural heritage of the nation, and therefore as a topic in school and university curriculum. As for its social impact, it will contribute to the growing interest in those cultural dynamics which are affected and/or shaped by gender issues as well as promoting a deeper social awareness of the cultural role of women in Italian society.

As for its methodology, the actual collaboration among different scientific areas, such as DH, Computer Science, Cinema Studies, Photography, Visual Studies, Gender Studies, Literary Theory, will constitute an example of an integrated and multi-layered methodology, offering itself as a possible benchmark for potential future projects. As for its objectives and expected potential impacts, this work is in line with the research goals and targets defined by the National Research Program (PNR) 2021-2027 and by the "Cultural Heritage" specific intervention area of Horizon Europe. In particular, it will mainly impact the cluster *"Humanistic culture, creativity, social transformation, society of inclusion"* of the PNR, specifically the sub-category *"Digital preservation and conservation of cultural heritage".* In fact, according to this sub-category research line, the priority should be given to the implementation of effective semantic modeling technologies, also through the construction of ontologies wherever necessary, which allow an effective aggregation of different information levels and types of data, in order to avoid redundancies or lack of data. The research should also include a shift from traditional databases to SW databases, in order to achieve interoperability between resources and set up the field for the transition to Big Data. All these research aspects will be considered within the development of the WOW Semantic Archive that will have an expected impact in supporting the digital transformation of the cultural sector, following the suggested focuses by the PNR related to *Digitization and Valorization of the Cultural Heritage*, but also strengthening the social inclusion, by reducing gender inequalities.

---

[5] https://www.europeanfilmgateway.eu/it

References

[1] E. Gardiner, R. G. Musto, The Digital Humanities: A Primer for Students and Scholars, Cambridge University Press, 2015.

[2] E. Hyvönen, Using the Semantic Web in digital humanities: Shift from data publishing to data-analysis and serendipitous knowledge discovery, Semantic Web 11 (2020) 187–193.

[3] F. Moretti, Distant reading, Verso Books, 2013.

[4] F. Ciotti, Modelli e metodi computazionali per la critica letteraria: lo stato dell'arte (2017).

[5] F. Ciotti, Distant reading in literary studies: a methodology in quest of theory, Testo e Senso (2021) 195–213.

[6] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American 284 (2001) 34–43.

[7] C. Bizer, T. Heath, K. Idehen, T. Berners-Lee, Linked Data on the Web (LDOW2008), in: Proceedings of the 17th international conference on World Wide Web, 2008, pp. 1265–1266.

[8] L. Pandolfo, L. Pulina, ARKIVO Dataset: A benchmark for ontology-based extraction tools., in: Proceedings of the 17th International Conference on Web Information Systems and Technologies, WEBIST 2021, October 26-28, 2021, SCITEPRESS, 2021, pp. 341–345.

[9] L. Pandolfo, S. Spanu, L. Pulina, E. Grosso, Understanding and modeling visitor behaviours for enhancing personalized cultural experiences, Int. J. Technol. Hum. Interact. 16 (2020) 24–38.

[10] N. Guarino, D. Oberle, S. Staab, What is an ontology?, in: Handbook on Ontologies, Springer, 2009, pp. 1–17.

[11] G. Adorni, M. Maratea, L. Pandolfo, L. Pulina, An ontology for historical research documents, in: Web Reasoning and Rule Systems - 9th International Conference, RR 2015, Berlin, Germany, August 4-5, 2015, Proceedings, volume 9209 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 11–18.

[12] G. Adorni, M. Maratea, L. Pandolfo, L. Pulina, An ontology-based archive for historical research, in: Proceedings of the 28th International Workshop on Description Logics, Athens,Greece, June 7-10, 2015, volume 1350 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015.

[13] L. Pandolfo, L. Pulina, M. Zieliński, Towards an ontology for describing archival resources, in: Proceedings of the Second Workshop on Humanities in the Semantic Web (WHiSe II) co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 22, 2017, volume 2014 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017, pp. 111–116.

[14] L. Pandolfo, L. Pulina, M. Zieliński, ARKIVO: an ontology for describing archival resources, in: P. Felli, M. Montali (Eds.), Proceedings of the 33rd Italian Conference on Computational Logic, Bolzano, Italy, September 20-22, 2018, volume 2214 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018, pp. 112–116.

[15] L. Pandolfo, L. Pulina, M. Zieliński, Exploring semantic archival collections: The case of Piłsudski Institute of America, in: Digital Libraries: Supporting Open Science - 15th Italian Research Conference on Digital Libraries, IRCDL 2019, Pisa, Italy, January 31 - February 1, 2019, Proceedings, volume 988 of *Communications in Computer and Information Science*, Springer, 2019, pp. 107–121.

[16] L. Pandolfo, L. Pulina, Building the semantic layer of the Józef Piłsudski digital archive with an ontology-based approach, International Journal on Semantic Web and Information Systems (IJSWIS) 17 (2021) 1–21.

[17] M. Rizzarelli, L'attrice che scrive, la scrittrice che recita. per una mappa della 'diva-grafia', Vaghe stelle. Attrici del/nel cinema italiano. Arabeschi, edited by Lucia Cardone, Giovanna Maina, Stefania Rimini, and Chiara Tognolotti 10 (2017) 366–371.

[18] G. Simi, L'occhio che palpita. monica vitti e gli scritti sull'arte, Cinergie–Il Cinema e le altre Arti (2021) 153–166.

[19] M. Rizzarelli, Il doppio talento dell'attrice che scrive. per una mappa delle "divagrafie", Cahiers d'études italiennes (2021).

[20] C. Tognolotti, Una diva fragrante. l'immagine divistica di sophia loren nei libri di ricette (2019).

[21] L. Pandolfo, L. Pulina, G. Adorni, A framework for automatic population of ontology-based digital libraries, in: AI*IA 2016: Advances in Artificial Intelligence - XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 - December 1, 2016, Proceedings, volume 10037 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 406–417.

[22] ADnOTO: a self-adaptive system for automatic ontology-based annotation of unstructured documents, in: Advances in Artificial Intelligence: From Theory to Practice - 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, Arras, France, June 27-30, 2017, Proceedings, Part I, volume 10350 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 495–501.

On Augmented Stochastic Submodular Optimization: Adaptivity, Multi-Rounds, Budgeted, and Robustness

Vincenzo Auletta[1,†], Diodato Ferraioli[1,*,†] and Cosimo Vinci[1,2,†]

[1]*Università degli Studi di Salerno, Fisciano, Salerno, Italy*
[2]*Università del Salento, Lecce, Italy*

Abstract

In this work we consider the problem of Stochastic Submodular Maximization, in which we would like to maximize the value of a monotone and submodular objective function, subject to the fact that the values of this function depend on the realization of stochastic events. This problem has applications in several areas, and in particular it well models basic problems such as influence maximization and stochastic probing.

In this work, we advocate the necessity to extend the study of this problem in order to include several different features such as a budget constraint on the number of observations, the chance of adaptively choosing what we observe or the presence of multiple rounds. We here speculate on the possible directions that this line of research can take. In particular, we will discuss about interesting open problems mainly in the settings of robust optimization and online learning.

Keywords
Influence Maximization, Stochastic Probing, Approximation Algorithms, Online Learning

Introduction

Consider the well-known problem of influence maximization: here we are given a (social) network, with edge probabilities encoding the strength of the relationships among network's components; these probabilities describe how easily influence (e.g., an information or an innovation) spreads over the network starting from a limited set of nodes, usually termed *seeds*; the goal is then to find the set of seeds that maximizes the number of "influenced" nodes. The non-adaptive version of the problem, in which seeds are chosen before that influence starts, has been widely studied, with numerous successful algorithms proposed [1, 2, 3] and applications ranging from viral marketing [1], to election manipulation [4, 5], to infective disease prevention [6]. Recently, focus moved to the adaptive variant of this problem (in which seeds can be added even when the influence is spreading from previously placed seeds) [7, 8, 9, 10].

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

[*] Corresponding author.

[†] These authors contributed equally.

auletta@unisa.it (V. Auletta); dferraioli@unisa.it (D. Ferraioli); cosimo.vinci@unisalento.it (C. Vinci)
https://docenti.unisa.it/001366/home (V. Auletta); https://docenti.unisa.it/023604/home (D. Ferraioli);
https://www.unisalento.it/scheda-utente/-/people/cosimo.vinci (C. Vinci)
0000-0002-7875-3366 (V. Auletta); 0000-0002-7962-5200 (D. Ferraioli); 0000-0001-7741-9342 (C. Vinci)

CEUR Workshop Proceedings (CEUR-WS.org)

This variant has strong relationship with some *Stochastic Probing* problems [11], in which there are n items whose states are stochastically defined according to (not necessarily equal) independent distributions and the goal is to choose a feasible subset of items maximizing the value of a non-negative, monotone and submodular set function. Several stochastic probing settings can be found in expert hiring [12], online dating and kidney exchange [13], Bayesian mechanism design [14], and sequentially posted pricing [15, 16]. Research both in influence maximization and stochastic probing mainly focused on the single-round setting. However, in several cases the optimization is run in a multi-round setting. For example, viral marketing or electoral campaigns are usually deployed in multiple rounds of (usually adaptive) influence; similarly, expert hiring sessions and sequentially posted pricing are run in multiple steps.

In this paper, we then advocate the necessity to extend the study of these problems in order to consider all these properties: budget, adaptivity, and multiple-rounds. Indeed, previous works only consider them separately.

Indeed, the analysis of optimization advantages of adaptive w.r.t. non-adaptive algorithms has been recently applied to stochastic maximization problems, including *single-round* influence maximization and stochastic probing. In particular, Golovin and Krause [7] showed that whenever the objective function satisfies a property, named *adaptive submodularity*, then a greedy algorithm achieves a constant approximation of the optimal greedy algorithm. Some results have been recently presented about (Adaptive) Multi-round Influence Maximization [17, 18]: these results however do not take into account budget.

Multiple rounds have been considered mainly in the setting of *online learning algorithms*, aka bandit algorithms, in which our algorithm are asked to work even if the input is only partially known, and to improve their performance over multiple round (by essentially learning the input). This class of algorithms has been introduced by Gittins [19], and it has been applied recently also to influence maximization and other submodular maximization processes [20, 21, 22, 23, 24]. While these works focus on multi-round settings, they do not consider the chance of adaptively choose observations, and often they also do not consider budget constraints.

Our contribution

In this work, we introduce the problem of *Stochastic Budgeted Multi-round Adaptive Submodular Maximization.* The problem is defined by a number T of rounds, a ground set of n items, whose state is drawn at each time step from a (node-specific) set Θ according to some time-dependent distribution, a time-dependent monotone and submodular state function that assigns a non-negative real-value to each realization of states upon specific item observations, and a budget B on the possible observations done within the time-horizon T. The problem is then to choose which observations to make in each of the T rounds, subject to the budget limit, in order to maximize the expected total value of the state function over the T rounds. It is not hard to see how this problem generalizes both the influence maximization and stochastic probing problems: in the first case, the state of an item v defines which nodes would be influenced whether v is selected as a seed, with the distribution of the state of v depending on edges' probabilities; an observation consists on choosing a given node as a seed, then running the influence spread, and finally evaluating the realized state by, for example, counting the number or the total weight of influenced nodes. In the second case, the state of an item can be active or inactive, and we can observe the state of selected items only; then, the value of the submodular set function depends on the set of selected items which are active. Note that SBMSm not only generalizes influence maximization and stochastic probing problems by extending them to multiple round settings, but it also allows to model more realistic settings where states can change over time: e.g., in the influence maximization setting, we may allow edge probabilities or node weights to change over time.

Nevertheless the complexity of the problem, we claim that the state of our knowledge is sufficient to face this kind of problems. In fact, we can prove that, by greedily allocating each unit of budget to the round in which it is expected to produce the largest expected value of the score function, given the amount of budget still available, and then running one of the well-known single-round adaptive budgeted approaches to choose observations to make at each round, we may achieve a constant approximation of the optimal algorithm (that adaptively allocates the budget among rounds, and the allocations at each round).

Let us now discuss few directions for the research about Stochastic Budgeted Multi-round Adaptive Submodular Maximization that we deem as very interesting and promising. In particular, we will henceforth focus on two main approaches.

Online Learning. Suppose that state distributions are unknown but all rounds share the same state distribution. Can we define an adaptive policy that maximizes the expected total value of the state function by "learning" the underlying probability distribution? In particular, we will consider an *agnostic (adaptive multi-round) policy*, where the choice of each picked item v is based on the observed partial state (both in the current round and the previous ones), on the initial instance, but not on the distribution of states. As each observation provides samples of several local states, such samples will be used to determine an *estimated distribution* of states at each round t, that is more accurate as t increases and more samples are provided. The estimated distribution will be used to "simulate" the real one with a good approximation (e.g., when computing the item maximizing the expected increment).

As benchmark quality metric for the proposed policy π, we will consider the α*-approximate regret* $Reg_\alpha(T) = \alpha \cdot \sigma_\mu(\pi^*) - \sigma_\mu(\pi)$, where π^* is an optimal non-agnostic adaptive policy (based on the knowledge of the distribution of states), and $\sigma_\mu(\pi)$ denotes the expected value of π.

We believe it would be interesting to evaluate whether there is a policy π with constant approximation and sublinear regret. In particular, it has been recently introduced an approach, named *Combinatorial Upper Confidence Bound* (CUCB), to design online learning algorithm with constant approximation and poly-logarithmic regret [20, 21, 22], that has been successfully used by Gabillon et al. [23] and by Das et al. [24] in setting that are similar but simpler than the one proposed in this work. Unfortunately, it is not hard to see that a naive application of CUCB to the model defined above cannot work, since we need not only to estimate which observations to make at each round, but also how many, and the latter can be largely affected by an imprecise estimation of states' distribution. However, this raises other interesting questions: are there settings in which we can be able to use the CUCB framework to achieve constant approximation and poly-logarithmic regret? Can we design another framework such that is able to return constant approximation and poly-logarithmic regret in our more general setting? Or, how high is the cost that we have to pay in term of approximation guarantee in order to achieve poly-logarithmic regret?

While the regret is an useful measure (and the standard de facto) in order to understand the learning abilities of an algorithm, other measures can be of some interest. For example, one may be interested in agnostic policies that are able to reach a given goal (e.g., the aggregate state function is above a given threshold) within a given deadline. Note that learning algorithm with low regret may fail in achieving this goal: indeed, regret may depend on other time-independent factors that may be often large. Moreover, low regret does not exclude long *exploration* phases in which a lot of “value” has been lost.

Robust Optimization. The second approach is also motivated by partial or imprecise knowledge of state distributions. However, while above we aimed to learn these distributions, we here are concerned on how and how much this limited knowledge affects our optimization goals. This approach is receiving large attention very recently, motivated by the necessity to run optimization algorithms not on real data but on predictions often generated by machine learning algorithms. The field has blossomed with applications to facility location [25], scheduling [26], clustering [27], influence maximization [28].

Here, natural interesting questions that we can ask are: how much the known optimization frameworks are sensible to distortion on input data? Or how much has to be lost in that framework if one want an algorithm that is robust to distortions within a given threshold?

Acknowledgments

This work is partially supported by the Italian MIUR PRIN 2017 Project ALGADIMAR “Algorithms, Games, and Digital Markets”, by “GNCS–INdAM” and by the PON R&I 2014-2020 Project TEBAKA "Sistema per acquisizione conoscenze di base del territorio”.

References

[1] D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, in: KDD, 2003, pp. 137–146.

[2] N. Chen, On the approximability of influence in social networks, SIAM Journal on Discrete Mathematics 23 (2009) 1400–1415.

[3] C. Borgs, M. Brautbar, J. Chayes, B. Lucier, Maximizing social influence in nearly optimal time, in: SODA, 2014, pp. 946–957.

[4] B. Wilder, Y. Vorobeychik, Controlling elections through social influence, in: AAMAS, 2018, pp. 265–273.

[5] M. Castiglioni, D. Ferraioli, N. Gatti, G. Landriani, Election manipulation on social networks: Seeding, edge removal, edge addition, Journal of Artificial Intelligence Research 71 (2021) 1049–1090.

[6] A. Yadav, B. Wilder, E. Rice, R. Petering, J. Craddock, A. Yoshioka-Maxwell, M. Hemler,

L. Onasch-Vera, M. Tambe, D. Woo, Influence maximization in the field: The arduous journey from emerging to deployed application, in: AAMAS, 2017, pp. 150–158.

[7] D. Golovin, A. Krause, Adaptive submodularity: Theory and applications in active learning and stochastic optimization, Journal of Artificial Intelligence Research 42 (2011) 427–486.

[8] W. Chen, B. Peng, On adaptivity gaps of influence maximization under the independent cascade model with full-adoption feedback, in: ISAAC, 2019, pp. 24:1–24:19.

[9] G. D'Angelo, D. Poddar, C. Vinci, Better bounds on the adaptivity gap of influence maximization under full-adoption feedback, in: AAAI, 2021, pp. 12069–12077.

[10] G. D'Angelo, D. Poddar, C. Vinci, Improved approximation factor for adaptive influence maximization via simple greedy strategies, in: ICALP, 2021, pp. 59:1–59:19.

[11] A. Asadpour, H. Nazerzadeh, A. Saberi, Stochastic submodular maximization, in: WINE, 2008, pp. 477–489.

[12] M. Hoefer, K. Schewior, D. Schmand, Stochastic probing with increasing precision, in: IJCAI, 2021, pp. 4069–4075.

[13] A. Gupta, V. Nagarajan, A stochastic probing problem with applications, in: IPCO, 2013, pp. 205–216.

[14] A. Asadpour, H. Nazerzadeh, Maximizing stochastic monotone submodular functions, Management Science 62 (2016) 2374–2391.

[15] S. Chawla, J. D. Hartline, D. L. Malec, B. Sivan, Multi-parameter mechanism design and sequential posted pricing, in: STOC, 2010, pp. 311–320.

[16] M. Adamczyk, A. Borodin, D. Ferraioli, B. d. Keijzer, S. Leonardi, Sequential posted price mechanisms with correlated valuations, in: WINE, 2015, pp. 1–15.

[17] L. Sun, W. Huang, P. S. Yu, W. Chen, Multi-round influence maximization, in: KDD, 2018, pp. 2249–2258.

[18] G. Tong, R. Wang, Z. Dong, X. Li, Time-constrained adaptive influence maximization, IEEE Transactions on Computational Social Systems 8 (2020) 33–44.

[19] J. C. Gittins, Bandit processes and dynamic allocation indices, Journal of the Royal Statistical Society: Series B (Methodological) 41 (1979) 148–164.

[20] S. Vaswani, L. Lakshmanan, M. Schmidt, et al., Influence maximization with bandits, arXiv preprint arXiv:1503.00024 (2015).

[21] W. Chen, Y. Wang, Y. Yuan, Q. Wang, Combinatorial multi-armed bandit and its extension to probabilistically triggered arms, The Journal of Machine Learning Research 17 (2016) 1746–1778.

[22] Z. Wen, B. Kveton, M. Valko, S. Vaswani, Online influence maximization under independent cascade model with semi-bandit feedback, NeurIPS 30 (2017).

[23] V. Gabillon, B. Kveton, Z. Wen, B. Eriksson, S. Muthukrishnan, Adaptive submodular maximization in bandit setting, NeurIPS 26 (2013).

[24] D. Das, S. Jain, S. Gujar, Budgeted combinatorial multi-armed bandits, in: AAMAS, 2022, pp. 345–353.

[25] Y. Azar, D. Panigrahi, N. Touitou, Online graph algorithms with predictions, in: SODA, 2022, pp. 35–66.

[26] Y. Azar, S. Leonardi, N. Touitou, Flow time scheduling with uncertain processing time, in: STOC, 2021, pp. 1070–1080.

[27] J. C. Ergun, Z. Feng, S. Silwal, D. Woodruff, S. Zhou, Learning-augmented k-means

clustering, in: ICLR, 2021.

[28] B. Wilder, N. Immorlica, E. Rice, M. Tambe, Maximizing influence in an unknown social network, in: AAAI, 2018, pp. 4743–4750.

Hedonic Games with Fixed-Size Coalitions

Vittorio Bilò[1,*,†], Gianpiero Monaco[2,†] and Luca Moscardelli[2,†]

[1]University of Salento, Provinciale Lecce-Arnesano P.O. Box 193, 73100, Lecce, Italy
[2]University of Chieti-Pescara, Viale Pindaro 42, 65127, Pescara, Italy

Abstract

In hedonic games, a set of n agents, having preferences over all possible coalition structures, needs to agree on a stable outcome. In this work, we initiate the study of hedonic games with fixed-size coalitions, where the set of possible coalition structures is restricted as follows: there are k coalitions, each coalition has a fixed size, and the sum of the sizes of all coalitions equals n. We focus on the basic model of additively separable hedonic games with symmetric valuations, where an agent's preference is captured by a utility function which sums up a contribution due to any other agent in the same coalition. In this setting, an outcome is stable if no pair of agents can exchange coalitions and improve their utilities. We analyse the fundamental questions of existence, complexity and efficiency of stable outcomes, and that of complexity of a social optimum.

Keywords

Coalition formation, Swap equilibria, Price of Anarchy and Stability

1. Introduction

We introduce the model of *hedonic games with fixed-size coalitions*, where n agents want to be assigned to coalitions in order to maximize their utilities. Differently from classical models of hedonic games [1, 2, 3, 4], both the number and sizes of coalitions are fixed: there are k coalitions, whose sizes sum up to n. The utility that agent i receives when being in the same coalition of agent j is given by a non-negative value v_{ij}. We assume that valuations are **symmetric**, i.e., $v_{ij} = v_{ji}$ for every distinct agents i and j. Valuations are *dichotomous* when they belong to $\{0, 1\}$.

We are interested in the existence and efficiency of outcomes which may be resistant to agents swaps. In particular, we distinguish among three different types of swap stability, depending on how we define the happiness of a pair of swapping agents. We say that an outcome is *swap stable under transferable utilities* (SSTU), if no pair of swapping agents can improve the sum of their utilities; it is *strictly swap stable* (SSS), if no swap can improve the utility of one agent without decreasing the utility of the other one; it is *swap stable* (SS), if no pair of swapping agents can improve both of their utilities simultaneously. It is immediate to see that, by definition, any outcome which is SSTU is also SSS and that any SSS outcome is also SS. So, positive results

---

Spirit'22: Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy, November 29, 2022, Udine, Italy

[*] Corresponding author.

[†] These authors contributed equally.

vittorio.bilo@unisalento.it (V. Bilò); gianpiero.monaco@unich.it (G. Monaco); luca.moscardelli@unich.it (L. Moscardelli)

CEUR Workshop Proceedings (CEUR-WS.org)

holding for SSTU outcomes (such as existence and efficient computation) extend to both SSS and SS ones, while negative results for SS outcomes (such as hardness of computation) apply to SSS and SSTU outcomes as well.

We prove that the *utilitarian social welfare* (USW), defined as the sum of the utilities of all agents, is a potential function which shows that, starting from any outcome, every sequence of swaps in which the sum of the utilities of the swapping agents increases converges to a SSTU outcome after a finite number of swaps. For dichotomous valuations, this number is at most $n(n-1)/2$, which yields a polynomial time algorithm for computing a SSTU outcome. For general valuations, however, we prove that computing a SS outcome is PLS-complete by reworking a reduction from the MAX CUT problem originally designed in [5].

To measure the efficiency of stable outcomes, we resort to two state-of-the-art metrics: the *price of anarchy* (PoA) and the *price of stability* (PoS). They compare the USW of a socially optimal outcome against the USW of, respectively, the worst and the best possible stable outcome. As a corollary of the fact that the USW is a potential function, the PoS of all three stability notions is equal to 1. The characterization of the PoA, instead, is much more challenging and constitutes one of the main results of our work. We first show that the PoA of SS outcomes can be unbounded even for games with dichotomous valuations. A similar negative result is also obtained for SSTU outcomes (and so also for SSS ones) for games with generic valuations and even for games with dichotomous valuations, as long as there exist agents who dislike all the other ones (i.e., *misanthropes*). In light of these negative results, our investigation will be limited to the characterization of the PoA of both SSS and SSTU outcomes in games with dichotomous valuations and no misanthropes. For SSS outcomes, we show that the PoA is exactly $2k-1$. To obtain the upper bound, we first derive a couple of technical lemmas. The first lemma lower bounds the USW of a SSS outcome as a function of the size of the largest coalition. The second one, instead, upper bounds the sum of the utilities of the agents belonging to any two coalitions c and c' in a SSS outcome which half of the total utility that the agents would get if they were all together in a coalition. By summing over all possible pairs of coalitions, the desired bound can be obtained. However, the technical lemma does not hold when both c and c' are singleton sets. Thus, one needs to deal with the presence of singletons in the coalition structure. We resolve this issue by means of an ingenious inductive argument on the number of singleton coalitions. For SSTU outcomes, for which the $2k-1$ upper bound carries over, we complement the analysis with a lower bound of $2k-2+\frac{1}{k-1}$. This lower bound is tight for $k=2$ and almost tight up to an additive factor of 1 for any other value of k.

We also investigate the problem of computing a social optimum (SO). On the negative side, we show that a SO cannot be approximated within $n^{o(1)}$, even for games with dichotomous valuations. Even for constant k, computing a SO remains NP-hard. On the positive side, by leveraging known approximability results for the DENSEST t-SUBGRAPH [6], we obtain an $O(k^2)$-approximation algorithm. For dichotomous valuations, we know that a SSTU outcome can be computed in polynomial time and that this outcomes is a $(2k-1)$-approximation of the SO. However, if all coalitions have size at least 3, we design an algorithm whose approximation guarantee is $\left(1+\frac{s}{s-1}(k-1)\right)$, where s is the size of the smallest coalition. In particular, as s increases, the performance guarantee tends to k.

References

[1] A. Bogomolnaia, M. O. Jackson, The stability of hedonic coalition structures, Games and Economic Behavior 38 (2002) 201–230. doi:`10.1006/game.2001.0877`.

[2] S. Banerjee, H. Konishi, T. Sönmez, Core in a simple coalition formation game, Social Choice and Welfare 18 (2001) 135–153. doi:`10.1007/s003550000067`.

[3] K. Cechlárová, A. Romero-Medina, Stability in coalition formation games, Int. J. Game Theory 29 (2001) 487–494. doi:`10.1007/s001820000053`.

[4] J. H. Dreze, J. Greenberg, Hedonic coalitions: Optimality and stability, Econometrica 48 (1980) 987–1003.

[5] A. Schaffer, M. Yannakakis, Simple local search problems that are hard to solve., SIAM J. Comput. 20 (1991) 56–87. doi:`10.1137/0220004`.

[6] Y. Asahiro, K. Iwama, H. Tamaki, T. Tokuyama, Greedily finding a dense subgraph, Journal of Algorithms 34 (2000) 203–221. doi:`https://doi.org/10.1006/jagm.1999.1062`.

Towards a Formal Verification of Attack Graphs

Davide Catta[1,†], Jean Leneutre[1] and Vadim Malvone[1]

[1]LTCI, Télécom Paris, Institut Polytechnique de Paris, Paris, France

Abstract
In this perspective paper, we propose different formalizations of games that are played over Attack Graphs between an Attacker and a Defender. In all such games we propose a formal approach (such as logics and automata theory) to check whether the Attacker has a strategy to win the game.

Keywords
Attack Graphs, Sabotage Games, Logics in Games

1. Introduction

Modern systems are inherently complex and security plays a crucial role. The main challenge in developing a secure system is to come up with tools able to detect vulnerability and unexpected behaviors at a very early stage of its life-cycles. These methodologies should be able to measure the grade of resilience to external attacks. Crucially, the cost of repairing a system flaw during maintenance is at least two order of magnitude higher, compared to a fixing at an early design stage [1]. In the past fifty years several solutions have been proposed and a story of success is the use of *formal methods* techniques [1]. They allow checking whether a system is correct by formally checking whether a mathematical model of it meets a formal representation of its desired behavior. Recently, classic approaches such as *model checking* and automata-theoretic techniques, originally developed for monolithic systems [2, 3], have been meaningfully extended to handle *multi-agent systems* [4, 5, 6, 7, 8]. These are systems that encapsulate the behavior of two or more rational agents interacting among them in a cooperative or adversarial way, aiming at a designed goal [9].

In system security checking, a malicious attack can be seen as an attempt of an Attacker to gain an unauthorized resource access or compromise the system integrity. In this setting, *Attack Graph* [10] is one of the most prominent attack model developed and receiving much attention in recent years. This encompasses a graph where each state represents an Attacker at a specified network location and edges represent state transitions, i.e., attack actions by the Attacker. In this paper, we sketch three different methodologies to reason about Attack Graphs by introducing game models in which a player (called Attacker) travels through adjacent edges of an Attack Graph and tries to reach a certain target state, and another player (called Defender)

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

[†] These authors contributed equally.

davide.catta@telecom-paris.fr (D. Catta); jean.leneutre@telecom-paris.fr (J. Leneutre);
vadim.malvone@telecom-paris.fr (V. Malvone)

tries to prevent to Attacker to reach his goal by dynamically deploying countermeasures. All the game models that we will introduce can be handled by means of formal methods, i.e., to reason about such games we will use logical methods or automata-theoretic approaches.

2. Attack Graphs

The term Attack Graph has been first introduced by Phillips and Swiler [11]. Attack Graphs represent the possible sequence of attacks in a system as a graph. An Attack Graph may be generated by using the following information: a description of the system architecture (topology, configurations of components, etc.), the list of the known vulnerabilities of the system and the Attacker's profile (his capabilities, password knowledge, privileges etc.) and attack templates (Attacker's atomic action, including preconditions and postconditions). An attack path in the graph corresponds to a sequence of atomic attacks. Several works have developed this approach and each work introduces its own Attack Graph model with its specificity, and thus there is no standard definition of an Attack Graph, see [12] for a survey. However, all introduced models can be mapped into a *canonical Attack Graph* as introduced in [13]. It is a labelled oriented graph, where:

1. each node represents both the state of the system (including existing vulnerabilities) and the state of the Attacker including constants (Attacker skills, financial resources, etc.) and variables (knowledge on the network topology, privilege level, obtained credentials, etc.);
2. each edge represents an action of the Attacker (a scan of the network, the execution of an exploit based on a given vulnerability, access to a device, etc.) that changes the state of the network or the states of the Attacker; an edge is labelled with the name of the action (several edges of the Attack Graph may have the same label).

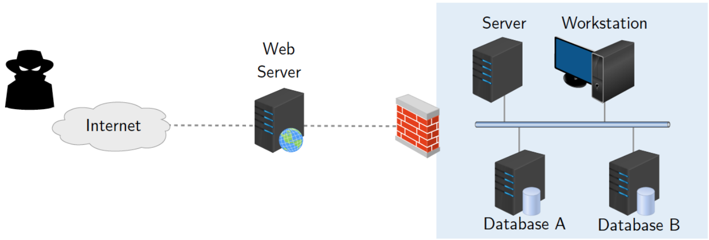
We recall that a *rooted* Kripke structure M is given by a non-empty set S of states, a designated initial state s_0, a finite-non empty set Σ of labels, a ternary relation $R \subseteq S \times \Sigma \times S$ and a labeling function V that maps any element of S to a set of atomic proposition. By abstracting all the data in the above discussion, one can define an Attack Graph as follows.

Definition 1. *Let* Σ *be a finite set of labels, and* $\mathcal{P}$ *a finite set of atomic propositions, such that* $\mathsf{win} \in \mathcal{P}$. *An* ***Attack Graph*** *is a tuple* $AG = \langle S, s_0, \Sigma, R, V, W\rangle$ *where:*

- $\langle S, s_0, \Sigma, R, V\rangle$ *is a rooted Kripke structure where the set* S *of states is finite and* $V(s) \subseteq \mathcal{P}$ *for any* $s \in S$*;*
- W *is a non-empty subset of* S *such that* $\mathsf{win} \in V(s)$ *for all* $s \in W$.

The set W *represents the set of target states of the Attacker.*

Let us make things more concrete. Consider the following scenario: an enterprise has a local area network (LAN) that features a *Server*, a *Workstation* and two databases A and B. The LAN also provides a *Web Server*. Internet's access to the LAN are controlled by a firewall. Such scenario is depicted in Figure 1. Suppose that we know some vulnerabilities and that we have established that a malevolent user can make the attacks listed in the Table of Figure 1, e.g., by making att_2, an Attacker can exploit a vulnerability related to the *Server*: as a precondition the Attacker needs to have root access to the *Web Server* and, as a postcondition, he will obtain root access to the *Server*. Then we can construct an Attack Graph built from this set of atomic attacks and collecting possible attack paths as depicted in Figure 1[1]. The Attacker's initial state is a node in the Attack Graph. Let us suppose that the Attacker is in state s_1 and wants to reach state s_4. To get to this target, he can perform the sequences of atomic attacks att_2, att_4 or att_3, att_2, att_4.



Attack	Location	Precondition	Postcondition	Counter measure
att_1	Web Server		$web\_server : root$	_
att_2	Server	$web\_server : root$	$server : root$	c_2
att_3	Workstation	$web\_server : root$	$password : 1234$	_
att_4	Database A	$server : root$	$databaseA : root$	c_4
att_5	Database B	$server : root \wedge$ $password : 1234$	$databaseB : root$	c_5

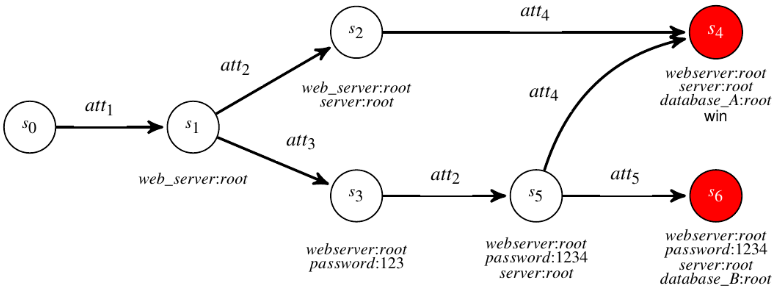


Figure 1: An illustrating LAN architecture example, a list of Atomic attacks and countermeasures over the LAN, and an Attack Graphs generated by such Atomic attacks.

---

[1] This Attack Graph is not complete w.r.t. our previous description, since some possible sequences of atomic attacks are not listed: for instance $att_1, att_2, att_3, att_5$ are not taken into account.

3. Attack Graph Games

In the previous section, we saw that an Attack Graph represents a sequence of possible actions made by an Attacker to reach a specific goal. Let us add another character to this story, the Defender, whose objective is to counter the attack. Suppose that she has the power to dynamically deploy a predefined set of countermeasures: for instance by reconfiguring the firewall filtering rules, or patching some vulnerabilities, that is by removing one or several preconditions of an atomic attack. A given countermeasure c will prevent the Attacker from longing a given attack att. One can imagine a game played over the Attack Graph in which the Attacker tries to reach one of its goal states, and the Defender tries to prevent the Attacker from reaching a goal state by deploying countermeasures. Given such a game, one natural question to ask is the following: is there a winning Attacker Strategy? We will now define three possible settings in which such games can be formalized.

3.1. Attack Graph Games & (Subset) Sabotage Modal Logic

Above, we introduced a Defender, whose power is to deploy certain countermeasures, and we introduced the concept of game over an Attack Graph. To make things less abstract, consider a two player turn-based game over Attack Graph in which: the Defender starts the game by selecting a certain countermeasure c. By choosing such a countermeasure, she deletes a subset of edges of the Attack Graph: all the edges that are labeled by c. The Attacker takes his turn and moves from s_0 to one of its successor along the edges that have not being erased (if any). The game evolves following this pattern. The Attacker wins if he can reach one of the states in W in a finite number of moves, the Defender wins otherwise. A run in such a game is nothing more than a run in a insidious sabotage game. Sabotage games were introduced by van Benthem in 2005 [14], with the aim of studying the computational complexity of a special class of graph-reachability problems. Namely, graph reachability problems in which the set of edges of the graph became thinner and thinner as long as a path of the graph is constructed. To reason about sabotage games, van Benthem introduced Sabotage Modal Logic. Such Logic is obtained by adding to the $\Diamond$-modality of classical modal logic another modality $\blacklozenge$.

Let G be a directed graph and s one of its vertex (or states); the intended meaning of a formula $\blacklozenge\varphi$ is " $\blacklozenge\varphi$ is true at a state s of G iff φ is true at s in the graph obtained by G by erasing an edge e". Our sabotage games differs from the one introduced by van Benthem because one of the players can erase *an entire subset of edges of a given graph.* To reason about such games, in [15] we introduce a variant of Sabotage Modal Logic that we call, for lack of wit, Subset Sabotage Modal Logic (SSML, for short). The logic SSML is obtained by adding a modality $\blacklozenge_a^{\subseteq}$ to the language of classical modal logic. Being more precise, let $\mathcal{P}$ be a non-empty set of atomic propositions and Σ a finite, non empty set of labels. Formulas of SSML are defined by the following grammar

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond_a\varphi \mid \blacklozenge_a^{\subseteq}\varphi$$

where p is any atomic proposition and a any label. One can define the boolean connectives $\vee$ and $\to$ in the usual way, $\Box_a\varphi$ as $\neg\Diamond_a\neg\varphi$, and $\blacksquare_a^{\subseteq}\varphi$ as $\neg\blacklozenge_a^{\subseteq}\neg\varphi$.

The intended meaning of a formula $\blacklozenge^{\subset}\varphi$ is " $\blacklozenge^{\subseteq}_{a}\varphi$ is true at a state s of a directed graph G iff φ is true at s in the graph G' that is obtained from G by erasing a non-empty set of edges that are labeled by a". We now precisely define the semantics of SSML. If $M = \langle S, s_0, \Sigma, R, V\rangle$ is a Kripke structure and $E \subseteq R$, we write $M \setminus E$ to denote the Kripke structure obtained by erasing the subset E from the relation R. If $e = (s, a, s') \in R$ we say that e is a *labeled edge* or simply an edge, and that a is the label of e. We denote by R_a the subset of labeled edges of R whose label is a, that is $R_a = \{e \in R \mid e \in S \times \{a\} \times S\}$.

The notion of satisfaction of a formula φ at a given state s of a Kripke structure M (written $M, s \models \varphi$) is inductively defined as follows:

$M, s \models \top$ for all $s \in S$;

$M, s \models p$ iff $p \in V(s)$;

$M, s \models \neg\varphi$ iff not $M, s \models \varphi$ (notation $M, s \not\models \varphi$);

$M, s \models \varphi_1 \wedge \varphi_1$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$;

$M, s \models \Diamond_a \varphi$ iff there is a $s' \in S$ such that $(s, a, s') \in R$ and $M, s' \models \varphi$;

$M, s \models \blacklozenge^{\subseteq}_{a} \varphi$ iff there is a non-empty $E \subseteq R_a$ such that $M \setminus E, s \models \varphi$.

We say that a formula φ is true in a rooted Kripke structure M (written $M \models \varphi$) iff φ is true at the initial state of M. Remark that $M, s \models \blacksquare^{\subseteq}_{a} \varphi$ if and only if for any non-empty subset E of R_a we have that $M \setminus E, s \models \varphi$, R_a included. This implies that if $M, s \models \blacksquare^{\subseteq}_{a} \varphi$ then $M \setminus R_a, s \models \varphi$.
A strategy is a plan of action. As it is logical, the plan is winning when it leads me to victory, whatever my opponent's plan of action. Thus, a winning strategy can be expressed as an alternance of universally quantified sentences and existentially quantified sentences "for all actions of my Opponent, there is an action that I can make that leads me to victory". Let us put ourselves in the villain's shoes: suppose that we are the Attacker, and that, by playing, we have reached a certain state s of an Attack Graph AG. It is now Defender's turn. If I have a winning strategy, I must be able to reach a successor state s' of s and this for any subset that is removed by the Defender. Said differently, we must have that $AG, s \models \blacksquare^{\subset} \Diamond \varphi = (\blacksquare^{\subseteq}_{a} \Diamond\varphi) \wedge \cdots \wedge (\blacksquare^{\subseteq}_{b} \Diamond \varphi)$ for some formula φ that expresses the winning condition. First, we define:

$$\blacksquare^{\subset} \varphi \doteq \bigwedge_{a\in\Sigma} \blacksquare^{\subseteq}_{a} \varphi \qquad \Diamond \varphi \doteq \bigvee_{a\in\Sigma} \Diamond_a \varphi$$

We can now define a family of formulas expressing the fact the the Attacker as a winning strategy to win a Subset Sabotage Game played over an Attack Graph.

Definition 2 (Winning Formulas). *The family $\{\psi_n\}_{n\in\mathbb{N}}$ of Winning formulas is defined by induction on n as follows:*

$$\psi_n = \begin{cases} \mathsf{win} & \textit{if}\, n = 0 \\ \Diamond \top \wedge \blacksquare^{\subset} \Diamond(\psi_{n-1} \vee \mathsf{win}) & \textit{otherwise} \end{cases} \tag{1}$$

The following thereom can be proved by induction on the size of the Attack Graph.

Theorem 1. *For any attack Graph* $AG = \langle S, s_0, \Sigma, R, V, W\rangle$ *such that* $|S| = n$ *we have that* $M \models \mathsf{win} \vee \psi_{n-1}$ *if and only if there is a Winning Attacker Strategy for the Subset Sabotage Game played over* AG*.*

The model-checking problem for SSML, consist in deciding if $M \models \varphi$ given a *finite* rooted Kripke Structure M and a SSML formula φ. The proof of the following theorem is obtained by defining a translation (parametrized by a given Kripke structure) from SSML formulas to SML formulas.

Theorem 2. *The model checking problem for SSML is decidable: if* M *is a finite rooted Kripke Structure and* φ *an SSML formula, we can decide whether* $M \models \varphi$ *or not.*

As a consequence of the two above theorems, we can decide if there is a winning Attacker Strategy for the Subset Sabotage Game played over AG, by checking if AG is a formal model of the formula ψ_{n-1} where n is the size of AG.

3.2. Attack Graph Games & Automata

In the SSML setting that we sketched in the previous section, we can decide if there is a winning Attacker Strategy over the subset sabotage game played over AG, by checking if AG is a formal model of a certain SSML formula ψ_n. Such a solution is quite ad-hoc: we have designed the logic SSML exactly to solve our problem. To reason about the kind of games that we described in the previous section, we can use another technique. We have presented such a technique in [16] by working together with other colleagues.

We define a game $\mathcal{G}$ as a structure $\langle V, v_I, T, G\rangle$ where $V = V_1 \cup V_2$ is a set of vertex such that $V_1 \cap V_2 = \emptyset$, $v_0 \in V_1$ is the initial vertex, $T = T_1 \cup T_2$ is a transition relation such that $T_1 \subseteq V_1 \times V_2$ and $T_2 \subseteq V_2 \times V_1$ and $G \subseteq S_2$ is a non-empty subset representing the set of Attacker's goal states. Given an Attack Graph $AG = \langle S, s_0, \Sigma, R, V, W\rangle$ together with a set of countermeasures $\mathcal{C}$ (as showed in Figure 1), we can easily convert the Attack Graph in a game $\mathcal{G}$ by using an algorithm. We briefly describe the idea of this algorithm with the following points:

1. Each vertex v of the game produced will be a pair $\langle s, L\rangle$, where s is an AG-state and L is a list of edge labels.
2. We start by setting as initial state of the game the pair $\langle s_0, \epsilon\rangle$ and for each s', such that $(s_0, a, s') \in R$ for some $a \in \Sigma$ we create an edge $(\langle s_0, \epsilon\rangle, \langle s', \epsilon\rangle) \in T_1$.
3. If one of the s' is in W we create a sink and we add $\langle s', \epsilon\rangle \in G$. Otherwise, for each countermeasure $c \in \mathcal{C}$ we create a vertex $\langle s', c\rangle \in V_2$ and we add a transition $(\langle s', \epsilon\rangle, \langle s', c\rangle) \in T_2$.
4. If in the Attack Graph AG there is an edge labeled with $d \neq c$ such that $(s', d, s'') \in R$ we add a vertex $\langle s'', d\rangle \in V_1$ and an edge $(\langle s', c\rangle, \langle s'', d\rangle) \in T_1$, otherwise we create a sink $(\langle s', c\rangle, \langle s', c\rangle) \in T_1$.

We repeat this procedure until there are states in AG. It is clear that if the Attack graph AG does not contain cycles, then the game $\mathcal{G}$ given by the procedure is a finite, labeled tree whose root is labeled by $v_I = \langle s_0, \epsilon \rangle$. We can thus define a winning strategy for the Attacker as a pruning of $\mathcal{G}$ in which each state in V_1 has at most one child and whose leaves are all in G, i.e., we can define winning strategies as follows.

Definition 3. *An attacker strategy for a game $\langle V, v_I, T, G \rangle$ is a finite tree whose root is v_I, whose branches are plays over $\mathcal{G}$ and that satisfy the following properties:*

1. *For each node s of the strategy: if $s \in V_1$ then s has at most one child.*
2. *For each node s of the strategy: if $s \in V_2$ and $s \notin G$ then s has as many child as there are nodes s' such that $(s, s') \in T_2$.*

An attacker strategy is winning whenever each leaf of the strategy belongs to G.

Finite trees can be recognized by *finite tree automaton.*

Definition 4. *A nondeterministic tree automaton (NTA, for short) is a tuple $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, F \rangle$, where: Q is a set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to 2^Q$ is a transition function mapping pairs of states and symbols to a set of states, and $F \subseteq Q$ is a set of the accepting states.*

A NTA $\mathcal{A}$ recognizes trees and works as follows. For a node tree labelled by a and $\mathcal{A}$ being in a state q, it sends different copies of itself to successors in accordance with δ. Given a non-deterministic tree automaton $\mathcal{A}$ (NTA, for short), we can check in linear time if the set of trees accepted by the automaton is empty. If such set is empty, there will be no winning strategies for the Attacker. Since the size (i.e. number of vertices) of a tree constructed by the NTA does not exceed the size of the game, we obtain the following result.

Theorem 3. *Given a game $\mathcal{G}$ it is possible to decide in linear time whether the Attacker has a strategy to win the game.*

3.3. Attack Graph Games & Interpreted System

The games we described in the two previous subsections are two-player turn-based games with perfect information. The two players move in turns and they have perfect information about each other's actions. These two assumptions are quite unrealistic. In particular, it is unrealistic to think that a Defender always have perfect information about Attacker's actions. Moreover, while in both models we have a clear view of what an Attacker is — we can identify him with the Attack Graph itself— the Defender is relegated to a subordinate role. Such a character does not have a clear ontological status has it is identified with a set of countermeasures that are given independently of the graph structure. If we want to think in terms of multi-agent system, we should try to give a clear agent's status to the Defender as well. To tackle all these problems we propose to think about Attacker and Defender(s) in a game played over an Attack Graph as *Agents* of an Interpreted System [17]. An Interpreted Systems (IS) is constructed starting from a finite set of Agents $Ag = \{1, \dots, n\}$, each agent i is specified by giving:

- a set L_i of local states;
- a set act_i of agent's action;
- a function P_i associating to each local state a non-empty set of agent's i actions;
- a transition function t_i that takes an agent i state l_i and an action a_k for each agent in Ag as argument and outputs an agent i state. The function $t_i(l_i, a_1, \dots, a_k)$ is defined if and only if we have that $a_i \in P_i(l_i)$.

If Ag is a set of k agents, a **global state** s is a tuple $s = \langle l_1, \dots, l_k \rangle$ where each l_i is an agent i state. The set of global states is denoted by G_I. Finally an **Interpreted System** is a tuple $I = \langle Ag, s_0, T, \Pi \rangle$ where Ag is a set of agents, $s_0 \in G_I$ is the global initial state, $T : G_I \times act^{Ag} \to G_I$ is the global transition function such that $T(s, a_1, \dots, a_k) = s'$ iff for every $i \in Ag$, $t_i(s, a_i) = s'$, and Π is a labeling function, associating to any global states a subset of atomic propositions. Interpreted systems can be used together with logics for the strategic reasoning such as Alternating-time temporal logic (ATL)[5] and Strategy Logic (SL)[7]. Moreover, interpreted systems came with a natural concept of imperfect information: if s is a global state, we denote by $s[i]$ its i-component. Two global states s and s', are **equivalent** for the agent i whenever $s[i] = s'[i]$. We denote such notion by $s \sim_i s'$. Two histories, i.e. finite sequences of global states, h and h', are equivalent for the agent i whenever they have the same length m and $h_j \sim_i h'_j$ for any $0 \leq j \leq m$.

Given an Attack Graph AG we can define an **Attacker Agent** whose states are the states of the attack graph, whose actions are the labels of the Attack Graph's edges (and eventually the idle action), and in which given two states s and s' of the AG if $(s, b, s') \in R$ then $t(s, (b, \vec{a})) = s'$ for some tuple of other's agent actions $\vec{a}$. The natural question is: how can we define the Defender? As we can see from the table in Figure 1, each attack in the Attack Graph, is identified with a set of preconditions and postconditions. The latter is nothing more than atomic propositions labeling states. If the Attacker is in a state s that is labeled with some preconditions, he can make an attack and move to a state s' labeled by the postconditions of the attack. We can suppose that the Defender's set $\mathcal{C}$ of countermeasures is defined symmetrically: each Defender countermeasure is identified with a pair of sets $\langle C_{pre}, C_{post} \rangle$ of atomic propositions that labels some states in the Attack Graph. Now, let $AG = \langle S, s_0, \Sigma, R, V, W \rangle$ be an attack graph. By definition, $V(s)$ is a set of atomic propositions, for any $s \in S$. In Section 2, we saw that the atomic propositions labeling a state represent two heterogeneous types of information: information about the *system* (vulnerabilities, state of the system, etc.), and information about the Attacker (his skills, knowledge of the network, access to device, etc.). We can imagine that a Defender has partial knowledge, e.g., she does not know exactly what are the skills of the Attacker, or she does not know to which devices the Attacker can access, or she has only information about the system or, even, of some subset of states of the system. We can thus imagine that the pairs $\langle C_{pre}, C_{post} \rangle$ in the set of countermeasures vary over the knowledge of the defender, i.e. $\mathcal{P}_D \subseteq \mathcal{P}$. Consider the directed graph $AG_D = \langle S_D, R_D \rangle$ where the set of states S_D is the set of equivalence classes of states of the Attack Graph modulo $\mathcal{P}_D$ (the knowledge of the defender). That is, $s \in S_D$ iff s is an equivalence class of states modulo the relation $s_i \equiv s_j \iff V(s_i) \cap \mathcal{P}_D = V(s_j) \cap \mathcal{P}_D$, where $s_i, s_j \in S$. The set of edges R_D is given by:

1. $(s, s') \in R_D$ if $(s_i, b, s_j) \in R$ for some states of the AG, s_i belonging to the equivalence class s and s_j belonging to the equivalence class s' or,
2. $(s, s') \in R_D$ if $V(s_i) = C_{pre}$ and $V(s_j) = C_{post}$ for some state of the AG, s_i belonging to the equivalence class s and s_j belonging to the equivalence class s'.

Over such a structure, we can construct a **Defender Agent**, whose set of states is S_D, having an action for each edge in R_D and in which the local transition functions mimics the relation R_D. We can impose that whenever the defender is in a state that is a precondition of a countermeasure, if she plays a certain action then the local transition functions will output the associated postconditions *no matter what are the actions of the other agents.*

Remark that by using such a construction, we can easily consider n-different defenders: we can imagine that each defender knows a different subset of $\mathcal{P}$ and as a consequence generates different defender's agent. This approach paves the way to the introduction of the strategic reasoning typical of the ATL logics into the Attack Graphs frame. In the two previous subsections, we only asked whether, given an AG one can find a winning Attacker strategy, i.e., we were obliged to consider only reachability goals for the Attacker. Thanks to the fact that Interpreted Systems are a model of the logic ATL, we can specify the strategic properties that we are interested to check via such a logic, and e.g., verify properties about the strategy ability of a coalition of Defenders.

4. Conclusion

We have sketched three possible formalization of Attacker-Defender(s) games that are played over an Attack Graph. All the approaches make use of formal methods techniques to estimate whether one of the players can win the proposed games. Two of the formalization consider two-players turn-based reachability games with perfect information, and are obtained by the means of, respectively, modal logic and automata theory. The third conceptualization, which in our opinion is the more promising, paves the way to a formalization of such games in terms of concurrent n-agents games with imperfect information and the use of formal verification techniques, e.g., model checking for the various strategic logics, to reason about such games.

The amount of work that still needs to be done to make such formalization attractive is enormous. However, we take the liberty of pointing out a research direction that we feel is a priority. It is known that under the hypothesis of uniformity, the model checking problem for ATL* with memoryfull strategies and imperfect information is undecidable [18]. Thus, if we want to use effectively the Interpreted System framework we sketched, we should restrict ourselves to use partial solutions to the problem, such as abstractions either on the information [19, 20, 21, 22] or on the strategies [23, 24, 25, 26, 27]. In the last decade, ATL has been extended to deal with the concept of strategies to obtain an objective under a limited bound of resources [28]. Such a concept is attractive for the security scenario that we are trying to take into account: an Attacker could have a limited amount of resources to make an attack and, equally, a Defender could have a limited amount of resources to make a countermeasure. To the best of our knowledge, quantitative extensions of ATL with imperfect information have not been studied in the literature. Furthermore, in this context, we can also study if an Attacker has some backup strategies to achieve his objectives by following the line on graded modalities as done in [29, 30, 31]. We believe that it would be interesting and useful for our goals to study such variants.

References

[1] E. M. Clarke, O. Grumberg, D. A. Peled, Model Checking, The MIT Press, Cambridge, Massachusetts, 1999.

[2] E. Clarke, E. Emerson, Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic., in: LP 1981 ,1981, pp. 52–71.

[3] O. Kupferman, M. Vardi, P. Wolper, An Automata Theoretic Approach to Branching-Time ModelChecking., Journal of the ACM 47 (2000) 312–360.

[4] O. Kupferman, M. Vardi, P. Wolper, Module Checking., Information and Computation 164 (2001) 322–344.

[5] R. Alur, T. Henzinger, O. Kupferman, Alternating-Time Temporal Logic., Journal of the ACM 49 (2002) 672–713.

[6] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: A model checker for the verification of multi-agent systems, in: (CAV09), 2009, pp. 682–688.

[7] F. Mogavero, A. Murano, G. Perelli, M. Y. Vardi, Reasoning about strategies: On the model-checking problem, ACM Trans. Comput. Log. 15 (2014) 34:1–34:47. URL: https://doi.org/10.1145/2631917. doi:10.1145/2631917.

[8] W. Jamroga, A. Murano, Module checking of strategic ability, in: AAMAS 2015, 2015, pp. 227–235.

[9] N. R. Jennings, M. Wooldridge, Application of intelligent agents, in: Agent Technology: Foundations, Applications, and Markets, Springer-Verlag, 1998.

[10] R. P. Lippmann, K. W. Ingols, An annotated review of past papers on attack graphs (2005).

[11] C. Phillips, L. P. Swiler, A graph-based system for network-vulnerability analysis, in: NSPW98, 1998, pp. 71–79.

[12] K. Kaynar, A taxonomy for attack graph generation and usage in network security, J. Inf. Secur. Appl. 29 (2016) 27–56.

[13] T. Heberlein, M. Bishop, E. Ceesay, M. Danforth, C. Senthilkumar, T. Stallard, A taxonomy for comparing attack-graph approaches, [Online] http://netsq.com/Documents/AttackGraphPaper. pdf (2012).

[14] J. van Benthem, An Essay on Sabotage and Obstruction, Springer Berlin Heidelberg, 2005, pp. 268–276. URL: https://doi.org/10.1007/978-3-540-32254-2_16. doi:10.1007/978-3-540-32254-2_16.

[15] D. Catta, J. Leneutre, V. Malvone, Subset sabotage games & attack graphs, in: WOA 2022, 2022, pp. 209–218. URL: http://ceur-ws.org/Vol-3261/paper16.pdf.

[16] D. Catta, A. Di Stasio, A. Murano, J. Leneutre, V. Malvone, A Game Theoretic Approach to Attack Graphs, 2023. To appear in ICAART 2023.

[17] R. Fagin, J. Halpern, Y. Moses, M. Vardi, Reasoning about Knowledge., MIT, 1995.

[18] C. Dima, F. Tiplea, Model-checking ATL under imperfect information and perfect recall semantics is undecidable, CoRR abs/1102.4225 (2011). URL: http://arxiv.org/abs/1102.4225.

[19] F. Belardinelli, A. Lomuscio, V. Malvone, An abstraction-based method for verifying

strategic properties in multi-agent systems with imperfect information, in: IAAI 2019, 2019, pp. 6030–6037. URL: https://doi.org/10.1609/aaai.v33i01.33016030. doi:`10.1609/aaai.v33i01.33016030`.

[20] F. Belardinelli, V. Malvone, A three-valued approach to strategic abilities under imperfect information, in: KR 2020, 2020, pp. 89–98. URL: https://doi.org/10.24963/kr.2020/10. doi:`10.24963/kr.2020/10`.

[21] A. Ferrando, V. Malvone, Towards the combination of model checking and runtime verification on multi-agent systems, in: PAAMS 2022, 2022, pp. 140–152. URL: https://doi.org/10.1007/978-3-031-18192-4_12. doi:`10.1007/978-3-031-18192-4\_12`.

[22] A. Ferrando, V. Malvone, Towards the verification of strategic properties in multi-agent systems with imperfect information, AAMAS (2023) (to appear).

[23] W. Jamroga, V. Malvone, A. Murano, Natural strategic ability, Artif. Intell. 277 (2019). URL: https://doi.org/10.1016/j.artint.2019.103170. doi:`10.1016/j.artint.2019.103170`.

[24] W. Jamroga, V. Malvone, A. Murano, Natural strategic ability under imperfect information, in: AAMAS 2019 , 2019, pp. 962–970. URL: http://dl.acm.org/citation.cfm?id=3331791.

[25] F. Belardinelli, A. Lomuscio, A. Murano, S. Rubin, Verification of multi-agent systems with public actions against strategy logic, Artif. Intell. 285 (2020) 103302. URL: https://doi.org/10.1016/j.artint.2020.103302. doi:`10.1016/j.artint.2020.103302`.

[26] R. Berthon, B. Maubert, A. Murano, S. Rubin, M. Y. Vardi, Strategy logic with imperfect information, ACM Trans. Comput. Log. 22 (2021) 5:1–5:51. URL: https://doi.org/10.1145/3427955. doi:`10.1145/3427955`.

[27] F. Belardinelli, A. Lomuscio, V. Malvone, E. Yu, Approximating perfect recall when model checking strategic abilities: Theory and applications, J. Artif. Intell. Res. 73 (2022) 897–932. URL: https://doi.org/10.1613/jair.1.12539. doi:`10.1613/jair.1.12539`.

[28] N. Alechina, B. Logan, H. N. Nguyen, F. Raimondi, L. Mostarda, Symbolic model-checking for resource-bounded ATL, in: AAMAS 2015, 2015, pp. 1809–1810.

[29] M. Faella, M. Napoli, M. Parente, Graded alternating-time temporal logic, Fundam. Informaticae 105 (2010) 189–210. URL: https://doi.org/10.3233/FI-2010-363. doi:`10.3233/FI-2010-363`.

[30] B. Aminof, V. Malvone, A. Murano, S. Rubin, Graded modalities in strategy logic, Inf. Comput. 261 (2018) 634–649. URL: https://doi.org/10.1016/j.ic.2018.02.022. doi:`10.1016/j.ic.2018.02.022`.

[31] V. Malvone, F. Mogavero, A. Murano, L. Sorrentino, Reasoning about graded strategy quantifiers, Information and Computation 259 (2018) 390 – 411.

Explicit and Symbolic Approaches for Parity Games

Antonio Di Stasio

Department of Computer Science, University of Oxford, UK

Abstract

In this paper, we review a broad investigation of the symbolic approach for solving Parity Games. Specifically, we implement in a tool, called `SymPGSolver`, four symbolic algorithms to solve Parity Games and compare their performances to the corresponding explicit versions for different classes of games. By means of benchmarks, we show that for random games, even for constrained random games, explicit algorithms actually perform better than symbolic algorithms. The situation changes, however, for structured games, where symbolic algorithms seem to have the advantage. This suggests that when evaluating algorithms for parity-game solving, it would be useful to have real benchmarks and not only random benchmarks, as the common practice has been.

Keywords

Parity Games, Symbolic Algorithms

Parity games (PGs) [1] are abstract games with a key role in automata theory and formal verification [2, 3, 4, 5, 6]. In the basic setting, parity games are two-player, turn-based, played on directed graphs whose nodes are labeled with priorities (also called, *colors*) and players have perfect information about the adversary moves. The two players, Player 0 and Player 1, take turns moving a token along the edges of the graph starting from a designated initial node. Thus, a play induces an infinite path and Player 0 wins the play if the smallest priority visited infinitely often is even; otherwise, Player 1 wins the play.

In formal system design [7, 3, 5, 8] parity games arise as a natural evaluation machinery for the automatic synthesis and verification of distributed and reactive systems [9, 10, 11], as they allow to express liveness and safety properties in a very elegant and powerful way [12]. Specifically, in model-checking, one can check the correctness of a system with respect to a desired behavior, that is, a *Kripke structure*, by checking whether a model of the system is correct with respect to a formal specification of its behavior. In case the specification is given as a μ-calculus formula [13], the model checking question can be rephrased, in linear-time, as a parity game [1]. Then, a parity game solver can be used as a model checker for a μ-calculus specification (and vice-versa), as well as for fragments such as CTL, CTL*, and the like.

In the automata-theoretic approach to μ-calculus model checking, under a linear-time translation, one can also reduce the verification problem to a question about automata. More precisely, one can take the product of the model and an alternating tree automaton accepting all tree models of the specification. This product can be defined as an alternating word parity automaton

---

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

antonio.distasio@cs.ox.uk (A. Di Stasio)
https://antoniodistasio.github.io/ (A. Di Stasio)
0000-0001-5475-2978 (A. Di Stasio)

CEUR Workshop Proceedings (CEUR-WS.org)

over a singleton alphabet, and the system is correct with respect to the specification iff this automaton is nonempty [5]. It has been proved there that the nonemptiness problems for nondeterministic tree parity automata and alternating word parity automata over a singleton alphabet are equivalent and that their complexities coincide. Hence, algorithms for the solution of the μ-calculus model checking problem, parity games, and the emptiness problem for parity automata can be interchangeably used to solve any of these problems, as they are linear-time equivalent.

The problem of deciding if Player 0 has a winning strategy (i.e., can induce a winning play) in a given parity game is known to be in UPTIME $\cap$ CoUPTIME [14]; whether a polynomial time solution exists is a long-standing open question [6]. Several algorithms to solve PGs have been proposed aiming to tighten the asymptotic complexity of the problem, as well as to work well in practice. Well known are *Recursive* (`RE`) [15], small-progress measures (`SPM`) [16], and `APT` [2, 17], the latter originated to deal with the emptiness of parity automata. Recently, Calude et al. [18] have given a major breakthrough providing a quasi-polynomial time algorithm for solving parity games that runs in time $O(n^{\lceil log(c)+6 \rceil})$. Previously, the best known algorithm for parity games was Dominion Decomposition [19] which could solve parity games in $O(n^{\sqrt{n}})$, so this new result represents a significant advance in the understanding of parity games. Notably, all these algorithms are *explicit*, that is, they are formulated in terms of the underlying game graphs. Due to the exponential growth of finite-state systems, and, consequently, of the corresponding game graphs, the state-explosion problem limits the scalability of these algorithms in practice. Hence for the analysis of large finite-state systems symbolic algorithms are necessary.

Symbolic algorithms are an efficient way to deal with extremely large graphs. They avoid explicit access to graphs by using a set of predefined operations that manipulate Binary Decision Diagrams (BDDs) [20] representing these graphs. This enables handling large graphs succinctly, and, in general, it makes symbolic algorithms scale better than explicit ones. For example, in hardware model checking symbolic algorithms enable going from millions of states to 10^{20} states and more [21, 22]. In contrast, in the context of PG solvers, symbolic algorithms have been only marginally explored. In this direction we just mention a symbolic implementation of `RE` [23, 24], which, however, has been done for different purposes and no benchmark comparison with the explicit version has been carried out. Other works close to this topic and worth mentioning are [25, 26], where a symbolic version of `SPM` has been theoretically studied but not implemented.

In [27, 28] a first broad investigation of the symbolic approach for solving PGs is provided. We implement four symbolic algorithms and compare their performances to the corresponding explicit versions for different classes of PGs [29]. Specifically, we implement in a new tool, called `SymPGSolver`[1], the symbolic versions of `RE`, `APT`, and two variants of `SPM`. The tool also allows to generate random games, as well as compare the performance of different symbolic algorithms.

Our analysis started from constrained random games [30]. The results show that on these games the explicit approach is better than the symbolic one, exhibiting a different behavior than the one showed in [30]. To gain a fuller understanding of the performances of the symbolic and the explicit algorithms, we have further tested the two approaches on structured games. Precisely, we have considered ladder games, clique games, as well as game models coming from

---

[1] The tool is available for download from https://github.com/antoniodistasio/sympgsolver

practical model-checking problems.

Ladder Games. In a ladder game, every node in P_i has priority i. In addition, each node $v \in \mathrm{P}$ has two successors: one in P_0 and one in P_1, which form a node pair. Every pair is connected to the next pair forming a ladder of pairs. Finally, the last pair is connected to the top. The parameter m specifies the number of node pairs. Formally, a ladder game of index m is $\mathcal{G} = (\mathrm{P}_0, \mathrm{P}_1, Mv, \mathsf{p})$ where $\mathrm{P}_0 = \{0, 2, \ldots, 2m-2\}$, $\mathrm{P}_1 = \{1, 3, \ldots, 2m-1\}$, $Mv = \{(v, w) | w \equiv_{2m} v + i$ for $i \in \{1, 2\}\}$, and $\mathsf{p}(v) = v \; mod \; 2$. Tables 1 and 2 reports the benchmarks.

m	SRE	SAPT	SSP	SSP2
1,000	0	0.00013	24.86	0.47
10,000	0.00009	0.00016	$\mathtt{abort}_T$	41.22
100,000	0.0001	0.00018	$\mathtt{abort}_T$	$\mathtt{abort}_T$
1,000,000	0.00012	0.00022	$\mathtt{abort}_T$	$\mathtt{abort}_T$
10,000,000	0.00015	0.00025	$\mathtt{abort}_T$	$\mathtt{abort}_T$

Table 1
Runtime executions of the symbolic algorithms on ladder games.

m	RE	APT	SPM
1,000	0.0007	0.0006	0.002
10,000	0.006	0.005	0.0017
100,000	0.057	0.054	0.18
1,000,000	0.59	0.56	1.84
10,000,000	6.31	5.02	20.83

Table 2
Runtime executions of the explicit algorithms on ladder games.

Benchmarks indicate that SRE and SAPT outperform their explicit versions, showing an excellent runtime execution even on fairly large instances. Indeed, while RE needs 6.31 seconds for games with index $m = 10M$, SRE takes just 0.00015 seconds. Tests also show that SSP and SSP2 have yet the worst performance.

Clique Games. Clique games are fully connected games without self-loops, where P_0 (*resp.,* P_1) contains the nodes with an even index (*resp., odd*) and each node $v \in \mathrm{P}$ has as priority the index of v. An important feature of the clique games is the high number of cycles, which may pose difficulties for certain algorithms. Formally, a clique game of index n is $\mathcal{G} = (\mathrm{P}_0, \mathrm{P}_1, Mv, \mathsf{p})$ where $\mathrm{P}_0 = \{0, 2, \ldots, n-2\}$, $\mathrm{P}_1 = \{1, 3, \ldots, n-1\}$, $Mv = \{(v, w) | v \neq w\}$, and $\mathsf{p}(v) = v$. Benchmarks on clique games are reported in Tables 3 and 4.

n	SRE	SAPT	SSP	SSP2
2,000	0.007	0.003	5.53	$\mathtt{abort}_T$
4,000	0.018	0.008	19.27	$\mathtt{abort}_T$
6,000	0.025	0.012	39.72	$\mathtt{abort}_T$
8,000	0.037	0.017	76.23	$\mathtt{abort}_T$

Table 3
Runtime executions of the symbolic algorithms on clique games

n	RE	APT	SPM
2,000	0.021	0.0105	0.0104
4,000	0.082	0.055	0.055
6,000	0.19	0.21	0.22
8,000	0.35	0.59	0.63

Table 4
Runtime executions of the explicit algorithms on clique games

The main result we obtain from our comparisons is that for random games, and even for constrained random games, explicit algorithms actually perform better than symbolic ones, most likely because BDDs do not offer any compression for random sets. The situation changes, however, for structured games, where symbolic algorithms sometimes outperform explicit algorithms. This is similar to what has been observed in the context of model checking [31].

n	Pr	Property	SRE	SAPT	SSP	SSP2	RE	APT	SPM	WS	DS
14,065	3	ND	0.00009	0.00006	3.30	0.0001	0.004	0.004	0.029	2	2
17,810	3	IORD1	0.0003	0.0005	$\mathtt{abort}_T$	85.4	0.006	0.006	0.037	2	2
34,673	3	IORW	0.0006	0.0008	164.73	56.44	0.015	0.014	0.053	2	2
2,589,056	3	ND	0.0002	$\mathtt{abort}_T$	$\mathtt{abort}_T$	0.29	1.02	0.93	9.09	4	2
3,487,731	3	IORD1	$\mathtt{abort}_T$	$\mathtt{abort}_T$	$\mathtt{abort}_T$	$\mathtt{abort}_T$	1.81	1.4	17.45	4	2
6,823,296	3	IORW	0.3	$\mathtt{abort}_T$	$\mathtt{abort}_T$	$\mathtt{abort}_T$	3.87	3.13	22.26	4	2

Table 5
SWP (Sliding Window Protocol)

n	Pr	Property	SRE	SAPT	SSP	SSP2	RE	APT	SPM	DS
81,920	3	ND	0.00002	31.69	1.37	0.0016	0.031	0.034	0.22	2
88,833	3	IORD1	0.0027	0.003	$\mathtt{abort}_T$	$\mathtt{abort}_T$	0.036	0.0038	0.27	2
170,752	3	IORW	14.37	98.4	$\mathtt{abort}_T$	$\mathtt{abort}_T$	0.07	0.07	0.47	2
289,297	3	ND	0.0001	154.89	12.3	0.0058	0.13	0.12	1.34	4
308,737	3	IORD1	0.0088	0.009	$\mathtt{abort}_T$	$\mathtt{abort}_T$	0.14	0.13	1.37	4
607,753	3	IORW	43.7	$\mathtt{abort}_T$	$\mathtt{abort}_T$	$\mathtt{abort}_T$	0.29	0.27	2.06	4

Table 6
OP (Onebit Protocol)

n	Pr	Property	SRE	SAPT	SSP	SSP2	RE	APT	SPM	DS
328	1	ND	0.00002	0.002	0.005	0.00002	0.0001	0.0001	0.0004	2
308	1	safety	0.00002	0.003	0.028	0.00002	0.0001	0.0001	0.0004	2
655	3	liveness	0.00008	0.0001	5.52	0.09	0.0003	0.0002	0.001	2
51.220	1	safety	0.0001	1.48	32.14	0.00002	0.01	0.01	0.09	4
53.638	1	ND	0.0001	0.2	4.67	0.0001	0.017	0.015	0.07	4
107,275	3	liveness	0.005	0.001	$\mathtt{abort}_T$	$\mathtt{abort}_T$	0.03	0.03	0.18	4

Table 7
Lift (Lifting Truck)

Finally, we evaluate the symbolic and explicit approaches on some practical model checking problems as in [32]. Specifically, we use models coming from: the Sliding Window Protocol (SWP) with window size (WS) of 2 and 4 (WS represents the boundary of the total number of packets to be acknowledged by the receiver), the Onebit Protocol (OP), and the Lifting Truck (Lift). The properties we check on these models concern: absence of deadlock (ND), a message of a certain type (d1) is received infinitely often (IORD1), if there are infinitely many read steps then there are infinitely many write steps (IORW), liveness, and safety. Note that, in all benchmarks, data size (DS) denotes the number of messages.

As we can see, by comparing Tables 5, 6, and 7, the experiments indicate more nuanced relationship between the symbolic and explicit approaches. Indeed, they show a different behavior depending on the protocol and the property we are checking. Overall, we note that

`SRE` outperforms the other symbolic algorithms in all protocols, although the advantage over `RE` is discontinued. Specifically, `SRE` is the best performing in checking absence of deadlock in all three protocols, but for IORD1 in the SWP protocol with $WS = 2$, or for IORW in the OP protocol, `RE` exhibits a significant advantage. Differently, `SAPT` and `SSP2` show better performances on a smaller number of properties. Moreover, the results highlights that `SSP` exhibits the worst performances in all protocols and properties.

We take this as an important development because it suggests a methodological weakness in this field of investigation, due to the excessive reliance on random benchmarks. We believe that, in evaluating algorithms for PG solving, it would be useful to have real benchmarks and not only random benchmarks, as the common practice has been. This would lead to a deeper understanding of the relative merits of PG solving algorithms, both explicit and symbolic.

Acknowledgments

We thank our co-authors on the publications mentioned in this communication: Aniello Murano and Moshe Y. Vardi. This work is partially supported by the ERC Advanced Grant WhiteMech (No. 834228), by the EU ICT-48 2020 project TAILOR (No. 952215), and by the PRIN project RIPER (No. 20203FFYLK).

References

[1] E. Emerson, C. Jutla, Tree Automata, μ-Calculus and Determinacy, in: 32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991, 1991, pp. 368–377.

[2] O. Kupferman, M. Y. Vardi, Weak Alternating Automata and Tree Automata Emptiness, in: Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, 1998, pp. 224–233.

[3] E. Clarke, E. Emerson, Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic, in: Logics of Programs, Workshop, Yorktown Heights, New York, USA, May 1981, LNCS 131, 1981, pp. 52–71.

[4] P. Cermák, A. Lomuscio, A. Murano, Verifying and synthesising multi-agent systems against one-goal strategy logic specifications, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, 2015, pp. 2038–2044.

[5] O. Kupferman, M. Vardi, P. Wolper, An Automata Theoretic Approach to Branching-Time Model Checking, J. ACM 47 (2000) 312–360.

[6] T. Wilke, Alternating Tree Automata, Parity Games, and Modal μ-Calculus, Bulletin of the Belgian Mathematical Society Simon Stevin 8 (2001) 359.

[7] E. Clarke, O. Grumberg, D. Peled, Model Checking., MIT Press, 2002.

[8] J. Queille, J. Sifakis, Specification and Verification of Concurrent Programs in Cesar, in: International Symposium on Programming, 5th Colloquium, Torino, Italy, April 6-8, 1982, Proceedings, LNCS 137, 1982, pp. 337–351.

[9] O.Kupferman, M.Vardi, P.Wolper, Module Checking, Information and Computation. 164 (2001) 322–344.

[10] W. Thomas, Facets of Synthesis: Revisiting Church's Problem, in: Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009., LNCS 5504, 2009, pp. 1–14.

[11] B. Aminof, O. Kupferman, A. Murano, Improved Model Checking of Hierarchical Systems, Inf. Comput. 210 (2012) 68–86.

[12] F. Mogavero, A. Murano, L. Sorrentino, On Promptness in Parity Games, in: Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013., LNCS 8312, 2013, pp. 601–618.

[13] D. Kozen, Results on the Propositional μ-Calculus, Theoretical Computer Science 27 (1983) 333–354.

[14] M. Jurdzinski, Deciding the Winner in Parity Games is in UP $\cap$ co-Up, Inf. Process. Lett. 68 (1998) 119–124.

[15] W. Zielonka, Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees, Theor. Comput. Sci. 200 (1998) 135–183.

[16] M. Jurdzinski,Small Progress Measures for Solving Parity Games, in: STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings, LNCS 1770, 2000, pp. 290–301.

[17] A. Di Stasio, A. Murano, G. Perelli, M. Y. Vardi, Solving parity games using an automata-based algorithm, in: Implementation and Application of Automata - 21st International Conference, CIAA 2016, Seoul, South Korea, July 19-22, 2016., 2016, pp. 64–76.

[18] C. S. Calude, S. Jain, B. Khoussainov, W. Li, F. Stephan, Deciding parity games in quasipolynomial time, in: STOC 2017, 2017, pp. 252–263.

[19] M. Jurdzinski, M. Paterson, U. Zwick, A deterministic subexponential algorithm for solving parity games, in: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006, ACM Press, 2006, pp. 117–123.

[20] R. E. Bryant, Graph-based algorithms for boolean function manipulation, IEEE Trans. Comput. (1986) 677–691.

[21] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, L. J. Hwang, Symbolic model checking: 10ˆ20 states and beyond, in: Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990, 1990, pp. 428–439.

[22] K. L. McMillan, Symbolic Model Checking, Kluwer Academic Publishers, 1993.

[23] G. Kant, J. van de Pol, Generating and solving symbolic parity games, in: Proceedings 3rd Workshop on GRAPH Inspection and Traversal Engineering, GRAPHITE 2014, Grenoble, France, 5th April 2014, 2014, pp. 2–14.

[24] M. Bakera, S. Edelkamp, P. Kissmann, C. D. Renner, Solving μ-calculus parity games by symbolic planning, in: Model Checking and Artificial Intelligence, 5th International Workshop, MoChArt 2008, Patras, Greece, July 21, 2008., 2008, pp. 15–33.

[25] D. Bustan, O. Kupferman, M. Y. Vardi, A measured collapse of the modal μ-calculus alternation hierarchy, in: STACS 2004, 21st Annual Symposium on Theoretical Aspects of

Computer Science, Montpellier, France, March 25-27, 2004, Proceedings, 2004, pp. 522–533.

[26] K. Chatterjee, W. Dvorák, M. Henzinger, V. Loitzenbauer, Improved set-based symbolic algorithms for parity games, in: 26th EACSL Annual Conference on Computer Science Logic, CSL 2017, August 20-24, 2017, Stockholm, Sweden, 2017, pp. 18:1–18:21.

[27] A. Di Stasio, A. Murano, M. Y. Vardi, Solving parity games: Explicit vs symbolic, in: Implementation and Application of Automata - 23rd International Conference, CIAA 2018, Charlottetown, PE, Canada, July 30 - August 2, 2018, Proceedings, 2018, pp. 159–172.

[28] A. D. Stasio, Reasoning about LTL Synthesis over finite and infinite games, Ph.D. thesis, University of Naples Federico II, Italy, 2018.

[29] T. van Dijk, Oink: An implementation and evaluation of modern parity game solvers, in: Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, LNCS 10805, Springer, 2018, pp. 291–308.

[30] D. Tabakov, Evaluation of Explicit and Symbolic Automata-Theoretic Algorithm, Master's thesis, Rice University, 2005.

[31] C. Eisner, D. A. Peled, Comparing symbolic and explicit model checking of a software system, in: Model Checking of Software, 9th International SPIN Workshop, Grenoble, France, April 11-13, 2002, Proceedings, 2002, pp. 230–239.

[32] J. A. Keiren, Benchmarks for parity games, in: FSEN 2015, 2015, pp. 127–142.

Give Me a Hand: How to Use Model Checking for Multi-Agent Systems to Help Runtime Verification and Vice Versa

Angelo Ferrando[1,*], Vadim Malvone[2]

[1]University of Genoa, Italy
[2]Telecom Paris, France

Abstract
In this paper, we review the history of model checking and runtime verification on multi-agent systems by recalling the results obtained in the two research areas. Then, we present some past, present and future directions to combine these techniques in the two possible sides, that is by using model checking for multi-agent systems to solve runtime verification problems and vice versa.

Keywords
LaTeX class, paper template, paper formatting, CEUR-WS

1. Introduction

Software systems cannot be trusted. Even though this sounds as a bold statement, it is most of the time the case. A software system, to be considered trustworthy, has to offer some guarantees to the end user. Amongst these guarantees, correctness is by far one of the most challenging to demonstrate. However, existent solutions to proof software correctness mainly focus on monolithic systems. So, systems that do not usually present any kind of autonomy, nor distribution, whatsoever. Unfortunately (in some sense), nowadays, artificial intelligent systems can be found everywhere. Thus, techniques to tackle their verification in order to establish their correctness also need to be revised (and adapted).

When we talk about software verification, we mainly refer to standard approaches such as: testing, simulation, and formal verification. Testing and simulation have one main issue: they can detect errors but can not determine their absence. To overcome this problem, *formal verification* results to be very useful. This approach provides a formal-based methodology to model systems, specify properties, and verify that a system satisfies a given specification.

In formal verification, the specification is usually based on temporal logics. The latter can describe the order of events without introducing the time explicitly. In temporal logics, we mainly distinguish between linear- and branching-time logics, which reflect the underlying

---

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

[*]Corresponding author.

angelo.ferrando@unige.it (A. Ferrando); vadim.malvone@telecom-paris.fr (V. Malvone)
0000-0002-8711-4670 (A. Ferrando); 0000-0001-6138-4229 (V. Malvone)

CEUR Workshop Proceedings (CEUR-WS.org)

nature of the time we consider. The most popular temporal logics are LTL (linear-time temporal logic) [34], CTL (computation tree logic) [14], and their extension CTL^* [18]. An outstanding development in the area of temporal logics has been the discovery of algorithmic methods to verify properties of finite-state systems represented by Kripke structures [28]. Hence, the formal verification of a system modelled by a Kripke structure M with respect to a temporal logic specification φ can be rephrased as "Is M a model of φ?", which explains the name *model checking* (MC), as it was coined by Clarke and Emerson in [14].

Naturally, model checking is not the only existent formal verification technique. Specifically, another verification approach focused on a more dynamic perspective (execution of the system rather than verification of an abstraction of the latter) is called Runtime Verification [32].

Runtime verification (RV) is being pursued as a lightweight verification technique bridging static verification techniques, such as MC, and testing. One of the main distinguishing features of RV is due to its nature of being performed at runtime, which opens up the possibility to act whenever incorrect behavior of a software system is detected. A fault is defined as the deviation between the current behavior and the expected behavior of the system [32, 16]. A fault might lead to a failure, but not necessarily. An error, on the other hand, is a mistake made by a human that results in a fault and possibly in a failure. Runtime verification is the discipline of computer science that deals with the study, development, and application of those verification techniques that allow checking whether a run of a system under scrutiny satisfies or violates a given correctness property.

In [20], we presented an initial vision on possible overlapping of RV and MC, especially in the area of Multi-Agent Systems (MAS); which are distributed systems comprised of intelligent components (called agents) that can be deployed to solve complex tasks in an autonomous fashion (which may, or may not, involve cooperation and communication amongst the agents). We focused on MAS for their implicit complexity, and consequently hard verification problem. Nonetheless, in [20] we mainly focused on the verification aspects of combining RV and MC. Instead, in here, we present some novel results in that direction, but we also focus on the synergy between RV and MAS. Thus, not only focusing on how RV can be deployed to verify MAS, but also on how the verification for MAS can be exploited to support RV.

In the rest of the paper we first discuss the state of the art of model checking and runtime verification for MAS (Section 2). Then, in Section 3 we present our results to use runtime verification to reduce the model checking complexity on MAS and conclude with future directions. Finally, in Section 4 we present model checking techniques to help runtime verification and conclude with some future works.

2. State of the art

Model Checking for MAS. One of the most important developments in this field is *Alternating-Time Temporal Logic* (ATL), introduced by Alur, Henzinger, and Kupferman [4]. Such a logic allows to reason about strategies of agents having the satisfaction of temporal goals as payoff criterion. More formally, it is obtained as a generalization of CTL, in which the existential E and the universal A *path quantifiers* are replaced with *strategic modalities* of the form $\langle\langle\Gamma\rangle\rangle$ and $[\![\Gamma]\!]$, where Γ is a set of *agents*. Despite its expressiveness, ATL suffers from the strong limitation that strategies are treated only implicitly in the semantics of such modalities. This restriction makes the logic less suited to formalize several important solution concepts, such as the *Nash Equilibrium.* These considerations led to the introduction of *Strategy Logic* (SL) [33], a more powerful formalism for strategic reasoning. As a key aspect, this logic treats strategies as *first-order objects* that can be determined by means of the existential $\exists x$ and universal $\forall x$ quantifiers, which can be respectively read as *"there exists a strategy x"* and *"for all strategies x"*. Notably, a strategy is a generic conditional plan that at each step of the game prescribes an action. With more detail, there are two main classes of strategies: memoryless and memoryful. In the former case, agents choose an action by considering only the current game state while, in the latter case, agents choose an action by considering the full history of the game. Therefore, this plan is not intrinsically glued to a specific agent, but an explicit binding operator (a, x) allows to link an agent a to the strategy associated with a variable x. Unfortunately, the high expressivity of SL comes at a price. Indeed, it has been proved that the model-checking problem for SL becomes non-elementary complete and the satisfiability undecidable. To gain back elementariness, several fragments of SL have been considered. Among the others, Strategy Logic with Simple-Goals [6] considers SL formulas in which strategic operators, bindings operators, and temporal operators are coupled. It has been shown that Strategy Logic with Simple-Goals strictly subsume ATL and its MC problem is P-COMPLETE, as it is for ATL. To conclude this section, we want to focus on a key aspect in MAS: the agents' visibility. Specifically, we distinguish between *perfect* and *imperfect* information games [35]. The former corresponds to a basic setting in which every agent has full knowledge about the game. However, in real-life scenarios it is common to have situations in which agents have to play without having all relevant information at hand. In computer science these situations occur for example when some variables of a system are internal/private and not visible to an external environment [29, 13]. In game models, the imperfect information is usually modelled by setting an indistinguishability relation over the states of the game [29, 35]. This feature deeply impacts on the MC complexity. For example, ATL becomes undecidable in the context of imperfect information and memoryful strategies [17]. To overcome this problem, some works have either focused on an approximation to perfect information [8, 11] or developed new notions on strategies [7, 26, 27, 10, 12, 9].

Runtime Verification for MAS. In [2], the authors presented a framework to verify at runtime agent interaction protocols (AIP). The formalism used in this work allows the introduction of variables, that are then used to constrain the expected behavior in a more expressive way. In [19], the same authors proposed an approach to verify at runtime AIP using multiple monitors. This is obtained by decentralizing the global specification (specified as a Trace Expression [1]), which is used to represent the global protocol, into partial specifications denoting the single agents' perspective. In [5, 36], other works on runtime verification of agent interactions are proposed, and in [30] a framework for dynamic adaptive MAS (DAMS-RV) based on an adaptive feedback loop is presented. Other approaches to MAS RV include the proposals spin-off from the SOCS project where the SCIFF computational logic framework [3] is used to provide the semantics of social integrity constraints. To model MAS interaction, expectation-based semantics specifies the links between observed and expected events, providing a means to test runtime conformance of an actual conversation with respect to a given interaction protocol [38]. Similar work has been performed using commitments [15].

3. Runtime verification gives a hand to Model checking for MAS

The model checking problem for ATL giving a generic MAS is known to be undecidable. Nonetheless, decidable fragments exist. Indeed, model checking ATL under perfect information is PTIME-complete [4], while under imperfect information and imperfect recall is PSPACE [37]. Unfortunately, MAS usually have imperfect information, and when memory is needed to achieve the goals, the resulting model checking problem becomes undecidable. Given the relevance of the imperfect information setting, even partial solutions to the problem are useful.

The only existent work on exploiting RV to formally verify the strategic behaviour in MAS is [24, 22]. In such work, given an ATL formula φ and a model of MAS M, the procedure extracts all the sub-models of M with perfect information that satisfy a sub-formula of φ. Then, runtime monitors are used to check if the remaining part of φ can be satisfied at execution time. If this is the case, we conclude at runtime the satisfaction of φ for the corresponding system execution. Note that, this does not imply that the system satisfies φ, indeed future executions may violate φ. The formal result over φ only concerns the current execution, and how it has behaved in it. However, the following preservation results holds.

Lemma 1. *Given a model M and an ATL formula φ, for any history h of M starting in s_I, we have that:*

$$Monitor_{\varphi_{LTL}}(h) = \top \ \Rightarrow \ M, s_I \models \varphi_{Ag}$$

$$Monitor_{\varphi_{LTL}}(h) = \bot \ \Rightarrow \ M, s_I \not\models \varphi_{\emptyset}$$

where φ_{LTL} is the variant of φ where all strategic operators are removed, φ_{Ag} is the variant of φ where all strategic operators are converted into $\langle\langle Ag\rangle\rangle$, $\varphi_{\emptyset}$ is the variant of φ where all strategic operators are converted into $\langle\langle\emptyset\rangle\rangle$.

Lemma 1 shows a preservation result from RV to ATL model checking that needs to be discussed. If our monitor returns true we have two possibilities:

1 The procedure found an under-approximation sub-model in which the original formula φ is satisfied then it can conclude the verification procedure by using RV only by checking that the atom representing φ holds in the initial state of the history h given in input;
2 A sub-formula φ' is satisfied in an under-approximation sub-model and at runtime the formula φ_{Ag} holds on the history h given in input.

While case (1) gives a preservation result for the formula φ given in input, case (2) checks formula φ_{Ag} instead of φ. That is, it substitutes Ag as coalition for all the strategic operators of φ but the ones in φ'. So, our procedure approximates the truth value by considering the case in which all the agents in the game collaborate to achieve the objectives not satisfied in the model checking phase. That is, while in [8, 11] the approximation is given in terms of information, in [7] is given in terms of memory of strategies, and in [21] the approximation is given by generalizing the logic, here we give results by approximating the coalitions. So, the main limitation of this approach concerns this aspect. Furthermore, we recall that the procedure produces always results, even partial. This aspect is strongly relevant in concrete scenario in which there is the necessity to have some sort of verification results. For example, in the context of swarm robots, with this procedure we can verify macro properties such as "the system works properly" since we are able to guarantee fully collaboration between agents because this property is relevant and desirable for each agent in the game. The same reasoning described above, can be applied in a complementary way for the case of over-approximation sub-models and the falsity.

Note that this is the first attempt of using runtime verification to verify strategic properties on MAS. Thus, even though the solution might not be optimal, it is a milestone for the corresponding lines of research. Additional works will be done to improve the technique and, above all, its implementation. For instance, we are planning to extend this work by considering a more predictive flavour.

4. Model checking for MAS gives a hand to Runtime verification

Runtime Verification is built on the assumption of perfect information over the system, that is, the monitor checking the system can perceive everything. Unfortunately, this is not always the case, especially when the system under analysis contains rational/autonomous components and is deployed in real-world environments with possibly faulty sensors. In [23], an extension of the standard Runtime Verification of Linear Temporal Logic properties to consider scenarios with imperfect information is presented; along with all the engineering steps necessary to update the verification pipeline. Moreover, a corresponding implementation is proposed and applied to a case study involving robotic systems. In particular, [23] defines the notion of imperfect information w.r.t. the monitor's visibility over the system, and then re-engineers the LTL monitor's synthesis pipeline to recognise such visibility information.

In [23], imperfect information is specified by means of a indistinguishability relation $\sim$ over the atomic propositions. Intuitively, given two atomic propositions p and q, it is said that they are indistinguishable if and only if $p \sim q$.

To handle the verification process in the imperfect information scenario, the standard RV approach needs to be extended. This extension is based on the notion of duplication of the atoms. The latter is used in order to make the truth value of each atomic proposition explicit.

The new monitor is defined as follows.

Definition 1 (Monitor with imperfect information). *Given an LTL formula φ and a visible* *history h_v, a monitor with imperfect information is so defined:*

$$
Monitor_{\varphi}^{v}(h_v) = \begin{cases} \top & h_v \in \mathcal{L}(\varphi) \wedge h_v \notin \mathcal{L}(\neg\varphi) \wedge h_v \notin \mathcal{L}(\otimes\varphi) \\ \bot & h_v \notin \mathcal{L}(\varphi) \wedge h_v \in \mathcal{L}(\neg\varphi) \wedge h_v \notin \mathcal{L}(\otimes\varphi) \\ uu & h_v \notin \mathcal{L}(\varphi) \wedge h_v \notin \mathcal{L}(\neg\varphi) \wedge h_v \in \mathcal{L}(\otimes\varphi) \\ ?_{\not\bot} & h_v \in \mathcal{L}(\varphi) \wedge h_v \notin \mathcal{L}(\neg\varphi) \wedge h_v \in \mathcal{L}(\otimes\varphi) \\ ?_{\not\top} & h_v \notin \mathcal{L}(\varphi) \wedge h_v \in \mathcal{L}(\neg\varphi) \wedge h_v \in \mathcal{L}(\otimes\varphi) \\ ? & h_v \in \mathcal{L}(\varphi) \wedge h_v \in \mathcal{L}(\neg\varphi) \wedge h_v \in \mathcal{L}(\otimes\varphi) \end{cases}
$$

Where $\mathcal{L}(\varphi)$ is the language of histories satisfying φ, $\mathcal{L}(\neg\varphi)$ is the language of histories violating φ, and finally, $\mathcal{L}(\otimes\varphi)$ is the language of histories making φ undefined (because of lack of information).

In what follows, we provide two preservation results from the monitor with imperfect information to the one with perfect information.

Lemma 2. *Given a finite history σ, a monitor with its visibility $Monitor_{\varphi}^{v}(h)$, and a general monitor $Monitor_{\varphi}(h)$, we have that:*

$$
\text{if } Monitor_{\varphi}^{v}(h_v) = \top \text{ then } Monitor_{\varphi}(h) = \top \\
\text{if } Monitor_{\varphi}^{v}(h_v) = \bot \text{ then } Monitor_{\varphi}(h) = \bot
$$

The above results can be extended to consider multi-monitors and solved by using formal verification for MAS. In fact, even though a single monitor does not have access to all the information it needs, multiple monitors might. As in standard distributed RV [25], we are planning to explore this idea as well, but focusing on the theoretical foundations of information sharing amongst monitors. Specifically, we are going to show how this can be specified as a multi-agent problem; where each monitor is denoted as an agent with the goal of gathering all the information needed to carry out the verification of its formal property. By representing the information sharing as a multi-agent problem, we could gain from different viewpoints. In particular, we could exploit existing techniques, used for strategic reasoning [31], to guide the information sharing amongst the monitors. Moreover, by denoting monitors as agents, we would recognise their autonomy on what concern the sharing of private information. For instance, a monitor could be the only one with the access to a certain resource, and it is not realistic to assume it will freely share such information. This because the act of sharing information is not free of charge, since it requires to both consume computation time and bandwidth. For this, and other reasons, it is of paramount importance to take into consideration the cost of sharing. We are going to show how such cost can be ported into the multi-agent problem, and how it can guide the selection of the information sharing strategy of the agents (*i.e.*, the monitors).

In particular, we are going to study the theoretical foundation of information sharing amongst runtime monitors. We will tackle this by porting the problem into a multi-agent setting. By dosing so, we apply existing formal verification techniques, such as model checking [31], to exploit the strategic reasoning of the agents to overcome the information sharing problem. We will not only present theoretical results, but we will also propose an implementation prototype, as a proof of concept.

References

[1] D. Ancona, A. Ferrando, and V. Mascardi. Comparing trace expressions and linear temporal logic for runtime verification. In *TPFM*, pages 47–64, 2016.

[2] D. Ancona, A. Ferrando, and V. Mascardi. Parametric runtime verification of multiagent systems. In *AAMAS*, pages 1457–1459, 2017.

[3] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. The *S*ciff abductive proof-procedure. In *AI*IA*, pages 135–147, 2005.

[4] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *JACM*, 49(5):672–713, 2002.

[5] N. Bakar and A. Selamat. Runtime verification of multi-agent systems interaction quality. In *ACIIDS*, pages 435–444, 2013.

[6] F. Belardinelli, W. Jamroga, D. Kurpiewski, V. Malvone, and A. Murano. Strategy logic with simple goals: Tractable reasoning about strategies. In IJCAI, pages 88–94, 2019.

[7] F. Belardinelli, A. Lomuscio, and V. Malvone. Approximating perfect recall when model checking strategic abilities. In *KR2018*, pages 435–444, 2018.

[8] F. Belardinelli, A. Lomuscio, and V. Malvone. An abstraction-based method for verifying strategic properties in multi-agent systems with imperfect information. In *AAAI*, 2019.

[9] F. Belardinelli, A. Lomuscio, V. Malvone, and E. Yu. Approximating perfect recall when model checking strategic abilities: Theory and applications. *JAIR*, 73:897–932, 2022.

[10] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multi-agent systems with public actions against strategy logic. *AIJ*, 285:103302, 2020.

[11] F. Belardinelli and V. Malvone. A three-valued approach to strategic abilities under imperfect information. In *KR*, pages 89–98, 2020.

[12] R. Berthon, B. Maubert, A. Murano, S. Rubin and, M. Y. Vardi. Strategy Logic with Imperfect Information. *TOCL*, 22(1):1–51, 2021.

[13] R. Bloem, K. Chatterjee, S. Jacobs, and R. Könighofer. Assume-guarantee synthesis for concurrent reactive programs with partial information. In *TACAS*, pages 517–532, 2015.

[14] E.M. Clarke and E.A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *LP*, pages 52–71, 1981.

[15] F. Chesani, P. Mello, M. Montali, and P. Torroni. Commitment tracking via the reactive event calculus. In *IJCAI*, pages 91–96, 2009.

[16] N. Delgado, A. Gates, and S. Roach. A taxonomy and catalog of runtime software-fault monitoring tools. *IEEE TSE*, 30(12):859–872, 2004.

[17] C. Dima and F.L. Tiplea. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. Technical report, arXiv, 2011.

[18] E.A. Emerson and J.Y. Halpern. “Sometimes” and “Not Never” Revisited: On Branching Versus Linear Time. *JACM*, 33(1):151–178, 1986.

[19] A. Ferrando, D. Ancona, and V. Mascardi. Decentralizing MAS monitoring with decamon. In *AAMAS*, pages 239–248, 2017.

[20] A. Ferrando and V. Malvone. Combine model checking and runtime verification in multi-agent systems. In ICTCS, pages 302–310, 2021.

[21] A. Ferrando and V. Malvone. Towards the verification of strategic properties in multi-agent systems with imperfect information. *CoRR*, abs/2112.13621, 2021.

[22] A. Ferrando and V. Malvone. Strategy RV: A Tool to Approximate ATL Model Checking under Imperfect Information and Perfect Recall. In AAMAS, pages 1764–1766, 2021.

[23] A. Ferrando and V. Malvone. Runtime verification with imperfect information through indistinguishability relations. In SEFM, pages 335–351, 2022.

[24] A. Ferrando and V. Malvone. Towards the combination of model checking and runtime verification on multi-agent systems. In PAAMS, pages 140–152, 2022.

[25] A. Francalanza, J. A. Pérez, and C. Sánchez. Runtime verification for decentralised and distributed systems. In LRVIAT, pages 176–210, 2018.

[26] W. Jamroga, V. Malvone, and A. Murano. Natural strategic ability. *AIJ*, 277:103170, 2019.

[27] W. Jamroga, V. Malvone, and A. Murano. Natural Strategic Ability under Imperfect Information. In *AAMAS*, 962–970, 2019.

[28] S.A. Kripke. Semantical Considerations on Modal Logic. *APF*, 16:83–94, 1963.

[29] O. Kupferman and M.Y. Vardi. Module checking revisited. In *CAV*, pages 36–47, 1997.

[30] Y. J. Lim, G. Hong, D. Shin, E. Jee, and D.-H- Bae. A runtime verification framework for dynamically adaptive multi-agent systems. In *BigComp*, pages 509–512, 2016.

[31] A. Lomuscio and F. Raimondi. Model checking knowledge, strategies, and games in multi-agent systems. In *AAMAS*, pages 161–168, 2006.

[32] M. Leucker and C. Schallhart. A brief account of runtime verification. *JLAP*, 78(5):293 – 303, 2009.

[33] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning About Strategies: On the Model-Checking Problem. *ACM TCL*, 15(4): 34:1-34:47, 2014.

[34] A. Pnueli. The Temporal Logic of Programs. In *FCS*, pages 46–57, 1977.

[35] J. H. Reif. The complexity of two-player games of incomplete information. *JCSS*, 29(2):274–301, 1984.

[36] C. Roungroongsom and D. Pradubsuwun. Formal verification of multi-agent system based on jade: A semi-runtime approach. In *RAICT*, pages 297–306, 2015.

[37] P.Y. Schobbens. Alternating-Time Logic with Imperfect Recall. 85(2):82–93, 2004.

[38] P. Torroni, P. Yolum, M. P. Singh, M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, and P. Mello. Modelling interactions via commitments and expectations. In *HRMASS-DOM*, 2009.

Logics for Reasoning about Auctions

Munyque Mittelmann[1]

[1]*Università degli Studi di Napoli Federico II, Italy*

Abstract

In this paper, I summarize the results obtained in my thesis, whose aim was to investigate the use of formal methods for the specification, design, and evaluation of mechanisms, with a focus on auctions. We address such challenges by introducing logic-based approaches for representing and designing auction-based markets with strategic players. Firstly, for providing a foundation for general and automated auction playing in MAS, we propose a framework for representing auctions, denoted Auction Description Language (ADL). ADL addresses important dimensions of auction-based markets and is general enough to represent most auction settings. Second, we propose a novel approach for reasoning and designing new auctions (and, in general, mechanisms) based on formal methods. Such an approach for Automated Mechanism Design aims to automatically generate mechanisms from their specifications and verify them in relation to target economical properties. We demonstrate how this approach can express key concepts from Economic Theory (including Nash equilibrium and strategyproofness) and how it enables automatically generating optimal mechanisms from a quantitative logical specification.

Keywords

Strategic Reasoning, Auctions, Formal Methods, Multi-Agent Systems, Mechanism Design

1. Introduction

An auction is a popular mechanism that aggregates participants' bids into a social decision, usually expressed in terms of allocations and payments. Automated agents are widely used in auction-based markets but software agents are usually designed to act on a specific context, which prevents them from switching between different kinds of markets. For doing so, they should be able to "understand" the auction rules and reason about their own valuations and also about other players private information valuations. This limitation inspires the development of a lightweight logic-based language for representing the rules of an auction market.

More than describing, the design and evaluation of new auctions (and, more generally, mechanisms) is a central problem in multiagent settings [1]. In such setting, we need to be able to aggregate individual preferences, which are conflicting when agents are self-interested. More importantly, the mechanism should choose a socially desirable outcome and reach an equilibrium despite the fact that agents can lie about their preferences [2]. Although logic-based languages have been widely used for verification [3] and synthesis [4]

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy, November 29th, 2022, Udine, Italy.

✉ munyque.mittelmann@ut-capitole.fr (M. Mittelmann)
🌐 https://sites.google.com/view/mittelmann (M. Mittelmann)

of Multi-Agent Systems (MAS), the use of formal methods for reasoning about auctions under strategic behavior as well as automated mechanism design has not been much explored yet. An advantage in adopting such perspective lies in the high expressivity and generality of logics for strategic reasoning [5]. Moreover, by relying on precise semantics, formal methods provide tools for rigorously analysing the correctness of systems, which is important to improve trust in mechanisms generated by machines. The problem of formally reasoning about mechanisms requires to consider quantitative information (e.g., utilities and payments), private information about the participant's preferences, and complex strategic concepts (such as strategy dominance and equilibria).

My thesis [6] addressed such challenges by introducing logic-based approaches for representing and designing auction-based markets with strategic players. Our motivation is two fold: first, we aim to provide a foundation for general and automated auction playing in MAS, by establishing a logical framework to create a good balance between expressive power and computational efficiency. Second, we propose a novel approach based on formal methods for (i) reasoning about auctions under strategic behavior and (ii) Automated Mechanism Design. Such approach aims to automatically generate auctions from their specifications and verify them in relation to target economical properties.

2. Background and Context

Representation of Auctions There are numerous variants of auctions depending on the parameters considered, including the number of distinct items and copies as well as the number of sellers and buyers [7, 8]. For a fixed set of parameters, the protocol, i.e., the bidding, payment and allocation rules, may also differ. Building intelligent agents that can switch between different auctions and process their rules is a key issue for building automated auction-based marketplaces. In this scenario, the auctioneer should at first allow participants to express their preferences and second describe the rules governing the market. In relation to the formal representation of auction rules, we recall the descriptive auction language [9], which allows the specification of auctions by allowing players to bid using the XOR language. Rule-based approaches have also being used for representing single-dimensional auctions [10] and negotiation protocols [11]. Similarly, negotiation protocols have being handled with meta-languages [12], the Extensible Markup Language [13] and rule-markup languages [14]. Since the above languages lack a precise semantics, Wooldridge and Parsons (2000, 2000) motivate and compare the use of different logical languages for specifying negotiation protocols. In the context of General Game Playing (GGP), the Game Description Language (GDL) was designed for specifying game rules while maintaining a tractable complexity [17]. This language has been extended to deal with integer values [18] and epistemic operators [19]. The use of languages inspired on GDL for describing market-based protocols have been studied in the context of negotiation [20, 21] and single-item markets [22]. Such approaches lack a clear link between the language, the mechanism formalization and the agents' preferences, which is a key aspect for enabling reasoning about auctions.

Automated Mechanism Design Designing an auction in such manner to ensure features of the outcome alongside with a desirable behavior of the participants is a key challenge in Economics. In fact, this problem is known as Mechanism Design: the formulation of game-like systems whose equilibria satisfy some desired properties, usually expressed in terms of incentive, utility or social welfare. Traditionally, mechanisms have been formulated and verified by human specialists, who use their knowledge and experience for defining the game rules. Creating and verifying mechanisms which will be played by strategic agents can be a very difficult and time-consuming task. Sandholm (2003) introduced Automated Mechanism Design (AMD), whose goal is to automatically create mechanisms for solving a specific preference aggregation problem. AMD is usually handled as an optimization and domain-oriented problem. Most solutions used on the literature are based on machine learning, which include, for instance, neural networks [24, 25] and statistical techniques [26]. A number of works explore computed-aided verification of auctions [27, 28, 29], where the process is assisted by a reasoner. In the context of fully-automatic verification, Pauly and Wooldridge (2003) and Wooldridge et al. (2007) advocate the use of Alternating-time Temporal Logic (ATL) [32] to reason about mechanisms. The main limitation in these works is the purely qualitative setting and the impossibility of expressing key strategic concepts such as dominance in the logic.

Logics for Strategic Reasoning This work is also related to the long-established logical approach to systems verification [3] and synthesis [4]. In the recent years much progress has been made in the field of logics for strategic reasoning. Pioneering work includes the Alternating-time Temporal Logic ATL [32], which uses coalition modalities to specify strategic abilities of groups of agents, an important notion in Mechanism Design. The Strategy Logic (SL) [33, 34] subsumes ATL and considers explicit manipulation of strategies. A recent quantitative extension of SL, denoted $\mathsf{SL}[\mathscr{F}]$ [35], introduces values in models and functions in the language, enabling the reasoning about key game-theoretic concepts such as utilities and preferences. Several works have also considered extensions of SL with imperfect information [36, 37, 38], which is also an important feature when modeling auctions with private valuations.

3. Contribution

We addressed the problem of modeling and analyzing auction mechanisms for MAS using logics and strategic reasoning. First, we provided a framework for representing auctions, denoted Auction Description Language (ADL) [39, 40, 41]. ADL addresses important dimensions of auction-based markets and is general enough to represent most auction settings. We illustrated the generality of ADL by modeling a number of representative auctions. We demonstrated how this language can be used for the automated verification of direct mechanisms and for automatically checking well-formedness of auction descriptions. We then extended ADL with knowledge operators and an action modality [42, 43] (denoted Epistemic ADL, or simply ADLK) for providing a ground for the design of general auction players and characterizing their rational behavior when reasoning about the effect of actions and other players rationality. We show that the model-checking problem for ADL-formulas belongs to Ptime when it involves functions that be computed in polynomial time. By the other hand, we show that the model-checking problem for ADLK is in Exptime.

In relation to Automated Mechanism Design, we investigate logical frameworks for strategic reasoning about mechanisms. We first propose a new variant of Strategy Logic with quantitative features, imperfect information and epistemic operators, denoted $\mathsf{SLK}[\mathscr{F}]$ [44]. We demonstrated how $\mathsf{SLK}[\mathscr{F}]$ can express the implementation of social choice functions and be used for automatically verifying a number of important concepts and properties often required in auctions, or more generally in mechanism design. We also show how we can express properties relating agents' revenues with respect to their beliefs about other agents' preferences. We showed that verifying a mechanism in relation to classical properties boils-down to model checking a $\mathsf{SLK}[\mathscr{F}]$ formula and we prove it can be done in Pspace for memoryless strategies.

We then introduced a quantitative semantics for SL with natural strategies and imperfect information (denoted $\mathsf{NatSL}[\mathscr{F}]$) [45], which provides a new perspective for formal verification and design of novel mechanisms based on the complexity of strategies. We show how to model popular strategies for repeated keyword auctions using $\mathsf{NatSL}[\mathscr{F}]$ and prove properties pertaining to this game; We proved that the model-checking problem for $\mathsf{NatSL}[\mathscr{F}]$ is Pspace-complete and that $\mathsf{NatSL}[\mathscr{F}]$ has incomparable distinguishing and expressive power to $\mathsf{SL}[\mathscr{F}]$. Finally, we offered a novel perspective on the design of mechanisms by rephrasing the AMD problem in terms of synthesis from $\mathsf{SL}[\mathscr{F}]$ specifications [46, 47]. This approach enables automatically generating optimal mechanisms from a quantitative logical specification, which may include not only game rules but also requirements over the strategic behavior of participants and quality of the outcome. We solved the synthesis problem for $\mathsf{SL}[\mathscr{F}]$ by investigating the related satisfiability problem in two cases: action-bounded and turn-based mechanisms.

4. Perspectives and Future Work

In a recent work [48], we considered the use of the probabilistic extension of SL [49] to handle stochastic features often present in auctions. Going from deterministic setting to a more general and probabilistic one is challenging due to several aspects. First, the wide and heterogeneous range of settings considered in the literature obscures the path for a general and formal approach to verification. The setting may consider deterministic or randomized mechanisms, incomplete information about agents' types (Bayesian mechanisms), mixed or pure strategies and iterative protocols (indirect mechanisms). Second, considering Bayesian mechanisms brings out different methods for evaluating a mechanism according to the time-line for revealing the incomplete information as the game is run.

We studied the verification of mechanisms under memoryless combinatorial strategies and Natural Strategies with bounded recall. This setting is enough to capture many kinds of auctions (such as one-shot or English auctions) where memoryless strategies are sufficient to represent the bidders behaviour since all the relevant information can be encoded in a state. However, when participating in repeated auctions, agents could gather information from other agents behaviour and act based on what happened in previous instances of the game. An interesting direction is, then, to investigate the use of strategies with recall for learning other players' valuations based on their behavior. For such situations we can study the model-checking problem for $\mathsf{SLK}[\mathscr{F}]$ with memoryful strategies. In the qualitative setting already, imperfect information yields undecidability, but known decidable cases exist [37, 36], which should be considered also in the quantitative case.

We believe the automated synthesis of mechanisms is a promising and powerful tool for AMD. However, the high expressiveness of $\mathsf{SL}[\mathscr{F}]$ may not always be needed for simple classes of mechanisms, and one may consider fragments of it to achieve better complexity. Therefore, an interesting direction for future work is to study the complexity of synthesizing from $\mathsf{SL}[\mathscr{F}]$-fragments, inspired from the SL-fragments One-Goal SL [50, 51] and Simple-Goal SL [52], for instance. These fragments are usually computationally easier than full SL, and we can hope that similar results can be established in the quantitative setting. On a related vein, we can study the translation of ADL to $\mathsf{SL}[\mathscr{F}]$-formulae, so as to include the auction description in the mechanism specification. In this setting, $\mathsf{SL}[\mathscr{F}]$ formulae can be used to express requirements of well-formed auction descriptions.

The problems contemplated in my thesis are also worth investigating from an empirical perspective. One direction is to explore the interplay between agents' bounded rationality and the auctioneer revenue so as to understand the impact of bounded rationality on mechanism design. An implementation of a model checker for $\mathsf{NatSL}[\mathscr{F}]$ would enable the empirical evaluation of natural strategies and auctions played by participants with restricted memory. Finally, experimental results could be used to assess the practical relevance of our proposed approaches, especially in relation to mechanism synthesis from $\mathsf{SL}[\mathscr{F}]$ specification due to the high theoretical complexity of the problem.

Acknowledgments

I am very grateful to my co-authors Laurent Perrussel, Aniello Murano, Bastien Maubert, Francesco Belardinelli, Wojtek Jamroga, Vadim Malvone, Andreas Herzig, and Sylvain Bouveret. This research is supported by the ANR project AGAPE ANR-18-CE23-0013 and the PRIN project RIPER (No. 20203FFYLK).

References

[1] V. Conitzer, T. Sandholm, Complexity of mechanism design, in: Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence (UAI 2002), University of Alberta, Edmonton, Morgan Kaufmann, 2002, pp. 103–110.

[2] F. Asselin, B. Jaumard, A. Nongaillard, A technique for large automated mechanism design problems, in: Proceedings of the International Conference on Intelligent Agent Technology (IAT 2006), 2006.

[3] E. Clarke, O. Grumberg, D. Kroening, D. Peled, H. Veith, Model checking, MIT press, 2018.

[4] C. David, D. Kroening, Program synthesis: challenges and opportunities, Philosophical Transactions of the Royal Society A 375 (2017) 20150403.

[5] M. Pauly, M. Wooldridge, Logic for mechanism design–a manifesto, in: Workshop on Game Theory and Decision Theory in Agent Systems (GTDT), 2003.

[6] M. Mittelmann, Logics for Representation and Design of Auctions, Ph.D. thesis, Universitè de Toulouse, 2022.

[7] P. Klemperer, Auction theory: A guide to the literature, Journal of Economic Surveys 13 (1999) 227–286.

[8] V. Krishna, Auction Theory, Academic Press, 2009.

[9] D. Rolli, S. Luckner, H. Gimpel, C. Weinhardt, A Descriptive Auction Language, Electronic Markets 16 (2006) 51–62.

[10] K. M. Lochner, M. P. Wellman, Rule-based specification of auction mechanisms, Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004) 2 (2004) 818–825.

[11] C. Bădică, M. Ganzha, M. Paprzycki, Rule-based automated price negotiation: Overview and experiment, in: Artificial Intelligence and Soft Computing (ICAISC 2006), Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1050–1059.

[12] S. Hudert, H. Ludwig, G. Wirtz, Negotiating SLAs-An approach for a generic negotiation framework for WS-agreement, Journal of Grid Computing 7 (2009) 225–246.

[13] S. Hudert, T. Eymann, H. Ludwig, G. Wirtz, A negotiation protocol description language for automated service level agreement negotiations, 2009 IEEE Conference on Commerce and Enterprise Computing (CEC 2009) (2009) 162–169.

[14] A. Dobriceanu, L. Biscu, C. Badica, Adding a declarative representation of negotiation mechanisms to an agent-based negotiation service, in: 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, 2007, pp. 471–474.

[15] M. Wooldridge, S. Parsons, Languages for negotiation, in: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000), IOS Press, NLD, 2000.

[16] M. Wooldridge, S. Parsons, On the use of logic in negotiation, in: Proceedings of the Autonomous Agents Workshop on Agent Communication Languages and Conversation Protocols, 2000.

[17] M. Genesereth, M. Thielscher, General game playing, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2014.

[18] M. Mittelmann, L. Perrussel, Game description logic with integers: A GDL numerical extension, in: Proceedings of the International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2020), 2020.

[19] G. Jiang, D. Zhang, L. Perrussel, H. Zhang, Epistemic GDL: A logic for representing and reasoning about imperfect information games, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2016), 2016.

[20] D. d. Jonge, D. Zhang, Using GDL to represent domain knowledge for automated negotiations, in: Proc. of Autonomous Agents and Multi-Agent Systems Workshops, AAMAS 2016 Workshops, 2016, pp. 134–153.

[21] D. de Jonge, D. Zhang, GDL as a unifying domain description language for declarative

automated negotiation, Autonomous Agents and Multi-Agent Systems 35 (2021) 13.

[22] M. Thielscher, D. Zhang, From General Game Descriptions to a Market Specification Language for General Trading Agents, Springer Berlin Heidelberg, 2010, pp. 259–274.

[23] T. Sandholm, Automated mechanism design: A new application area for search algorithms, in: Principles and Practice of Constraint Programming – CP 2003, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 19–36.

[24] W. Shen, P. Tang, S. Zuo, Automated mechanism design via neural networks, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2019), 2019.

[25] P. Dütting, Z. Feng, H. Narasimhan, D. Parkes, S. S. Ravindranath, Optimal auctions through deep learning, in: Proceedings of the International Conference on Machine Learning (ICML 2019), 2019.

[26] H. Narasimhan, S. B. Agarwal, D. C. Parkes, Automated mechanism design without money via machine learning, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2016), 2016.

[27] M. Caminati, M. Kerber, C. Lange, C. Rowat, Sound auction specification and implementation, in: ACM Conference on Economics and Computation, 2015.

[28] G. Barthe, M. Gaboardi, E. Arias, J. Hsu, A. Roth, P.-Y. Strub, Computer-aided verification for mechanism design, in: Conf. on Web and Internet Economics, 2016.

[29] M. Kerber, C. Lange, C. Rowat, An introduction to mechanized reasoning, Journal of Mathematical Economics 66 (2016) 26 – 39.

[30] M. Pauly, R. Parikh, Game logic-an overview, Studia Logica 75 (2003) 165–182.

[31] M. Wooldridge, T. Ågotnes, P. Dunne, W. Van der Hoek, Logic for automated mechanism design-a progress report, in: Proceedings of AAAI Conference on Artificial Intelligence (AAAI 2007), 2007.

[32] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, Journal of the ACM 49 (2002) 672–713.

[33] K. Chatterjee, T. A. Henzinger, N. Piterman, Strategy logic, Information and Computation 208 (2010) 677–693.

[34] F. Mogavero, A. Murano, G. Perelli, M. Y. Vardi, Reasoning about strategies: On the model-checking problem, ACM Transactions on Computational Logic (TOCL) 15 (2014) 1–47.

[35] P. Bouyer, O. Kupferman, N. Markey, B. Maubert, A. Murano, G. Perelli, Reasoning about Quality and Fuzziness of Strategic Behaviours, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2019), 2019.

[36] F. Belardinelli, A. Lomuscio, A. Murano, S. Rubin, Verification of multi-agent systems with public actions against strategy logic, Artificial Intelligence 285 (2020).

[37] R. Berthon, B. Maubert, A. Murano, S. Rubin, M. Vardi, Strategy logic with imperfect information, ACM Transactions on Computational Logic 22 (2021).

[38] P. Cermák, A. Lomuscio, F. Mogavero, A. Murano, Practical verification of multi-agent systems against SLK specifications, Information and Computation 261 (2018) 588–614.

[39] M. Mittelmann, L. Perrussel, Auction description language (ADL): general framework for representing auction-based markets, in: Proceedings of the European Conference

on Artificial Intelligence (ECAI 2020), volume 325, IOS Press, 2020, pp. 825–832.

[40] M. Mittelmann, S. Bouveret, L. Perrussel, A general framework for the logical representation of combinatorial exchange protocols, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2021), ACM, 2021, pp. 1602–1604.

[41] M. Mittelmann, S. Bouveret, L. Perrussel, Representing and reasoning about auctions, Autonomous Agents and Multi-Agent Systems 36 (2022) 20. URL: https://doi.org/10.1007/s10458-022-09547-9. doi:10.1007/s10458-022-09547-9.

[42] M. Mittelmann, L. Perrussel, An epistemic logic for reasoning about strategies in general auctions, in: Proceedings of the Workshops of the International Conference on Logic Programming, volume 2678, 2020.

[43] M. Mittelmann, A. Herzig, L. Perrussel, Epistemic reasoning about rationality and bids in auctions, in: Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA 2021), volume 12678, Springer, 2021, pp. 116–130.

[44] B. Maubert, M. Mittelmann, A. Murano, L. Perrussel, Strategic reasoning in automated mechanism design, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR 2021), 2021, pp. 487–496.

[45] F. Belardinelli, W. Jamroga, V. Malvone, M. Mittelmann, A. Murano, L. Perrussel, Reasoning about human-friendly strategies in repeated keyword auctions, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2022), IFAAMAS, 2022, pp. 1602–1604.

[46] M. Mittelmann, B. Maubert, A. Murano, L. Perrussel, Automated synthesis of mechanisms, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2022), 2022.

[47] M. Mittelmann, B. Maubert, A. Murano, L. Perrussel, Synthesis of mechanisms with strategy logic, in: ICTCS, volume 3284 of CEUR Workshop Proceedings, CEUR-WS.org, 2022, pp. 47–52.

[48] M. Mittelmann, B. Maubert, A. Murano, L. Perrussel, Formal verification of bayesian mechanisms, in: Proc. of the AAAI Conference on Artificial Intelligence, AAAI 2023, AAAI Press, 2023.

[49] B. Aminof, M. Kwiatkowska, B. Maubert, A. Murano, S. Rubin, Probabilistic strategy logic, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019), 2019.

[50] F. Mogavero, A. Murano, G. Perelli, M. Y. Vardi, Reasoning about strategies: on the satisfiability problem, Logical Methods in Computer Science 13 (2017).

[51] P. Cermák, A. Lomuscio, A. Murano, Verifying and synthesising multi-agent systems against one-goal strategy logic specifications, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2015), 2015.

[52] F. Belardinelli, W. Jamroga, D. Kurpiewski, V. Malvone, A. Murano, Strategy logic with simple goals: Tractable reasoning about strategies, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2019), 2019.