Formalisation of Action with Durations in Answer Set Programming

Etienne Tignon
University of Potsdam, Germany

In this paper, I will discuss the work I am currently doing as a Ph.D. student at the University of Potsdam, under the tutoring of T. Schaub. I'm currently looking into action description in ASP. More precisely, my goal is to explore how to represent actions with durations in ASP, in different contexts. Right now, I'm focused on Multi-Agent Path Finding (MAPF), looking at how to represent speeds for different agents and contexts.

Before tackling duration, I wanted to explore and compare different representations of action taking in ASP. For this, I started comparing different simple encodings tackling the MAPF problem. Even in simple code, choices and assumptions have been made in their creations. The objective of my work is to present the consequences of those design decisions in terms of performance and knowledge representation. As far as I know, there is no current research on this topic.

Besides that, I'm also exploring different ways to represent duration and to solve related problems. I planed to compare them the same way I described before. I also want this to help me find innovative and effective ways to solve problems with duration.

1 Introduction

ASP is already used in time-related solving in a lot of different contexts, from planning trains [2] to moving robots in wharehouse [25] [18]. There exist some dedicated tools like Telingo [6] that are used for planning. However, most of the existing works make an assumption: that the events you are trying to place in time are instantaneous (in the sense that they are defined in a point in time, opposed to an interval). This is not always what you want to represent. For example: what if you want to create a schedule for the lectures in a university? The lectures are not instantaneous, they take time. How to represent things who take times? The goal of my Ph.D. is to study how to represent timed events with duration in ASP. I'm looking at different implementations of timed events in ASP, comparing them in terms of knowledge representation and performances. From that, I hope to be able to deduce performant and easy-to-read models of timed events with duration.

For now, even if I hope my results to be versatile, I'm mostly focused on solving MAPF with duration. In MAPF with duration, agents can take actions to move from one point to another like in classical MAPF. But the time needed to do this action can change depending on several parameters. But I would like to use ASP and its strengths to find innovative ways to represent and solve MAPF instances where different durations of actions have to be taken into consideration.

2 Background

2.1 Answer Set Programming

Answer Set Programming [15] [28] is a versatile declarative problem-solving paradigm. It combines a high-level modeling language with effective grounding and solving technology. On top of that, its success is also due to effective systems, like the solver *clasp* [13] and the grounder *Gringo* [14] from the Potsdam Answer Set Solving Collection (aka Potassco) [12].

Research has already been made that uses ASP to tackle time-related problems. For exemple, ASP is explored to solve MAPF [25] [18] or train scheduling [2]. Also, there exist systems dedicated to this, like telingo [6]. ASP implementations of some of the mentioned representations exists (for action languages [23],[8],[11], event calculus [20], PDDL+ [5] to name a few).

2.2 Multi-Agent Path Finding

As I said before, during this Ph.D., I am mostly interested in applying techniques and concepts of the formalisms on MAPF [27]. The Multi-Agent Path Finding problem is an extension of the pathfinding one for which the path of multiple agents has to be computed while taking into consideration the interactions and possible collisions among the agents. Most of the current work on MAPF defines a discrete environment where the space to explore is defined by nodes on which the agent can be at any point in time. The edges between two nodes represent the fact that an agent can move from one to another. In this context, a move doesn't really take time. This implies that if at time t an agent is on the node n_1, if it moved in the edge n_1, n_2, at time $t+1$ it will be on the node n_2. What I am interested in during this Ph.D. is to explore ways to represent the concept of "this agent takes X time to move from n_1 to n_2".

Several works deal with similar subjects, like with continuous time [4], or with geometric agents [29]. However, most of these works are not related to ASP. Moreover, I want to study actions with duration in particular. Most of the literature about MAPF when movements take time also makes other assumptions, such as space and/or time being continuous for example.

2.3 Timed Events Formalisms

The will to express timed events rigorously and practically is far from new in the field of knowledge representation. Here is a brief, non-exhaustive list of formalizations and models existing in the literature which are planned to be explored in this Ph.D.

- **Action languages** [26][30][17][16] are formal models used for talking about actions and their effects. They use a transition system composed of fluents which have different values that can be modified by actions. Depending on the exact language used, different advanced concepts can be expressed, like conditional consequences on an action, redefinition of inertia, action with indirect effects, and many others.

- **Allen's interval algebra** [3] is a calculus made for temporal reasoning. It allows the description of relationships between intervales. Thanks to this formalizations, it is possible to represent the relations between timed events (such as "the event e_1 occur before the event e_2"). This calculus doesn't represent any quantitative notion like the time of the intervals or the distance between two intervals in time; only qualitative relations are exprimed.

- **Event calculus** [21], like action languages, is a formalism able to represent events and their effects. In this formalism, you have fluents which can either hold or not at each point in time. Furthermore, you have events which can occur and change fluent which holds. You can, for example, express that an action α makes a fluent β hold at time τ under the condition that another fluent β' hold at the same time.

- **PDDL+** [9] is a moddeling language destined to represent mixed discrete-continuous planning domains. It is an extention of **PDDL** [24]. Inside this formalism, it is possible to represent events and actions which have effect on continuous variables.
- **Temporal logic** [10] [1] [7] is a way to extend logic with temporal symbols, such as next ($\circ$), eventually ($\diamond$), or until (U). In this paradigm, you consider that atoms are true or false at each point in time. With thoses temporal symbols, you can write the relationships between different timepoints. For example to say that, if the atom a_1 holds, a_2 hold in the next timepoint, you can write $a_1 \to \circ a_2$.
- **Metric temporal logic** [22] further extends temporal logic by using time-constrained version of the temporal operators. With this, it is possible to represent quantified timed relationships, like "if the atom a_1 hold, a_2 hold until in 10 a_3 hold (wich mean that a_3 must hold at least once in the next 10 timepoints) : $a_1 \to a_2 \, \mathsf{U}_{10} \, a_3$"
- **Temporal network** [19] is a formalism that allow the representation of graphs that have edges defined for each point in time (for example, you can have the edge $(n_1, n_3, 4)$ to represent the edge from n_1 to n_3 at time 4).

3 Current status

For now, most of the work has been dedicated to the study of actions representations, durations, MAPF, and ASP. A solid understanding of the state of the art in those fields is needed to ensure that the work of this Ph.D. will lead to a significant advance.

Besides that, I have started a comparative analysis of different encodings based on action languages in ASP, dedicated to the solving of MAPF instances. In particular, two aspects are observed:

- Knowledge representation: Some implementations are easier to read than others. It is important to not underestimate the importance of it, so we can create understandable and maintainable programs.
- Performance: One line of difference can have a great impact on the behavior of the grounder and the solver. Most of the time, the same concept can be modeled in many ways. Each of them will lead to a different number of atoms and rules, which will define how long the grounder will need to create the grounded program. This, in turn, will impact how the solving will be tackled.

I hope this structured comparison between encodings will lead me to a better understanding of the best way to model this kind of problem in ASP.

I also started thinking about how to represent MAPF problems in the differents paradigms discussed earlier. I intend to have the same example in all those formalizations so that in the future, when I will look at their implementations in ASP, I will have a common ground to comparison.

References

[1] M. Abadi & Z. Manna (1989): *Temporal Logic Programming*. Journal of Symbolic Computation 8, pp. 277–295, doi:10.1016/s0747-7171(89)80070-7.

[2] D. Abels, J. Jordi, M. Ostrowski, T. Schaub, A. Toletti & P. Wanko (2009): *Train scheduling with hybrid ASP*. In C. Boutilier, editor: *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09)*, AAAI/MIT Press, pp. 3–17, doi:10.1007/978-3-030-20528-7_1.

[3] J. Allen (1983): *Maintaining knowledge about temporal intervals*. *Communications of the ACM* 26(11), pp. 832–843, doi:10.1016/b978-1-4832-1447-4.50033-x.

[4] A. Andreychuk, K. Yakovlev, D. Atzmon & R. Stern (2019): *Multi-Agent Pathfinding with Continuous Time*. In S. Kraus, editor: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI'19)*, ijcai.org, pp. 39–45, doi:10.24963/ijcai.2019/6.

[5] M. Balduccini, D. Magazzeni & M. Maratea (2016): *PDDL+ Planning via Constraint Answer Set Programming*. In B. Bogaerts & A. Harrison, editors: *Proceedings of the Ninth Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP'16)*, pp. 1–12. Available at https://sites.google.com/site/aspocp2016/.

[6] P. Cabalar, R. Kaminski, P. Morkisch & T. Schaub (2009): *telingo = ASP + Time*. In C. Boutilier, editor: *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09)*, AAAI/MIT Press, pp. 256–269, doi:10.1007/978-3-030-20528-7_19.

[7] S. Demri, V. Goranko & M. Lange (2016): *Temporal Logics in Computer Science: Finite-State Systems*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press.

[8] S. Dworschak, S. Grell, V. Nikiforova, T. Schaub & J. Selbig (2008): *Modeling biological networks by action languages via answer set programming*. *Constraints* 13(1-2), pp. 21–65, doi:10.1007/s10601-007-9031-y.

[9] M. Fox & D. Long (2006): *Modelling Mixed Discrete-Continuous Domains for Planning*. *Journal of Artificial Intelligence Research* 27(1), pp. 235–297, doi:10.1613/jair.2044. Available at https://dl.acm.org/doi/10.5555/1622572.1622580.

[10] D. Gabbay (1987): *The Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems*. In B. Banieqbal, H. Barringer & A. Pnueli, editors: *Proceedings of the Conference on Temporal Logic in Specification*, *Lecture Notes in Computer Science* 398, Springer-Verlag, pp. 409–448, doi:10.1007/3-540-51803-7_36.

[11] M. Gebser, T. Grote & T. Schaub (2010): *Coala: A Compiler from Action Languages to ASP*. In T. Janhunen & I. Niemelä, editors: *Proceedings of the Twelfth European Conference on Logics in Artificial Intelligence (JELIA'10)*, *Lecture Notes in Artificial Intelligence* 6341, Springer-Verlag, pp. 360–364, doi:10.1007/978-3-642-15675-5_32.

[12] M. Gebser, R. Kaminski, B. Kaufmann & T. Schaub (2012): *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers.

[13] M. Gebser, B. Kaufmann, A. Neumann & T. Schaub (2007): *clasp: A Conflict-Driven Answer Set Solver*. In C. Baral, G. Brewka & J. Schlipf, editors: *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, *Lecture Notes in Artificial Intelligence* 4483, Springer-Verlag, pp. 260–265, doi:10.1007/978-3-540-72200-7_23.

[14] M. Gebser, T. Schaub & S. Thiele (2007): *Gringo: A New Grounder for Answer Set Programming*. In C. Baral, G. Brewka & J. Schlipf, editors: *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, *Lecture Notes in Artificial Intelligence* 4483, Springer-Verlag, pp. 266–271, doi:10.1007/978-3-540-72200-7_24.

[15] M. Gelfond & V. Lifschitz (1988): *The Stable Model Semantics for Logic Programming*. In R. Kowalski & K. Bowen, editors: *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, MIT Press, pp. 1070–1080.

[16] M. Gelfond & V. Lifschitz (1998): *Action languages*. *Electronic Transactions on Artificial Intelligence* 3(6), pp. 193–210. Available at http://www.ep.liu.se/ej/etai/1998/007/.

[17] E. Giunchiglia & V. Lifschitz (1998): *An Action Language Based on Causal Explanation: Preliminary Report*. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 623–630. Available at citeseer.nj.nec.com/article/giunchiglia98action.html.

[18] R. Gómez, C. Hernández & J. Baier (2021): *A Compact Answer Set Programming Encoding of Multi-Agent Pathfinding*. *IEEE Access* 9, pp. 26886–26901, doi:10.1109/access.2021.3053547.

[19] Márton Karsai, Nicola Perra & Alessandro Vespignani (2014): *Time varying networks and the weakness of strong ties*. *Scientific Reports* 4(1), doi:`10.1038/srep04001`. Available at `https://doi.org/10.1038%2Fsrep04001`.

[20] T. Kim, J. Lee & R. Palla (2009): *Circumscriptive Event Calculus as Answer Set Programming*. In C. Boutilier, editor: *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09)*, AAAI/MIT Press, pp. 823–829.

[21] R. Kowalski & M. Sergot (1986): *A Logic-based Calculus of Events*. *New Generation Computing* 4(1), pp. 67–95, doi:`10.1007/bf03037383`.

[22] R. Koymans (1990): *Specifying Real-Time Properties with Metric Temporal Logic*. *Real-Time Systems* 2(4), pp. 255–299, doi:`10.1007/bf01995674`.

[23] V. Lifschitz (1999): *Action languages, answer sets, and planning*. In K. Apt, V. Marek, M. Truszczyński & D. Warren, editors: *The Logic Programming Paradigm: a 25-Year Perspective*, Springer-Verlag, pp. 357–373, doi:`10.1007/978-3-642-60085-2_16`.

[24] D. McDermott (1998): *PDDL — The Planning Domain Definition Language*. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

[25] V. Nguyen, P. Obermeier, T. Son, T. Schaub & W. Yeoh (2017): *Generalized Target Assignment and Path Finding Using Answer Set Programming*. In C. Sierra, editor: *Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence (IJCAI'17)*, IJCAI/AAAI Press, pp. 1216–1223, doi:`10.24963/ijcai.2017/169`.

[26] E. Pednault (1987): *Formulating multi-agent dynamic-world problems in the classical planning framework*. In M. Georgeff & A. Lansky, editors: *Reasoning about actions and plans*, Morgan Kaufmann Publishers, pp. 47–82, doi:`10.1016/b978-0-934613-30-9.50006-8`.

[27] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, R. Barták & E. Boyarski (2019): *Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks*. In P. Surynek & W. Yeoh, editors: *Proceedings of the Twelfth International Symposium on Combinatorial Search (SOCS'19)*, AAAI Press, pp. 151–159.

[28] V. Subrahmanian & C. Zaniolo (1995): *Relating Stable Models and AI Planning Domains*. In: *Proceedings of the Twelfth International Conference on Logic Programming*, MIT Press, pp. 233–247, doi:`10.7551/mitpress/4298.003.0030`.

[29] P. Surynek (2019): *Multi-Agent Path Finding with Continuous Time and Geometric Agents Viewed through Satisfiability Modulo Theories (SMT)*. In P. Surynek & W. Yeoh, editors: *Proceedings of the Twelfth International Symposium on Combinatorial Search (SOCS'19)*, AAAI Press, pp. 200–201.

[30] H. Turner (1997): *Representing Actions in Logic Programs and Default Theories: A Situation Calculus Approach*. *Journal of Logic Programming* 31(1-3), pp. 245–298, doi:`10.1016/s0743-1066(96)00125-2`.