Situation Calculus Temporally Lifted Abstractions for Generalized Planning – Extended Abstract

Giuseppe De Giacomo[1,3], Yves Lespérance[2] and Matteo Mancanelli[3,*]

[1]University of Oxford, Oxford, UK
[2]York University, Toronto, ON, Canada
[3]Sapienza University, Rome, Italy

Abstract

We present a new formal framework for generalized planning (GP) based on the situation calculus extended with LTL constraints. The GP problem is specified by a first-order basic action theory whose models are the problem instances. This low-level theory is then abstracted into a high-level propositional nondeterministic basic action theory with a single model. A refinement mapping relates the two theories. LTL formulas are used to specify the temporally extended goals as well as assumed trace constraints. If all LTL trace constraints hold at the low level and the high-level model can simulate all the low-level models with respect to the mapping, we say that we have a *temporally lifted abstraction*. We prove that if we have such an abstraction and the agent has a strategy to achieve a LTL goal under some trace constraints at the abstract level, then there exists a refinement of the strategy to achieve the refinement of the goal at the concrete level. We use LTL synthesis to generate the strategy at the abstract level. We illustrate our approach by synthesizing a program that solves a data structure manipulation problem.

Keywords

Generalized Planning, Situation Calculus, Nondeterministic Domains, Abstractions, Strategy Synthesis

1. Overview

In generalized planning (GP), one tries to generate a typically iterative policy that solves an infinite set of similar planning problem instances [2, 3, 4]. For example, we may want to synthesize a program for finding the minimum value in a list, for lists of any lengths. Many approaches to generalized planning involve constructing an abstraction and finding a solution for this abstraction which handles all the actual problem instances [5, 6]. We propose a new formal framework for generalized planning based on the situation calculus [7, 8] that allows one to provide an abstract description of the domain and associated LTL trace constraints [6, 9], and prove that a controller synthesized for the abstract theory can be refined into one that achieves the goal at the concrete level.

Our framework is based on the *nondeterministic situation calculus* [10] (DL21) where each agent action $A(\vec{x})$ is accompanied by an environment reaction e outside the agent's control that determines the action's outcome, e.g., a flipped coin may fall head or tail. A nondeterministic basic action theory (NDBAT) can be seen as a special kind of action theory, where we have *system* actions $A(\vec{x}, e)$, successor state axioms $\mathcal{D}_{ssa}$, describing how predicates and functions change after system actions are performed, and action precondition axioms $\mathcal{D}_{poss}$, stating when each system action can occur. [11] (BDL23) have proposed an account of abstraction for NDBATs. They relate a high-level NDBAT to a low-level NDBAT through a *refinement mapping* that specifies how a high-level action is implemented at the low level by a ConGolog program [12, 13]. They then define notions of sound and/or complete abstraction for such NDBATs in terms of a notion of bisimulation between their models. [14, 15] have adapted and extended this kind of approach to solve GP problems, focusing on QNP abstractions.

AI4CC-IPS-RCRA-SPIRIT 2024: International Workshop on Artificial Intelligence for Climate Change, Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy. November 25-28th, 2024, Bolzano, Italy [1].
*Corresponding author.
giuseppe.degiacomo@cs.ox.ac.uk (G. D. Giacomo); lesperan@eecs.yorku.ca (Y. Lespérance); mancanelli@diag.uniroma1.it (M. Mancanelli)

Here, we assume that the modeler specifies a propositional high-level (HL) action theory/model with a limited set of HL fluents and nondeterministic actions, which abstracts over a concrete low-level (LL) action theory with multiple models, with a given refinement mapping *m*. At the LL, in each model we have complete information about the state of the world, while at the HL, we have actions that may have several outcomes, e.g., after advancing to the next item in a list, we may or may not reach the list's end. We extend the HL theory with LTL trace constraints to impose fairness assumptions on the possible sequences of nondeterministic actions, e.g., ensuring that if we keep advancing we will eventually reach the list's end. Finally, we define a notion of *temporally lifted abstraction*, where every LL trace that is a refinement of a sequence of HL actions is *m*-similar to a trace involving this action sequence in the HL model, and where the LTL trace constraints are satisfied by the LL theory. The NDBATs represent our GP problem, where each LL model specifies the planning problem instances, and the HL model abstracts away the LL details, retaining only the shared features. We then provide a method for solving all the planning problem instances simultaneously. In particular, we show that given such an abstraction, if we can use LTL synthesis on the HL model to obtain a HL strategy that achieves a LTL goal under the given trace constraints, then we can automatically refine it to get a LL strategy that achieves the mapped LTL goal in all concrete instances of the problem.

We illustrate how our approach works by using it to synthesize a program to find the minimum value of a list. This application is inspired by [16], which proposed an approach for solving program synthesis tasks [17, 18, 19, 20, 21] that involve the manipulation of data structures such as lists, trees, and graphs by viewing them as instances of generalized planning. They provide several examples of how their method can be applied, but they do not provide complete formal specifications of the data structures used and formal proofs that the assumed temporal constraints and goal specifications hold for them.

2. Methodology

Our methodology involves the following main steps:

1. *Formalize the concrete planning problem instances in situation calculus* - this consists of writing the specification for the domain of interest; hence, it is straightforward
2. *Specify a propositional temporally lifted abstraction as a HL NDBAT* - this abstracts over some details and includes nondeterministic actions; some LTL trace constraints will also be introduced to capture restrictions on the possible future histories; obtaining the HL NDBAT is similar to the previous step and, in many cases, we can reuse the specification of one GP task for other similar tasks (i.e., involving the same data structure)
3. *Write the LTL goals* - this step is domain-dependent
4. *Run a LTL synthesis engine on the abstraction* - this automatically derives a HL strategy to reach the goals; note that our HL propositional abstraction can always be interpreted as an LTL specification
5. *Translate the HL strategy to a LL program* - this step can be simply addressed by using the refinement mapping

This methodology yields provably correct solutions with strong formal guarantees. Note that there should be no need to generate the entire situation calculus specifications from scratch. Instead, one could build a library of specifications and reuse them in a modular way. Thus, the modeler can just specify her problem in terms of HL trace and goal constraints, exploiting this library, and then run the automatic synthesis engines.

Acknowledgments

This work has been partially supported by the ERC Advanced Grant WhiteMech (No. 834228), the PNRR MUR project FAIR (No. PE0000013), the Italian National Ph.D. on Artificial Intelligence at Sapienza University of Rome, the National Science and Engineering Research Council of Canada, and York University.

References

[1] D. Aineto, R. De Benedictis, M. Maratea, M. Mittelmann, G. Monaco, E. Scala, L. Serafini, I. Serina, F. Spegni, E. Tosello, A. Umbrico, M. Vallati (Eds.), Proceedings of the International Workshop on Artificial Intelligence for Climate Change, the Italian workshop on Planning and Scheduling, the RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and the Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (AI4CC-IPS-RCRA-SPIRIT 2024), co-located with 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[2] S. Srivastava, N. Immerman, S. Zilberstein, Learning generalized plans using abstract counting, in: AAAI, 2008, pp. 991–997.

[3] Y. Hu, G. De Giacomo, Generalized planning: Synthesizing plans that work for multiple environments, in: IJCAI, 2011, pp. 918–923.

[4] V. Belle, H. J. Levesque, Foundations for generalized planning in unbounded stochastic domains, in: KR, 2016, pp. 380–389.

[5] B. Bonet, H. Geffner, Policies that generalize: Solving many planning problems with the same policy., in: IJCAI, volume 15, 2015, pp. 2798–2804.

[6] B. Bonet, G. De Giacomo, H. Geffner, S. Rubin, Generalized planning: non-deterministic abstractions and trajectory constraints, in: IJCAI, 2017, pp. 873–879.

[7] J. McCarthy, P. J. Hayes, Some Philosophical Problems From the Standpoint of Artificial Intelligence, Machine Intelligence 4 (1969) 463–502.

[8] R. Reiter, Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems, The MIT Press, 2001.

[9] B. Aminof, G. De Giacomo, A. Murano, S. Rubin, Planning under LTL environment specifications, in: ICAPS, 2019, pp. 31–39.

[10] G. De Giacomo, Y. Lespérance, The nondeterministic situation calculus, in: M. Bienvenu, G. Lakemeyer, E. Erdem (Eds.), Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021, 2021, pp. 216–226. URL: https://doi.org/10.24963/kr.2021/21. doi:10.24963/KR.2021/21.

[11] B. Banihashemi, G. De Giacomo, Y. Lespérance, Abstraction of nondeterministic situation calculus action theories, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, 2023, pp. 3112–3122. URL: https://doi.org/10.24963/ijcai.2023/347. doi:10.24963/IJCAI.2023/347.

[12] G. De Giacomo, Y. Lespérance, H. J. Levesque, ConGolog, a concurrent programming language based on the situation calculus, Artificial Intelligence 121 (2000) 109–169. URL: https://doi.org/10.1016/S0004-3702(00)00031-X. doi:10.1016/S0004-3702(00)00031-X.

[13] G. De Giacomo, Y. Lespérance, A. R. Pearce, Situation calculus based programs for representing and reasoning about game structures, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010, AAAI Press, 2010. URL: http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1388.

[14] Z. Cui, Y. Liu, K. Luo, A uniform abstraction framework for generalized planning., in: IJCAI, 2021, pp. 1837–1844.

[15] Z. Cui, W. Kuang, Y. Liu, Automatic verification for soundness of bounded qnp abstractions for generalized planning, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, 2023, pp. 3149–3157.

[16] B. Bonet, G. De Giacomo, H. Geffner, F. Patrizi, S. Rubin, High-level programming via generalized planning and ltl synthesis, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, volume 17, 2020, pp. 152–161.

[17] C. C. Green, Application of theorem proving to problem solving, in: IJCAI, 1969, pp. 219–240.

[18] R. J. Waldinger, R. C. T. Lee, PROW: A step toward automatic program writing, in: IJCAI, 1969, pp. 241–252.

[19] A. Church, Logic, arithmetics, and automata, in: Proc. Int. Congress of Mathematicians, 1963, pp. 23–35.

[20] M. Abadi, L. Lamport, P. Wolper, Realizable and unrealizable specifications of reactive systems, in: ICALP, volume 372 of *Lecture Notes in Computer Science*, Springer, 1989, pp. 1–17.

[21] A. Pnueli, R. Rosner, On the synthesis of a reactive module, in: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 1989, pp. 179–190.