Abstraction in Situation Calculus Action Theories — Presentation Abstract*

Yves Lespérance
York University
Toronto, Canada
lesperan@cse.yorku.ca

Intelligent agents often operate in complex domains and have complex behaviors. Reasoning about such agents and even describing their behavior can be difficult. One way to cope with this is to use *abstraction* (Saitta and Zucker 2013). In essence, this involves developing an abstract model of the agent/domain that suppresses less important details. The abstract model allows us to reason more easily about the agent's possible behaviors and to provide high-level explanations of the agent's behavior. To efficiently solve a complex reasoning problem, e.g. planning, one may first try to find a solution in the abstract model, and then use this abstract solution as a template to guide the search for a solution in the concrete model. Systems developed using abstractions are typically more robust to change, as adjustments to more detailed levels may leave the abstract levels unchanged.

In recent joint work, Bita Banihashemi, Giuseppe De Giacomo, and I have proposed a general framework for *agent abstraction* (Banihashemi, De Giacomo, and Lespérance 2017) based on the situation calculus (SitCalc) (McCarthy and Hayes 1969; Reiter 2001) and the ConGolog (De Giacomo, Lespérance, and Levesque 2000) agent programming language. We assume that one has a high-level/abstract action theory, a low-level/concrete action theory, and a *refinement mapping* between the two. The mapping associates each high-level primitive action to a (possibly nondeterministic) ConGolog program defined over the low-level action theory that "implements it". Moreover, it maps each high-level fluent to a state formula in the low-level language that characterizes the concrete conditions under which it holds.

In this setting, we define a notion of a high-level theory being a *sound abstraction* of a low-level theory under a given refinement mapping. The formalization involves the existence of a suitable *bisimulation* relation (Milner 1971; 1989) between models of the low-level and high-level theories with respect to the mapping. With a sound abstraction, whenever the high-level theory *entails* that a sequence of actions is executable and achieves a certain condition, then the low level must also entail that there exist an executable refinement of the sequence such that the "translated" condition holds afterwards. Moreover, whenever the low level thinks that a refinement of a high-level action (perhaps involving exogenous actions) can occur (i.e., its executability is satisfiable), then the high level does also. Thus, sound abstractions can be used to perform effectively several forms of reasoning about action, such as planning, agent monitoring, and generating high-level explanations of low-level behavior. We also provide a "proof-theoretic" characterization that gives us the basis for automatically verifying that we have a sound abstraction.

In addition, we define a dual notion of *complete abstraction* where whenever the low-level theory *entails* that some refinement of a sequence of high-level actions is executable and achieves a "translated" high-level condition, then the high level also *entails* that the action sequence is executable and the condition holds afterwards. Moreover, whenever the high level thinks that an action can occur (i.e., its executability is satisfiable), then there exists a refinement of the action that the low level thinks can happen as well.

Many different approaches to abstraction have been proposed in a variety of settings such as planning (Sacerdoti 1974), automated reasoning (Giunchiglia and Walsh 1992), model checking (Clarke, Grumberg, and Long 1994), and data integration (Lenzerini 2002). Most of these do not deal with dynamic domains. Previous work on hierarchical planning generally makes major simplifying assumptions (Nau, Ghallab, and Traverso 2016). In contrast, our approach deals with agents represented in an expressive first-order framework.

For simplicity, we have focused on a single layer of abstraction, but the framework supports extending the hierarchy to more levels. Our approach can also support the use of ConGolog programs to specify the possible behaviors of the agent at both the high and low level, as we can follow (De Giacomo et al. 2016) and "compile" the program into the basic action theory (BAT) $\mathcal{D}$ to get a new BAT $\mathcal{D}'$ whose executable situations are exactly those that can be reached by executing the program. We have also identified a set of BAT constraints that ensure that for any low-level action sequence, there is a unique high-level action sequence that it refines. This is useful for providing high-level explanations of behavior and monitoring (De Giacomo, Reiter, and Soutchanski 1998).

More recently, we have examined how our abstraction

*This abstract describes joint work with Bita Banihashemi and Giuseppe De Giacomo; see the referenced co-authored papers for more details.

framework can be used to reason about an agent's online executions where the agent acquire new knowledge through sensing as it executes (Banihashemi, De Giacomo, and Lespérance 2018a). As well, we have used the framework to show how one can gain efficency in performing agent supervision, i.e., obtaining a supervisor/controller that minimally restricts an agent's behaviour to ensure that it satisfies some specification (Banihashemi, De Giacomo, and Lespérance 2018b).

In future work, we plan to investigate methodologies for obtaining abstractions for given objectives, as well as automated synthesis techniques to support this. We are also examining how to implement a hierarchical agent supervision tool for the case where abstract theory is propositional. We also plan to study how our abstraction framework can be used for online agent supervision. Finally, we will also explore how agent abstraction can be used in verification, where there is some related work (Mo, Li, and Liu 2016; Belardinelli, Lomuscio, and Michaliszyn 2016).

Acknowledgments

We acknowledge the support of Sapienza 2015 project "Immersive Cognitive Environments" and the National Science and Engineering Research Council of Canada.

References

Banihashemi, B.; De Giacomo, G.; and Lespérance, Y. 2017. Abstraction in situation calculus action theories. In *Proc. of the 31st AAAI Conference on Artificial Intelligence*, 1048–1055. AAAI Press.

Banihashemi, B.; De Giacomo, G.; and Lespérance, Y. 2018a. Abstraction of agents executing online and their abilities in the situation calculus. In Lang, J., ed., *Proc. of the 27th International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 1699–1706. ijcai.org.

Banihashemi, B.; De Giacomo, G.; and Lespérance, Y. 2018b. Hierarchical agent supervision. In André, E.; Koenig, S.; Dastani, M.; and Sukthankar, G., eds., *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, 1432–1440. IFAAMAS.

Belardinelli, F.; Lomuscio, A.; and Michaliszyn, J. 2016. Agent-based refinement for predicate abstraction of multi-agent systems. In *Proc. of the 22nd European Conference on Artificial Intelligence*, 286–294.

Clarke, E. M.; Grumberg, O.; and Long, D. E. 1994. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems* 16(5):1512–1542.

De Giacomo, G.; Lespérance, Y.; Patrizi, F.; and Sardiña, S. 2016. Verifying ConGolog programs on bounded situation calculus theories. In *Proc. of the 13th AAAI Conference on Artificial Intelligence*, 950–9568.

De Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 2000. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence* 121(1–2):109–169.

De Giacomo, G.; Reiter, R.; and Soutchanski, M. 1998. Execution monitoring of high-level robot programs. In *Proc. of the 6th International Conference on Principles of Knowledge Representation and Reasoning*, 453–465.

Giunchiglia, F., and Walsh, T. 1992. A theory of abstraction. *Artificial Intelligence* 5(2):323–389.

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 233–246.

McCarthy, J., and Hayes, P. J. 1969. Some Philosophical Problems From the Standpoint of Artificial Intelligence. *Machine Intelligence* 4:463–502.

Milner, R. 1971. An algebraic definition of simulation between programs. In *Proc. of the 2nd International Joint Conference on Artificial Intelligence*, 481–489.

Milner, R. 1989. *Communication and concurrency*. PHI Series in computer science. Prentice Hall.

Mo, P.; Li, N.; and Liu, Y. 2016. Automatic verification of Golog programs via predicate abstraction. In *Proc. of the 22nd European Conference on Artificial Intelligence*, 760–768.

Nau, D.; Ghallab, M.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

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

Sacerdoti, E. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 5(2):115–135.

Saitta, L., and Zucker, J.-D. 2013. *Abstraction in Artificial Intelligence and Complex Systems*. Springer-Verlag New York.