Recent Developments in Action Languages Based on Extensions of Answer Set Programs

Joohyung Lee
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, USA

Abstract

Action languages are formalisms designed for describing actions and their effects. Many of these languages can be viewed as high-level notations of answer set programs structured to represent transition systems, and, as such, they are limited to Boolean decision problems. We present recent developments in action languages that are based on extensions of answer set programs. A generalization of $\mathcal{C}+$ called $\mathcal{C}+$ *modulo theories* is based on an extension of answer set programs to the first-order level similar to the way satisfiability modulo theories extends propositional satisfiability. The formalism can represent hybrid transition systems in which both continuous and discrete changes co-exist, in an elaboration tolerant way. Another action language called $p\mathcal{BC}+$ is based on a probabilistic extension of answer set programs, and allows for probabilistic reasoning about transition systems, as well as probabilistic diagnosis for dynamic domains. These languages are implemented using ASP, SMT, and Markov Logic solvers. [1]

1 Action Languages

Action languages, such as $\mathcal{A}$ (Gelfond and Lifschitz 1993), $\mathcal{B}$ (Gelfond and Lifschitz 1998), $\mathcal{C}$ (Giunchiglia and Lifschitz 1998), and $\mathcal{C}+$ (Giunchiglia et al. 2004), are formalisms designed for actions and their effects. The expressive possibility of action languages, such as indirect effects, triggered actions, and additive fluents, has been one of the main research topics. Many of these languages can be viewed as high-level notations of answer set programs structured to represent transition systems.

Among them, action language $\mathcal{BC}+$ (Babb and Lee 2015) is highly expressive and general enough to embed several previous action languages. Building upon language $\mathcal{BC}$ (Lee and Meng 2013), language $\mathcal{BC}+$ is a proper generalization of both $\mathcal{B}$ and $\mathcal{C}+$, so it can express Prolog-style recursive definitions available in $\mathcal{B}$ as well as fluents whose behavior is described by defaults other than inertia as in $\mathcal{C}+$. Computational problems involving $\mathcal{BC}+$ descriptions can be reduced to computing answer sets. This fact led to an implementation of $\mathcal{BC}+$ by modifying system CPLUS2ASP (Babb and Lee 2013), which was originally designed to compute $\mathcal{C}+$ using ASP solvers.

However, $\mathcal{BC}+$, as well as many other action languages, are essentially propositional as they are based on answer set programs, which restricts its applicability to deterministic and discrete domains only.

This note describes two action languages recently introduced to overcome the limitations of the propositional setting, each of which based on different extensions of the stable model semantics.

2 Action Language Modulo Theories

In (Lee and Meng 2013), action language $\mathcal{C}+$ was extended to handle continuous changes based on the framework of *Answer Set Programming Modulo Theories (ASPMT)* (Bartholomew and Lee 2013).

2.1 Answer Set Programming Modulo Theories

ASPMT was designed for efficient computation of expressive nonmonotonic first-order reasoning. It is a tight integration of ASP with Satisfiability Modulo Theories (SMT), thereby employing the expressiveness of the ASP modeling language while leveraging efficient constraint/theory solving methods available in SMT. Unlike the traditional stable model semantics, it allows for the concept of "intensional" functions, which is described by a logic program rather than pre-fixed. Using SMT solving techniques involving functions, ASPMT can be applied to domains containing real numbers and alleviates the "grounding explosion " problem.

2.2 Action Language Modulo Theories

In (Lee and Meng 2013), we proposed a generalization of action language $\mathcal{C}+$ by defining it as a shorthand for ASPMT theories. The language $\mathcal{C}+$ *modulo theories* allows us to achieve the advantages of both hybrid automata and action languages, where the former provides an effective way to represent continuous changes, and the latter provides an elaboration tolerant way to represent (discrete) transition systems. In other words, the formalism gives us an elaboration tolerant way to represent hybrid transition systems. Unlike hybrid automata, the structured representation of states allows for expressing complex relations between fluents, such as recursive definitions of fluents and indirect effects of actions, and unlike the original $\mathcal{C}+$, which is propositional, the transitions described by the extended $\mathcal{C}+$ are not limited to discrete ones only; the advanced modeling capacity of action languages, such as additive fluents, statically defined fluents, and action attributes, can be achieved in the context of hybrid reasoning. We implemented a prototype system CPLUS2ASPMT, which

[1] This is a collaboration with Vladimir Lifschitz, Fangkai Yang, Michael Bartholomew, Yunsong Meng, Joseph Babb, Yi Wang, Samidh Talsania, and Nikil Loney.

allows for a succinct representation of hybrid transition systems in language $\mathcal{C}$+ modulo theories that can be compiled into the input languages of SMT solvers Z3 and DREAL. $\mathcal{C}$+ modulo theories takes advantage of ongoing developments in SMT. Based on the concept of "Satisfiability Modulo Ordinary Differential Equations (ODEs)" (Gao, Kong, and Clarke 2013), we embrace the concept into action language $\mathcal{C}$+ by introducing two new abbreviations of causal laws, one for representing the evolution of continuous variables as specified by ODEs and another for describing invariants that the continuous variables must satisfy when they progress. The general class of hybrid automata involving non-convex invariants can be expressed in the extended $\mathcal{C}$+ modulo ODE.

For example, the derivatives of differentiable fluent constants for turning left are declared in the language of CPLUS2ASPMT as follows:

```
derivative of x is cos(theta) if mode=turning_left.
derivative of y is sin(theta) if mode=turning_left
derivative of theta is tan(pi/18) if mode=turning_left.
```

Invariants for avoiding the collision with an obstacle are represented such as:

```
constraint x=X & y=Y ->> ((X-9)*(X-9) + Y*Y > 9).
always_t (x=X & y=Y ->> ((X-9)*(X-9) + Y*Y > 9)) if mode=V.
```

3 Probabilistic Extension of $\mathcal{BC}$+

In (Lee and Wang 2018a), we introduced a probabilistic extension of action language $\mathcal{BC}$+ based on the foundation of LP^{MLN}.

3.1 LP^{MLN}: Combination of ASP + MLN

LP^{MLN} is a probabilistic extension of answer set programs with the concept of weighted rules, whose weight scheme is adopted from that of Markov Logic (Richardson and Domingos 2006). Like Markov Logic, not all LP^{MLN} rules have to be true but, roughly speaking, the more rules are true, the larger weight is assigned to the corresponding stable model.

It is shown in (Lee and Wang 2016; Lee, Meng, and Wang 2015) that LP^{MLN} is expressive enough to embed Markov Logic and several other probabilistic logic languages, such as ProbLog (De Raedt, Kimmig, and Toivonen 2007) and Pearls' Causal Models (Pearl 2000).

It is also shown in (Lee and Yang 2017; Lee and Wang 2016) that LP^{MLN} can be reduced to ASP and Markov Logic. This finding was leveraged to implement LP^{MLN} using existing implementations of ASP and Markov Logic solvers.

3.2 $p\mathcal{BC}$+

In (Lee and Wang 2018a), we presented a probabilistic extension of $\mathcal{BC}$+, which we call $p\mathcal{BC}$+. Just like $\mathcal{BC}$+ is defined as a high-level notation of answer set programs for describing transition systems, $p\mathcal{BC}$+ is defined as a high-level notation of LP^{MLN} programs. Language $p\mathcal{BC}$+ inherits expressive logical modeling capabilities of $\mathcal{BC}$+ but also allows us to assign a probability to a sequence of transitions.

In (Lee and Wang 2018a), we show how probabilistic reasoning about transition systems, such as prediction, postdiction, and planning problems, can be modeled in $p\mathcal{BC}$+ and computed using an implementation of LP^{MLN}. Further, we show that it can be used for probabilistic abductive reasoning about dynamic domains, where the likelihood of the abductive explanation is derived from the parameters manually specified or automatically learned from the data based on the learning algorithm described in (Lee and Wang 2018b).

For example, the following causal laws describe that the abnormality *EnterFailed* has 0.1 chance to occur when the action $Goto$ is executed:

caused $\{\sim$*EnterFailed*$\}^{\text{ch}}$ **if** $\sim$*EnterFailed*
caused *Pf_EnterFailed* $= \{\mathbf{t} : 0.1, \mathbf{f} : 0.9\}$
caused_ab *EnterFailed* **if** $\top$ **after** *pf_EnterFailed* $\wedge$ $Goto(r)$.

References

Babb, J., and Lee, J. 2013. Cplus2ASP: Computing action language $\mathcal{C}$+ in answer set programming. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 122–134.

Babb, J., and Lee, J. 2015. Action language $\mathcal{BC}$+: Preliminary report. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Bartholomew, M., and Lee, J. 2013. Functional stable model semantics and answer set programming modulo theories. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, volume 7, 2462–2467.

Gao, S.; Kong, S.; and Clarke, E. 2013. Satisfiability modulo ODEs. *arXiv preprint arXiv:1310.8278*.

Gelfond, M., and Lifschitz, V. 1993. Representing action and change by logic programs. *Journal of Logic Programming* 17:301–322.

Gelfond, M., and Lifschitz, V. 1998. Action languages. *Electronic Transactions on Artificial Intelligence* 3:195–210.

Giunchiglia, E., and Lifschitz, V. 1998. An action language based on causal explanation: Preliminary report. In *Proceedings of AAAI*, 623–630. AAAI Press.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153(1–2):49–104.

Lee, J., and Meng, Y. 2013. Answer set programming modulo theories and reasoning about continuous changes. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Lee, J., and Wang, Y. 2016. Weighted rules under the stable model semantics. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 145–154.

Lee, J., and Wang, Y. 2018a. A probabilistic extension of action language $\mathcal{BC}$+. *Theory and Practice of Logic Programming*. To appear.

Lee, J., and Wang, Y. 2018b. Weight learning in a probabilistic extension of answer set programs. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*. To appear.

Lee, J., and Yang, Z. 2017. LPMLN, weak constraints, and P-log. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 1170–1177.

Lee, J.; Meng, Y.; and Wang, Y. 2015. Markov logic style weighted rules under the stable model semantics. In Technical Communications of the 31st International Conference on Logic Programming.

Pearl, J. 2000. *Causality: models, reasoning and inference*, volume 29. Cambridge Univ Press.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.