Towards Answer Set Prolog Based Architectures for Intelligent Agents

Sandeep Chintabathina
Texas Tech University
Department of Computer Science
Lubbock, TX 79409, USA
sandeep.chintabathina@ttu.edu

An important research area in the field of AI is the design of intelligent agents acting in a changing environment. Research in this area has led to development of agent architectures that support various tasks such as planning, diagnosis, learning etc. One such architecture is based on the agent repeatedly executing the *observe-think-act-loop* (Baral and Gelfond 2000; Kowalski and Sadri 1999). This architecture is applicable if the world including the agent and its environment is viewed as a transition diagram whose states correspond to possible physical states of the world and whose arcs are labeled by actions. A transition, $\langle \sigma, a, \sigma' \rangle$, of a diagram denotes that action a in executable in state σ and that after the execution of a the system may move to state σ'. The diagram consists of all possible trajectories of the system. One of the approaches to describing these diagrams is a theory based on *action languages* - formal models of parts of natural language used for reasoning about actions and their effects (Gelfond and Lifschitz 1998). A theory in an action language (often called an *action description*) describes a transition diagram that contains all possible trajectories of a given dynamic system.

Currently there are several action languages that are used to study different features of dynamic domains. For instance action language $\mathcal{AL}$ (Baral and Gelfond 2000; Turner 1997) is simply an extension of action language $\mathcal{A}$ (Gelfond and Lifschitz 1993) by state constraints which express causal relations between *fluents* - functions whose values depend on state and may change as a result of actions. The semantics of $\mathcal{AL}$ formalize McCarthy's *Principle of inertia* which says that "*Things tend to stay the same unless they are changed by actions*" (McCarthy and Hayes 1969). Similarly $\mathcal{A}_C$ (Baral and Gelfond 1997) expands $\mathcal{A}$ by allowing concurrent actions in the causal laws (which are prohibited in $\mathcal{A}$). Another extension of $\mathcal{AL}$ is the action language H (Chintabathina, Gelfond, and Watson 2005) which is capable of modeling domains containing *continuous processes* - properties of the domain whose values change continuously with time. Semantics of this language are based on a generalization of McCain-Turner equation (McCain and Turner 1995) which allows us to define processes using state constraints. (Chintabathina, Gelfond, and Watson 2007) introduces a new action language $\mathcal{CARD}$ which extends both $\mathcal{AL}$ and $\mathcal{A}_C$. This language is capable of dealing with concurrent and non-deterministic effects of actions, default fluents and actions which use resources. In some respects it is more powerful than action language $\mathcal{C}+$ (Giunchiglia et al. 2004) although they are based on different underlying assumptions. Currently we are working on refining the syntax and semantics of $\mathcal{CARD}$. At some point in the future we want to expand $\mathcal{CARD}$ by continuous processes.

The next step to accomplishing various tasks of an agent is to be able to reason about the knowledge. In order to do so action descriptions are usually encoded in to equivalent logic programs. The class of logic programs that we consider here are called *Answer Set Prolog* (ASP) programs which are defined under answer set semantics (Gelfond and Lifschitz 1991). ASP is a declarative logic programming language quite suitable for knowledge representation, reasoning and declarative problem solving (Baral 2003). Previous works (Baral and Gelfond 2000; Balduccini and Gelfond 2003a; Balduccini 2007) have shown that ASP can be used for a variety of AI tasks such as planning, diagnosis, learning etc. ASP's ability to build elaboration tolerant knowledge bases and its non-monotonicity makes it suitable for modeling commonsense behavior of agents. It is capable of expressing defaults, causal relations and dealing with incomplete knowledge.

By translating action descriptions in to ASP programs, various tasks of the agent are reduced to computing answer sets of the ASP programs. (Balduccini and Gelfond 2003a) shows an encoding of action descriptions of $\mathcal{AL}$ in to ASP programs and thereby how various diagnostic tasks are reduced to computing answer sets of programs. (Chintabathina, Gelfond, and Watson 2005) presents an approach to translating action descriptions of H in to ASP. We will talk more about it later. Currently we are investigating issues with encoding action descriptions of $\mathcal{CARD}$.

There are several inference engines that are capable of computing answer sets. Such engines are called *answer set solvers*. They start by grounding the program i.e. instantiating variables by ground terms and computing the answer sets of the resulting ground program. Answer sets are computed using substantially modified and expanded satisfiability checking algorithms. Although such solvers are capable of finding solutions to a variety of problems, there are a number of deficiencies. According to (Mellarkod, Gelfond,

and Zhang 2008), these solvers are not efficient when dealing with programs that contain variables ranging over large domains. The ground instantiations of such programs are huge, leading to memory and time problems. In order to overcome this problem researchers have developed new and improved solvers from time to time. Recently (Mellarkod, Gelfond, and Zhang 2008) has developed a powerful extension of ASP, $\mathcal{AC}(\mathcal{C})$, and defined an algorithm, $\mathcal{AC}solver$, for computing answer sets in the new language. The solver partially avoids grounding of variables over large domains and replaces grounding with the use of constraint solving techniques. Another limitation of the answer set solvers is that they are unable to keep up with advancements in the field. The input languages of many of the solvers do not support features that researchers would like them to have. This is the reason why researchers are working on extending ASP and developing the corresponding solvers. One such extension of ASP is CR-Prolog (Balduccini and Gelfond 2003b) which is very useful for diagnostic tasks and generating high quality plans.

Coming back to encoding action descriptions in to ASP, let us talk about language H. We know that statements of H contain continuous functions. Translating these statements into ASP rules is straight forward. However, due to issues involved with grounding it is not possible to run the resulting programs under current answer set solvers. This is why (Chintabathina, Gelfond, and Watson 2005) proposes to discretize functions occuring in the statements of H and then translate the resulting action descriptions in to ASP. According to (Chintabathina, Gelfond, and Watson 2005) there is a provably correct translation for discretized action descriptions of H. However, this approach suffers from drawbacks because answer set solvers are inefficient whenever the translated programs contain variables ranging over large domains. $\mathcal{AC}solver$ is not useful here because the translated programs do not comply with the necessary conditions for input programs of the $\mathcal{AC}solver$. Hence, there is a need for development of new and more advanced solvers.

We encounter a similar situation with our current version of action language $\mathcal{CARD}$. Subclasses of action descriptions of $\mathcal{CARD}$ that do not contain any non-deterministic laws can be translated directly in to rules of CR-Prolog. If the resulting programs do not contain variables ranging over large domains then answer sets of these programs are computed efficiently. For action descriptions that contain non-deterministic laws, in the general case, there is no translation that captures the semantics of these laws. This is because CR-Prolog was designed prior to $\mathcal{CARD}$. Therefore, we now have a situation where we need a new logic programming language and a solver to implement $\mathcal{CARD}$.

In order to meet our needs one possible approach would be to investigate ASP and its extensions and see if we can add new features or remove restrictions to accomodate new action languages. Any additions or modifications will also prompt changes in the corresponding solvers and algorithms. We will investigate whether language $\mathcal{AC}(\mathcal{C})$ can be modified to accept programs obtained by translating action descriptions of H. Later on we will investigate if we can make any modifications to CR-Prolog in order to capture the semantics of $\mathcal{CARD}$. Finally we will develop solvers for these modified languages.

References

Balduccini, M., and Gelfond, M. 2003a. Diagnostic reasoning with A-Prolog. *TPLP* 3(4-5):425–461.

Balduccini, M., and Gelfond, M. 2003b. Logic programs with consistency-restoring rules. In *Proc. of AAAI-03 Spring Symposium Series*, 9–18.

Balduccini, M. 2007. Learning action descriptions with A-Prolog: Action language C. In *Proc. of CommonSense'07*, 13–18.

Baral, C., and Gelfond, M. 1997. Reasoning about effects of concurrent actions. *Journal of Logic Programming* 31(1-3):85–117.

Baral, C., and Gelfond, M. 2000. Reasoning agents in dynamic domains. In Minker, J., ed., *Logic-Based Artificial Intelligence*, 257–279. Kluwer Academic Publishers.

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press.

Chintabathina, S.; Gelfond, M.; and Watson, R. 2005. Modeling hybrid domains using process description language. In *Proc. of ASP-05*, 303–317.

Chintabathina, S.; Gelfond, M.; and Watson, R. 2007. Defeasible laws, parallel actions, and reasoning about resources. In *Proc. of CommonSense'07*, 35–40. AAAI Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3/4):365–386.

Gelfond, M., and Lifschitz, V. 1993. Representing action and change by logic programs. *Journal of Logic Programming* 17:301–321.

Gelfond, M., and Lifschitz, V. 1998. Action languages. *Electronic Transactions on AI* 3(16).

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

Kowalski, R., and Sadri, F. 1999. From logic programming towards multi-agent systems. *Annals of Mathematics and Artificial Intelligence* 25(3-4):391–419.

McCain, N., and Turner, H. 1995. A causal theory of ramifications and qualifications. In Mellish, C., ed., *Proc. of IJCAI-95*, 1978–1984. Morgan Kaufmann.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence* 4:463–502.

Mellarkod, V.; Gelfond, M.; and Zhang, Y. 2008. Integrating Answer Set Programming and Constraint Logic Programming. In *Proc. of ISAIM'08*.

Turner, H. 1997. Representing actions in logic programs and default theories: A situation calculus approach. *Journal of Logic Programming* 31(1-3):245–298.