A General Approach to the Implementation of Action Theories

Gerardo I. Simari   Diego R. García   Gabriel R. Filocamo

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial
Departamento de Ciencias e Ingeniería de la Computación

Universidad Nacional del Sur. Av. Alem 1253, (8000) Bahía Blanca, Argentina
Tel: ++54 291 4595135 - Fax: ++54 291 4595136
{gis,drg,grf}@cs.uns.edu.ar

Abstract

Much effort has been dedicated to provide a general model of agents working in complex environments. This research focuses on the high level cognition used in determining the behavior of the agent.

The language of the Situation Calculus represents a very useful way of modeling an agent's knowledge of its environment. One of its advantages is that there exist methods to derive an executable program from a basic set of axioms. This program can then be used to determine the actions that are necessary in order to accomplish certain goal states. The main objective of this line of work is to obtain an automatic way of deriving such an executable program.

1 Introduction and Background

An important goal in the research programme of Artificial Intelligence is to develop autonomous entities called *agents*. This autonomy is of utmost relevance for an agent to be considered intelligent in the environment that it occupies; the study of the relationship among beliefs, actions, and their consequences is one way to go in obtaining this objective.

The research we are conducting in the LIDIA lies within the area of Cognitive Robotics. Work in this field is directed towards the provision of a uniform theoretical and implementation framework for autonomous robotic or software agents that reason, act, and perceive in complex environments. The difference between "traditional" robotics and cognitive robotics is the emphasis placed on higher level cognition in determining agent behaviors.

One way of modeling an agent's beliefs about the world, the possible actions it can take, and its effects on the environment is the *Situation Calculus* [11, 12].

1.1 A Brief Introduction to the Situation Calculus

We will briefly introduce the Situation Calculus in order to provide the framework necessary in the rest of the presentation.

The Situation Calculus is a second order language, quite suitable for representing dynamically changing worlds. Each change in the world is the result of a specific *action*, which is defined in the language. A *situation* stands for a history of the actions taken in the world (simply a sequence of actions), and is represented as a first order formula. A special situation, the constant s_0, is used to represent the initial state. Every other situation is denoted by the special functional symbol $do(\alpha, s)$; this means that action α executed in situation s leads to a new situation, called $do(\alpha, s)$, where s is a situation built in the same way, or s_0. For example, for action $grab(x)$ (which denotes grabbing object x), $do(grab(x), s)$ is the situation which results from grabbing object x in situation s.

There exists a certain type of relations that may vary from situation to situation, which are called *Relational Fluents*. These are represented by predicates that take a situation term as their last argument. Functions whose values depend on the situation are called *Functional Fluents*, and also take a situation term as their last argument. For example, the relational fluent $nextTo(x)$, which holds in s_0, no longer holds in $do(walk(y), s_0)$, where object x is different from object y.

Another type of relations, which do not depend on the situation, are represented by predicates without a situation parameter, and are called *Non-Fluent predicates*. Similarly, those functions that are situation independent are called *Non-Fluent functions*.

The theory is completed by a variety of axioms, which capture the different aspects of the world that is being modeled. These axioms are:

- **Action Precondition Axioms:**

  The Action Precondition Axioms specify the requirements that must be satisfied for an action to be executed. For this reason, there will be *one* axiom for each action. In the definition of these axioms the predicate $poss(a, s)$ is used, meaning that action a is possible (can be executed) in situation s.

  An Action Precondition Axiom will have the following general form $poss(a, s) \equiv \Upsilon$, where Υ is a first order formula.

- **Effect Axioms** The Effect Axioms specify how the relational and functional fluents' values change after the execution of an action. In this case, there will be *at least one* axiom for each fluent.

  An Effect Axiom for a relational fluent f will have the following general form: $\phi_f \rightarrow f(t^n, do(\alpha, s))$, and for a functional fluent g, $\gamma_g \rightarrow g(t^n, do(\alpha, s)) = r$, where $t^n = t_1, t_2, ..., t_n$ are terms, and ϕ_f,γ_g are first order formulas. The Effect Axioms for relational fluents can be subdivided into positive and negative axioms. An axiom is *positive* if the relational fluent f is not negated in

- **Successor State Axioms**

  The Successor State Axioms provide a complete way of describing how the world evolves as a response to the execution of actions. Each axiom describes how to compute a the value of a fluent for the next situation, given its value for the current situation. Intuitively the successor state axiom for a fluent f has the following structure:

$$\begin{array}{lll} \text{True afterwards} & \Leftrightarrow & [\text{An action made it true} \vee \\ & & \text{True already and no action made it false}] \end{array}$$

The use of $\Leftrightarrow$ means that the axiom will be true after the execution of the action if it is made true or it stays true; and that it will be false otherwise. For each relational fluent f, the successor state axiom has the following form:

$$f(x^n, do(a,s)) \equiv \gamma_f^+(x^n, a, s) \vee f(x^n, s) \wedge \neg\gamma_f^-(x^n, a, s)$$

where $\gamma_f^+(x^n, a, s) \rightarrow f(x^n, do(a,s))$ and $\gamma_f^-(x^n, a, s) \rightarrow f(x^n, do(a,s))$ are the positive and negative normal form effect axioms for the fluent f, respectively.

2 Objectives

Central in this work is the goal of making the process of obtaining an executable implementation of the action theory an automatic one. Therefore, what we want is to be able to compile the language of the Situation Calculus into an object language such as Prolog. There are many advantages to having such a compiler; the implementation of an action theory can be quite tedious, and having an automatic processor can ease the design of such a theory. For example, if the initial axioms suffer modifications, the final result can be quickly visualized.

There exists a process, based on work done by Lloyd and Topor [10], that can be used to obtain a prolog program from the initial Situation Calculus axioms. They propose a set of transformations suggested by Clark's theorem [1] which reformulate the Action Precondition and Successor State axioms in a way that results in a direct translation into Prolog clauses.

Once the basic axioms are specified, the automatic process should generally consist of the following steps:

- **Transform the Action Precondition Axioms:**
  - Obtain the if-halves from the Action Precondition Axioms.
  - Apply the Lloyd-Topor transformation rules to the if-halves in order to obtain formulas that are easy to implement (Lloyd-Topor normal form).
- **Transform the Successor State Axioms:**
  - Obtain the if-halves from the Successor State Axioms.
  - Apply the Lloyd-Topor transformation rules to the if-halves in order to obtain formulas that are easy to implement (Lloyd-Topor normal form).
- **Initial Database definition:** Obtain the if-halves from the initial database definition. These definitions obey the form:

  $$F(x_n, s_0) \equiv \Psi_F(x_n, s_0)$$

  where $F(x_n, s_0)$ will not be negated in this case. This is due to the fact that negative information is represented in Prolog by the *Closed World Assumption* (CWA). It should also be noted that F cannot be a functional fluent in this case because this type of fluent cannot be directly implemented in Prolog. Notwithstanding, an n-ary functional fluent can be implemented as a $(n+1)$-ary Prolog predicate, where the last argument represents the value of the fluent.

- **Non-Fluent Predicate definition:** Obtain the if-halves from the definitions, and then apply the Lloyd-Topor transformation rules to these if-halves.
- **Obtain a program in a given object language:** Using the formulae obtained in the previous steps, derive a fully functional program that models the specified action theory.

3 Current Research Line

Current work is being dedicated to the study of the applications of the Situation Calculus. One way of understanding the implementation of an action theory seems to be the resolution of a simple problem that covers all of the details of the process. One such problem is the *Monkey and Bananas Problem*

The Monkey and Bananas Problem is a classic planning problem. It was proposed by John McCarthy in 1963, and later reprinted in [11]. A monkey is in a room that contains a bunch of bananas hanging from the ceiling, and a chair. The monkey can't reach the bananas from the floor. Nevertheless, if the chair were below the bananas, and the monkey were standing on top of it, then the bananas would become reachable. One possible setup for an initial state is the monkey not standing next to the chair, and the chair not sitting below the banana bunch.

One advantage to using this problem as a first approach is that, with a simple world like the one described, there shouldn't be obstacles that obscure the process of obtaining an executable program from an initial formulation. Because there are few objects and actions, the number of axioms needed to formalize an action theory will be reduced, which is useful in dealing with errors committed in the initial stages of the resolution.

Once the program is obtained, it can be used to obtain plans in which a set of fluents are valid (or not). For example, it could be used to obtain a plan which the monkey can use to grab the bananas and get down from the chair.

Having solved a simple problem like the one described, it will be easier to approach broader problems that could contain special cases that must be taken into account in the construction of an automated tool.

References

[1] Clark, K. L. Negation as failure. In *Logic and Databases* (New York, 1978), H. Gallaire and J. Minker, Eds., Plenum Press, pp. 293–322.

[2] Elkan, C. Reasoning about action in first-order logic. In *Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence* (Vancouver, British Columbia, May 1992).

[3] Finzi, A. I., Pirri, F., and Reiter, R. Open world planning in the situation calculus. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)* (Menlo Park, CA, July 30– 3 2000), AAAI Press, pp. 754–760.

[4] Gelfond, M., and Lifschitz, V. Representing action and change by logic programs. *Journal of Logic Programming 17*, 2,3,4 (1993), 301–323.

[5] Gelfond, M., Lifschitz, V., and Rabinov, A. What are the limitations of the situation calculus? In *Automated Reasoning: Essays in Honor of Woody Bledsoe*, R. Boyer, Ed. Kluwer Academic Publishers, Dordrecht, 1991, pp. 167–179.

[6] Hintikka, J. *Knowledge and Belief.* Cornell University Press, Ithaca, New York, 1977.

[7] Kowalski, R. A., and Sergot, M. J. A logic-based calculus of events. *New Generation Computing 4* (1986), 67–95.

[8] Lespérance, Y., Levesque, H. J., Lin, F., Marcu, D., Reiter, R., and Scherl, R. B. A logical approach to high-level robot programming—A progress report. In *Control of the Physical World by Intelligent Systems, Papers from the 1994 AAAI Fall Symposium* (Menlo Park, California, 1994), B. Kuipers, Ed., American Association for Artificial Intelligence, American Association for Artificial Intelligence, pp. 79–85.

[9] Lespérance, Y., Levesque, H. J., and Reiter, R. A situation calculus approach to modeling and programming agents. In *Foundations of Rational Agency*, M. J. Wooldridge and A. Rao, Eds. Kluwer Academic Pub;ishers, Dordrecht, 1999, pp. 275–299.

[10] Lloyd, J. W. *Foundations of Logic Programming, Second Edition.* Springer-Verlag, 1987.

[11] McCarthy, J. Situations, actions, and causal laws. In *Semantic Proceedings of the tri-annual IFIP Conf, Minsky (ed), Machine Intelligence, eds: Meltzer, and Michie, vars. PublishersT Press.* 1968.

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

[13] Moore, R. Reasoning about knowledge and action (technical report 191). Tech. rep., SRI AI Center, 1980.

[14] Pylyshyn, Z. W. *Computation and cognition : toward a foundation for cognitive science.* Cambridge, Mass. : MIT Press, 1984, 292 p., CALL NUMBER: BF311 .P93 1984, 1984.