Ray Reiter's *Knowledge in Action*

A Review

Drew McDermott

This is a beautiful book, in conception and execution. What Ray Reiter has done has taken a set of ideas worked out by him and his collaborators over the last 11 years and recrystallized them into a sustained and consistent presentation.[1] This is not a collection of those papers but a complete rewrite that avoids the usual repetition and notational inconsistency that one might expect. It makes one wish everyone as prolific as Reiter would copy his example—but because that's unlikely, we must be grateful for what he has given us.

In case you haven't heard, Reiter and his crew, starting with the publication of Reiter (1991), breathed new life into the situation calculus (McCarthy and Hayes 1969) that had gotten the reputation of being of limited expressiveness. The basic concept of the calculus is, of course, the *situation,* which we can think of as a state of affairs, that is, a complete specification of the truth values of all propositions (in a suitable logical language), although that's closer to McCarthy's and Hayes's traditional formulation than the analysis Reiter settles on (which I describe later). Any atomic formula whose truth value can change over time must be written with a predicate that takes a situation as argument. We might write child($person_1$, $person_2$) if we analyze a domain as not allowing the set of children or parents of a person to change. However, if the analysis must take into account time scales in which new children can be born, we

■ *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems,* Ray Reiter, MIT Press, 448 pp., 2001, ISBN 0-262-18218-1.

must write instead child($person_1$, $person_2$, s), where s is a term denoting a situation. The meaning is that in s, $person_2$ is a child of $person_1$; in other situations, this relationship might not hold true.

The basic notation for expressing change in the situation calculus is do(a, s), meaning the situation resulting from executing the action a starting in situation s. To express the fact that adopting a person makes him/her one's child, we might write

$$(\forall p_1, p_2, s)(\text{child}(p_1, p_2, \text{do}(\text{adopt}(p_1, p_2), s)))$$

"In the situation resulting from executing the action adopt(p_1, p_2) in s, p_2 is a child of p_1." Note that the notation does not make explicit who is executing the action; I use the term *target agent* to denote the implicit executor, connoting the agent who might actually carry out the actions being reasoned about.

The notation do(a, s) seems to raise two troublesome questions. The first is, What happens between s and do(a, s)? The answer seems to be "Nothing" because traditional axioms specify truth values of propositions in do(a, s) as a function of their values in s, which doesn't leave much of a role for situations, if there are any, between the two.

The second problem is how to deal with the notorious frame problem and its various relatives (McCarthy and Hayes 1969; Pylyshyn 1987). The problem is that there is no implicit relationship between the truth values of a proposition in s and do(a, s), or do(a_2, do(a_1, s)). We can only state these relationships by adding axioms of some sort. Getting the axioms right, and getting algorithms based on them right, has not been easy.

As a result, the basic situation calculus lived as a textbook curiosity for several years. Researchers interested in practical applications of temporal reasoning, such as automated planning, kept the basic ontology of the calculus but focused on efficient algorithms to the neglect of logical foundations. Researchers interested in the underlying logic looked for ways of getting the logic to work, most ingeniously by resorting to nonmonotonic techniques.

Reiter began putting things back on track by addressing both these questions. He adopted and extended the solution to the frame problem of Haas (1987), Schubert (1990), and Davis (1990) and noticed that the basic do(a, s) notation does not in fact rule out the occurrence of interesting events between s and do(a, s). The result has been a stream of important papers that have given new life to the attempt to formalize realistic reasoning in temporal contexts.

Knowledge in Action is a recapitulation of this work. It begins by explaining the framework and the solution to the frame problem. Chapters 4 and 5 then give a rigorous basis for the situation calculus and for logic-programming reasoning in the important special case where the closed-world assumption holds. This is the assumption that any atomic formula that does not follow from a given set of axioms is false. A key result of Chapter 4 is the precise specification of how to regress a formula Q, that is, to derive the weakest formula P whose truth in a situation will make Q true in a later one. A formula can be regressed through any number of actions, so that to deduce whether Q is true in S_k = do($action_k$, do($action_{k-1}$, ..., do($action_1$, S_0))), one can use regression to produce a formula Q_0 such that if Q_0 is true in S_0, Q is true in S_k. This fact allows one to use a logic program for the contents of S_0 to answer queries about all later situations.

Chapter 6 introduces the GOLOG language, a language for describing actions in terms of the situation calculus. It allows for conditional; iterative; and above all, nondeterministic actions. The nondeterminism is important because a key reasoning task using GOLOG is to specify a nondeterministic program and then prove (using Prolog) that there is a successful way to execute it. This is an attractive way to find *classical plans* (that is, sequences of actions) to achieve goals; the GOLOG program is a hint about what the plan looks like, and a successful execution trace is a plan.

Chapter 7 is the eye-opener for those who thought that the situation calculus can't talk about the passing of continuous time. All you have to do is require that every action instance have a time of occurrence. Instead of jumpOver(x), you might have the action jumpOver(x, t). Even better, you can have two actions, startJumpOver(x, t_1) and endJumpOver(x, t_2), with the proposition jumpingOver(x) true in the interval $[t_1, t_2]$. It's not hard to generalize this idea to continuous changes, so that altitude(s) is the real number denoting the jumper's (that is, the target agent's) height above the ground in situation s. A function such as altitude, with a situation argument, is called a *fluent.* (A predicate augmented with a situation argument, such as child, is called a *relational fluent.* Sometimes the term *fluent* is applied to the terms or atomic formulas constructed with functional and relational fluents.)

It is important to realize that Reiter's ontology does not mention a situation—let alone a continuum of situations—between do(startJumpOver(x, t_1), s) and do(endJumpOver(x, t_2), do(startJumpOver(x, t_1), s)). In his framework, a situation is the result of a sequence of events separated by noninfinitesimal time intervals. Thus far, we have talked only about event sequencess of the form do($action_k$, do($action_{k-1}$, ..., do($action_1$, S_0))), that is, actions taken by a single agent. However, in Chapter 7, the scope is broadened to include actions taken by nature, that is, events that occur when their preconditions are true, without any agent having to decide to make them occur. We can use these autonomous events to provide a more natural representation of the jumpOver example in which the end of the jump occurs automatically when the jumper hits the ground.

In spite of the addition of such autonomous events, it is still the case that a situation is a sequence of discrete entities. It might sound as if there is a problem here: How can we reason about continuous change if quantities jump from one value to another from one situation to the next? The answer is that a different formal language must be used in talking about change between situations, namely, the language of equations with elapsed time as an unknown variable. In the jumpOver example, the jump altitude could be

$$\text{jumpvel}(s) \times (t - \text{start}(s)) - 1/2G(t - \text{start}(s))^2$$

Here start(s) is the time a situation starts; t is some subsequent time. The possible values of t are the positive real numbers, but these are not the times of situations. Instead, constraints on t are used to specify when the next action might, or the next autonomous event must, occur and, hence, when the next situation can be constructed. In our continuing example, the event landAfterJump(t) occurs when t is such that the jump altitude gets back to zero. The net result is that there are no situations between do(jumpOver(x, t_1), s) and do(landAfterJump(t_2), do(jumpOver(x, t_1), s)). However, there don't have to be! All the reasoning about t – start(do(jumpOver(x, t_1), s)) is for the purpose of finding the value of t_2–t_1, so that the situation resulting from landAfterJump can be constructed.[2]

Hence, all situations can be located in a tree rooted at the distinguished *initial situation* S_0. The situation do(a_n, do(a_{n-1}, ...(do(a_1, S_0)))) corresponds to the branch labeled a_1, ..., a_n. The number of children in each situation seems to be tacitly assumed to be finite but only because actions with numeric-valued parameters are not in Reiter's focus.

This ontology of situations suffices for various applications and ramifications of the basic calculus. In Chapter 11, "Sensing and Knowledge," some further evolution becomes necessary. To this point, actions have been characterized in terms of their effects, meaning the changes they cause in the values of fluents. Actions have not had any effect on the target agent's knowledge of the values of fluents; more precisely, they have not had any unpredictable effect on knowledge. In the situation do(buy (ob21), s), the target agent owns ob21 and implicitly knows it owns ob21.

Sensing actions require a change because the only purpose of sensing is to change what you know in unpredictable ways; as Reiter points out, if the reasoner knows the result of a sensing action, there is no point in actually carrying it out. The change is to adopt the idea that for an agent A to believe proposition P is for P to be true in all possible worlds that are compatible with A's beliefs. This approach is originally owed to Hintikka (1961) and Kripke (1963) and was applied to temporal reasoning by Moore (1980). Reiter adopts Moore's formulation and integrates it with his situation-calculus framework.

More formally, Reiter identifies possible worlds with situations and provides a relation K such that K(s', s) means that in situation s, the set of worlds compatible with the target agent's beliefs includes s'. In this scheme, the statement that in S_0 (the initial situation), the target agent believes that it's in Kansas would be represented thus:

$$(\forall s')(K(s', S_0) \supset in(Kansas, s'))$$

The statement that the target agent does not believe that it's in Kansas comes out as

$$(\exists s')(K(s', S_0) \wedge \neg in(Kansas, s'))$$

Note that this formula is distinct from a statement that the agent believes it is not in Kansas, whose formulation is left as an exercise for the reader.

The only change that is required in the ontology to accommodate the K

relation is to allow for multiple trees, rooted at multiple initial situations. However, there is still a distinguished S_0; all other initial situations are related to it through the *K* relation.

On this ontological foundation, Reiter builds a theory of *knowledge-producing actions,* those that narrow the set of alternative possible worlds. The hard part is to characterize regression for such actions, that is, how facts of the form know(…) in later situations depend on the truth of facts in the initial situation. Obviously, if knowledge-producing actions are involved, the contents of the initial situation are not the only determinant of the truth values of propositions in later situations. However, there are subtle problems about exactly what the target agent knows and does not know in the initial situation and, hence, how its knowledge changes over time. Reiter's exploration of these problems is clear and deep.

There are many more topics covered in *Knowledge and Action,* including databases, progression (forward inference from situations to their successors), planning, open worlds, exogenous events, and decision theory. You have to read it to appreciate its richness.

The book is not without weaknesses. There are many theorems but few proofs. Many are left as exercises for the reader; for others, one must go back to the original papers.

Much of the book is concerned with extending GOLOG to accommodate each new feature added to the basic situation calculus. I don't quite understand why GOLOG is so important. Apparently, the main reasons are that the meaning of any GOLOG program can be given as a translation into the situation calculus and that GOLOG interpreters can be written in Prolog in a natural way (or "ways," depending on exactly which version of GOLOG we are talking about). The first reason turns out not to be too surprising; for every GOLOG program ρ, all we have to do is define the relation Do(ρ, *s*, *s′*), specifying that *s′* is a possible situation resulting from executing ρ starting in *s.*

I believe that the second reason for the importance Reiter attaches to GOLOG is that it is possible, with a little ingenuity, to write Prolog programs—GOLOG interpreters—to find *s′* given ρ and a description of *s.* In the early going, the purpose of the GOLOG interpreter is to find a deterministic sequence of actions that implements a nondeterministic ρ. By *nondeterministic,* I mean that the program contains actions of the form *A*1 | *A*2, meaning "Either *A*1 or *A*2." Because *A*1 and *A*2 can be elaborate actions with hard-to-predict consequences, it can be nontrivial and useful to verify that there is a way to make all the choices that arise in the course of executing a program without running into a dead end. However, in later chapters, the purpose keeps changing. For example, in the chapter on decision theory, the purpose is to compute the probability that a given proposition is true after executing a deterministic GOLOG program (one with no occurrences of |). This reasoning task is quite different, and every chapter seems to discuss yet another one. Some of the tasks are natural, but others seem to have been chosen because they can be carried out by relatively simple Prolog programs. If one inquires what reasoning tasks are actually studied by researchers in planning, knowledge theory, and decision theory, one comes up with a rather different set. It might have been more interesting if Reiter had thought about how his formalism would apply in these contexts.

"If Reiter had …" is a hard phrase to write. It grieves me deeply to know that we will never find out what Ray Reiter would have come up with as he continued to work in the area of temporal reasoning. We will have to be content with the research presented in his wonderful book.

Notes

1. As most readers know, Ray Reiter died in September 2002 during the writing of this review. I decided to keep references to him in the tense one would use for someone still living because it's so hard not to think of him that way.

2. In the programs presented in the book, the value is found by using the built-in constraint solver of a particular Prolog implementation. One topic that is not discussed is the use of more general differential equations to describe autonomous processes; they would be necessary for many practical applications.

References

Davis, E. 1990. *Representations of Commonsense Knowledge*. San Francisco, Calif.: Morgan Kaufmann.

Haas, A. R. 1987. The Case for Domain-Specific Frame Axioms. In *The Frame Problem in Artificial Intelligence,* ed. F. R. Brown, 343–348. San Francisco, Calif.: Morgan Kaufmann.

Hintikka, J. 1961. Modalities and Quantification. *Theoria* 27(61): 119–128.

Kripke, S. 1963. A Semantical Analysis of Modal Logic I: Normal Modal Propositional Logic. *Zeitschrift für Mathematische Logik under Grundlagen der Mathematik* 9:67–97.

McCarthy, J., and Hayes, P. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence* 4, eds. B. Meltzer and D. Michie, 463–502. Edinburgh, U.K.: Edinburgh University Press.

Moore, R. C. 1980. Reasoning about Knowledge and Action. Technical Report 191, SRI AI Center, Menlo Park, Calif.

Pylyshyn, Z. 1987. *The Robot's Dilemma: The Frame Problem and Other Problems of Holism in Artificial Intelligence*. Greenwich, Conn.: Ablex.

Reiter, R. 1991. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy,* ed. V. Lifschitz, 359–380. San Diego, Calif.: Academic.

Schubert, L. K. 1990. Monotonic Solution of the Frame Problem in the Situation Calculus: An Efficient Method for Worlds with Fully Specified Actions. In *Knowledge Representation and Defeasible Reasoning,* eds. H. E. Kyburg, R. P. Loui, and G. N. Carlson, 86–95. New York: Kluwer Academic.

Drew McDermott is a professor of computer science at Yale University. He was educated at the Massachusetts Institute of Technology, where he received a Ph.D. in 1976. He was chair of the Yale Computer Science Department from 1991 to 1996. He is coauthor of two texbooks in AI, serves on the editorial board of *Artificial Intelligence,* and is a fellow of the American Association for Artificial Intelligence. McDermott's research is in robot navigation, planning, and interagent communications.