# Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems



Raymond Reiter

The MIT Press
Cambridge, Massachusetts
London, England

# Preface

The situation calculus has been with us since 1963, when John McCarthy first introduced it as a way of logically specifying dynamical systems in artificial intelligence, but for most of that time, it was not taken very seriously as a realistic formalism. True, it was the language of choice for investigating technical issues, like the frame problem, that arise in axiomatizing dynamics, but most AI researchers viewed it as just that—a theoretical tool without much practical importance. About nine years ago, Hector Levesque and I came to realize that the situation calculus had a lot more potential than was commonly believed, and we and our collaborators set out to demonstrate this, mainly by extending the language to incorporate features like time, concurrency, procedures, probability, etc., while taking care to do so in ways that provide for efficient implementations. This book is largely the result of that activity.

Insofar as its subject matter concerns the modeling of dynamical systems, this book crosses traditional academic boundaries. Its intended audience consists of graduate students and researchers in AI, databases, robotics, software agents, simulation, decision and control theory, computer animation, and, indeed, in any discipline whose central concern is with specifying and implementing systems that evolve over time. The academic world doesn't lack for books about dynamical systems, so what distinguishes this one? The simple answer is that its theoretical and implementation foundations rest on mathematical logic. In a nutshell, the central idea of the book is this: When faced with a dynamical system that you want to simulate, control, analyze, or otherwise investigate, first axiomatize it in a suitable logic. Through logical entailment, all else will follow, including system control, simulation, and analysis. Such a claim is by no means obvious, and to a large extent, this book is an exploration of this idea—in our case, using the situation calculus as the underlying logic.

This book is as much about implementing dynamical systems as it is about their theoretical and representational foundations. Therefore, it provides a large number of examples and, perhaps unusually for books of its kind, it includes all the code for these examples. This turned out to be feasible because the implementation language, Prolog, is so elegant and close to logic that once one gets the logical specification right, compilation into extremely compact Prolog code is in most cases absolutely trivial. How to perform this compilation—and the justification for it—is the subject of Chapter **??**, and I believe that learning how to do this is one of the most important lessons of the book. This methodological theme pervades the book and can be captured by the slogan:

| No implementation without a sitcalc specification |
|---|

To keep faith with this slogan I have been careful, throughout the book, to accompany all code with its logical specification in the situation calculus, even if, on occasion, this may seem a bit on the pedantic side. The payoffs are many: A logical specification states clearly and unambiguously what the modeling assumptions are, it helps enormously in coding and debugging an implementation, and it allows you to prove properties of the system model.

I have used this material on several occasions as the basis of a graduate course at the University of Toronto. Students were drawn from virtually all branches of computer science, but also included control theorists, electrical and computer engineers, and the occasional mathematician. The course ran for 13 weeks, two hours per week, which was enough time to comfortably cover most, but not all of the book. I think that Chapters 1–8, 11, and 12 contain the essential ideas and should form the core of any such course. Time permitting, Chapter 10 on planning can be a lot of fun. Chapter 9 on progression is perhaps the least important to include in a course, even though it tells rather a nice story about how STRIPS and the situation calculus are related. I assigned exercises from each chapter covered, trying for a balance between theoretical and implementation-based questions. The final course component was a project that gave expression to the students' individual interests and backgrounds. Frequently, these projects were inventive and ambitious; often, they led to graduate theses. Project topics ranged all over the map, including databases, computer animation, simulation of physical systems, program verification, theoretical foundations, and high-level robotics. My experience in teaching this course has been that once they learn how to use the situation calculus, students quickly realize how pervasive dynamics is in their own areas of interest, and relevant project topics immediately present themselves.