UNIVERSITY OF CRETE - HERAKLION GREECE

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF SCIENCES AND ENGINEERING

A Formal Theory for Reasoning About Action, Knowledge and Time

by

Theodore Patkos

In partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

September 2010

UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE

A Formal Theory for Reasoning About Action, Knowledge and Time

Dissertation submitted by
Theodore Patkos
in partial fulfillment of the requirements for
the PhD degree in Computer Science

Author:

Theodore Patkos, University of Crete

Examination Committee:

Dimitris Plexousakis, Professor, University of Crete

Grigoris Antoniou, Professor, University of Crete

Ioannis Tsamardinos, Assistant Professor, University of Crete

George Vouros, Professor, University of Aegean

Antonis Argyros, Associate Professor, University of Crete

Nick Bassiliades, Associate Professor, Aristotle University of Thessaloniki

Manolis Koubarakis, Associate Professor, National and Kapodistrian University of Athens

Approved by:

Panagiotis Trahanias
Chairman of Graduate Studies

Heraklion, September 2010

A Formal Theory for Reasoning About Action, Knowledge and Time

Abstract: Aiming at achieving a proper regulation of their behavior in real-world environments, participating agents need to reason not only about the specifications of the environment they inhabit, but also about their own knowledge concerning its current state by exploiting information acquired at run-time. Considering the highly dynamic nature of most complex domains, the study of knowledge evolution over time is a critical aspect. In this thesis, we develop a *unified formal theory of action, knowledge and time* using the language of the Event Calculus and automate the process of reasoning about a wide range of commonsense phenomena.

Traditionally, epistemic reasoning has been structured around the highly expressive but computationally expensive "possible worlds" specifications. Recent theories adopt alternative representations for knowledge, dismissing the accessibility relation of possible worlds and promising more efficient reasoning in classes of restricted expressiveness. The framework we propose combines the full expressive power of the possible worlds semantics with the benefits of alternative approaches, building on a proper handling of a type of causal dependencies that emerge among partially known world aspects. We investigate the properties of these so called *hidden causal dependencies* and develop a provably sound and complete axiomatization that is independent of the underlying formalism. We show correctness properties by studying the correlation of the theory with an epistemic formalism that implements the standard definition for knowledge, based on a recently proposed branching version of the Event Calculus.

Furthermore, we investigate a number of different extensions of the basic axiomatization augmenting the mental state of intelligent agents with essential cognitive skills, such as the ability to remember and forget, to reason about physical actions, to handle complex ramifications in partially observable domains, and others. We demonstrate the potential of the theory by modeling complex benchmark problems proposed in relevant literature, as well as scenarios that emerge in the highly demanding nascent field of Ambient Intelligence. Finally, we also describe the design of a reasoner that can accommodate both epistemic and online reasoning and present a way to implement the framework using logic programming languages.

Keywords: Reasoning about Change and Causality, Epistemic Reasoning, Event Calculus, Cognitive Robotics, Ambient Intelligence

Supervisor:
Dimitris Plexousakis
Professor of Computer Science
University of Crete

Μια Τυπική Θεωρία Συλλογιστικής με βάση τη Δράση, τη Γνώση και το Χρόνο

Περίληψη: Με στόχο την επίτευξη μιας ορθά ορισμένης συμπεριφοράς σε ρεαλιστικά περιβάλλοντα, οι μετέχοντες πράκτορες απαιτείται να εκτελούν διεργασίες συλλογιστικής όχι μόνο λαμβάνοντας υπόψη τις προδιαγραφές του περιβάλλοντος στο οποίο επιδρούν, αλλά και με βάση τη γνώση που διαθέτουν για την τρέχουσα κατάσταση, αξιοποιώντας πληροφορία που αποκτάται σε χρόνο εκτέλεσης. Συνυπολογίζοντας την ιδιαίτερα δυναμική φύση των περισσότερων σύνθετων πεδίων, η μελέτη της κλιμάκωσης της γνώσης σε σχέση με το χρόνο συνιστά μια εξίσου κρίσιμη παράμετρο. Στα πλαίσια της παρούσας διατριβής αναπτύσσουμε μια *ενοποιημένη τυπική θεωρία δράσης, γνώσης και χρόνου* με χρήση του Λογισμού Συμβάντων και αυτοματοποιούμε τη διαδικασία συλλογιστικής για ένα ευρύ φάσμα φαινομένων κοινής λογικής.

Παραδοσιακά, η συλλογιστική γνώσης προδιαγράφεται γύρω από την ιδιαίτερα εκφραστική, αλλά δαπανηρή υπολογιστικά μέθοδο των πιθανών κόσμων. Η τρέχουσα έρευνα υιοθετεί εναλλακτικές αναπαραστάσεις για τη γνώση, απαλείφοντας τη σχέση προσβασιμότητας των πιθανών κόσμων και παρέχοντας εγγυήσεις αποδοτικής εξαγωγής συμπερασμάτων σε κλάσεις περιορισμένης εκφραστικότητας. Το πλαίσιο εργασίας που προτείνουμε συνδυάζει την πλήρη εκφραστική δύναμη της σημασιολογίας των πιθανών κόσμων με τα οφέλη των εναλλακτικών προσεγγίσεων, βασιζόμενο στον αποδοτικό χειρισμό μιας κατηγορίας αιτιατών εξαρτήσεων που ανακύπτουν μεταξύ παραμέτρων κόσμου που είναι μερικώς αντιληπτές. Εξετάζουμε τις ιδιότητες αυτών των επονομαζόμενων *κρυφών αιτιακών εξαρτήσεων* και αναπτύσσουμε μια ορθή και πλήρη αξιωματοποίηση, ανεξάρτητη του υποκείμενου φορμαλισμού μοντελοποίησης. Η απόδειξη των ιδιοτήτων ορθότητας επιτυγχάνεται μέσω της συσχέτισης με ένα νέο φορμαλισμό που υλοποιεί τον καθιερωμένο ορισμό γνώσης και βασίζεται στην προσφάτως προταθείσα έκδοση ενός διακλαδιζόμενου Λογισμού Συμβάντων.

Επιπρόσθετα, μελετάμε ένα πλήθος επεκτάσεων της βασικής αξιωματοποίησης που επαυξάνουν τη νοητική ικανότητα ευφυών πρακτόρων με σημαντικές γνωστικές δεξιότητες, όπως την ικανότητα διατήρησης ή απώλειας γνώσης, τη δυνατότητα συλλογιστικής ως προς φυσικές ενέργειες, το χειρισμό σύνθετων παρενεργειών σε μερικώς παρατηρήσιμα περιβάλλοντα

κ.α. Παρουσιάζουμε τη δυναμική της θεωρίας μέσω μοντελοποίησης προβλημάτων αναφοράς σε συναφή πεδία, καθώς επίσης μέσω σεναρίων που ανακύπτουν στην ιδιαίτερα απαιτητική περιοχή της Διάχυτης Νοημοσύνης. Τέλος, παρουσιάζουμε το σχεδιασμό ενός εργαλείου συλλογιστικής που μπορεί να υποστηρίξει απόκτηση γνώσης σε πραγματικό χρόνο και περιγράφουμε τον τρόπο που μπορεί να υλοποιηθεί το ευρύτερο πλαίσιο εργασίας με χρήση γλωσσών λογικού προγραμματισμού.

Λέξεις-κλειδιά: Συλλογιστική Αλλαγής και Αιτιότητας, Επιστημική Συλλογιστική, Λογισμός Συμβάντων, Γνωσιακή Ρομποτική, Διάχυτη Νοημοσύνη

Επόπτης Καθηγητής:

Δημήτρης Πλεξουσάκης
Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Dedication

To my parents Lazaros and Eygenia

Στους γονείς μου Λάζαρο και Ευγενία

Acknowledgements

Throughout this endeavor towards defending a thesis, a Ph.D. candidate has to face issues that are not exclusively related to research. While some important mental challenges are of academic nature, others that emerge in everyday life may equally have impact on the final objective. What seems to be a lonely process often requires the collective and harmonious participation of people that contribute to their own "area of expertise". These people that have affected my life in Crete I wish to thank next.

Probably the only person who played a decisive role to almost all aspects that determined my progress over the last years is my supervisor Professor Dimitris Plexousakis. His academic advices, his reasoning and directions always created new enthusiasm for investigating the problems that we have set and stimulated extra motivation for harder work. But in addition, on certain difficult occasions I have also been taught from his paradigm the essence of being supportive, discreet and to show trust in oneself. I own a huge debt of gratitude to Professor Plexousakis for giving me the opportunity to work with him.

I would also wish to thank Professor Grigoris Antoniou not only for the time he has devoted in reviewing my dissertation, but also for entrusting me with many research activities from which I have gained valuable experiences. Our inspiring meetings and his excellent theoretical knowledge have let me profit a lot.

I extent my deep appreciation also to Assistant Professor Ioannis Tsamardinos, the third member of my advisory committee, for our collaboration from the beginning, his guidance and also his friendly will to discuss a variety of topics.

Furthermore, I feel fortunate to have collaborated and discussed relevant matters with Associate Professors Nick Bassiliades and Antonis Argyros. Their way of seeing science and approaching problems have influenced my view of Artificial Intelligence. I am also indebted to the other members of my examination committee, Professor George Vouros and Associate Professor Manolis Koubarakis, for their care in my work and their constructive comments.

This work was supported by a graduate fellowship from the Institute of Computer Science (ICS) of the Foundation for Research and Technology Hellas (FORTH), as well as by the Maria Manasaki legacy's fellowship of the University of Crete, which I want to acknowledge for providing financial assistance and technical equipment. In addition, I wish to acknowledge the help and support from the secretariat and the administrators of the department of Computer Science and ICS-FORTH.

I would also like to thank all my friends and colleagues at FORTH and the University of Crete with which I have shared hours of work and joy. Special reference should be made to Antonis Bikakis, a talented researcher, an exceptional character and a dear friend. From the most stringent research problem to the most philosophical consideration and from any kind of sport to any kind of -unhealthy- Greek habit Antonis has always been best company. Big thanks should also go to George Flouris and Alex Artikis, two of the most dedicated and intelligent young researchers I have met. They were always willing to consider my questions and suggest ideas.

Many other friends in Crete and Thessaloniki have offered big help and this dissertation would not have been completed without their generous assistance in all kinds of matters. I own much to Kassy with her unique personality and the special way of judging matters. I also thank Adamantia for the memorable times and her endless support. Antonis, Kassy and Adamantia have stood by me on good and difficult occasions and often tolerated me when things got rough. Also thanks to Thomas Skylogiannis, Adam Arvelakis, Nick Xanthopoulos, Anastasis Oulas, Amalia Foka, "Tzortzis" Konstantinides, "Panaes" Papadakos, Kwstas Varsos and Manos Kalaitzakis, as well as to Stelios Bagios, Thomas Kappas and Eirini Nestori for all the times we spent hanging out.

Finally, my family deserves my deepest gratitude and appreciation. My parents, Lazaros and Eygenia, along with my sister Annie and her husband George, have offered tremendous mental and physical support and materialized their sincere love in many ways over all these years. Their persistence in showing commitment, integrity and devotion under all circumstances has set a perfect example for my future life. This dissertation is dedicated to them.

Let us dare to face the situation. Man has become superman. He is a superman because he not only has at his disposal innate physical forces, but also commands, thanks to scientific and technological advances, the latent forces of nature which he can now put to his own use...

...However, the superman suffers from a fatal flaw. He has failed to rise to the level of superhuman reason which should match that of his superhuman strength. He requires such reason to put this vast power to solely reasonable and useful ends and not to destructive and murderous ones...

...the essential fact which we should acknowledge in our conscience, and which we should have acknowledged a long time ago, is that we are becoming inhuman to the extent that we become supermen.

Albert Schweitzer - Nobel Peace Prize Lecture, 1952

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Contents

In his 1890 classic "The sign of the four" Sir Arthur Conan Doyle opens with another characteristic demonstration of Sherlock Holmes' keen deduction skills that always left Watson amazed - and a bit irritated by the sense of naturalness that Holmes usually assumed: some coin or key marks on an expensive watch seem enough to reveal a careless and untidy owner who kept hard objects in the same pocket; the hundreds of scratches around the key-hole of the inner plate indicate that the key usually slipped, a customary for drunkards; worse, the four pin-pointed scratched numbers upon the inside of the watch case, obviously made by pawnbrokers, leave few doubts about the owner's life habits; finally, the initials, suggesting Watson's long deceased father and dated as old as the watch, disclose Watson's eldest brother as its owner, as jewelry usually descent to the eldest son!

Even in this short passage, Sherlock justifies why he appreciated highly three virtues for ingenious commonsense inferencing, the powers of *observation*, *deduction* and *knowledge*. It seems that Doyle was echoing the sentiments of future AI researchers who, faced with the task of creating intelligent autonomous agents for solving complex problems, struggle to endow their creations with such cognitive skills. Agents operating in complex dynamic worlds often need to achieve control over partially observable and uncertain environments and to reason about the knowledge at hand - much like an investigator. Managing information acquired through sensing has proven to be a substantial skill. This thesis proposes

a formal framework for commonsense reasoning in dynamic and partially observable domains based on knowledge about the -direct and indirect- effects of actions and of physically triggered events, as well as on world observations obtained at run-time.

1.1 Motivation

The formal account of knowledge has been a fascinating active field of research in philosophy, computer science and in other disciplines, as well. The common objective has been the investigation of various manifestations of epistemic concepts, such as knowledge and belief, and the ability to reason by exploiting their inherent properties. Within AI, a multitude of logic formalisms have extended epistemic modal logics for the single and multi agent case, suggesting axiom systems to provide solutions for well-known issues, such as the problem of logical omniscience or the matter of belief change.

The study of these issues has resulted in significant accomplishments, but also in the emergence of novel and more demanding challenges that have enormous impact on the directions that AI research is adopting. One such field, which has concentrated the focus of attention recently, studies the relationship between knowledge and action. While formalisms for reasoning about knowledge tell agents how to adjust their beliefs given observations about changes of an environment, they do not deal with issues, such as reasoning about the properties of the actual actions that cause the changes, which introduce high-level agent programming and planning requirements. In dynamic systems, where one or more agents manage different skills and coordinate their actions to achieve a certain state of affairs, action theories allow for reasoning about the state and changes of the environment, the notion of causality, about potential effects of actions, action preconditions, qualifications and many others. Incorporating a formal account of knowledge in action theories is essential for modeling subtle real-world applications, where agents need to acquire knowledge and decide upon their actions at execution time.

Within this context research in the field of reasoning about action and knowledge has made quantum leaps in the last decade. Powerful frameworks with very expressive formal accounts for knowledge and change and elegant mathematical specifications have been developed, motivated primarily by the adaptation of the standard possible worlds semantics in formal action theories. Still, their application to practical implementations raises legitimate concerns, due to their dependence on a computationally highly demanding structure:

according to the possible worlds specifications, in a domain of n atomic formulae determining whether a formula is known potentially requires 2^n distinguishable worlds to check truth in. Aiming at efficiency, contemporary progress explores alternative characterizations of knowledge. An increasing number of recent approaches either focuses on classes of restricted expressiveness in order to perform tractable reasoning or sacrifices logical completeness with respect to the standard possible worlds semantics.

However, the restrictions imposed on these alternative approaches fail to cover certain important aspects of commonsense reasoning. Unknown preconditions of context-dependent effects of actions introduce a type of uncertainty that is difficult to handle efficiently with alternative knowledge representations. Upon action execution knowledge about the effect is lost if the preconditions are unknown, still it becomes contingent on them; obtaining knowledge about the context through sensing may provide information about whether the effect was actually affected. This dependency, which we name *hidden causal dependency* (HCD), is inherently modeled in possible worlds by appropriately decreasing the number of accessible worlds, without necessarily causing any explicit change to the agent's epistemic state. Alternative approaches concentrating primarily on explicit knowledge effects and complete context knowledge seem less susceptible to such causal dependencies. The situation becomes even harder if physically occurring events are considered that are only triggered when the world is at a particular state. In this case the agent should also be in position to reason about the potential triggering of such events if its partial knowledge about the world neither precludes nor justifies the event occurrence.

Furthermore, previous work in literature has mainly concentrated on reasoning about knowledge when sense actions occur, but did not thoroughly elaborated on the temporal aspect of knowledge. In real-world systems, though, temporal constraints are usually ubiquitous and for the agent to perform effectively an explicit representation of time and the ability to deal with information about the actual course of events are crucial [Chittaro 2000]. The very nature of most environments involves dynamically changing context, information becoming outdated as time passes and events occurring at specified time instances, more often than not concurrently with other events. In order to ensure that the behavior of rational agents is properly regulated and also to prove the ability to achieve assigned objectives in a dynamic, real-world environment, a formal account of knowledge, action and time is essential.

1.2 Thesis Contribution and Technical Results

In this thesis we propose a knowledge theory[1], based on a highly influential action formalism, the Event Calculus, that does not manipulate possible worlds. The theory, named Discrete Event Calculus Knowledge Theory (DECKT), is a unified framework of action, knowledge and time devised for reasoning non-monotonically about conditional and indirect knowledge effects, knowledge-producing (sense) actions, ramifications of knowledge, as well as loss of knowledge caused by non-deterministic events. Moreover, exploiting the linear time structure provided by the Event Calculus, the framework builds on the interaction of knowledge and time, in order to express temporal knowledge and delayed knowledge effects, cumulative and canceling knowledge effects of concurrent events or knowledge about continuously changing world aspects. Derivations involving disjunctive or existentially quantified formulae are still supported as in possible worlds-based theories, but when reasonably restricted the proposed theory can also perform efficient reasoning. The result is a theoretical framework with well defined correctness properties and also a theory that can be applied to practical implementations to enhance the mental state of intelligent agents.

Furthermore, we extend the basic axiomatization in several ways. An account for physically-triggered events is provided, for the case of agents that are uncertain whether the events actually occur or not, still having to reason about their potential effects. Moreover, sensing is extended to consider not only inertial aspects as in most related studies, but also aspects that may change their value in a continuous fashion. An investigation of the notion of ability within such domains is also provided. The resultant framework of action and knowledge is applicable to far more expressive domains as compared to current alternative approaches, still being suitable for practical implementations. To demonstrate its potential, we apply the theory to one of the most challenging ramification benchmark problems proposed in literature, as well as to the emerging field of Ambient Intelligence that introduces novel and highly demanding requirements. Finally, we develop a tool and a methodology to enable the automation of the process of reasoning with the epistemic notions introduced in the framework.

From a more technical standpoint, this thesis contributes the following results with re-

[1]The difference to a theory that handles belief is that knowledge is always correct, while beliefs may not reflect the actual state of the world and an agent should possess a mechanism to revise beliefs that turn out to be false. This issue is further exemplified in Sections 2.1.5 and 2.2.1.

spect to the line of research in the broader field of knowledge representation and reasoning:

1. We develop a unified framework for automated temporal, epistemic and causal reasoning in very expressive classes of dynamic and uncertain domains enabling the representation of a multitude of commonsense phenomena. To this end we exploit the reasoning potential of the Event Calculus and extend its basic ontology with epistemic features. The formalism is suitable for practical implementations as it relies on an efficient alternative treatment of knowledge as opposed to the standard possible worlds specifications.

2. We establish a translation of possible worlds into a generic form of implication rules, i.e., HCDs, so that all information obtained by adding or removing worlds (i.e., set of world properties) can equally be acquired by reasoning with HCDs. We provide evidence that HCDs are more appropriate for practical implementations for reasoning in partially observable domains in terms of computational complexity and resources required. Revisiting initial attempts to formulate dependencies among world aspects without possible worlds (e.g., [Thielscher 2005b]), we investigate the intuition behind the evolution of knowledge with possible worlds over time and end up with an axiomatization that is independent of the underlying logic language.

3. We provide formal evidence that the proposed framework derives sound and complete conclusions with respect to the standard definition for knowledge that implements the possible worlds semantics. To obtain this result, we create an epistemic version of the Event Calculus that adopts the latter approach, building on the recently developed branching version of the calculus by Mueller [Mueller 2007b], and investigate the correlation between the two epistemic theories. In addition, we also axiomatize extensions that go beyond the expressiveness of the possible worlds, such as the ability to forget.

4. Setting off from existing approaches to model state knowledge in action theories, we investigate advanced aspects of commonsense reasoning, such as complex knowledge ramifications and indeterminate occurrences and effects of events. To demonstrate the potential of the theory we study problems of both theoretical and practical value. We model a well known benchmark problem of the field of cognitive robotics, namely Shanahan's circuit [Shanahan 1999a], extending it with a treatment of partial knowledge and progressing the current state of research. Furthermore, we

apply the theory to scenarios that emerge in the nascent and highly demanding field of Ambient Intelligence that introduce real-world challenges.

5. Finally, an elaboration on how HCDs and the axiomatization in general can be implemented is provided, discussing also a modeling that is generic enough to be translated to any prolog- or lisp-like syntax. An online reasoner for modeling the mental state of autonomous agents has been designed adopting this approach that supports both epistemic and online reasoning tasks, two highly desirable features for our domains of interest.

1.2.1 Application Domain

The present study has been primarily motivated by the challenges raised by the emerging research field of Ambient Intelligence. The Ambient Intelligence paradigm, formulated by a number of projects and initiatives that were announced at the early 2000s, proposes a shift in computing towards a multiplicity of communicating devices disappearing into the background, providing an intelligent, augmented environment, where the emphasis is on the human factor. For decades, humans had to continuously adapt themselves to their surrounding technology, in order to make the best out of it. Ambient Intelligence technologies have the potential to create intelligent environments with the ability to proactively adapt to humans, serve their needs and goals, and communicate with them utilizing novel means. This paradigm implies a seamless medium of interaction, advanced networking technology and efficient knowledge management, in order to deploy an environment that is aware of the characteristics of human presence and the diversities of personalities, and also capable of responding intelligently to spoken or gestured indications of need or desire.

The Ambient Intelligence paradigm has generated an enabling multidisciplinary research field that envisages to bring intelligence to everyday environments and facilitate human interaction with devices and the surrounding. Artificial Intelligence has a decisive role to play for the realization of this vision promising commonsense reasoning and better decision making in dynamic and highly complex conditions, as advocated by recent studies [Ramos 2008]. Within Ambient Intelligence environments human users do not experience passively the functionalities of smart spaces, instead they participate actively in it by performing actions that change its state in different ways. At the same time, the smart space itself and its devices are expected to perform actions and generate plans either in response

to changes in the context or to predict user desires and adapt to user needs.

A key concept for such environments is the notion of context. Context describes any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and application, including the user and applications themselves, while a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task [Abowd 1999]. Therefore, contextual information may refer to a multitude of different dimensions, which can be classified as external and internal [Prekop 2003], or else physical and logical [Hofer 2003], depending on whether hardware sensors or high-level inference is used to specify them. The former (external or physical) may involve location, light, sound, movement, touch, temperature, pressure, identity, time, device and many others, while the latter (internal or logical) may refer to a user's task/activity, preferences, intentions, emotional state, social environment and other.

It is reasonable to expect that context in large-scale Ambient Intelligence systems may often be ambiguous or completely unknown, especially considering the highly dynamic nature of such domains. Therefore, reasoning needs to incorporate both epistemic and temporal dimensions; whenever sensing is not available, knowledge needs to be derived through commonsense inference, considering the most recent information about the world and the effects of known actions. The causal description of a domain in particular is essential for enabling planning tasks in order to achieve a desirable state of affairs, especially when the required information is not available at plan time, but can only be obtained at run-time. A formal treatment of these notions is imperative, in order to ensure that the behavior of such systems is properly regulated and prove their properties.

1.3 Thesis Outline

This thesis is organized as follows. In Chapter 2 a review of the three main fields of research is provided, namely action theories, epistemic modal logics and Ambient Intelligence, whose convergence constitutes the domain of interest of this work. At the same time, we also attempt an outline of the wider context within which this study contributes. In Chapter 3 we focus on relevant literature and discuss the advantages and limitations of action theories that implement epistemic reasoning. In Chapter 4 we lay the theoretical foundations for the Discrete Event Calculus Knowledge Theory. We first introduce the

preliminaries of the theory and then present the formal axiomatization along with numerous explanatory examples. Chapter 5 provides an analysis of the properties of the theory in terms of soundness, completeness and complexity matters. Towards this end, a possible worlds-based Event Calculus theory is also developed at the beginning of the chapter. In Chapter 6 we investigate significant extensions of the framework, in order to accommodate more expressive domains. In particular, we consider sense actions about aspects of the world that may change in a continuous fashion, we integrate a treatment about triggered actions, which, in the face of incomplete knowledge, require potential actions to be introduced and, finally, we formally define the ability of an agent to achieve its objectives in a partially observable environment. In Chapter 7 we examine different use case scenarios for complex domains. We further discuss the means to implement the knowledge theory both in terms of existing reasoning tools for the Event Calculus, as well as with the help of a recently proposed reasoner developed during the course of this thesis. Finally, Chapter 8 summarizes the main points of the thesis and discusses future research directions.

The main conclusions and limitations identified from surveying the various formalisms presented in Chapters 2 and 3 for practical application appear in [Patkos 2007a] and [Patkos 2007b] and motivated the present research. A preliminary version of the basic axiomatization of the theory as given in Chapter 4 has been published in [Patkos 2008], whereas the axiomatization of Chapter 4 in conjunction with the main conclusions of Chapter 5, which constitute the core part of the thesis have been published in [Patkos 2009a]. The extension of knowledge-producing actions for non-inertial world aspects presented in Chapter 6 appear in [Patkos 2009b], while the extension of trigger actions along with the treatment of HCDs of Chapter 4 and the representation of Shanahan's circuit given in Chapter 7 are currently under consideration for publication in [Patkos 2010b]. Finally, the basic formulation of the use cases for the field of Ambient Intelligence presented in Chapter 7 appear in [Patkos 2010a] and [Patkos 2011].

CHAPTER 2

Background Material and Literature Review

Contents

The present chapter provides background material necessary to follow the main concepts and theories of this thesis, reflecting also the wider context within which the study contributes. It comprises three main topics, the intersection of which constitute the domain of interest: action theories and their extensions, epistemic modal logics and Ambient Intelligence. For each of them we introduce the issues and problems that need to be tackled, the general directions of research and also discuss prominent approaches along with their limitations.

2.1 Action Theories for Complex Environments

To study the dynamics of changing worlds, research in AI has established techniques of different form, ranging from purely space searching and procedural to probabilistic, to deductive. Within a logical knowledge representation dynamic domains are formalized around the notions of action and causality, based on which AI attempts to automate the process of reasoning about commonsense knowledge, i.e., knowledge of how the world works. Reasoning about action and change is a fundamental field of research within AI, placed -but not restricted- in the area of cognitive robotics, that studies the logical characterization of the concepts of action, change and causality, as well as the planning of action sequences to accomplish a given task. Cognitive robotics is a reply to the criticism that knowledge representation and reasoning has been overly concerned with reasoning in abstract and not concerned enough with the dynamic world of an embodied agent [van Harmelen 2007].

Action theories have made significant progress beyond the *classical planning paradigm* [Ghallab 2004], relaxing many of its simplifying assumptions and defining theoretical foundations to model more complex domains. This section elaborates on knowledge representation and reasoning techniques from the action-based point of view, providing a review of several formalisms that have been developed, driven mainly by theorem proving techniques. The objective is to present the distance that has been traveled by the different approaches in addressing the general problems that AI faces in implementing commonsense reasoning theories for dynamic domains with increasing complexity. Alongside the survey, we also provide justification for our choice to use the Event Calculus as the underlying formalism for the knowledge theory we develop in this thesis.

2.1.1 Introducing the Field: Fundamental Problems

The design of declarative approaches to commonsense reasoning and planning is a relevant field of research in AI since its beginnings. The general idea is to create intelligent autonomous agents that are able to represent all kinds of knowledge about the world and use this knowledge to infer commonsense conclusions about a wide range of phenomena, such as preconditions and effects of actions, concurrent actions, natural events, nondeterminism and others. As it turns out, there are significant difficulties that need to be overcome in order to formalize a domain with some logical language, before even considering axiomatizing complex phenomena. A simple description of the effects of an action with a formal

language requires not only the representation of those world aspects that are affected, but also those that do not change their truth value. The straightforward solution of declaring explicitly all effects and non-effects of actions does not scale well with the size of the domain, therefore any action theory needs to incorporate some decent solution. This issue is commonly known as the *frame problem* introduced by McCarthy and Hayes [McCarthy 1987]. To respond to the explosive number of axioms introduced by theorem proving-based planners in the straightforward approach, several alternatives have been proposed along the way, some of which will be discussed in the following subsections.

Axiomatizing the effects of actions is hardly restricted to a solution to the frame problem. In the context of reasoning about action, one needs not only to concentrate on the effects explicitly described by their associated effect axioms, but also infer indirect effects, derived by some general knowledge of dependencies among world aspects. Capturing efficiently the potentially unbounded number of consequential effects of actions defines the essence for a solution to the *ramification problem*. Furthermore, one must specify the conditions under which the effects will take place after executing an action, without exhaustively enumerating all potential preconditions. This is known as the *qualification problem*, stated by McCarthy [McCarthy 1977]. In particular, an agent needs not consider unexpected qualifications for an action, unless there is evidence that they may arise.

In their well known study Lin and Reiter [Lin 1994] argue that ramification and qualification constraints are closely related, not only syntactically, as they can both be expressed in terms of state constraints, but also by the fact that the ones may cause the others to arise and vice versa. Incorporating a uniform solution for all three problems is a challenging task, in order to implement action frameworks for commonsense reasoning. Indeed, many solutions require precise assumptions; for instance, while many existing approaches to the frame problem are monotonic, the qualification problem inherently requires a non-monotonic solution that enables the agent to retract its initial expectations when things do not proceed as expected. Additionally, complex ramifications are hardly limited to state constraints, but may refer for example to triggers and delayed effects. A recent survey of these issues can be found in the introductory part of [Vo 2005], while a brief discussion on the development of solutions to the ramification problem is also provided in Section 7.1.1 of this thesis.

2.1.2 Review of Formalisms for Reasoning about Action and Change

Axiomatizations of domains using action theories can serve several types of reasoning tasks, such as predicting the outcome of given action sequences, explaining observations or finding a situation in which certain goal conditions are met. Depending on the desirable outcome, the inputs may involve a description of the world, a description of the agent's goal and/or a description of the possible actions that can be performed, in some formal language. Quoting Thielscher's informal definition, *an action theory consists of a formal language that allows adequate specifications of action domains and scenarios, and it tells us precisely what conclusions can be drawn from these specifications* [Thielscher 2000b]. As the objective is to express the dynamics of the world, most theories include a more or less implicit general notions of time, change and causality. Below, we review some of the most important formalisms, along with their extensions on handling important common-sense phenomena (Table 2.1 at the end of this section summarizes the results). The theories are usually variations of the predicate calculus; yet, the propositional STRIPS language can be considered as a predecessor in the field.

STRIPS. STRIPS [Fikes 1971] is probably an action representation, rather than an action description language. The STRIPS representation describes the initial state of world with a complete set of ground propositions. It provides a set of operators, their associated operator descriptions and a data structure called a database that acts as a snapshot of the world. Operators serve as representations of actions and they map databases into databases. For a given STRIPS operator, this mapping is intended to capture the effects of the action represented by the operator; when applied to a given world state (database), it produces a new world state that is intended to represent the way the world will be after the action corresponding to the operator is performed. It is restricted to *goals of attainment* that are defined as propositional conjunctions. All world states satisfying the goal formula are considered equally good. A domain theory, i.e., a formal description of the actions that are available to the agent, completes a planning problem. The description is a function from action names to operators that define a propositional conjunction of fluent names (preconditions) and a consistent conjunction of literals (effects). As a result, each action is specified by so called add- and delete-lists containing fluents that express the preconditions and effects, accordingly.

Although STRIPS enjoys the simplicity of its semantics, it is very restrictive in terms of expressiveness, severely limiting the type of actions that an agent wishes to represent.

The propositional STRIPS planning language has been formalized and the complexity of the problems that can be specified in this language has been analyzed by Lifschitz in [Lifschitz 1986]. Note, however, that STRIPS constitutes the core of PDDL (Planning Domain Description Language), the standardized syntax used in the AIPS planning competitions, and many current planners still employ the STRIPS formulation of planning problems and variations of it, such as the action description language ADL [Pednault 1989].

Situation Calculus. The Situation Calculus, first proposed in [McCarthy 1968] and formalized in [Levesque 1998] and [Reiter 2001a], is a first-order language with some second-order features, designed for the representation of dynamically changing worlds, in which all changes are the result of named *actions* performed by agents. A possible world history, which is simply a sequence of actions, is represented by a first-order term called a *situation*. A distinguished binary function $do(\alpha, s)$ denotes the successor situation to s resulting from performing the action a. Relations and functions whose truth values vary from situation to situation are called *relational* and *functional fluents*, respectively. Actions, situations and fluents are the main ingredients of the Situation Calculus formalism that provide a complete treatment of reasoning about action.

Each action $\alpha(\vec{x})$ is described by two axioms. The *Action Precondition Axiom*, which states under what conditions is the action executable and the *Successor State Axiom*, which describes how fluents change when the action is performed. These axioms characterize all the conditions under which action a causes fluent f to become true (respectively, false) in the successor situation and completely describe the causal laws for f. The Successor State and the Action Precondition Axioms, in conjunction with unique name axioms for actions compose the proposed solution for handling the representational frame problem in Situation Calculus applied in the general case (i.e., where actions are deterministic and primitive), as originally shown in [Reiter 1991]. This axiomatization requires $(|f| + |a|)$ axioms in total, compared with the roughly $(2 \times |a| \times |f|)$ axioms that would otherwise be required to explicitly state each frame axiom, where $|f|$ is the number of fluents and $|a|$ the number of actions. Finally, a complete Situation Calculus theory D also includes a set of domain independent foundational axioms that define situations, as well as axioms describing what is true in the initial situation.

To utilize action theories in practice, a family of high-level programming languages for intelligent agents has been developed, where each language implements a corresponding formalism. The intuition is to abstract the process of planning, where, instead of per-

forming a search among all possible action sequences that might achieve a goal state, a high-level domain-dependent program can be applied to guide the planning procedure through certain paths. This program acts as a sketch of a plan that gives strong clues or restrictions about the intended solution. For the Situation Calculus this language is Golog [Levesque 1997] that combines elements from classical programming (conditionals, loops, etc.) with expressions that denote action formulas and non-deterministic constructs, such as choice between two actions, choice of action arguments and iteration. Existing implementations use successor state axioms when evaluating conditional statements in a Golog program, as well as pure regression to evaluate a fluent condition. As a consequence, the evaluation of a condition in general depends on the length of the history and the number of fluents, whose past values have an influence on the conditional statement.

Fluent Calculus. Although successor state axioms of the Situation Calculus suffice to solve the representational frame problem, which concerns the efforts to specify the non-effects of actions, they fail to address the inferential frame problem for actually computing these non-effects. The solution to the inferential frame problem was first introduced in the equational logic programming formalism of [Hölldobler 1990] that later reformulated by Thielscher and became known as Fluent Calculus [Thielscher 1998, Thielscher 1999b].

The Fluent Calculus is a many-sorted predicate logic language that in addition to *actions*, *situations* and *fluents* also defines a sort for *states*. Unlike the Situation Calculus, the Fluent Calculus distinguishes the notions of state and situation; the latter contains a history of actions that have been performed, while the former refers to the actual fluents that hold. A function $State(s)$ denotes the state in situation s. Each situation has an associated state, i.e., the world can be in the same state in different situations, but the state in every situation is unique. States are reified by representing fluents and states as terms, instead of atoms. Based on this notion, the frame problem is solved by *State Update Axioms*, which define the effects of an action as the difference between the state prior to the action and the successor one, formalizing an equational relation between states at consecutive situations. Under the assumption that positive and negative effects are disjoint, state update axioms are a provably correct solution to the frame problem. In addition, extensions of the Fluent Calculus to accommodate the ramification and the qualification problems have been presented, e.g. in [Thielscher 2000b], along with a provably correct axiomatization.

To design intelligent agents that reason and plan on the basis of the Fluent Calculus, a high-level programming method, called FLUX [Thielscher 2005a], has been developed,

which uses the paradigm of constraint logic programming. It comprises a method for encoding incomplete states along with a technique for updating these states according to a declarative specification of the elementary actions and sensing capabilities of an agent. The main difference between the Flux and Golog languages is that the former adopts the progression principle to evaluate conditions in agent programs, in contrast to regression applied in Golog. Progression through state update axioms enables Flux programs to scale up well to long action sequences performed by agents, thus managing more efficiently high computational load.

Event Calculus. Another logic-based framework for representing and reasoning about actions that has recently attracted much attention, is the Event Calculus, originally proposed by Kowalski and Sergot [Kowalski 1986]. Like the previous calculi, the Event Calculus considers actions (also called *events*) and *fluents*. In addition, this first-order language also defines a sort for *timepoints*, which are used to axiomatize expressions, such that fluents are true if they have been initiated by an event occurrence at some earlier timepoint. A central feature of the Event Calculus is a possibly incomplete specification of a set of actual event occurrences, called a *narrative*. Based on narratives and the use of circumscription [Lifschitz 1994] to tackle the frame problem, the calculus is capable of representing a variety of phenomena more naturally and also to perform non-monotonic reasoning, as described in many works, such as [Mueller 2006] and [Shanahan 1999b]. The frame problem, in particular, is avoided by application of the *commonsense law of inertia*, which states that fluents maintain their truth value unless affected by some event.

To date, the Event Calculus has been reformulated in a variety of dialects and applied in a multitude of contexts, such as cognitive robotics, web service composition, pervasive computing, education, video games and many others. An extensive list of references of such formulations and extensions can be found in [Miller 2002] and [Mueller 2006], which also summarize how variants of the calculus may be expressed as classical logic axiomatizations to handle a broad set of phenomena. Moreover, tools exists that use the Event Calculus for automated reasoning about action and change, such as the DECReasoner solver presented in [Mueller 2004] that performs deduction and abduction reasoning through satisfiability applying a method for encoding discrete event calculus problems into propositional conjunctive normal form. Another system is the $\mathcal{E}$-RES system [Kakas 2000] that implements the $\mathcal{E}$ action language [Kakas 2002, Kakas 1997], which is based on the calculus ontology. In fact, Miller and Shanahan [Miller 2002] have defined conditions un-

der which an $\mathcal{E}$ domain description matches an Event Calculus domain description, in that they entail the same fluent truth values. Finally, recently a solver that casts Event Calculus descriptions into Answer Set Programming (ASP) reasoning task has been released [Kim 2009].

The Event Calculus differs in many ways from the Situation and the Fluent Calculus, but most significantly by the representation of time. The Event Calculus uses a linear time structure, where all events are considered to be actual, in contrast to the branching time structure of Situation and the Fluent Calculi, where each event may give rise to a different possible future and events are considered to be hypothetical. Another distinctive feature is that the basic ontology of the Event Calculus does not support functional fluents, rather they can only be represented by means of appropriately constrained relational fluents. A more elaborate comparison of these formalisms is discussed below in Section 2.1.6.

Action Languages. Motivated by an increasing number of initial formulations of action formalisms being erroneous or permitting unintended and counterintuitive conclusions, especially regarding causality matters, a class of action theories has been developed that is independent of a specific axiomatization. Building on formal validation methods, the intension of this family of languages was to ensure that the encoding of any domain will yield correct results, thus permitting the assessment and comparison of the variety of different approaches that coexist in the field. Although initially they were used to established results for a restricted class of domains, recent formalisms have high expressiveness, natural language-like syntax and clear formal semantics.

The semantics are based on the theory of causal explanation proposed in [McCain 1997], which distinguishes between the claim that a formula is true and the stronger claim that there is a *cause* for it to be true. The result of such causal reasoning is that some properties of classical implication do not hold for causal implication. For instance, causality is non-monotonic, not reflexive, contraposition cannot be applied to causal implication, while transitivity can neither be allowed not forbidden [Schwind 1999]. Gelfond and Lifschitz [Gelfond 1998] provide an overview of causal languages, starting with the action description language $\mathcal{A}$, the first high-level action language introduced in the literature, which only supports one sort of expressions of the form A **causes** L **if** F, where A is an action name, L a literal and F a conjunction of literals (possibly empty). One extension of $\mathcal{A}$ is the action language C [Giunchiglia 1998], which provides additional language expressions, besides direct effects of actions, such as dependencies between fluents.

2.1.3 Time and Concurrency

The temporal aspect is an important property for action and change, therefore extensions for most actions theories have been proposed that include actions occurring at specific timepoints, have duration or occur concurrently.

Situation Calculus. The pure Situation Calculus handles actions sequentially and atemporally. Although restrictive, this theory is sufficient enough to express interleaved concurrency by introducing instantaneous actions that initiate and terminate the relational fluent, thus representing the process -but not the explicit duration- of performing an action [Reiter 2001a]. Interleaved occurrence of concurrent actions is appropriate if the outcome is independent of the order, in which the actions are interleaved. Still, no explicit representation of time is considered.

The extension of the sequential Situation Calculus with an explicit representation of time has been accomplished by adding a temporal argument to all instantaneous actions, denoting the actual time at which they occur [Pinto 1994]. Moreover, to represent concurrent actions and natural event occurrences (exogenous actions) in the Situation Calculus, the sort *action* of simple actions is distinguished from a new sort of *concurrent* actions [Reiter 1996]. Thus, an additional set of foundational axioms for the new sort, analogous to situation terms, is defined and the successor state axioms are generalized in a straightforward manner. The work in [De Giacomo 2000] axiomatizes the transition semantics leading to an extension of the implementation language Golog for concurrent actions. In [Papadakis 2002] Papadakis and Plexousakis propose an extension to the temporal Situation Calculus to handle concurrent instantaneous actions and actions with duration with effects occurring in any of the possible future situations resulting from an action's execution.

Fluent Calculus. The corresponding, to the classical Situation Calculus, notions for concurrent and continuous actions, applied to the Fluent Calculus, are presented in [Thielscher 1999a, Thielscher 2001b], where, in a similar fashion, time is parameterized as an argument of fluents and actions and new sorts for concurrent actions and arithmetic operations are added to the original language. State update axioms for concurrent actions are recursive, terminating with the base case of the empty action. Using recursive axioms, the effect of actions that occur simultaneously can be inferred by considering each of them independently, along with any additional positive or negative effects. Based on this formal

theory, an extension of the FLUX programming language to integrate both deliberate and natural actions into a single method for the planning and execution of actions, is presented in [Martin 2003].

Event Calculus. All axiomatizations of the Event Calculus reify the temporal argument to represent the time instant of event occurrences, enabling a more natural handling of temporal propositions. For instance, among the predicates defined in the calculus are the $HoldsAt(f,t)$ denoting that fluent f is true at timepoint t, $Happens(e,t)$ expressing that event e occurs at timepoint t, $Initiates(e,f,t)$ ($Terminates(e,f,t)$) representing that, if event e occurs at t, then fluent f will be true (resp. false) after t, and others. This handling of time is inherently represented in the formalism's basic ontology, in contrast to the Situation or Fluent Calculus, permitting any Event Calculus domain description to accommodate a wide range of temporal phenomena, such as direct, indirect and delayed effects of actions, concurrent events, events with duration, continuous change, triggered events etc. Furthermore, the time variable may belong either to the integers or to the reals sort.

Similarly, the Event Calculus-based action description language $\mathcal{E}$ incorporates this time structure to reason about partially ordered event occurrences. Yet, it does not employ the full expressive power of the formalism; $\mathcal{E}$ cannot declare effects that persist over a time span or represent delayed effects, as argued in [Papadakis 2002].

Action Languages. The dynamic causal law of action language C described in the previous section captures the interaction of concurrent actions as a set of simultaneous actions, without an explicit representation of the temporal dimension. The action language $C+$ [Giunchiglia 2004] that has evolved from C, is an important recent formalism that provides a uniform model for supporting, in addition to indirect effects and non-deterministic actions, also conditionals and concurrent actions. It can express both boolean and multi-valued, additive fluents and provides a more compact representation of several practical problems. Closely related is the recent planning language $\mathcal{K}$ [Eiter 2003a, Eiter 2004].

2.1.4 Non-determinism and Uncertainty

There is another critical dimension that should not be neglected when facing reasoning about actions for agents in dynamic real-world environments, that of uncertainty. Uncertainty may refer both to the initial state of the world and to the result of actions performed by an agent. The former aspect, which also deals with issues of partial observability, is

studied in the next subsection, whereas below we review how non-determinism can be used to model ambiguous action effects.

Situation Calculus. Pinto et al. [Pinto 2000] have proposed an extension to the standard Situation Calculus to incorporate indeterminism and actions with uncertain effects, based on a formalism-independent model for representing actions and change. The model decomposes indeterminate actions into a non-deterministic part and its possible outcomes and also associates a probability distribution to the outcomes. In [Mateus 2001] this work is further extended to consider domains in which the set of outcomes of an action are discrete, continuous or mixed, resulting to the Probabilistic Situation Calculus. A probabilistic variant of the action language Golog, called pGolog, has been proposed in [Grosskreutz 2000].

Fluent Calculus. The Fluent Calculus has been extended in a more straightforward manner to handle non-deterministic actions, as described in [Thielscher 2000a]. In order to model actions with alternative, but finite, outcomes a generalization of simple state update axioms that uses disjunction of possible equational relations between a state and its successor is generated. Moreover, there are actions, whose effect may involve a certain degree of vagueness. This uncertainty about effects can be expressed by existentially quantifying (and possibly restricting) one or more parameters of them that model an uncertainty factor.

Event Calculus. A convenient way to represent nondeterministic effects in the Event Calculus is to use deterministic fluents that follow the solution pattern specified for handling the frame problem. As already mentioned, within this calculus the frame problem is addressed by enforcing the commonsense law of inertia. Specifically, when a fluent is subject to this law, its truth value is preserved as time progresses, unless the fluent is affected by some event. By releasing a fluent from the law of inertia, its truth value is allowed to fluctuate in an arbitrary fashion and therefore may or may not be the same as it was in previous timepoints. As a result, releasing effects of actions from the law of inertia enables the introduction of uncertainty to domain theories. Apart from this approach, there are other alternatives, such as the use of *disjunctive event axioms* of the form $Happens(e,t) \Rightarrow Happens(e_1,t) \vee ... \vee Happens(e_n,t)$ with apparent results [Mueller 2006].

Action Languages. The action language $C+$ introduced before is expressive enough to support non-deterministic effects of actions. Recent extensions of $C+$ aim at combining both qualitative and quantitative uncertainty in a uniform framework for reasoning

about actions. Qualitative uncertainty is represented by forming a set of possible alternatives, while quantitative uncertainty is expressed through a probability distribution on a set of possible alternatives. Representative examples are the $\mathcal{PC}+$ [Eiter 2003b] and $\mathcal{GC}+$ [Finzi 2005] action languages. Moreover, Ferraris and Giunchiglia [Ferraris 2000] have proposed a simple extension to the STRIPS formalism allowing for specifying actions with non-deterministic effects. All these languages are highly usable in modeling uncertain domains, but are not flexible enough to incorporate treatment for other aspects.

2.1.5 Sensing, Knowledge and Belief Revision

Significant research effort in the field has concentrated on extending action theories to incorporate *sensing* or *knowledge-producing actions*, i.e., actions whose effects change the mental state of an agent rather that the actual state of the world. Since these studies are highly relevant to this thesis' work, related literature is discussed in detail in the next chapter. As an introductory remark we just mention that most of the frameworks are based on the possible worlds semantics, described in Section 2.2.1 of this chapter, and present complete solutions to the frame problem for knowledge. As a result, memory emerges as a side-effect: a fluent remains known, unless something relevant has changed.

Reasoning about knowledge assumes that the agent's knowledge and sensor results are always correct and newly acquired information does not contradict with existing knowledge. This is often a very strong assumption, therefore some studies have suggested extensions that include some sort of belief revision. In [Shapiro 2000] the Situation Calculus model is extended to reason about belief rather than knowledge and a similar approach concerning the Fluent Calculus has been formalized in [Jin 2004]. Still, these efforts do not propose solutions to the analogue of the frame problem in the context of belief. This is developed in [Scherl 2005] where it is argued that greater expressivity, than offered by the Situation Calculus, is needed, in order to be able to quantify over fluents and states. The Fluent Calculus, on the other hand, can be utilized to form a successor state axiom for beliefs that also models the case where the result of the sensing action contradicts the current state of knowledge generating a set of states that are both consistent with the result of the sensing action and minimally close to a state which was belief accessible prior to the sensing action.

2.1.6 Linear vs Branching Time Representation

In this section we elaborate on the properties of the different time representations and how they influence reasoning with the corresponding formalisms. As already stated, the classical Situation Calculus, as well as the Fluent Calculus, use a branching time representation, where all actions are considered hypothetical. A point in time in the Situation Calculus is represented by a situation (sequence of actions), and each new action may give rise to a different possible future. In the Event Calculus, on the other hand, there is a single time line on which events occur and all events are considered to be actual events. A temporal ordering of event occurrence formulae in the Event Calculus constitutes a narrative of actual events. In other words, the Situation Calculus relates fluents to situations, while the Event Calculus relates fluents to timepoints belonging to the real time line.

In the mid nineties the subject of reasoning with the different temporal representations and, as a consequence the relation between the two calculi, was a vivid research topic. In [Van Belleghem 1995, Van Belleghem 1997] and latter in [Chittaro 2000] a detailed analysis of the relation between the Situation and Event Calculus was attempted, concluding that they are indeed very similar, but, in contrast to previous to the time studies, with important differences, as well. The problem is on counterfactual or hypothetical reasoning. For instance, statements of the form "If A had happened, then B would have held", can be correctly represented in theories with branching time structure, where one can simultaneously talk about several possible evolutions of the world. Still, the Situation Calculus suffers from other forms of restrictions for counterfactual reasoning, as shown in [Van Belleghem 1997] (e.g., when non-deterministic actions are used).

On the other hand, the Event Calculus has the advantage of facilitating representation of continuous change, with respect to the Situation Calculus; in the Event Calculus we only need to add an extra axiom describing such change, without modifying the existing axioms. It is possible to extend the Situation Calculus to deal with continuous change as well, though the extension requires more substantial modifications to the formalism than in Event Calculus [Van Belleghem 1995].

Circumstances in which the different formulations of the two calculi are important do occur sometimes even within a discrete time structure, as, for instance, when confronting ramifications. The Event Calculus is better suited to deal with at least a large class of ramifications in a correct way, without other additions or modifications to the framework,

as opposed to the Situation Calculus and despite the existence of several ways to overcome these problems, such as the assumption of a notion of previous state of the world.

There are many studies that incorporate the time representation of the one calculus to the other. Pinto and Reiter [Pinto 1995], for instance, endow the branching structure of time, implicit in the Situation Calculus formalism, with a time line, presenting a formalization of the Situation Calculus enriched with a predicate "actual" that identifies a path of situations describing the world's true evolution. An initial attempt for a branching time Event Calculus was presented in [Levy 1998]. A more elaborate study has been conducted by Mueller [Mueller 2007b] who modified the classical logic Event Calculus to yield a new formalism that instead of linear, uses branching time and supports reasoning about hypothetical events, just like Situation Calculus. Finally, one should also mention the recent work of Thielscher for a unifying action calculus [Thielscher 2010], which is independent of a specific solution to the Frame Problem and is shown to be general enough to encompass a variety of different action representation formalisms. Most notably, it abstracts from the underlying time structure (branching or linear) and thus can be instantiated with both Situation Calculus-style approaches as well as Event Calculus-like languages. In so doing, this general calculus provides a uniform method for translating a variety of specific formalisms into each other.

To conclude, most comparative studies argue that a linear time structure is often better suited for modeling real world domains. The conclusion is based on the fact that it is in general more efficient to state explicitly when events occur or define partial orderings among them, than reasoning on several different action sequences that may require to update the global state of the world whenever something happens, even if the narratives are completely independent from each other.

2.1.7 Discussion and Comparative Study

The problem of representing and analyzing the dynamics of complex domains and managing certain of its properties has proven to be an essential, as well as a difficult one for AI scientists. So far, a multitude of formal methods have been presented that can be used to capture different aspects, such as goals, actors and actions, responsibilities and constraints. It has also been described how they adapt to the diversity of real-world conditions that constitute the domain, such as to express temporal constraints, uncertainty and sensing.

We have emphasized on the use of intuitive and mathematically formal approaches, since they allow IS engineers to produce detailed, formal specifications of ambient computing processes. Formal models, in which concepts are defined rigorously and precisely, permit the use of mathematics, in order to verify that the specifications possess certain correctness properties, such as that constraints are maintained during process execution or that goals can be satisfied given the knowledge available. Table 2.1 below, although not exhaustive, summarizes the properties of the three main formalisms for reasoning about action, presenting a compact comparative overview of their abilities to deal with the different aspects of commonsense reasoning.

Nonetheless, despite their advancements, most approaches suffer from serious impediments when applied to real-world conditions, raising skepticism and creating a topic of enduring debate among computer science researchers. The increased complexity of the reasoning mechanisms forces most approaches to trade expressiveness for simplicity and broadness for efficiency, limiting the impact of their applicability to small-scale implementations. While the progress accomplished on reasoning about action and automated planning has been rapid, researchers have found great difficulty in integrating separate models for handling aspects such as nondeterminism and uncertainty, concurrency, natural actions, knowledge and beliefs, sensing actions and continuous change in one unified model. As argued by Thielscher [Thielscher 2000c], most extensions to the problems have in fact been investigated in isolation and combining co-existing models for different phenomena is a very challenging task.

An essential step towards refining state-of-the-art approaches for complex domains and facilitating the sharing of reasoning tools and libraries, is to understand the space of possible formalisms and where each formalism is situated in it. The literature is rich with notable relevant surveys and comparative studies concerning planning techniques, e.g., [Pollack 1999, Bresina 2002, Boutilier 1999, Blythe 1999, Gelfond 1998, de Weerdt 2005, Ghallab 2004]. Furthermore, there is real interest in identifying mappings and logical equivalences between the different formalisms. Schiffel and Thielscher in [Schiffel 2006] develop a formal translation between domain axiomatizations of the Situation Calculus and the Fluent Calculus and present a Fluent Calculus semantics of Golog programs, thus facilitating the comparison of extensions made separately for the two calculi, such as concurrency handling, or even to translate extensions made only for one of the calculi to the other. Other studies, such as [Baader 2005] and [Liu 2006], attempt to de-

velop an action language based on description logics that can be viewed as a fragment of the Situation Calculus. The objective is to find fragments of action theories that are decidable, while maintaining the well-established solutions to the frame problem and being sufficiently expressive to be useful in applications. This topic is further investigated in Section 5.4. In [Classen 2006], a translation of an ADL problem description, which is a fragment of PDDL, into a basic theory of ES, a variant of the Situation Calculus, where situations occur only in the semantics, is presented. Furthermore, a version of Fluent Calculus, sufficiently expressive to capture the broad formalized semantics of the Features-and-Fluents framework, is presented in [Witkowski 2006], that provides a translation function that maps any scenario expressed in this ontological class into a Fluent Calculus axiomatization. Much work has also been conducted on showing how the Event Calculus corresponds to the Situation Calculus, as in [Kowalski 1997, Kristof Van Belleghem and Marc Denecker and Danny De Schreye 1995].

Summarizing, the Situation Calculus is one of the most widely adopted formalisms that for many years has steered research in the field and still is the source of novel ideas, since many inspiring researchers work on it. However, acknowledging the effort required in applying theory to practice when confronting the complexity of real-world domains, one can identify restrictions that limit its applicability to reasoning with short action sequences. Already, newer extensions replace entirely regression with progression to achieve more efficient reasoning and provide solutions to problems for which regression is in general inappropriate, such as the projection problem [Vassos 2008]. The Fluent Calculus, on the other hand, owes much of its good reputation to the use of progression that, in conjunction with the explicit representation of states, can be accorded a high level of expressiveness and efficiency for the different extensions that usually advance those of the Situation Calculus. Moreover, its implementation by means of the Flux programming language has achieved significant results. Nevertheless, both these calculi cannot easily provide unified ontologies that integrated solutions for most of the problems encountered in commonsense domains. The Event Calculus, on the other hand, is more flexible in expressing a multitude of domains in its core ontology. Also the temporal aspect should not be disregarded. The explicit treatment of time is crucial for real-world applications. Arguably, each approach has different strong assets with respect to the others; in this thesis we employ the Event Calculus, as it is highly usable, comprehensive and handles inherently most of the important phenomena of reasoning about action and change, as underscored in recent studies (e.g., [Mueller 2007a]). Furthermore, active research in implementing reasoners for this calcu-

lus is combined with recent achievements in other fields of logic programming, resulting in highly efficient new reasoners that exploit fast satisfiability or answer-set programming planners.

Table 2.1: Comparing Calculi for Reasoning About Action

	Situation Calculus		**Fluent Calculus**		**Event Calculus**	
	Basic Ontology	Extensions	Basic Ontology	Extensions	Basic Ontology	Extensions
Nature	monotonic second-order	first-order	monotonic first-order	first- and second-order	non-monotonic first order	-
Frame Problem	successor state axioms	-	state update axioms	-	inertia, circumscription	-
Ramification	state, causal, trigger constraints	-	state, causal, trigger constraints	-	state, causal, trigger constraints, delayed effects	
Qualification	Limited [Lin 1994]	-	Abnormal fluents	-	Abnormal fluents	Degrees of Abnormality
Temporal Representation	implicit (sequential actions)	explicit [Pinto 1995, Papadakis 2002]	implicit	explicit [Thielscher 1999a, Thielscher 2001b]	explicit	-
Time Structure	branching	actual timeline [Pinto 1995]	branching		linear time	branching time [Mueller 2006]
Counterfactual Reasoning	excluding nondeterminism	with time [Pinto 1995]	can be supported	-	-	BDEC [Mueller 2006]
Concurrency	interleaved	explicit [Reiter 1996, Papadakis 2002]	interleaved	explicit [Thielscher 2001b]	explicit	

	Situation Calculus		Fluent Calculus		Event Calculus	
	Basic Ontology	Extensions	Basic Ontology	Extensions	Basic Ontology	Extensions
Actions with Duration	implicit [Reiter 2001a]	explicit [Papadakis 2002]	-	explicit [Thielscher 2001b, Witkowski 2006]	implicit	3-argument *Happens*
Non-determinism & Uncertainty	-	probabilistic [Pinto 2000]	-	disjunctions with degrees [Thielscher 2000a]	explicit	
Knowledge & Sensing	-	K fluent [Scherl 2003]	-	$KState$ fluent [Thielscher 2000d]	-	partial solution [Forth 2004]
Belief Update	-	restricted [Shapiro 2000]	-	reliability degree [Jin 2004, Scherl 2005]	-	-
Relational/ Functional Fluents	Yes/Yes	-	Yes/Yes	-	Yes/No (constructed)	-
Implementation	Golog	Con-, Indi-, cc-, p-Golog	Flux	-	SAT-based DECReasoner, ASP-based, online reasoner	$\mathcal{E}$-RES

2.2 Reasoning about Knowledge with Epistemic Modal Logic

Epistemic modal logic is a subfield of modal logic that is concerned with reasoning about knowledge. Knowledge modeling is usually structured around the possible-worlds specifications and the application of Kripke semantics, which provide a very intuitive approach to understanding the process of reasoning and analyzing complex systems. The general syntax and semantics of this logic of knowledge are succicently described next, along with the set of properties that will later be of importance to the theory that we develop in the thesis.

2.2.1 Possible Worlds Semantics

The idea of applying the possible worlds semantics to model knowledge and belief was originally due to Hintikka [Hintikka 1962]. The intuitive idea of this paradigm is to acknowledge in the semantics that *things might have happened differently from the way they did in fact happen*. Thus, besides the true state of affairs (actual world), there are other possible states. Under this interpretation, an agent is said to know a fact if this is true in all the states that it considers possible. The language that is employed to formalize these ideas is typically some variation of propositional modal logic[1]. Such a language **PL** uses a set **P** of primitive propositions $(p, q, r, ...)$ to represent basic facts about the world. Technically, a *language* is just a set of formulae $(\phi, \psi ...)$. To express knowledge statements, we augment the language by modal operators $K_1, .., K_n$, where n (≥ 1) is used to declare the knowledge obtained by different agents. Starting from the primitive propositions in **PL**, we form more complicated formulae by closing off under negation, conjunction and the modal operators for knowledge or belief. The statement that an agent does not know whether ϕ holds means that it considers both ϕ and $\neg\phi$ possible.

To give semantics to sentences in **PL**, a Kripke structure is usually employed. A Kripke structure **M** for a system of n agents is a tuple $< \mathbf{S}, \pi, \mathbf{R}_1, .., \mathbf{R}_n >$, where **S** is a set of possible worlds or states and π is an interpretation that associates each world in **S** with a truth assignment. That is, for each state $w \in \mathbf{S}$, $\pi(w)(p)$ is a mapping from a primitive letter $p \in \mathbf{P}$ to {true, false}. Each $\mathbf{R}_i$ is a binary (accessibility) relation on **S** that captures the possibility relation according to agent i. Thus, $w\mathbf{R}_i w_1$ holds if agent i considers world w_1

[1]The material in this section is largely taken from [Fagin 2003].

possible when in world w. In addition, to capture what it means for a formula to be true at a given world in a structure the relation $\models$ is be defined, where $(\mathbf{M}, w) \models \phi$ means "ϕ is true at $\mathbf{M}, w$" or "ϕ holds at $\mathbf{M}, w$". Some of the clauses defined using $\models$ are as follows:
$(\mathbf{M}, w) \models p$ iff $\pi(w)(p) = \text{true}$
$(\mathbf{M}, w) \models \neg\phi$ iff $(\mathbf{M}, w) \not\models \phi$
$(\mathbf{M}, w) \models \phi \wedge \psi$ if both $(\mathbf{M}, w) \models \phi$ and $(\mathbf{M}, w) \models \psi$
$(\mathbf{M}, w) \models K_i\phi$ iff $(\mathbf{M}, w_i) \models \phi$ for all w_1 such that $w\mathbf{R}_i w_1$

2.2.2 Basic Knowledge Axioms

The Distribution Axiom. One important property of the definition of knowledge is that each agent knows all the logical consequences of its knowledge. If an agent knows a and knows that a implies b, then both a and $(a \Rightarrow b)$ are true at all worlds it considers possible. This axiom allows us to distribute the K operator over implication and it follows that

(K) $\models (Ka \wedge K(a \Rightarrow b) \Rightarrow Kb)$

The Necessitation (or Knowledge Generalization) Rule. Agents know all the formulae that are valid in a given structure. If a is true at all the possible worlds of structure $\mathbf{M}$, then a must be true at all the worlds that an agent considers possible in any given world in $\mathbf{M}$, so it must be the case that Ka is true at all possible worlds of $\mathbf{M}$. More formally,

(RN) For all structures $\mathbf{M}$, if $M \models a$, then $M \models Ka$

From this we can deduce that if a is valid, then so is Ka. This rule is very different from the formula $(a \Rightarrow Ka)$, which says that if a is true, then the agent knows it. An agent does not necessarily know all things that are true, however, agents do know all valid formulae. Intuitively, these are the formulae that are necessarily true, as opposed to the formulae that just happen to be true at a given world.

The Knowledge (or Truth) Axiom. Although an agent may not know facts that are true, it is the case that if an agent knows a fact, then it is true. This property has been taken by philosophers to be the major one distinguishing knowledge from belief. Formally:

(T) $\models Ka \Rightarrow a$

	Axioms	
Name	Axiom Schema	Restriction on R_i
K	$\mathbf{K}_i\alpha \wedge \mathbf{K}_i(\alpha \supset \beta) \supset \mathbf{K}_i\beta$	none
T	$\mathbf{K}_i\alpha \supset \alpha$	reflexive
D	$\mathbf{K}_i\alpha \supset \neg\mathbf{K}_i \neg \alpha$	serial
4	$\mathbf{K}_i\alpha \supset \mathbf{K}_i\mathbf{K}_i\alpha$	transitive
5	$\neg\mathbf{K}_i\alpha \supset \mathbf{K}_i \neg \mathbf{K}_i\alpha$	euclidean
Rules of Inference		
Name	Rules	
MP (Modus Ponen)	From α and $\alpha \supset \beta$ infer β	
RN (Necessitation)	From α infer $\mathbf{K}\alpha$	

Table 2.2: Characteristic axioms and rules of inference of knowledge.

Consistency Axiom (D). If we want to model the notion of belief, then we ought to drop axiom (T), but add an axiom capturing the fact that an agent does not believe false:

(D) $\neg K(false)$, or equally

$Ka \Rightarrow \neg K \neg a$

The Positive Introspection Axiom. Agents can do introspection regarding their knowledge, i.e., they know what they know.

(4) $\models Ka \Rightarrow KKa$

The Negative Introspection Axiom. Agents know what they do not know.

(5) $\models \neg Ka \Rightarrow K \neg Ka$

Although a number of additional properties follow from the basic axioms, in a precise sense these properties completely characterize the definition of knowledge, as far as the K operators are concerned. There exists a very strong relationship between these axioms and the properties of the accessibility relation $\mathbf{R}_i$ between worlds of the Kripke structure. Specifically, different axiom systems that combine subsets of the aforementioned axioms can be defined, according to the properties that we wish to infuse to the accessibility relation (Table 2.2). Yet, in any of them the Distribution Axiom and the Necessitation Rule are two properties that seem forced on us by the definition for knowledge of the possible-worlds approach.

To conclude, the construction of an axiom system for knowledge consists of the following axioms and inference rules, as well:
(A1) All tautologies of propositional calculus
(MP) From a and $(a \Rightarrow b)$ infer b (modus ponens)

2.2.3 The Problem of Logical Omniscience

The notion of possible worlds as a means to represent knowledge and belief has been extensively studied in the last decades, giving rise to formal ways of reasoning with epistemic properties. The success attained by the resulting epistemic and doxastic modal logics is based on the fact that it provides a very intuitive and expressive approach to model the mental state of intelligent agents. On the other hand, the commitment to Kripke semantics introduces serious side-effects, such as the well known *problem of logical omniscience*. In brief, logical omniscience requires an agent to know all logical consequences of its beliefs (i.e., the agent is a perfect reasoner and its beliefs are closed under implication) and all valid sentences (including tautologies). Apparently, these properties are too pretentious to expect for practical agents with limited resources (computational intractability) and may also lead to unrealistic cognitive capabilities (irrelevant beliefs).

The *partiality* or *incompleteness* of possible worlds has been traditionally accepted in the AI literature as an intuitive property, in order to alleviate logical omniscience-related problems, since the agent may be unaware of certain facts, may have limited resources or may ignore some relevant rules (e.g., the agent may have not been told what the rule of Modus Ponens is). *Inconsistency* is a totally different matter: the agent may focus in a subset of its beliefs (context) to draw conclusions that are consistent within the context, but inconsistent otherwise. Human believers are rarely consistent, they often have beliefs a and b, where $a \models \neg b$, without being aware of the implicit inconsistency. Konolige [Konolige 1986] argues that logical consistency is much too strong a property for resource bounded reasoners, she thinks that being non-contradictory (not believing a and $\neg a$ at the same time) is the most one can reasonably demand; if a theory is expressed in first-order logic, it is not even decidable in general whether it is consistent or not. As McArthur points out in [McArthur 1988], computationally logical omniscience results in intractability or, for the first-order case, undecidability. As a result, many approaches to mitigate the logical omniscience problem have been suggested, based either on a syntactic or a semantic

treatment of the epistemic notions (relevant surveys and comparative studies can be found in [Moreno 1998, Sim 1997, Halpern 2007, Halpern 2008]).

2.3 Ambient Intelligence

In the early 2000s, a number of initiatives, projects and research reports were announced around the globe that even their titles suggested the fact that Information Technology was on the verge of a shift towards a new research trend. The EU-funded "*Disappearing Computer*" initiative[2], the W3C "*Device Independence*" activity[3] and the National Academy of Science "*Embedded Everywhere*" research agenda [National Research Council Staff 2001], all made public at 2001, were indicative of an attempt to articulate a more technology-transparent and user-centered research endeavor. This attempt was mainly driven by the *ubiquitous computing* paradigm, a term coined by Mark Weiser's 1991 vision of a new generation of computer systems [Weiser 1991]. Today, the main tenets of this nascent field have been clearly formulated, leading to the emergence of the highly-evolving multi-disciplinary domain of *Ambient Intelligence* following the trends of ubiquitous and context-aware computing. Indicative of this attempt to articulate a more technology-transparent and user-centered reality is the intensive research that is being conducted to realize pragmatic infrastructures and novel technologies, as well as the increasing volume of journal and conference papers that explore the frontiers of computing under this context.

2.3.1 Characteristics of Ambient Intelligence Environments

For decades, humans had to continuously adapt themselves to their surrounding technology, in order to make the best out of it. The vision of Ambient Intelligence assumes a shift in computing towards a multiplicity of communicating devices disappearing into the background, providing an intelligent, augmented environment, where the emphasis is on the human factor.

Ambient Intelligence stems from the convergence of a multitude of disciplines: distributed intelligence, ubiquitous computing, dynamic networks and ubiquitous communications, intelligent human-computer interaction and intuitive user-friendly interfaces. The

[2]The Disappearing Computer: http://www.disappearing-computer.net/

[3]Device Independence: http://www.w3.org/2001/di/

field was initially promoted by a number of reports published by the European Information Society Technologies Advisory Group[4] (ISTAG) and has expanded significantly since. It visualizes an intelligent ubiquitous computing space, where a distributed network of potentially hidden sensors, intelligent devices and interfaces provides presence- and context-aware services to humans, adapting and responding to their needs, preferences, habits and gestures in a seamless, unobtrusive notion. The vision of worlds embedded with smart components that can understand human behavior and respond intelligently to spoken or gestured indications of desire signalled the emergence of numerous projects that emphasized on the use of *sensing technology* and the need to collect and interpret detected data. Profound projects, such as IBM's BlueEyes[5] and MIT's Project Oxygen[6], have made essential initial contributions on this difficult task.

Of dominant relevance in the attempt to efficiently interpret and act upon sensed data is the exploitation and management of context-sensitive information, which characterizes the interaction between humans and their surrounding environment. In close collaboration with sensing, recent advancements in network technology and wireless communications have enabled the rapid growth of *context-awareness*, not solely restricted on location related information, but extended to other parameters that determine a particular situation, such as movement, light, audio, humidity and others. The objective of this technological development is the creation of a highly personalized future society, where both humans and devices will be characterized by semantic profiles in order to model and share their desires, abilities and specifications. The MyCampus [Sadeh 2006] is an indicative project that aims at implementing a context-aware computing infrastructure for providing services adaptable to user needs and based on semantic information stored in profiles, called e-Wallets [Gandon 2004].

Beneath this high level consideration of the Ambient Intelligence vision, IS scientists can identify a multitude of computer science challenges that such device- and computation-rich environments entail. The accomplishment of the desirable level of "smartness" and the implementation of *intelligent computing* has been set as a high priority from the very beginning. Projects such as Philips HomeLab [Marzano 2003], Georgia Tech's Aware Home[7] and Gator Tech Smart House [Helal 2009], are being conducted under the guideline of

[4]IST Advisory Group (ISTAG): http://cordis.europa.eu/fp7/ict/istag/home_en.html
[5]IBM Blue Eyes Project: http://www.almaden.ibm.com/cs/BlueEyes/index.html
[6]MIT Project Oxygen: http://oxygen.lcs.mit.edu/
[7]Georgia Aware Home: http://awarehome.imtc.gatech.edu/

building smart systems to augment and respond intelligently to human behavior. The expert can recognize numerous issues introduced by intelligent computing: the need to deploy algorithms that enable mobile devices to engage in ad hoc communications; to collaborate in order to achieve common and complex objectives; to reason about their actions and knowledge potentially in a distributed manner; to contribute resources in a conservation-efficient manner. This is just a subset of the facets that synthesize the field.

Of course, the very idea of ambient computing has also raised a number of social and ethical concerns, as well. The humanistic notion of machine interaction and the privacy and trust issues of Ambient Intelligence require significant and long term underpinning research that is subject to focused research communities.

2.3.2 Challenges for AI

The Ambient Intelligence paradigm has generated an enabling multidisciplinary research field that envisages to bring intelligence to everyday environments and facilitate human interaction with devices and the surrounding. Artificial Intelligence has a decisive role to play for the realization of this vision promising commonsense reasoning and better decision making in dynamic and highly complex conditions, as advocated by recent studies [Ramos 2008]. Within AmI environments human users do not experience passively the functionalities of smart spaces, instead they participate actively in it by performing actions that change its state in different ways. At the same time, the smart space itself and its devices are expected to perform actions and generate plans either in response to changes in the context or to predict user desires and adapt to user needs.

Ambient Intelligence follows on from work in AI. While there have been significant advancement in many of the involved disciplines for computational intelligence individually over the last years, arranging a physical environment where mobile and stationary devices communicate and cooperate to achieve common objectives has proven to be a laborious task for the research community. Although much success has been achieved in defining theoretical frameworks for the fields of distributed AI, agent teamwork and coalition formation, planning and reasoning about actions and cooperative problem solving, the advent of ubiquitous and context-aware computing has introduced new, more practical challenges, pushing research in these fields to its limit. Often, current algorithms cannot accommodate the burgeoning complexity inherent in real-world ambient systems, since they typically

rely on restricted models and simplifying assumptions, which do not apply in realistic conditions. The majority of deployed Ambient Intelligence systems do not achieve the desired level of intelligence in understanding and reacting to human behavior, stumbling on the restrictions of current methodologies and technology used. Already works are being published that question the logic and rational behind some of our larger expectations; Rogers [Rogers 2006], for instance, argues that the progress in Ubiquitous Computing research has been hampered by intractable computational and ethical problems and that the field needs to broaden its scope, setting and addressing other goals that are more attainable.

When research on planning within the AI community established the *classical planning problem* for the development of techniques for agents to generate courses of action in order to reach a desirable world state, it adopted a number of simplifying assumptions to delimit the domain. Some of these assumptions were very restrictive. The planning agent was considered omniscient, possessing knowledge about all relevant facts of its environment. Its actions were deterministic, atomic and simultaneous, they had neither temporal extent nor fixed times of occurrence. The environment was assumed static and the only source of change in the environment was the agent itself, while no other exogenous event occurred. In addition, the agent possessed perfect domain and initial state knowledge. The goals presented to it remained unchanged throughout the processes of planning and execution and were categorical, i.e., they were either achieved or not. Of course, these simplifications do not persist in realistic planning situations and must be relaxed or completely eliminated when adapting planners to Ambient Intelligence systems.

We can recognize certain challenges and limitations that must be addressed for realizing the Ambient Intelligent paradigm. We can no longer rely on a complete description of the world domain that will allow us to have well-defined theoretical foundations for reasoning about action and planning. Participating agents have limited perceptions to acquire knowledge about the initial and running state of the world, the availability of actions and their effects, and the accuracy of sensing actions, leading to contradicting belief states between situated agents that, nevertheless, must cooperate in performing joint activities. Moreover, the closed world assumption is a luxury that tends to be abandoned, since exogenous actions are important cause of world state change.

Distributed planning is an intricate part of the ambient computing environment. Unfortunately, several requirements restrain the applicability of planning algorithms, such as limited computing resources, issues related to trust and privacy about sharing common

plans and contributing available resources to others and also the inability in sharing common plan representations and action decompositions. Even the nature of communication between entities designates different approaches according to various conditions; we cannot adopt a centralized plan coordination scheme when mobile devices negotiate in ad hoc manner to achieve a state of affairs, nor can we suggest broadcasting messages among team members when power preservation is a valuable metric.

Mobility is another aspect that renders an Ambient Intelligence system highly complex and dynamic. Mobility refers to situations where computing devices are constantly moving in the environment and their expectancy to participate in teamwork is completely depended on the duration of their connection to the network. This challenge has side effects on many parameters that influence planning tasks constituting planning an ongoing, dynamic process requiring plan and execution interleaving, as well as efficient monitoring and backtracking mechanisms.

This is just a brief introduction to the listing of problems that the research community tries to harness, some more attainable than other. If we take into consideration the variety of Ambient Intelligence scenarios that necessitate both user-initiated and automatically generated service provisioning, potentially requiring proactive planning and real-time execution monitoring, we might understand the reason why our current efforts, as considerable as they have been, they still do not match up to our expectations, as Greenfield also argues in [Greenfield 2006]. In this thesis we construct a theory that can contribute to many of theses issues, while laying the ground for future extensions to accommodate even broader requirements. In particular, focusing on the single agent case, we assume reasoning about a great variety of commonsense phenomena occurring in an Ambient Intelligence environment under partial observability and run-time knowledge update. Although we treat sensing as an accurate process, this assumption can be lifted in future extensions. Furthermore, having a complete and formal characterization of an agent's mental state, we provide the ingredients for studying multi-agent collaboration based on introspection and common knowledge.

CHAPTER 3

Review of State-of-the-Art

Contents

The previous chapter surveyed the progress in the broader field of action theories, and also described the general characterization of epistemic notions within modal logics. As the present thesis is targeted on the integration of these two fields into a unified epistemic action theory, this chapter studies in more detail frameworks that contribute towards this line of research. Initial attempts to formalize the epistemic effects of actions on the agent's mental state adopted the standard possible worlds approach. Highly expressive frameworks have been developed, treating sensing as a form of action to enable high-level cognitive tasks. Nevertheless, these frameworks introduce serious computational issues that render them inappropriate for practical implementations. Contemporary progress explores alternative knowledge representations disengaged from the possible worlds specifications, promising more efficient reasoning. In the following two sections we review frameworks of both categories presenting a historical development of the field up until state-of-the-art approaches and discuss main features and limitations.

3.1 Possible worlds-based Epistemic Action Theories

Probably the first work that motivated research for the creation of epistemic action theories is due to Moore [Moore 1985], who presented an adaptation of the possible worlds semantics in formal action theories treating the accessibility relation between possible worlds as a fluent. Based on initial versions of the Situation Calculus, Moore's axiomatization describes how the knowledge of an agent may change after executing sensing or ordinary

actions. In particular, the number of accessible related worlds remains unchanged upon ordinary action occurrences, but reduces as appropriate when sense actions occur, in order to reflect the new knowledge; after sensing only those worlds where the sensed fluent has the same truth value as the result obtained by the action remain accessible related.

This approach has influenced a multitude of studies since. Scherl and Levesque further investigated knowledge and sensing actions in the context of the Situation Calculus, providing a solution to the frame problem for this type of actions [Scherl 1993, Scherl 2003]. Introducing the epistemic fluent K and defining its semantics, they proved a number of properties that the specification enjoys. For instance, they showed that sense actions only affect the mental state of the agent and do not change the state of the world, they elaborated on the result of memory preservation of inertial aspects and they also showed how regression can be applied to sense actions. The same framework has been further extended to accommodate the notion of *ability* to achieve an objective [Lespérance 2000]. One of the impediments of the approach, though, was that it did not distinguish between the actual effects of actions and what the agent knows about these effects.

This issue has been investigated more thoroughly in the work of Thielsher within the context of the Fluent Calculus [Thielscher 2000d]. Providing a relatively similar definition of knowledge, based on states instead of situations, i.e., collection of fluents rather than action sequences, Thielscher also proposed a solution to the inferential, in addition to the representational frame problem for knowledge. In a similar to the Situation Calculus style, for the definition of knowledge no single initial situation in the tree of alternative worlds is maintained, instead a forest of trees is structured each with its own initial situation. The theory can employ an inference scheme for specifying what is known and not known about the successor situation in an elegant way distinguishing between the *de dicto* and the *de re* interpretation, along with an extension of the notion of ability defined for the Situation Calculus.

A framework with increased expressive capabilities was presented by Lobo et al. [Lobo 2001] that introduced the action description language $\mathcal{A}_k$, an extension of the high level language $\mathcal{A}$ to also handle sense actions and knowledge. In contrast to the previous approaches, this framework also incorporated conditional sensing, actions with non-deterministic effects that cause knowledge to be removed from the set of facts known by the agent, as well as preconditions of effects rather than just on actions. The semantics of epistemic states defined in $\mathcal{A}_k$ resembles Kripke structures, therefore it is comparable to

the previous calculi. On the other hand, the framework is more general and it is not possible to differentiate between the actual effects of actions and what an agent knows of them nor does it investigate complex ramifications and triggered actions, as we study in this thesis.

A similar approach has also been suggested in [Baldoni 2001, Baldoni 2004], but with the difference that epistemic evolution is studied from the modal logic standpoint. Specifically, Baldoni et al. described a modal action theory, in which actions were represented by modalities and belief (rather than knowledge) operators were ruled by the KD modal logic, resulting in the traditional three-valued Kripke-based models for state interpretations. The formalism could accommodate non-determinism and effect ambiguity, loss of knowledge and complex actions and could be used to provide provably correct conditional plans for execution with DyLOG. Still, in contrast to the previous approaches and the theory we develop in this thesis, it could not be used to represent disjunctive knowledge or to distinguish between the agent's mental state and the actual world state.

Extensions of epistemic action theories also deal with issues related to multiple agents [Shapiro 2002], group-level epistemic modalities [Kelly 2008] and others. Nonetheless, none of the aforementioned frameworks study the interaction of sensing and time, which is essential in most real-world applications. The results of sensing occur at a particular point in time, may have a predefined duration of validity and may even be different depending on other, potentially concurrent, sensing or non-sensing actions. This has been acknowledged in the work of Zimmerbaum and Scherl [Zimmerbaum 2001], who proposed a unified logical theory of knowledge, sensing, time and concurrency within Situation Calculus. The framework introduced interesting issues, but its applicability was limited, as the theory followed an alternative model for the treatment of time than the more widely used approach proposed by Pinto and Reiter [Pinto 1995]. This was due to the difficulty the authors encountered in combining the different extensions in this calculus. The Event Calculus, on the other hand, provides a unified framework that handles many aspects of commonsense reasoning, especially those related to time. It uses a linear time representation, where all events are considered to be actual, which is more suitable for real-world implementations, as opposed to the branching time representation of the Situation and the Fluent Calculi.

In addition, probably the most significant limitation of the approaches mentioned above is related to the representation of knowledge. They all share a common definition of semantics based on the possible worlds model and perform reasoning by adapting the accessibility relation over possible worlds, in order to determine if a formula is true or not in each of

them. Inevitably, the efficiency of theorem proving of knowledge formulae raises many concerns; with n atomic formulae, there are potentially 2^n distinguishable worlds to check truth in[1]. Despite the fact that all these rigorous frameworks provide very expressive formal accounts for knowledge and change, from the computational standpoint their application to practical implementations is less promising. Aiming at tractability, contemporary progress in the field explores alternative characterizations of knowledge, focusing on restricting expressiveness or sacrificing logical completeness, as regards to the standard possible worlds specifications.

Son and Baral [Son 2001], for instance, investigated a way to alleviate complexity issues by proposing the representation of an agent's state of knowledge at different levels of approximations. As with Lobo et al., they extended the action language $\mathcal{A}$ with a treatment of knowledge and the introduction of sense actions (they too named their resultant language $\mathcal{A}_k$). Instead of using the full expressiveness of Kripke structures, they used a simpler formulation to define their semantics leading to a smaller and more manageable state space. Moreover, they presented a type of approximate progression that leads to sound but incomplete future states. Following a rather similar motivation Claßen and Lakemeyer [Claßen 2009] adapted the logic of limited belief $\mathcal{SL}$ [Liu 2004] in the context of the Golog language. The authors exploit the idea of iterative deepening on levels of implicit beliefs based on an agent's explicit beliefs, thus achieving efficient reasoning with incomplete first-order theories. The approach enjoys tractability properties when restricting the KB in a proper form (described in the next subsection) and is based on regression, with progressive reasoning being in the author's future plans. In relation to our work, each approximation can be considered as augmenting a state with HCDs of different complexity. Still, the approaches are less general and do not consider knowledge-loosing actions or non-determinism. $\mathcal{A}_k$ in particular was intended as a high-level language to act as a benchmark formalism among epistemic theories and is restricted to the propositional case. On the other hand, the distinction between explicit and implicit beliefs that is attempted in these two latter approaches has also been shown to be an important leverage for addressing aspects of the logical omniscience problem; this line of research is an interesting direction for our work as well, where HCDs can assist in building iterative levels of derived implicit knowledge.

Recently, Son et al. [Son 2005] also proposed a set of approximations, with emphasis

[1] A more elaborate complexity analysis can be found in Section 5.3.

on handling domains that incorporate state constraints, a parameter not extensively investigated by previous approaches. Their study approached the field from the point of view of conformant planning and used the action language $\mathcal{AL}$. In contrast to previous conformant planners, their intension was not to compile away state constrains, in order to preserve expressiveness. They managed to reduce the complexity of planning to time polynomial with respect to the number of fluents (the state space remains exponential to the number of fluents) by means of two approximations that are sound (but incomplete), even in the presence of incomplete initial situations. The approximations investigate the inertial part of the domain focusing on what possibly holds or what possibly changes, respectively. The approach can also be extended to support concurrent and non-deterministic actions. Without introduction of these approximations, a standard conformant planner would require a double exponential state space to the number of fluents, which renders model checking to a co-**NP** complete problem, even without state constraints. Again, the main source of complexity is in handling incomplete information, which the planner models as a set of alternative possible states. Moreover, the epistemic part of a domain is not explicitly modeled nor are more complex ramifications beyond state constraints, as we do in this thesis.

3.2 Alternative Approaches

To alleviate the computational intractability of reasoning under the possible worlds specifications, as well as to address other problematic issues, such as the logical omniscience side-effect, alternative approaches for handling knowledge change have been proposed that are disengaged from the accessibility relation. These theories are rapidly moving from specialized frameworks to an important research direction, still they adopt certain restrictions about the type of knowledge formulae or domain classes that they can support.

A preliminary attempt to provide new insight to epistemic action theories was based on redefining knowledge in terms of *interval arithmetics* [Funge 1999]. Interval-valued epistemic fluents were incorporated in the Situation Calculus to express uncertainty about the value of ordinary fluents confined within (sometimes) maximally restricted and dynamically altered intervals, based on interval arithmetic operations.

In [Petrick 2002a] a knowledge-based planner for domains with incomplete knowledge and sense actions was presented. The framework was based on a first-order generalization of STRIPS in order to perform inferencing tasks by modeling actions as database updates

in a forward-chaining fashion and was later extended to accommodate postdiction inferencing, along with other representational enhancements [Petrick 2004]. The actions were considered as knowledge-level modifications to the agent's mental state, rather than as physical-level updates to the world state, thus lost their relation to their causal implications. Instead, in our approach actions maintain their full physical-level specifications, whereas knowledge modification is treated at a meta-level for extending their causal specifications with epistemic notions. To improve computational efficiency the expressiveness of the framework was restricted, so that only certain fixed types of disjunctive knowledge could be represented, while the inferential mechanism was incomplete. In particular, only "exclusive-or" knowledge of literals were permitted; exactly one of the literals of a disjunctive knowledge formula could be true. Despite the limitations, the planner could represent a wide range of interesting problems providing useful insight in the problem of planning under incomplete knowledge.

Maybe the most influential approach towards an alternative formal account for reasoning about knowledge and action is due to Demolombe and Pozos-Parra [Demolombe 2000] who introduced two different *knowledge fluents* to explicitly represent the knowledge that an ordinary fluent is true or false. Working on the Situation Calculus, they treated knowledge change as changing each of these fluents individually, the same way ordinary fluent change is performed in the calculus, thus reducing reasoning complexity by linearly increasing the number of fluents. Nevertheless, the expressive power of the representation was limited to knowledge of literals, while it enforced knowledge of disjunctions to be broken apart into knowledge of the individual disjuncts.

Petrick and Levesque [Petrick 2002b] proved the correspondence of this approach to the possible worlds-based Situation Calculus axiomatization for successor state axioms of a restricted form. Specifically, equivalence is obtained only when the conditions under which a fluent changes its truth value contain no fluent (context-free theories) or fluents in a restricted disjunctive normal form (literal-based theories). Moreover, in the same study the authors defined a combined action theory that extended knowledge fluents to also account for first-order formulae when disjunctive knowledge is tautology-free, still enforcing it to be broken apart into knowledge of the individual parts. Recently, a decomposition property of even more expressive classes of action theories has been suggested, based on the notion of a *Cartesian situation*, that simplifies certain complex types of disjunctive formulae into equivalent components that only mention fluent literals [Petrick 2008]. Intuitively, from

the situations that an agents considers as possible cartesian are pairs that differ on the value of only one instantiated fluent, while the values of all other fluents are identical. The price to pay is a requirement of definite knowledge of some of the disjunction components.

Regression used by standard Situation Calculus is considered impractical for large sequences of actions and introduces restrictive assumptions, such as closed-world and domain closure, which are problematic when reasoning with incomplete knowledge. Recent approaches deploy different forms of progression. Liu and Levesque [Liu 2005] for instance, study a class of incomplete knowledge that can be represented in so called *proper KBs* (i.e., KBs consisting of a finite set of formulae of the form $\forall(e \rightarrow \rho)$ or $\forall(e \rightarrow \neg\rho)$, where ρ ranges over atoms excluding equality and e ranges over quantifier-free formulae whose only predicate is equality) and perform progression on them. The idea is to focus on domains where a proper KB will remain proper after progression, so that an efficient evaluation-based reasoning procedure can be applied. Domains where the actions have local effects (i.e., when the properties of fluents that get altered are contained in the action) provide such a guarantee. The approach is efficient and sound for local effect action theories, still proper KBs under this *weak progression* do not permit some general forms of disjunctions to emerge, such as HCDs that we investigate in this thesis. The solution may even be complete when queries are in a certain normal form and the theory is context-complete (i.e., when there is complete knowledge about the context of context-dependent actions, at least at the time of action occurrence). The latter restriction is raised in [Vassos 2007], but with an extra cost of explicitly listing all possible values of each fluent, essentially rendering the theory propositional. Still, the local effect assumption seems too restrictive for realistic scenarios, while it is not clear how simple or more complex ramifications can be supported. Recently, Vassos et al. [Vassos 2009] investigated an extension to theories with incomplete knowledge in the Situation Calculus where the effects are not local and progression is still appropriate for practical purposes.

Beyond the Situation Calculus, knowledge update that resembles the previous approach is also suggested by Amir and colleagues [Amir 2003] that perform progression of belief states, rather than enumerating all possible worlds in every belief state and update each of those states separately in order to generate the updated belief state. The difference is that they present algorithms for complete or approximate recursive state estimation (or logical filtering as it is also named) in the context of non-deterministic actions as well, both for the propositional [Amir 2003] and the first-order case [Shirazi 2005]. Correctness of the

approach for answering queries uniform in the situation term is proved. Still, in contrast to our framework, the objective is not to develop a complete theory of knowledge and action where both the mental states of an agent and the real state of the world can be represented, but rather to focus on the epistemic inference capabilities of the agent. Moreover, complex ramifications and potential actions are not investigated.

Dependencies between unknown preconditions and effects have been incorporated in an extension of the FLUX programming language [Thielscher 2005a], which is used to implement efficient action theories with incomplete states based on the Fluent Calculus semantics. The extension, presented in [Thielscher 2005b], handles dependencies by appending implication constraints to the existing store of constraint handling rules, in a spirit very similar to the HCDs proposed in the present study. The emphasis is on building an efficient constraint solver, thus restricting expressiveness in a less broad class of domains. Moreover, it is not clear how the extensive set of complicated implication rules defined follow on from the properties of possible worlds. Initiating from a common ground, in the present study we provide a more intuitive and extensive mapping of the two approaches, revealing how their characteristics correlate.

Apart from Thielscher's work, the recent study by Forth and Shanahan [Forth 2004] is highly related to the framework proposed in this thesis. Extending the preliminary investigation of [Shanahan 2001], Forth and Shanahan utilized knowledge fluents in the Event Calculus to specify when an agent possesses enough knowledge to execute an action in a partially observable environment. Although their theory was based on a different Event Calculus axiomatization than we use here, described in [Shanahan 1999b], which treats nondeterminism in an alternative manner, the introduction of epistemic axioms share the same objective, i.e, to capture knowledge change as ordinary fluent change. Still, their intention was to handle ramifications in order to determine which fluents to sense and when, focusing on a closed and controlled environment; they did not provide a complete theory about knowledge within the Event Calculus. Specifically, an agent was only assumed to perform "safe" actions, i.e., actions for which enough knowledge about its preconditions was available. In an open environment the occurrence of exogenous actions might also fall under the agent's attention, whose effects are dependent on -unknown to it- preconditions. It is not clear how knowledge evolves in terms of such uncertain effects, neither how knowledge about disjunction of fluents can be modeled.

Furthermore, the theory proposed by Forth and Shanahan cannot handle situations

when an action might have multiple effects, dependent on different sets of preconditions. For instance, the action of opening a switch may cause a light to become *lit* if it was not lit before and also a relay to become *activated* if it was not activated before. These effects are the result of a single action occurrence and are independent to each other, but only dependent to the preconditions given. It is not clear how the theory would behave if the agent knew one of the preconditions, but not the other; would the action be considered a "possible" one or not? In either case, it is evident that an agent's knowledge should not only refer to a specific action, but also to the possible effects that can be achieved by it. Finally, in their approach the fundamental axioms of the Event Calculus about fluent persistence have been extended to accommodate the occurrence of non "possible" actions, action for which an agent does not have enough knowledge about. The choice of basing the persistence of ordinary fluents on what the agent knows or does not know seems less intuitive, since knowledge, by definition, has no effect on the domain. Our proposed theory, on the other hand, attempts a broader treatment of knowledge evolution within open environments, unifying a wide range of commonsense reasoning phenomena that might happen.

CHAPTER 4

Discrete Event Calculus Knowledge Theory

Contents

In this chapter we develop the basic axiomatization of the Knowledge Theory for agents operating in dynamically changing, partially observable and uncertain worlds. The theory uses the Event Calculus as an underlying formalism. We start by describing the basic tenets of the discrete time Event Calculus axiomatization and continue with an in depth analysis of the knowledge theory, splitting it in two sections: one defining the foundational meta-axioms for knowledge effects and another capturing causal dependencies among unknown fluents, called hidden causal dependencies. The formal definition of a domain description with epistemic concepts is given in Section 4.4. The complete axiomatization is summarized in the last section. Throughout the chapter various examples are given to exemplify concepts where necessary.

4.1 Preliminaries

4.1.1 General Notational Conventions

In the sequel of this thesis predicate symbols, function symbols and constants start with an uppercase letter, while variables start with a lowercase letter. Moreover, as we are going to define different sorts, all variables belonging to one sort are represented with the same letter with subscripts where necessary. Within all formulae free valuables are implicitly universally quantified.

The expression $\bigwedge^{f_i \in S}[P(f_i)]$ stands for $P(f_1) \wedge ... \wedge P(f_n)$ for all $f_i \in S$, where S a set of n fluents and $n \geq 0$ unless otherwise stated. In the particular case where $n = 0$ then $S = \emptyset$, meaning that no $P(f_i)$ predicate is included in the formula containing the initial conjunctive expression. Similar is the treatment of $\bigvee^{f_i \in S}[P(f_i)]$. In certain situations reference to the set S is omitted completely, as it will be clear from the context. Finally, the notation $P_1 \equiv P_2$ defines P_1 as an abbreviation for P_2; all occurrences of the expression P_1 are to be replaced with the expression P_2.

4.1.2 Discrete Time Event Calculus

The Event Calculus is a narrative-based first-order predicate calculus for reasoning about action and change. A number of different dialects have been proposed that are summarized in [Shanahan 1999b] and [Miller 2002]. All versions share the same ontology, in which events cause changes in the environment within an independently defined time structure. Intuitively, changes in the world may be due to the direct result of named actions performed by agents, external stimulus or the indirect result of state constraints. The Event Calculus applies the *principle of inertia*, which captures the property that things tend to persist over time unless affected by some event; when released from inertia, a fluent may have a fluctuating truth value. It also uses *circumscription* [Lifschitz 1994] to solve the frame problem and support default reasoning, i.e., reasoning in which a conclusion is reached based on limited information and later retracted when new information is available.

Our account of action and knowledge is formulated within the circumscriptive linear Discrete Event Calculus (Definition 4.1) that is extensively described in [Mueller 2006][1].

[1]The syntax given in [Mueller 2006] defines some additional predicates and axioms that allow for broader domains; we restrict our consideration only to those aspects accounted for in the rest of the thesis.

It is a discrete version of the classical Event Calculus [Miller 2002] and assumes a many-sorted first-order language, which uses *events* to indicate changes in the environment, *fluents* to denote time-varying properties and a *timepoint* sort, which is a subsort of the integer number sort. A set of predicates is defined to express which fluents hold when ($HoldsAt$), which events happen ($Happens$), what their effects are ($Initiates$, $Terminates$, $Releases$) and whether a fluent is subject to the law of inertia or released from it ($ReleasedAt$). If a fluent is initiated or terminated it becomes inertial at the next time instant. In the sequel, variables of the sort event are represented by e, fluent variables by f and variables of the timepoint sort by t, with subscripts where necessary.

Definition 4.1 (Discrete Event Calculus) *The domain-independent Discrete Event Calculus axiomatization (DEC) restricts the timepoint sort to the integers and consists of the following 8 axioms*

Influence of Events on Fluents

(DEC1) $Happens(e,t) \wedge Initiates(e,f,t) \Rightarrow$
$HoldsAt(f,t+1)$

(DEC2) $Happens(e,t) \wedge Terminates(e,f,t) \Rightarrow$
$\neg HoldsAt(f,t+1)$

(DEC3) $Happens(e,t) \wedge Releases(e,f,t) \Rightarrow$
$ReleasedAt(f,t+1)$

(DEC4) $Happens(e,t) \wedge (Initiates(e,f,t) \vee Terminates(e,f,t)) \Rightarrow$
$\neg ReleasedAt(f,t+1)$

Inertia of HoldsAt

(DEC5) $HoldsAt(f,t) \wedge \neg RealeasedAt(f,t+1) \wedge$
$\neg \exists e(Happens(e,t) \wedge Terminates(e,f,t)) \Rightarrow$
$HoldsAt(f,t+1)$

(DEC6) $\neg HoldsAt(f,t) \wedge \neg RealeasedAt(f,t+1) \wedge$
$\neg \exists e(Happens(e,t) \wedge Initiates(e,f,t)) \Rightarrow$
$\neg HoldsAt(f,t+1)$

Inertia of ReleasedAt

(DEC7) $RealeasedAt(f,t) \wedge$
$\neg \exists e(Happens(e,t) \wedge (Initiates(e,f,t) \vee Terminates(e,f,t))) \Rightarrow$

$$ReleasedAt(f, t+1)$$

(DEC8) $\neg RealeasedAt(f,t) \wedge \neg\exists e(Happens(e,t) \wedge Releases(e,f,t)) \Rightarrow$
$\neg ReleasedAt(f, t+1)$

□

Axioms (DEC1-4) express the influence of event occurrences on the state of fluents, axioms (DEC5,6) enforce the commonsense law of inertia for the truth values of fluents, while (DEC7,8) enforce inertia for the *ReleasedAt*() predicate. According to the axiomatization, if a fluent has been initiated or terminated by means of an effect axiom it cannot change its truth value indirectly (e.g., due to a state constraint), unless it is released beforehand.

It should be noted here that although the knowledge theory we develop in this thesis is based on the discrete time variant of the Event Calculus, no particular restriction would prevent us from using the continuous time axiomatization instead. In fact, the two formalisms are proved logically equivalent if the timepoint sort is restricted to the integers ([Mueller 2004]). Our choice was mainly motivated by reasons of simplicity and because reasoners for the former formalism are widely available. Furthermore, only the timepoint sort is restricted to the integers in the discrete time variant; other sorts may freely use the real number sort.

A particular domain description, as will be formally defined in Section 4.4, consists of an axiomatization describing the commonsense domain of interest, observations of world properties at various times and a narrative of known world events. For instance, *positive*, *negative* and *release effect axioms* describe the conditions under which an event e initiates, terminates or releases a fluent f at timepoint t, respectively:

$$\bigwedge\nolimits^{i}[HoldsAt(f_i,t)] \Rightarrow Initiates(e,f,t)$$
$$\bigwedge\nolimits^{i}[HoldsAt(f_i,t)] \Rightarrow Terminates(e,f,t)$$
$$\bigwedge\nolimits^{i}[HoldsAt(f_i,t)] \Rightarrow Releases(e,f,t)$$

Fluents $\vec{f_i}$ express the set of preconditions. Even when the conjunction of preconditions does not hold, event e may still happen, but it will not have the intended effect.

4.2 Core DECKT

The Discrete Event Calculus Knowledge Theory (DECKT) assumes agents acting in dynamic, non-deterministic environments, having accurate but potentially incomplete knowledge about the state of world aspects and able to perform *knowledge-producing* actions [Scherl 1993], actions that cause loss of knowledge, as well as actions with context-dependent effects. Knowledge is treated as a fluent, namely the *Knows* fluent, expressing which ordinary fluents and fluent formulae are known to the agent. To employ a theory of knowledge the *Knows* fluent is ruled by the knowledge (T) and the distribution (K) axioms of modal logic, with no addition of positive or negative introspection axioms, i.e., we concentrate on the epistemic state of an agent about dynamically changing facts of the world. The formulation of these axioms using Event Calculus syntax is as follows:

(T) $HoldsAt(Knows(f), t) \Rightarrow HoldsAt(f, t)$
(K) $HoldsAt(Knows(f_1), t) \wedge HoldsAt(Knows(f_1 \Rightarrow f_2), t) \Rightarrow HoldsAt(f_2, t)$

Intuitively, knowledge about a fluent is always correct and knowledge derivation is closed under logical consequence, i.e., it is complete and consistent. Full adoption of modal logic's S4 or S5 axiomatic systems is desirable for studying the theoretical properties of our knowledge theory, but not necessary for implementation purposes. In fact, certain strong assumptions imposed by such systems, such as the necessitation rule or even the reflexivity property, can be relaxed, sacrificing completeness for efficiency or resulting in a theory about belief. As we see later on, these axioms need not always be explicitly stated in the theory, rather they can be implicitly expressed by appropriately structuring the axiomatization. Furthermore, for any fluent f_i it is the case that:

(P1) $\bigwedge^i [HoldsAt(Knows(f_i), t)] \Leftrightarrow HoldsAt(Knows(\bigwedge^i f_i), t)$
(P2) $\bigvee^i [HoldsAt(Knows(f_i), t)] \Rightarrow HoldsAt(Knows(\bigvee^i f_i), t)$

The treatment of knowledge in our theory is based on the *closed-world assumption on knowledge*, which is interpreted as follows; an agent initially possesses a collection of sentences in its knowledge base which describes truths about the external world. Beginning with these sentences, everything that the agent knows involves this collection, as well as the sentences that follow logically from it. Whatever knowledge does not follow logically from the initial collection of knowledge is regarded as *lack of knowledge*. As the agent

interacts with the environment and performs ordinary and sensing actions, its knowledge base is continuously augmented by sentences asserting knowledge about the outcomes of those actions. Thus, lack of knowledge does not only appeal to the initial state, but is also relative to the agent's behavior. This is characterized by Reiter as a *dynamic* closed-world assumption ([Reiter 2001a], chapter 11).

As already mentioned, direct action effects, modeled as positive or negative effect axioms, cause the effect fluent to become inertial (DEC4), while indirect effects, expressed as state constraints, assume that the effect fluent be released. In order to enable the application of epistemic inferences for both direct and indirect action effects in an elaboration tolerant manner, i.e., without having to write additional rules for each new axiom added to the theory, the *Knows* fluent is always released from inertia in DECKT. Nevertheless, its state is never allowed to fluctuate, rather the axiomatization ensures that knowledge about a fluent is restricted by some state constraint at all times. To model knowledge about inertial world aspects, the KP fluent (for "knows persistently") is introduced. In brief, direct action effects affect the KP fluent, while indirect effects and ramifications of knowledge, owed to state constraints, may interact with the *Knows* fluent explicitly.

Notation: Before proceeding with the axiomatization we specify some notational conventions used for the DECKT axiomatization. Let C denote the context of a positive, negative or release effect axiom (the set of precondition fluents), i.e. $C = \{f_1, ..., f_n\}$, $n \geq 0$ (we omit to specify the axiom it refers to as it will be clear from the context). Let $C(t)^+$ be the subset of known preconditions from C at a given time instant t, i.e., $C(t)^+ = \{f \in C | HoldsAt(Knows(f), t)\}$. Finally, let $C(t)^- = C \setminus C(t)^+$ be the set of precondition fluents that the agent either does not know or knows that they do not hold at time instant t. Consequently, $C(t)^+ \cap C(t)^- = \emptyset$ and $C = C(t)^+ \cup C(t)^-$ for any time instant t. We also define a number of abbreviations. A fluent f is known whether it holds iff it is known to be true or known to be false (similarly for KPw):

(Kw) $HoldsAt(Kw(f), t) \equiv HoldsAt(Knows(f), t) \vee HoldsAt(Knows(\neg f), t)$
(KPw) $HoldsAt(KPw(f), t) \equiv HoldsAt(KP(f), t) \vee HoldsAt(KP(\neg f), t)$

Moreover, an event e affects or may affect a fluent f if there is some positive, negative or release effect axiom none of whose preconditions $\vec{f_i}$ is known false:

(KmA) $KmAffect(e, f, t) \equiv$

$$KmInitiate(e, f, t) \vee KmTerminate(e, f, t) \vee KmRelease(e, f, t)$$

where

(KmI) $KmInitiate(e, f, t) \equiv Initiates(e, f, t) \wedge \neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t)$

and similarly for $KmTerminate(e, f, t)$ and $KmRelease(e, f, t)$. These latter abbreviations define epistemic predicates that do not cause any actual effect to f.

4.2.1 Axiomatization

DECKT consists of the axiom sets described below (see also Table 4.1 in Section 4.5).

Knowledge and the law of inertia.
Knowledge is released from inertia at all times.

(KT1) $ReleasedAt(Knows(f), t)$

Knowledge persistence.
This axiom captures the correlation between the *Knows* and the *KP* fluent, introduced as a state constraint to the theory. *KP* is always subject to inertia.

(KT2) $HoldsAt(KP(f), t) \Rightarrow HoldsAt(Knows(f), t)$

Events with known preconditions.
If an agent knows all preconditions of a deterministic action, then it also knows its effect. $KP(f)$ and $KP(\neg f)$ cancel one another to preserve consistency.
Specifically, for a positive effect axiom $\bigwedge^{f_i \in C}[HoldsAt(f_i, t)] \Rightarrow Initiates(e, f, t)$:

(KT3.1) $Happens(e, t) \wedge \bigwedge^{f_i \in C}[HoldsAt(Knows(f_i), t)] \Rightarrow Initiates(e, KP(f), t)$
(KT3.2) $Happens(e, t) \wedge \bigwedge^{f_i \in C}[HoldsAt(Knows(f_i), t)] \Rightarrow Terminates(e, KP(\neg f), t)$

Similarly, for a negative effect axiom $\bigwedge^{f_i \in C} HoldsAt(f_i, t) \Rightarrow Terminates(e, f, t)$:

(KT3.3) $Happens(e, t) \wedge \bigwedge^{f_i \in C}[HoldsAt(Knows(f_i), t)] \Rightarrow Initiates(e, KP(\neg f), t)$
(KT3.4) $Happens(e, t) \wedge \bigwedge^{f_i \in C}[HoldsAt(Knows(f_i), t)] \Rightarrow Terminates(e, KP(f), t)$

Knowledge-producing (sense) events.
Sense actions provide knowledge about the truth state of fluents.

(KT4) $Initiates(sense(f), KPw(f), t)$

From (KT4), $Kw(f)$ is derived, due to (KT2), (Kw). By definition, sensing actions affect only the mental state of an agent and cause no effects on the domain. Intuitively, a sense action is a non-deterministic binary choice; from the point of view of the agent there are two possible outcomes and consequently epistemic states, one informing the agent that f is true and another that it is false.

Events with uncertain effects.
If an action with *deterministic* effects occurs, which (*a*) has at least one precondition whose truth value the agent does not know (hence, the agent does not know whether the effect axiom is triggered), (*b*) there is no precondition that the agent knows it does not hold (otherwise, the agent would have been certain that the effect axiom would not be triggered) and (*c*) the agent does not already know the potential new truth value of the effect fluent, then the agent dismisses its knowledge about the state of the effect. This is the case of deterministic fluents with unknown preconditions. Note that we implicitly assume $C \neq \emptyset$; if there are no precondition fluents, then the effect is always deterministically affected, falling under the scope of the (KT3) axioms. Formally, for positive effect axioms:

(KT5.1) $\bigvee^{f_i \in C}[\neg HoldsAt(Kw(f_i), t)] \wedge \neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t) \wedge$
$\neg HoldsAt(Knows(f), t) \wedge Happens(e, t) \Rightarrow$
$Terminates(e, KPw(f), t)$

Similarly, for negative effect axioms:

(KT5.2) $\bigvee^{f_i \in C}[\neg HoldsAt(Kw(f_i), t)] \wedge \neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t) \wedge$
$\neg HoldsAt(Knows(\neg f), t) \wedge Happens(e, t) \Rightarrow$
$Terminates(e, KPw(f), t)$

Moreover, for actions with *non-deterministic* effects, i.e., whenever $\bigwedge^{f_i \in C}[HoldsAt(f_i, t)] \Rightarrow Releases(e, f, t)$, axiom (KT5.3) below expresses that if none of the preconditions is known not to hold (either the effect's preconditions are unambiguously satisfied or the agent does not know if they are satisfied; in either case knowledge about the effect is lost), then the agent cannot infer the state of the effect:

(KT5.3) $\neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t)] \wedge Happens(e, t) \Rightarrow Terminates(e, KPw(f), t)$

Knowledge minimization.

In DECKT knowledge is derived either from direct actions effects (axiom sets (KT3-6)) that affect the KP fluent or indirectly from state constraints of the form $HoldsAt(\phi_1, t) \Rightarrow HoldsAt(\phi_2, t)$ that affect the $Knows$ fluent according to (K). As knowledge about a formula is always released from inertia, we need a way to prevent the $Knows$ fluent from fluctuating whenever knowledge cannot be inferred, i.e., whenever no domain state constraint is known to be triggered to produce either $HoldsAt(Knows(\phi_2), t)$ or $HoldsAt(Knows(\neg\phi_2), t)$. To obtain this result, we apply a form of default reasoning, assuming that by default at any timepoint knowledge about a fluent formula does not hold. Axiom (KT7) below performs exactly such a minimization to the extension of the $Knows$ fluent (in a style similar to performing circumscription to a formula for the purpose of minimizing the extension of a predicate).

Let $\phi_1(\vec{f_i})$, $\phi_2(\vec{f'_j})$ denote arbitrary formulae whose only free variables are fluents $\vec{f_i}, \vec{f'_j}$ and which do not mention epistemic fluents. Axiom (KT7) is structured as follows:

(KT7) $HoldsAt(Kw(\phi_1(\vec{f_i})), t) \Leftrightarrow$
$\exists f'_1, ..., f'_n (HoldsAt(Knows(\phi_2(\vec{f'_j})), t) \wedge$
$HoldsAt(Knows(\phi_2(\vec{f'_j}) \Rightarrow (\neg)\phi_1(\vec{f_i})), t)) \vee$
$HoldsAt(KPw(\phi_1(\vec{f_i})), t)$

where $f'_1, ..., f'_n$ $(0 \leq n \leq j)$ are those fluents in $\phi_2(\vec{f'_j})$ that do not appear in $\phi_1(\vec{f_i})$. The intuition is that an agent knows a formula iff there exists some state constraint known to be triggered at that particular time instant (therefore (KT2) is also accounted for).

By grounding (KT7) to the set of available state constraints of a particular domain axiomatization, its instantiation can significantly simplify the whole complexion and complexity of the axiom. This set is well defined, still it may be modified online according to occurring events and context. Proposition 3 in Section 7.3.1 prescribes exactly how the axiom can be grounded to a particular domain. As an example suppose that the only available to an agent state constraints are $HoldsAt(f_1, t) \Rightarrow HoldsAt(f, t)$ and $\neg HoldsAt(f_2, t) \Rightarrow HoldsAt(f, t)$, then (KT7) for f will be formulated as follows:
$HoldsAt(Kw(f), t) \Leftrightarrow HoldsAt(Knows(f_1 \vee \neg f_2), t) \vee HoldsAt(KPw(f), t)$
Notice that the disjunction of the bodies of the state constraints is required to be known, not

just knowledge about the individual disjuncts: according to (P2), this is a much stronger statement that needs to be accounted for that may lead to knowledge about f.

The following example creates a flavor of how the previously introduced axiom sets work together to achieve commonsense behavior given a domain axiomatization:

Example 4.1. Consider a domain where a robot can open a door if it stands in front of it. The positive effect axiom $HoldsAt(f', t) \Rightarrow Initiates(e, f, t)$ is applied to describe this situation, where fluent f denotes that a door is open, f' denotes that a robot stands in front of that door and e is the action of pushing gently anything that stands in front of the robot. Imagine that initially the agent only knows that the door is closed, i.e., $HoldsAt(KP(\neg f), 0)$ (or $HoldsAt(Knows(\neg f), 0)$ due to (KT2)). If $Happens(e, 0)$, the robot will lose its knowledge about f at t=1, due to (KT5.1). Instead, if it first senses its position with regard to the door, i.e., $Happens(sense(f'), 0) \wedge Happens(e, 1)$, then (KT4) will provide definite knowledge whether the robot stands in front of the door or not at t=1 and, if it does then the agent will also know the state of the door at t=2, due to (KT3.1,2), i.e., $HoldsAt(Kw(f'), 1) \wedge HoldsAt(Kw(f), 2)$. □

4.3 Hidden Causal Dependencies

The DECKT axiomatization presented so far enables the derivation of sound knowledge about context-dependent effects of actions, even when the conditions are unknown. Nevertheless, these epistemic inferences are not complete in certain situations, as compared to the possible worlds specifications. Completeness breaks down in situations that are not always apparent in the immediate epistemic state after an action execution, but may be detected in future timepoints. Such situations emerge when the effect or action preconditions are unknown to the agent. Consider the following example:

Example 4.2. Continuing from Example 4.1, assume now that the agent is unaware of both f' and f initially. Its epistemic state is illustrated in Figure 4.1 after different action occurrences[2]. Sensing f' at $t = 0$ leads to two sets of two possible worlds depending on the truth value of f'; either s'^1_{1a}, s'^1_{2a} where f' becomes known to be true and f remains

[2]We adopt a schematic representation to illustrate the set of worlds (or states) that an agent considers possible at a particular time instant, according to the possible-worlds approach. Recall that a fluent is known if it has the same truth value in all possible worlds. Below the dashed line the known to the agent formulae are shown and in certain cases fluents that are unknown are displayed for emphasis. Each possible world s is

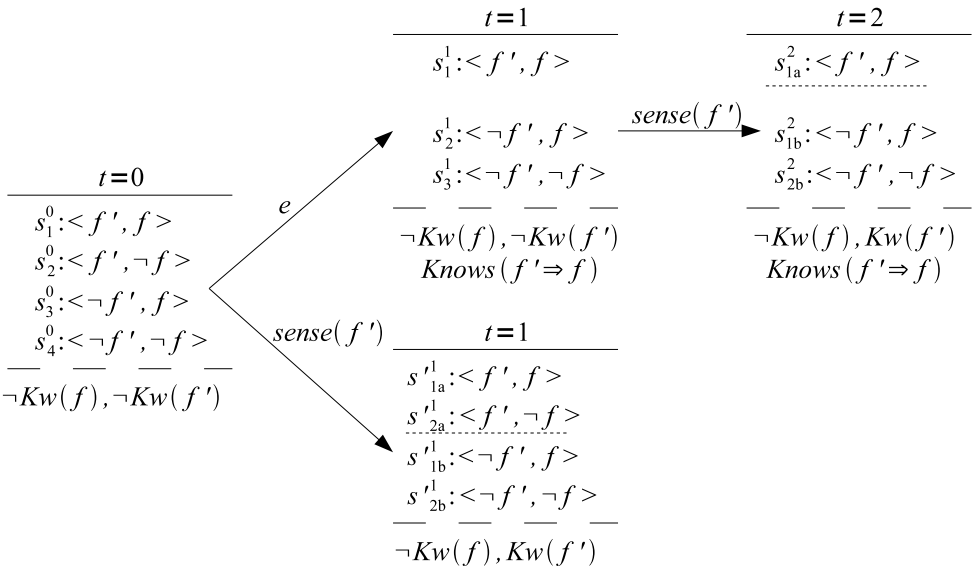

Figure 4.1: Action e initiates fluent f if f' holds.

unknown, or s'^1_{1b}, s'^1_{2b} where f' becomes known to be false and f remains unknown. On the other hand, if the agent executes action e before sensing f' three possible worlds remain at $t = 1$. Sensing f' after that will not affect its epistemic state concerning the individual fluents that will again be the same as when sensing f' at $t = 0$. Notice, though, that now there is only one world at the contingency resulting from f' being true, namely world s^2_{1a}. This means that although in the general case the agent does not know for sure whether f holds at $t = 2$, in the particular situation where f' was true initially, the agent has sufficient information to also infer that f is known to be true as well (if it finds out that it was standing in front of the door then it can infer that the door is open after the action). In other words, although we cannot claim that f is known at $t = 2$ after the two-action narrative, we should still express the fact that if f' were known to hold then f would also be known to hold. □

What the previous example reveals is the inherent ability of the possible worlds approach to incorporate information that is not always evident on the agent's epistemic state, but may appear at future timepoints as the number of possible worlds changes. In example 4.2 a dependency between f and f' is implicitly created after e; knowing f' results in knowing f as well. We name such an implicit dependency *hidden causal dependency* (HCD) and represent it as a disjunction (knowledge about disjunctions of fluents), rather

characterized by a serial number (subscript) and the timepoint that it refers to (superscript). If there are two alternative sets of possible worlds, for instance after a sense action, they are also declared in the subscript and are schematically separated by a dotted line.

than an implication. HCDs are primarily caused by the execution of actions with unknown preconditions, therefore most approaches that dispensed the use of possible worlds choose to restrict the expressiveness of the underlying action theory to either context-complete domains, i.e., when the context of occurring actions is completely known at the time of execution, or context-free domains.

Instead, in DECKT we integrate an approach to deal with such dependencies. We model the epistemic constraint that a HCD expresses as epistemic disjunctions of fluents and define a set of axioms that determine how the agent should treat these clauses of disjunctive knowledge. For instance, in example 4.2 at $t = 2$ the fact that $HoldsAt(Knows(f' \Rightarrow f), 2)$ which is implicitly inferred, can be represented by writing $HoldsAt(KP(\neg f' \vee f), 2)$. The disjunctive knowledge encodes a notion of epistemic causality in the sense that if future knowledge brings about f' (resp. $\neg f$) it also brings about f (resp. $\neg f'$).

Notice that we use the auxiliary KP fluent to capture this type of temporal knowledge. This way we can manipulate any disjunctive knowledge in a uniform manner, as described in the following subsections[3]. In the sequel we study the procedure of reasoning with HCDs; considerations will be given as to when they are created, when they are destroyed and what knowledge should be preserved when an HCD is destroyed. Knowledge preservation, in particular, is of critical importance, as HCDs may be involved in complicated interactions, potentially producing knowledge about other fluents. It is crucial to understand that although HCDs denote temporary implication relations, the knowledge that they produce is accurate and may be subject to inertia; actions affecting fluents of a HCD may destroy the dependency, but will not affect any information that might have been produced. This is the main difference between HCDs and domain state constraints, which have permanent validity and any change in the state of fluents involved in the body of the constraint will definitely affect its head. The former characterize how the state of the world has evolved at a particular time instant, while the latter how it should be at all times.

4.3.1 Creation of HCDs

Whenever an action with a context-dependent effect occurs, with preconditions unknown to the agent then a HCD is created. For brevity, we assume in the rest that no action affects

[3]This study develops a more general approach than the one proposed in [Patkos 2009a], dismissing completely the *Implies* fluent introduced there to exclusively characterize HCDs.

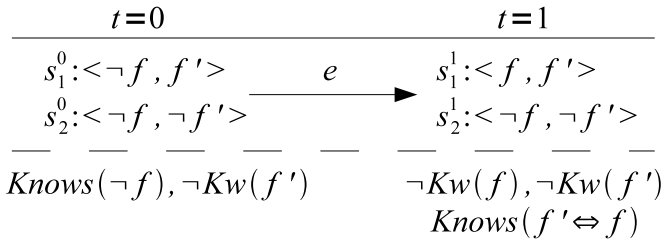

Figure 4.2: Event e initiates the previously known to be false fluent f, under the condition that f' holds.

the preconditions at the same time, except, of course, if one of the effect's precondition is the effect fluent itself. We distinguish the following situations.

4.3.1.1 Positive effect axioms: $\bigwedge^i[HoldsAt(f_i,t)] \Rightarrow Initiates(e,f,t)$

If an action with a positive effect occurs none precondition of which is known to be false, yet some are unknown to the agent, and the effect is not already known to be true, then a HCD is created between the unknown preconditions and the effect:

(KT6.1.1) $\neg HoldsAt(Knows(f),t) \wedge$
$\neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i),t) \wedge \bigvee^{f_i \in C}[\neg HoldsAt(Kw(f_i),t)] \Rightarrow$
$Initiates(e, KP(f \vee \bigvee^{f_j \in C(t)^-} \neg f_j),t)$

In other words, we augment the theory with a disjunctive knowledge formula that, considering axiom (KT2), is equivalent to $HoldsAt(Knows(\bigwedge^{f_j \in C(t)^-} f_j \Rightarrow f), t+1)$ but is more conveniently handled. See, for instance, Example 4.2 above. At timepoint $t = 1$, after the execution of e (KT6.1.1) is triggered to produce $HoldsAt(KP(\neg f' \vee f), 1)$ which results in $HoldsAt(Knows(f' \Rightarrow f), 1)$. Notice also that the conditions that trigger axiom (KT6.1.1) also trigger (KT5.1); this way, although the latter leads to loss of knowledge concerning the state of the effect, the former delimits this loss according to the occurring action.

Example 4.3. Consider now the case illustrated in Figure 4.2. Now the agent knows initially that the effect does not hold before the action, e.g., that the door is closed. This has as a result that after the action execution, the door is open *if and only if* the robot had been standing in front of the door at $t = 0$. In other words, at $t = 1$ we have that $(f' \Leftrightarrow f)$, i.e., in addition to the HCD between $\neg f'$ and f, another one is created between f' and $\neg f$.

Notice how this is confirmed by the possible worlds approach as even fewer worlds remain at the resulting state. □

From the latter example it becomes evident that a HCD is also created between an effect fluent and its unknown preconditions when an action initiates that effect, given that the agent knew that the effect did not hold before the action. For instance, by sensing that a door, which was known to be closed, has become open after performing the *PushGently* action, the robot can infer that it must have been standing in front of the door, i.e., the action's preconditions must be true:

(KT6.1.2) $HoldsAt(Knows(\neg f), t) \wedge$
$\neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t) \wedge \bigvee^{f_i \in C}[\neg HoldsAt(Kw(f_i), t)] \Rightarrow$
$\bigwedge^{f_j \in C(t)^-} Initiates(e, KP(\neg f \vee f_j), t)$

As before, instead of augmenting the theory with $\bigwedge^{f_j \in C(t)^-}[HoldsAt(Knows(f \Rightarrow f_j), t+1)]$, we write $\bigwedge^{f_j \in C(t)^-}[HoldsAt(Knows(\neg f \vee f_j), t+1)]$. Axiom (KT6.1.2) is triggered along with (KT6.1.1), resulting in the creation of an epistemic biimplication relation among the preconditions and the effect fluent. Returning to example 4.3, these two axioms model exactly the agent's epistemic state after the action execution, namely that the agent knows that f is true iff f' is true[4].

4.3.1.2 Negative effect axioms: $\bigwedge^{i}[HoldsAt(f_i, t)] \Rightarrow Terminates(e, f, t)$

The situation is similar for negative effect axioms with the difference that instead of f the HCDs are created for $\neg f$.

(KT6.1.3) $\neg HoldsAt(Knows(\neg f), t) \wedge$

[4] At this point it is appropriate to comment on the structuring of formulae for HCDs in the general case. In this study we implicitly assume one effect axiom per effect fluent in order to facilitate readability of the HCD formulae. In the general case, there might be multiple effect axioms with different contexts. In this case, a HCD is created if none of the precondition sets is completely known (otherwise, there is no point in creating the HCD) and there is at least one set where some precondition is unknown while no precondition in the same set is known to be false. One HCD is created for each such set between its unknown preconditions and the effect. In addition, if the effect is known not to hold, one more HCD is created comprising the effect and the disjunction of the conjunction of each set's unknown preconditions. By gathering all effect axioms of the same fluent into one axiom with preconditions expressed in DNF, the previous analysis becomes more clear. Still, we avoid this generalization by the simplifying assumption.

$$\neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t) \wedge \bigvee^{f_i \in C} [\neg HoldsAt(Kw(f_i), t)] \Rightarrow$$
$$Initiates(e, KP(\neg f \vee \bigvee^{f_j \in C(t)^-} \neg f_j), t)$$

(KT6.1.4) $HoldsAt(Knows(f), t) \wedge$
$$\neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t) \wedge \bigvee^{f_i \in C} [\neg HoldsAt(Kw(f_i), t)] \Rightarrow$$
$$\bigwedge^{f_j \in C(t)^-} Initiates(e, KP(f \vee f_j), t)$$

Example 4.4. Let the negative effect axiom $HoldsAt(f_1, t) \wedge HoldsAt(f_2, t) \Rightarrow Terminates(e, f, t)$ and imagine that an agent knows initially that f, f_1 hold but does not know whether f_2 holds, i.e., $HoldsAt(Knows(f), 0) \wedge HoldsAt(Knows(f_1), 0) \wedge \neg HoldsAt(Kw(f_2), 0)$. In this case, $C = \{f_1, f_2\}$, while $C(0)^- = \{f_2\}$. After $Happens(e, 0)$ both (KT6.1.3,4) will be triggered resulting in $HoldsAt(KP(\neg f_2 \vee \neg f), 1) \wedge HoldsAt(KP(f \vee f_2), 1)$, which is equivalent to $HoldsAt(Knows(f_2 \Leftrightarrow \neg f), 1)$. □

4.3.1.3 Release axioms: $\bigwedge^i [HoldsAt(f_i, t)] \Rightarrow Releases(e, f, t)$

It is trivial to see that in the case of non-deterministic effects a HCD is only created if the agent has prior knowledge about the effects. Specifically, only if it initially knows f's truth value and at some future time after the action it senses that some preconditions do not hold, then this knowledge should be preserved (or, in other words, only if it senses that the previously known effect fluent has changed its truth value will the agent be certain that the preconditions must have been true):

(KT6.1.5) $HoldsAt(Knows((\neg)f), t) \wedge$
$$\neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t) \wedge \bigvee^{f_i \in C} [\neg HoldsAt(Kw(f_i), t)] \Rightarrow$$
$$\bigwedge^{f_j \in C(t)^-} Initiates(e, KP((\neg)f \vee f_j), t)$$

4.3.1.4 Trigger axioms: $\bigwedge^i [HoldsAt(f_i, t)] \Rightarrow Happens(e, t)$

A trigger axiom causes an event to occur whenever the world is at a particular state. The event may have different effects described by positive, negative or release axioms. Therefore, we can expect e's preconditions to be added to the effect preconditions and treat the creation of HCDs in the way described in the previous subsections. Note though that, as we explain in Section 6.2, whenever the preconditions of a trigger action are unknown to the agent, instead of e a new action is executed, called e_{pot} (potential e) causing the effect to become unknown.

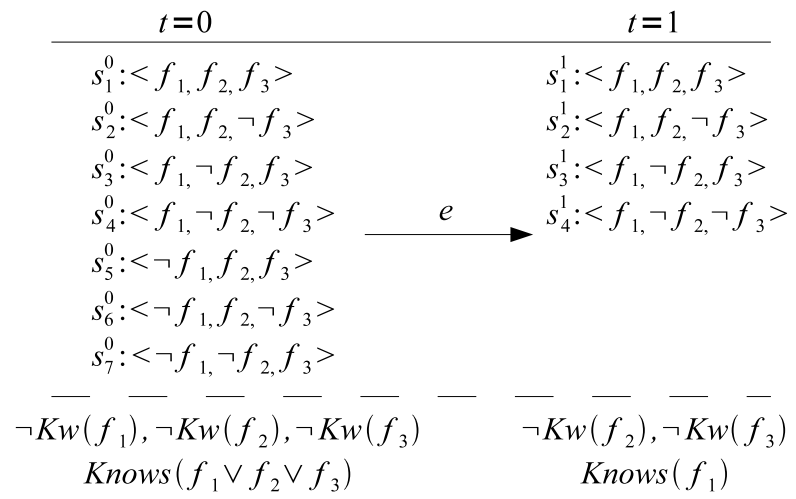

Figure 4.3: Event e initiates f_1, resulting to the expiration of the HCD.

4.3.2 Expiration of HCDs

In contrast to state constraints that introduce implication relations that must be satisfied at all times, HCDs are valid only for limited time periods, as they are created due to the agent's given epistemic state. Specifically, the dependency is valid for as long as the involved fluents remain unaffected by occurring events; if some event may modify them the dependency may expire (still, some knowledge can be preserved depending on the situation, as we see later on).

Example 4.5. Let an event e such that $Initiates(e, f_1, t)$ and let an agent knowing initially a HCD expressing that $HoldsAt(Knows(f_1 \vee f_2 \vee f_3), 0)$ (Figure 4.3). After the execution of action e, the agent's epistemic state dictates that f_1 is known to be true and moreover the HCD expires, since the remaining possible worlds cannot support any epistemic inferences concerning fluents f_2 and f_3. This should come as no surprise, since by modifying any unknown fluent of a disjunction, the agent cannot be certain if this was the one to contribute to its truthfulness. □

4.3.2.1 HCD Termination

If an event occurs that affects or may affect any of the fluents involved in a HCD then this HCD is no longer valid:

(KT6.2.1) $HoldsAt(KP(\bigvee^d f_d), t) \wedge Happens(e, t) \wedge \bigvee^d [KmAffect(e, f_d, t)] \Rightarrow$

$$Terminates(e, KP(\bigvee^{d} f_d), t)$$

4.3.2.2 HCD Reduction

Although all fluents of a newly created HCD are unknown to the agent, some of them may become known either through sensing or indirectly due to state constraints, as time progresses (direct fluent modification would trigger (KT6.2.1), leading to the termination of the HCD). Let D denote the set of fluents involved in an HCD and $D'(t)$ denote those fluents that are not directly known at t. For instance, suppose $HoldsAt(KP(\bigvee^{d} f_d), t)$ then $D = \{f_1, .., f_d\}$ and $D'(t) = \{f \in D | \neg HoldsAt(KPw(f), t)\}$. This set is critical for the DECKT axiomatization as it captures knowledge that may change in an implicit manner. To see why imagine that at some timepoint t we have a HCD stating that $HoldsAt(Knows(f_1 \vee f_2), t)$ and also $HoldsAt(KP(\neg f_1), t)$. This means that $\neg f_1$ is known explicitly and f_2 is known indirectly, i.e., $D = \{f_1, f_2\}$ and $D'(t) = \{f_2\}$. If an event e affects f_2 at timepoint t, the HCD will expire at $t + 1$, still knowledge about f_1 will remain since the KP fluent is inertial. On the other hand, if an event e' affects f_1 the HCD will expire, but knowledge about f_2 will be lost, as there will be no other rule to support it. This is inappropriate, therefore we need to acknowledge the existence of the $D'(t)$ set and treat its fluents with special care.

Depending on the type of action and the related context there are situations where although an HCD becomes invalidated there may still be knowledge that should be preserved. Specifically, if before the action the agent knows that the fluent that may be affected does not hold, then this fluent did not contribute to the HCD in the first place, therefore the remaining unknown components of the disjunction should create a new HCD, assuming of course that none of them is concurrently affected:

(KT6.2.2) $HoldsAt(KP(\bigvee^{d} f_d), t) \wedge Happens(e, t) \wedge$
$\bigvee^{d}[KmAffect(e, f_d, t) \wedge HoldsAt(Knows(\neg f_d), t)] \wedge$
$\neg \exists e'(Happens(e', t) \wedge \bigvee^{d}[KmAffect(e, f_d, t) \wedge \neg HoldsAt(Kw(f_d), t)]) \Rightarrow$
$Initiates(e, KP(\bigvee^{f'_d \in D'(t)} f'_d), t)$

That is, if an action affects or may affect a fluent of a disjunction that is known to be false, then the remaining components of the disjunction form a new HCD.

Example 4.6. Slightly modifying Example 4.5 above, imagine that now the agent initially knows that f_1 does not hold (Figure 4.4). The agent is aware that f_1 does not

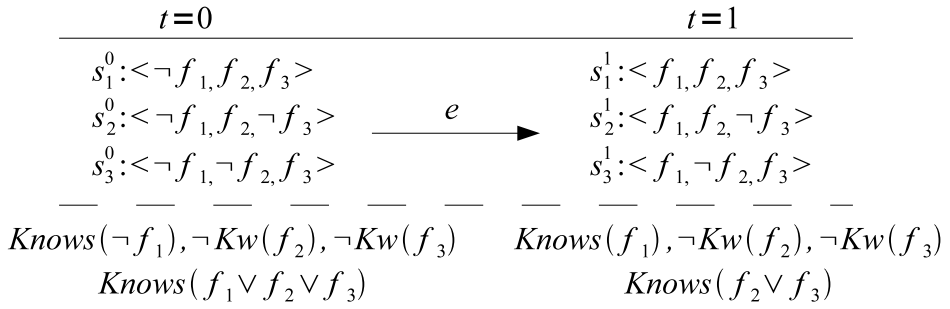

Figure 4.4: Event e initiates fluent f.

contribute to the HCD, therefore after an event e destroys the HCD, a new dependency is created between the remaining unknown fluents. This case is treated by axiom (KT6.2.2). □

4.3.2.3 HCD Expansion

Notice now the particular situation where a context-dependent event occurs, the preconditions of which are unknown to the agent and its effect is part of an HCD. In this case, the agent cannot be certain whether the HCD has been affected by the event or not, as this depends on the truth value of the preconditions. In fact, the HCD becomes contingent on the set of precondition fluents; if the preconditions prove to be false, the original HCD should still be applicable, otherwise it must be invalidated, according to the previous analysis. In order to handle this situation we need to dismiss the original HCD, based on (KT6.2.1), but in addition to create a new one that also incorporates the negation of the action's unknown preconditions. As a result, by obtaining knowledge about the preconditions (through sensing for instance) the agent can distinguish whether the original dependency should persist or not.

Example 4.7. Let actions e_1, e_2 such that $HoldsAt(f_1, t) \Rightarrow Initiates(e_1, f, t)$ and $HoldsAt(f_2, t) \Rightarrow Initiates(e_2, f_1, t)$. Let also an agent that initially knows none of f, f_1, f_2. As shown in Figure 4.5, after executing action e_1 at $t = 0$ and e_2 at $t = 1$, the agent still does not have knowledge about any of the involved fluents, despite the fact that the number of possible worlds has been reduced. More specifically, after execution of e_1 fluent f becomes contingent on its precondition f_1, due to (KT6.1.1). Therefore, we have that $HoldsAt(Knows(f_1 \Rightarrow f), 1)$. For the same reason, after e_2, we have that $HoldsAt(Knows(f_2 \Rightarrow f_1), 2)$. But now, the implication relation between f and f_1 becomes

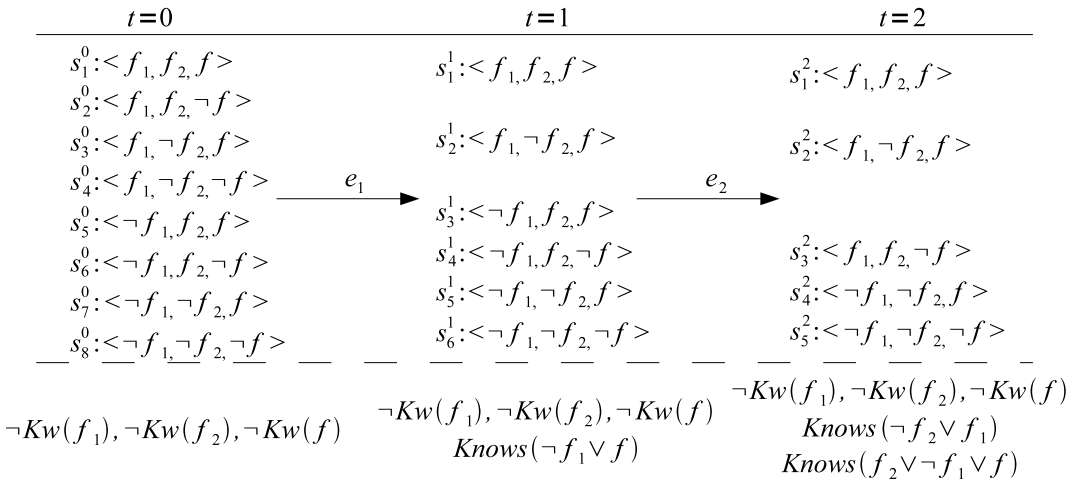

Figure 4.5: Event e_1 initiates fluent f if f_1 holds, while e_2 initiates f_1 if f_2 holds.

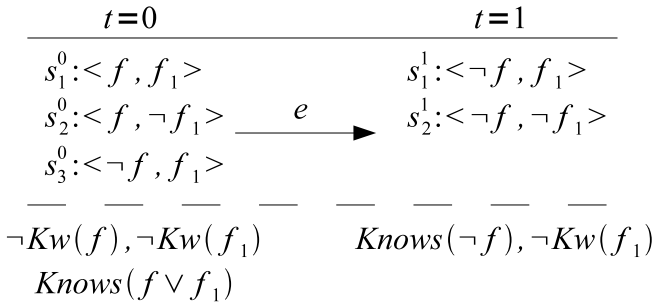

Figure 4.6: Event e terminates f given that f holds before the event's execution.

contingent on f_2. If f_2 holds, the execution of e_2 affects f_1. If it does not hold, e_2 will not have the intended effect, i.e., f_1 will not be altered, and the original dependency should remain valid. □

It becomes clear that the unknown preconditions of a context-dependent effect should result in the expansion of any HCD that includes that effect. Before modeling this situation though, one must notice an important contingency: the agent uses these preconditions to determine whether the original HCD should be applicable or not; what if this distinction cannot be made? Such a situation may be, for instance, the result of an action occurrence that leads to a world state where the precondition fluents have the same truth value regardless of the world state before the action.

Example 4.8. Imagine the effect axiom $HoldsAt(f,t) \Rightarrow Terminates(e,f,t)$ (eg. the action of closing a door if it is open) and an agent knowing initially $(f \vee f_1)$, where f_1

an arbitrary fluent, without knowing any of these fluents individually (Figure 4.6). After the execution of action e fluent f will not hold regardless of its previous state: if initially $HoldsAt(f,t)$ the effect axiom will be triggered resulting in $\neg HoldsAt(f,t+1)$; on the other hand, if $\neg HoldsAt(f,t)$ the effect axiom will not be triggered and $\neg HoldsAt(f,t+1)$ will again emerge due to inertia (the door will be closed after the action). Since f may be affected by e the original HCD must be abandoned, still no expanded HCD is created as the truth value of f cannot be used to distinguish the different contingencies. Nevertheless, notice that in such a setting the agent should know that $\neg HoldsAt(f,t+1)$, which is indeed derived by the axiomatization, due to (KT6.1.3): $HoldsAt(Knows(\neg f \vee \neg f), t+1)$. □

Considering the above discussion, we can now formalize the appropriate axiomatization for HCD expansion. First, we introduce the following abbreviation stating that a fluent may become inverted by an occurring event:

(INV) $KmInverted(f,t) \equiv$
$\exists e(Happens(e,t) \wedge (EffectPredicate(e,f,t) \vee KmRelease(e,f,t)))$

where, for a fluent literal f and its corresponding atom F, $EffectPredicate(e,f,t)$ denotes $KmTerminate(e,F,t)$ when $f = F$, and $KmInitiate(e,F,t)$ when $f = \neg F$. Notice that the *KmInverted* predicate is completely independent of the truth value that a fluent may have at any time instant. For example, for an effect axiom of the form $HoldsAt(f_1,t) \Rightarrow Initiates(e,f,t)$ we are interested whether $KmInverted(f_1,t)$ is true, while for the axiom $\neg HoldsAt(f_1,t) \Rightarrow Initiates(e,f',t)$ we should seek whether $KmInverted(\neg f_1,t)$ holds.

Hence, for any action e that initiates, terminates or releases a fluent of a HCD, if some of its preconditions are unknown to the agent, i.e., $C(t)^- \neq \emptyset$, and the unknown preconditions $\vec{f_j}$ of the effect fluent are not or may not be inverted, then a new HCD is created that considers all the components of the original HCD along with the unknown preconditions of e's effect axiom:

(KT6.2.3) $HoldsAt(KP(\bigvee^d f_d),t) \wedge Happens(e,t) \wedge$
$\bigvee^d [KmAffect(e,f_d,t) \wedge \neg HoldsAt(Kw(f_d),t)] \wedge$
$\neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i),t) \wedge \neg(\bigwedge^{f_i \in C(t)^-} [KmInverted(f_i,t)]) \Rightarrow$
$\bigwedge^{f_i \in C(t)^-} [Initiates(e, KP(f_i \vee \bigvee^{f'_d \in D'(t)} f'_d), t)]$

Intuitively, since any HCD represents an epistemic implication relation, axiom (KT6.2.3)

creates a nested implication relation with head the original HCD and body the negated unknown preconditions of the effect axiom that may affect that HCD.

Example 4.7. (cont'd) After event e_2 occurs, the HCD between f and f_1 expires, due to (KT6.2.1), still a new one is being created (or more accurately, the old HCD becomes contingent on f_2), due to (KT6.2.3). This results in $HoldsAt(Knows(f_2 \vee \neg f_1 \vee f), 2)$ (more clearly $HoldsAt(Knows(\neg f_2 \Rightarrow (f_1 \Rightarrow f)), 2)$). □

Example 4.8. (cont'd) On the other hand, for the HCD $(f \vee f_1)$ of this example, all conditions in the body of (KT6.2.3) are satisfied at t, except one: it holds that $KmInverted(f, t)$, due to the effect axiom. Therefore, no expanded HCD is created (the original HCD expires though, due to (KT6.2.1)). □

Yet another more general example is the following.

Example 4.9. Let $HoldsAt(f_1, t) \Rightarrow Initiates(e, f_2, t)$ and $HoldsAt(f_1, t) \Rightarrow Terminates(e', f_2, t)$ and initially $HoldsAt(Knows(\neg f_2 \vee f_1), 0)$ but neither f_1 nor f_2 are known.
If $Happens(e, 0)$ we get:
$HoldsAt(Knows(\neg f_1 \vee f_2), 1)$ due to (KT6.1.1) and
$HoldsAt(Knows(f_1 \vee \neg f_2 \vee f_1), 1)$ due to (KT6.2.3) because f_1 is unknown and the original HCD becomes dependent on it. The formulae are equivalent to $HoldsAt(Knows(f_2 \Leftrightarrow f_1), 1)$.
If $Happens(e', 0)$ occurred instead, then similarly we get:
$HoldsAt(Knows(\neg f_1 \vee \neg f_2), 1)$ due to (KT6.1.3) and
$HoldsAt(Knows(f_1 \vee \neg f_2 \vee f_1), 1)$ due to (KT6.2.3). These formulae are equivalent to $HoldsAt(Knows(\neg f_2), 1)$, i.e., the agent knows that f_2 will not hold regardless of f_1's state.
□

4.3.2.4 Transitivity

Finally, we also need to consider the transitivity property of implication relations. Whenever an agent knows that f_1 implies f_2 and f_2 implies f_3 there is an implicit relation stating that also f_1 implies f_3. If an action happens that affects f_2 the two original HCDs will expire due to (KT6.2.1), still the relation between f_1 and f_3 that has been established should persist. This is formalized in the following axiom:

(KT6.2.4) $HoldsAt(Knows(f \vee (\bigvee^{i} f_i)), t) \wedge HoldsAt(Knows(\neg f \vee (\bigvee^{j} f_j)), t) \wedge$
$Happens(e, t) \wedge KmAffect(e, f, t) \Rightarrow$
$Initiates(e, KP(\bigvee^{f_{i'} \in D'(t)} f_i \vee \bigvee^{f_{j'} \in D'(t)} f_j), t)$

4.3.2.5 Application of the HCD Axiomatization

The following example is illustrative of the axiomatization's potential in broad domains, combining HCDs with general epistemic disjunctions treated as HCDs.

Example 4.10. Let $HoldsAt(f_1, t) \Rightarrow Initiates(e, f_2, t)$ and $HoldsAt(f_3, t) \Rightarrow Terminates(e, f_4, t)$ and initially $HoldsAt(KP(f_1 \vee f_3), 0)$, but no single fluent is known whether it holds. After $Happens(e, 0)$ both effect axioms might be triggered, still the agent cannot know which. Therefore, although none effect is known to hold, their disjunction is known. Specifically:
$HoldsAt(Knows(\neg f_1 \vee f_2), 1)$, due to (KT6.1.1) and (KT2),
$HoldsAt(Knows(\neg f_3 \vee \neg f_4), 1)$, due to (KT6.1.3) and (KT2),
as well as $HoldsAt(Knows(f_1 \vee f_3), 1)$, due to inertia.
From these formulae $HoldsAt(Knows(f_2 \vee \neg f_4), 1)$ is also inferred as a derivation.

Imagine now that an event e' occurs at t=1 that terminates f_1 (unconditionally). Then, at t=2 all HCDs that include f_1 will expire. Specifically, we have that $\neg HoldsAt(Knows(f_1 \vee f_3), 2)$, due to (KT6.2.1), as well as $\neg HoldsAt(Knows(\neg f_1 \vee f_2), 2)$, for the same reason. Here is the tricky part. At this point, knowledge about $(f_2 \vee \neg f_4)$ is not supported by any rule and could get lost, which would be an incorrect result, as this is established knowledge and no event affected it. Axiom (KT6.2.4) avoids the inconvenience by retaining the correlation among the remaining fluents. In particular, from the expired HCDs that included f_1 and $\neg f_1$, it obtains $HoldsAt(Knows(f_3 \vee f_2), 2)$. From this formula and the fact that $HoldsAt(Knows(\neg f_3 \vee \neg f_4), 2)$ due to inertia, again $HoldsAt(Knows(f_2 \vee \neg f_4), 2)$ can be inferred. □

4.4 Formal Definition of Epistemic Domain Descriptions

We now formally define an epistemic domain description to represent both domain knowledge and also the agent's mental state. First, we extend the standard Event Calculus signature describing the commonsense domain to also accommodate the notion of knowledge.

Definition 4.2 (Event Calculus Epistemic Axiomatization) *We define an Event Calculus axiomatization with knowledge and sensing actions taking the form*[5]

$$D = E \cup M \cup \Xi \cup \Sigma \cup \Delta_2 \cup \Theta \cup \Omega \cup \Psi$$

where

1. E is a conjunction of domain independent Event Calculus axioms, such as (DEC1-8)
2. M is a conjunction of
   - knowledge axioms, resulting from the properties of the *Knows* relation, and
   - the necessitation rule, if required[6]
3. Ξ is the conjunction of the foundational axioms of the Discrete Event Calculus Knowledge Theory introduced in this chapter, as well as axioms for trigger axioms that are introduced in subsequent chapters, as extensions of the basic theory. The set Ξ comprises Ξ_1, which is the conjunction of (KT3-6) knowledge axioms and (TR3) for the effects of potential actions, Ξ_2 denoting the conjunction of (KT1,2,7) axioms, and Ξ_3 for the (TR1,2) axioms on potential action occurrences.
4. Σ is a conjunction of
   - positive effect axioms, i.e., $\bigwedge^i HoldsAt(f_i, t) \Rightarrow Initiates(e, f, t)$
   - negative effect axioms, i.e., $\bigwedge^i HoldsAt(f_i, t) \Rightarrow Terminates(e, f, t)$
   - release axioms, i.e., $\bigwedge^i HoldsAt(f_i, t) \Rightarrow Releases(e, f, t)$
   - positive cumulative/canceling effect axioms, i.e., $\bigwedge^i HoldsAt(f_i, t) \wedge \bigwedge^j (\neg) Happens(e_j, t) \Rightarrow Initiates(e, f, t)$
   - negative cumulative/canceling effect axioms, i.e., $\bigwedge^i HoldsAt(f_i, t) \wedge \bigwedge^j (\neg) Happens(e_j, t) \Rightarrow Terminates(e, f, t)$
   - effect constraints, i.e., $\bigwedge^i HoldsAt(f_i, t) \wedge \pi_1(e, f_1, t) \Rightarrow \pi_2(e, f_2, t)$, where $\pi_1, \pi_2 = Initiates$ or $Terminates$ predicates
5. Δ_2 is a conjunction of
   - trigger axioms, i.e., $\bigwedge^i HoldsAt(f_i, t) \Rightarrow Happens(e, t)$

[5] We extend the original notation from [Mueller 2006].

[6] *"If a formula is a theorem, then so is knowledge about that formula".*

- disjunctive event axioms, i.e., $Happens(e,t) \Rightarrow \bigvee^i Happens(e_i,t)$

6. Θ is a conjunction of cancellation axioms containing predicates $Ab_1, ..., Ab_n$ of the form $\bigwedge^i HoldsAt(f_i,t) \Rightarrow Ab_j(...,t)$

7. Ω is a conjunction of unique names axioms

8. Ψ is a conjunction of

   - state constraints, i.e., $\bigwedge^i HoldsAt(f_i,t)$ or $\bigwedge^i HoldsAt(f_i,t) \Rightarrow HoldsAt(f_j,t)$ or $\bigwedge^i HoldsAt(f_i,t) \Leftrightarrow \bigwedge^j HoldsAt(f_j,t)$
   - action precondition axioms, i.e., $Happens(e,t) \Rightarrow \bigwedge^i HoldsAt(f_i,t)$ and
   - event occurrence constraints, i.e., $Happens(e_1,t) \wedge \bigwedge^i HoldsAt(f_i,t) \Rightarrow (\neg)Happens(e_2,t)$ □

The M set employs an axiomatic system similar to a corresponding epistemic modal logic system. Since we talk about a theory of knowledge, the knowledge axiom in M is a minimal requirement. Moreover, the set Ξ of axioms prescribes how knowledge should evolve according to a given observation and narrative and, also, introduces the new type of knowledge-producing actions as mapping a fluent expression to a knowledge-producing action. The set of cancellation axioms Θ is used to provide a solution to the qualification problem.

At this point it is appropriate to consider more carefully the behavior and syntactic form of sensing actions and expressions. We have already mentioned that a sense action affects only the agent's state of knowledge. In particular, we define a sensing action as an action that causes no changes in the state of the world, but only affects epistemic fluents. Of course, in a real world system a sensing action may indeed have side-effects on the environment, thus, one may argue that it is reasonable to allow for sensing actions to produce changes in the state of ordinary fluents, especially when sensing is interpreted as communicating information between agents. Still, we can get around these situations by properly handling side-effects as preconditions of sensing actions, requiring non-knowledge-producing actions to establish the appropriate conditions for the sense action to occur.

Furthermore, we need to carefully restrict the permissible syntactic form of the fluent expressions that a *sense* action may consider. Specifically, for the formal specification of sense actions, we require that they concern expressions without free variables that contain

only ordinary relational fluent atoms, and not the *Knows* fluent. In other words, an agent can sense sentences about the external world, but not truths about its own knowledge, as also pointed out in [Reiter 2001b].

Definition 4.3 (Epistemic domain description) *Given an axiomatization D, an Event Calculus* epistemic domain description Φ *is constructed as follows*

$$CIRC[\Sigma \wedge \Xi_1; Initiates, Terminates, Releases] \wedge$$
$$CIRC[\Delta_1 \wedge \Delta_2 \wedge \Xi_3; Happens] \wedge$$
$$CIRC[\Theta; Ab_1, ..., Ab_n] \wedge E \wedge M \wedge \Xi_2 \wedge \Omega \wedge \Psi \wedge \Gamma_1 \wedge \Gamma_2$$

where

1. Δ_1 is the narrative, consisting of a conjunction of
   - event occurrence formulae, i.e., $Happens(e, t)$ and
   - temporal ordering formulae, i.e., conjunction of timepoint comparisons, such as $t_1 \leq t_2$, $t_1 = t_2$, $t_1 \neq t_2$ etc.
2. Γ_1 and Γ_2 are the observations, consisting of a conjunction of formulae of the form $HoldsAt(f, t)$ or $ReleasedAt(f, t)$. Γ_1 is the set of (objective) sentences describing the state of the world being axiomatized, while Γ_2 is the set of (subjective) sentences describing the state of knowledge of the agent. □

That is, the domain description is given by the parallel circumscription (denoted as *CIRC*) of the basic predicates, along with the observations of world properties at various times. More specifically, fluent expressions in Γ_2 consist of sentences of the form $HoldsAt(Knows(f'), t)$ or $HoldsAt(KP(f'), t)$, where f' are objective fluent terms. The Γ_2 database consists exclusively of sentences declaring what the agent knows about the world it inhabits, but there are no sentences declaring what it knows about what it knows and so forth. In a real-world implementation, the set of observations may begin with subjective sentences only, and Γ_1 will be populated by the logical consequences of Γ_2.

4.5 Summary

Table 4.1 below summarizes the full DECKT axiomatization for commonsense reasoning in dynamic and uncertain domains with actions having context-dependent effects. Axioms (KT1,2,4,7) introduce the necessary epistemic notions for supporting reasoning about

knowledge, while the rest extend a domain theory with a set of meta-axioms for distinguishing what is true in the world and what an agent knows to be true. In particular, the (KT3) and (KT5) sets concentrate on epistemic inferences that can be made about the effect, considering the available knowledge an agent has about the context. The (KT6) set axiomatizes the life-cycle of HCDs, i.e., which conditions justify their creation and expiration and what knowledge must be preserved when actions affect fluents involved in an HCD.

In a nutshell, the procedure of reasoning about knowledge evolves as follows (see also the examples given in Chapter 7): for any deterministic action with no preconditions or when the agent knows all preconditions, (KT3.1) and (KT3.2) or (KT3.3) and (KT3.4) are triggered and knowledge persists until some other event changes it. If the agent happens not to know at least one precondition while, at the same time, there is no precondition that it knows it does not hold, then it cannot be certain about the action's effects, the action is considered non-deterministic and (KT5.1) or (KT5.2) are triggered. At the same time, a causal dependency is create between the unknown preconditions and the effect, according to (KT6.1) axioms. This dependency may interact with other similar relations facilitating epistemic derivations and resulting potentially in definite knowledge about other fluents (see Example 7.1.2 about Shanahan's circuit as an illustrative use case). The dependency is valid until some event directly changes any of the involved fluents through (KT6.2.1), still leading to the creation of new dependencies that may either be reduced (KT6.2.2), expanded (KT6.2.3) or interlaced due to their transitive nature (KT6.2.3). Finally, for ordinary non-deterministic actions, axiom (KT5.3) is applied.

When deterministic effects occur, the KP fluent determines the agent's state of knowledge by means of the (KT2) axiom. In addition, knowledge can also be inferred using state constraints. If, however, knowledge about a fluent formula can neither be directly established nor indirectly, then axiom (KT7) that defines the necessary and sufficient condition for knowledge to hold, dictates the agent's lack of knowledge about this formula. Any change in the agent's KB may cause knowledge to be gained. In subsequent chapters we also investigate extensions of the theory to accommodate the potential triggering of physical actions and the ability for an agent to sense continuously changing world aspects, leading to the construction of a framework that enables the creation of agents that can not only remember but also forget.

Table 4.1: DECKT Axiomatization Overview

Name	Axiom	Description
Knowledge and Inertia	(KT1)	Knowledge is always released (but constrained).
Knowledge Persistence	(KT2)	Knowledge about inertial world aspects is persistent.
Knowledge Minimization	(KT7)	A fluent is known if and only if there is direct or indirect evidence to justify this knowledge.
Knowledge Producing Actions	(KT4)	Sensing Inertial Parameters.
Meta-axioms		
Known Preconditions	(KT3.1) (KT3.2)	Positive Effect Axioms
	(KT3.3) (KT3.4)	Negative Effect Axioms
Hidden Causal Dependencies		
Creation	(KT6.1.1) (KT6.1.2)	Positive Effect Axioms
	(KT6.1.3) (KT6.1.4)	Negative Effect Axioms
	(KT6.1.5)	Release Axioms
Termination	(KT6.2.1)	A fluent of a HCD is/may be altered.
Preservation	(KT6.2.2)	A fluent of a HCD, known not to contribute, is/may be altered.
Expansion	(KT6.2.3)	A fluent of a HCD may be altered by an event, whose preconditions are unknown at t, and it is not inverted at (t+1).
Transitivity	(KT6.2.4)	A fluent of transitive HCDs is/may be altered.

CHAPTER 5

Property Analysis

Contents

The knowledge theory we developed in the previous chapter adopts an alternative representation for knowledge, dismissing completely the use of the accessibility relation over possible worlds. Although the latter approach enables the creation of highly expressive epistemic frameworks, it suffers serious computational inefficiencies raising concerns about the applicability in real-world implementations. In this chapter we study the correspondence of our theory with the standard possible worlds semantics and prove correctness results. In order to show that the fluent formulae known are the same in both approaches after any sequence of actions, we construct a knowledge theory for the Event Calculus that is based on the possible worlds semantics and provide the means to achieve logical equivalence.

An evaluation of the two epistemic approaches is also provided in terms of computational complexity. Since the main objective of DECKT has been to enable epistemic reasoning in a way that resembles ordinary non-epistemic reasoning, we present in this chapter both a complexity analysis of different fragments of the Event Calculus, as well as an elaboration of the worst and the more realistic situation that specify the computational effort of model checking tasks. Finally, decidability issues of first-order action theories in general are discussed. This chapter intends among others to enable the reader to draw a clearer picture on the issues that DECKT and other similar theories need to face in order to be implemented in practice.

5.1 A Possible Worlds-based Theory for the Event Calculus

To prove correctness of the DECKT treatment of knowledge in terms of standard possible worlds semantics we need to show its correspondence to a theory that utilizes the latter approach. Yet, no such theory existed for the Event Calculus. Moreover, the classical Event Calculus is based on a linear time representation, where parallel worlds cannot be imitated. Therefore, in the following sections we construct a knowledge theory that is based on the recently developed branching discrete Event Calculus (BDEC) [Mueller 2007b] and study equivalence with DECKT.

5.1.1 Branching Discrete Event Calculus

The branching discrete Event Calculus (BDEC) is a modified version of the linear discrete Event Calculus (LDEC); in fact, the two formalisms are proved logically equivalent when appropriately restricted, as we see in Section 5.2. BDEC replaces the timepoint sort with the sort of situations, lifting the requirement that every situation must have a unique successor state. This way, it maintains all properties of the classical Event Calculus, such as the law of inertia and the ability to reason about a multitude of commonsense phenomena of action and change, but also gains advantages due to the branching time structure, supporting, for instance, hypothetical reasoning.

The transition from LDEC to BDEC is done by adding an extra argument to predicates *Happens*, *Releases*, *Initiates* and *Terminates* specifying the successor situation and also by introducing the relation $S(s_1, s_2)$ to express that situation s_2 is a successor of s_1. We

take S_0 to denote the initial situation. BDEC's axiomatization shares the same set of axioms with LDEC (see Section 4.1.2), with the addition of the notion of situations and a second order induction axiom similar to the one for the Situation Calculus [Reiter 1993]. Specifically, in addition to the 8 original axioms, where the timepoint sort is replaced by situations, the new formalism's axiomatization, also includes:

(BDEC1) $\neg S(s, S_0)$
(BDEC2) $S(s_1, s) \wedge S(s_2, s) \Rightarrow s_1 = s_2$
(BDEC3) $\forall P((P(S_0) \wedge \forall s_1, s_2(S(s_1, s_2) \wedge P(s_1) \Rightarrow P(s_2))) \Rightarrow \forall s P(s))$
(BDEC12) $Happens_B(e, s_1, s_2) \Rightarrow S(s_1, s_2)$

The first three are generalized Peano axioms, allowing each situation to have one or more successors. The remaining (BDEC4-11) axioms are trivially formulated from LDEC; for instance axiom (DEC5) is transformed into:

(BDEC4) $S(s_1, s_2) \wedge HoldsAt(f, s_1) \wedge \neg RealeasedAt_B(f, s_2) \wedge$
$\neg \exists e(Happens_B(e, s_1, s_2) \wedge Terminates_B(e, f, s_1, s_2)) \Rightarrow$
$HoldsAt(f, s_2)$

5.1.2 Branching Time Event Calculus Knowledge Theory

The Branching Event Calculus Knowledge Theory (BDECKT) that we develop follows on from Moore's [Moore 1985] formalization of possible world semantics in action theories, where the number of K-accessible worlds remains unchanged upon ordinary event occurrences and reduces as appropriate when sense actions occur. Similar to Scherl and Levesque's approach for the Situation Calculus [Scherl 2003], BDECKT generalizes BDEC in that there is no single initial situation in the tree of alternative situations, but rather a forest of trees each with its own initial situation. Note, also, that for reasons that will become clear in the next section, we dismiss non-deterministic effects and only allow for events with unknown preconditions to occur; although release effect axioms are still included, we implicitly assume that whenever a known fluent becomes released it remains known, subject to some state constraint.

To axiomatize knowledge in BDECKT we introduce the predicate $K(s', s)$ denoting that world (or situation) s' is accessible from s. The theory proceeds as follows:

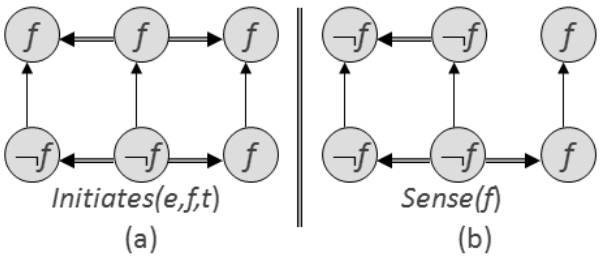

Figure 5.1: Worlds accessible by the successor situation after (a) ordinary or (b) sense actions.

Knowledge Definition

(BKT1) $HoldsAt(Knows_B(f), s) \equiv$
$\forall s' K(s', s) \Rightarrow HoldsAt(f, s')$

(BKT2) $K(s'_1, s_1) \Rightarrow$
$\exists s_2, s'_2(Happens_B(e, s_1, s_2) \Leftrightarrow Happens_B(e, s'_1, s'_2))$

(BKT3) $Initiates_B(e, f, s_1, s_2) \Leftrightarrow Initiates_B(e, f, s'_1, s'_2)$

(BKT4) $Terminates_B(e, f, s_1, s_2) \Leftrightarrow Terminates_B(e, f, s'_1, s'_2)$

(BKT1) is the standard definition for knowledge according to the possible worlds semantics; a fluent is known iff it has the same truth value in all worlds considered possible. Axiom (BKT2) states that the same event happens in all K-related situations, while axioms (BKT3) and (BKT4) require for an event to have the same effect, regardless of the situation it occurs in.

Ordinary events

When an ordinary event occurs in a situation, then all successor situations of the K-related to it situations (and only them) are K-related to its successor (see Figure 5.1(a), where vertical arrows denote successor situations after event occurrences, while horizontal double-lined arrows denote situations accessible by the originating situation).

(BKT5) $Happens_B(e, s_1, s_2) \Rightarrow$
$(K(s'_2, s_2) \Leftrightarrow \exists s'_1(S(s'_1, s'_2) \wedge K(s'_1, s_1)))$

Knowledge-producing (sense) events

The action of sensing a fluent ensures that it will be known in the successor situation, i.e., it will have the same truth value in all possible worlds (Figure 5.1(b)).

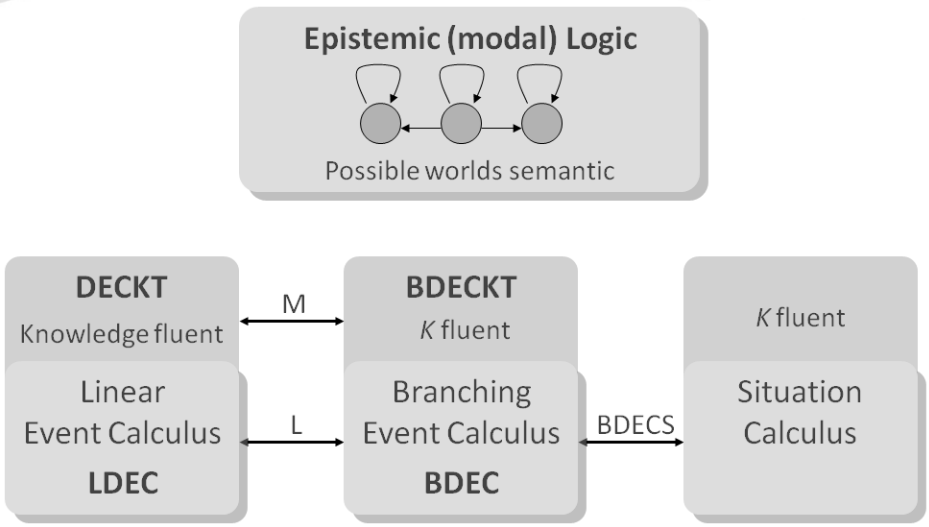

Figure 5.2: Relation between the action formalisms and their epistemic extensions under different bridging sets of axioms.

(BKT6) $Happens_B(sense(f), s_1, s_2) \Rightarrow$
$(K(s'_2, s_2) \Leftrightarrow \exists s'_1(S(s'_1, s'_2) \wedge K(s'_1, s_1) \wedge (HoldsAt(f, s_1) \Leftrightarrow HoldsAt(f, s'_1))))$

5.2 Correctness Properties

The DECKT axiomatization is based on the linear Event Calculus that treats knowledge as a fluent and uses a set of axioms to determine the way this epistemic fluent changes its truth value as a result of event occurrences and the knowledge already obtained about relevant context. BDECKT on the other hand is based on a branching time version of the Event Calculus where knowledge is understood as reasoning about the accessibility relation over possible situations. Mueller has established a set L of mapping rules between the underlying linear and branching versions of the Event Calculus that we use in our knowledge theories and proved that these two versions can be logically equivalent [Mueller 2007b]. Our objective in this section is to show that knowledge evolves the same way in both DECKT and BDECKT, i.e., the set of known formulae is the same after a sequence of actions.

The set L, defined by Mueller, comprises the following axioms, where $S_L(s_1)$ denotes the successor situation of s_1 in LDEC:

(L1) $S(s_1, s_2) \Leftrightarrow S_L(s_1) = s_2$

(L2) $Happens_B(e, s_1, s_2) \Leftrightarrow (Happens(e, s_1) \wedge S_L(s_1) = s_2)$
(L3) $Initiates_B(e, f, s_1, s_2) \Leftrightarrow Initiates(e, f, s_1)$
(L4) $Terminates_B(e, f, s_1, s_2) \Leftrightarrow Terminates(e, f, s_1)$
(L5) $Releases_B(e, f, s_1, s_2) \Leftrightarrow Releases(e, f, s_1)$

This set restricts BDEC to a linear past, i.e., only one branch at a time can have an equivalent linear axiomatization, which introduces an important limitation to our attempt to relate DECKT and BDECKT; non-determinism due to release effect axioms cannot be supported, as it requires new worlds to emerge from a single one. The result is that (BDEC $\wedge$ L) is logically equivalent to (LDEC $\wedge$ L) (Figure 5.2). Based on this corollary established by Mueller, our intension now is to show that the two knowledge theories manipulate knowledge change the same way (in contrast to the underlying theories, whose equivalence is based on an one-to-one mapping of all their axioms). In particular, we concentrate on proving logical equivalence between axiom sets (KT3,4,5,6) and (BKT5,6) and define a set M that serves as a bridge between DECKT and BDECKT for that purpose:

(M1) $HoldsAt(Knows(f), s) \Leftrightarrow HoldsAt(Knows_B(f), s)$
(M2) $S_L(s_1) = s_2 \Rightarrow (K(s'_2, s_2) \Rightarrow \exists s'_1(S(s'_1, s'_2) \wedge K(s'_1, s_1)))$
(NR) If $\vdash HoldsAt(\phi, t)$, then $\vdash HoldsAt(Knows(\phi), t)$

Axiom (M2) relates event occurrences in DECKT with BDECKT's accessibility relation. Specifically, it disallows a world to be K-related to worlds other than those whose precedents were K-related to its own precedent (apart from the initial state, of course, which has no precedent). In other words, it prohibits unexpected world appearances and averts worlds to be accessibly related to others that belong to future or past situations (Figure 5.3)[1]. The necessitation rule (NR) is produced as a side-effect in BDECKT, due to the definition of knowledge, therefore we need to explicitly include it in the M set in order to accomplish equivalence.

We can now prove that the conjunction of DECKT, BDEC, LDEC, L and M can provide all BDECKT epistemic derivations leading to completeness with respect to the possible worlds semantics and respectively, the conjunction of BDECKT, BDEC, LDEC, L and M can provide all DECKT epistemic derivations resulting in soundness of DECKT inferences

[1]The inverse of (M2) is not necessarily true, as it would require for all worlds (s_1) that are K-related to the same world (s'_1) to have the same successor (s_2)

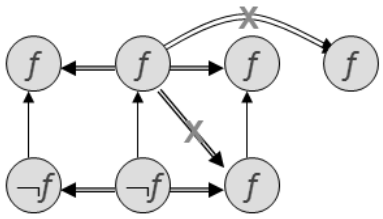

Figure 5.3: Axiom M2 constraints the accessibility relation among situations permissible by Moore's formulation of possible worlds in action theories.

(Figure 5.2). For our equivalence results we will use a number of lemmas. The proofs for the lemmas and theorems that appear in this section are provided in Appendix A.1.

Lemma 1. $DECKT \wedge (BKT1-4) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (BKT5)$
Lemma 2. $DECKT \wedge (BKT1-4) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (BKT6)$

Using Lemmas 1 and 2 we can conclude in the following theorem:
Theorem 1. (Completeness) *The DECKT axiomatization produces all BDECKT epistemic derivations.*

Lemma 3. $BDECKT \wedge (KT1,2,7) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (KT3.1) \wedge (KT3.2) \wedge (KT3.3) \wedge (KT3.4)$
Lemma 4. $BDECKT \wedge (KT1,2,7) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (KT4)$
Lemma 5. $BDECKT \wedge (KT1,2,7) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (KT5.1) \wedge (KT5.2)$
Lemma 6. $BDECKT \wedge (KT1,2,7) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (KT6)$

Using Lemmas 3 to 6 we can conclude in the following theorem:
Theorem 2. (Soundness) *All epistemic derivations produced by the DECKT axiomatization are also produced by BDECKT.*

As a result, from Theorems 1 and 2 we have established the following corollary
Corollary 1 *After any ground sequence of actions with deterministic effects but with potentially unknown preconditions, a fluent formula ϕ is known whether it holds in DECKT if and only if it is known whether it holds in BDECKT, under the bridging set of axioms L and M.*

5.3 Complexity Analysis

One characteristic feature of the possible worlds approach is that by reducing the number of worlds it obtains information about fluents and their temporal relations (see for example Fig. 4.5). The main disadvantage, though, is the high computational costs, as it requires a vast number of fluents to be stored and one reasoning task to be performed for each world after every action. DECKT provides the same information in an alternative way significantly reducing computational effort: instead of inferring the relations among fluents from the number of worlds that remain after an action, the DECKT axiomatization follows the opposite direction by applying a provably correct and complete way to create, maintain and destroy the relations, progressing a single world. The efficiency of manipulating fluent dependencies explicitly rather than by means of preserving multiple world instances stems from the fact that these dependencies are modeled as ordinary fluents and fall under the influence of classic Event Calculus reasoning.

In this section we provide an analysis of the computational complexity of DECKT for the process of deriving knowledge formulae and compare it with the standard possible worlds specifications. A consideration of the worst case scenario, as well as of the more realistic situation is provided. In addition, since knowledge within DECKT is treated as an ordinary fluent, we also elaborate on the different complexity classes of the Event Calculus for general model checking tasks.

5.3.1 On Event Calculus Query Processing

So far we have shown that DECKT can be equivalent in the treatment of knowledge with possible worlds-based theories; we now show that it is also more suitable for practical implementations. The objective is to study the complexity of checking whether a fluent or fluent formula holds after the occurrence of an event sequence in total chronological order, given a domain theory comprising a fixed set of context-dependent effect axioms and a set of implication rules (either in the form of state constraints or in the form of HCDs).

We can split the reasoning process for each action into two sub-processes that take place in two successive stages (Fig. 5.4); first, direct action effects to fluents, specified by the effect axioms, are determined and then inferences based on the implication rules are derived. During the first stage, changes to the truth value of fluents refer to the successor

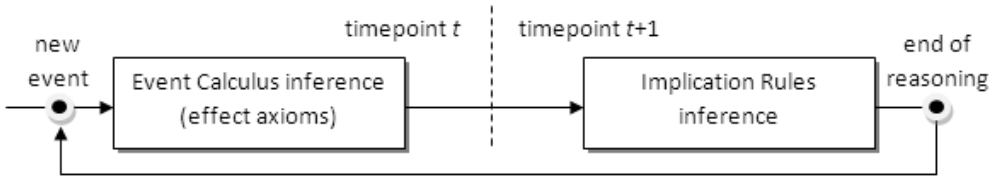

Figure 5.4: Reasoning process with the Event Calculus

timepoint exclusively, therefore cannot affect the preconditions of other effect axioms (no cycling triggering). Implication rules, on the other hand, implement classical logic inference. For each of the sub-processes worst case complexity is given by the complexity of checking whether $P \cup D_{in} \models F$, where P denotes the program (effect axioms or implication rules, respectively), D_{in} the input database (truth values and inertia state of fluents, as well as occurring events, i.e., $HoldsAt()$, $ReleasedAt()$ and $Happens()$ predicates, respectively) and F a ground set of fluents that constitute the query that needs to be answered in order to progress reasoning to the next stage. In particular, for all timepoints before the final we are only interested in atomic fluents that may act as preconditions. For the final timepoint we could instead seek for deriving whether a formula of ground atoms holds.

Regarding the second inference stage, we note that in order to perform all derivations from the set of rules, we can apply standard techniques from classical logic inference, such as resolution. To preserve generality of results, we denote the complexity of this stage as $O(INF^{SC})$ or $O(INF^{HCD})$, based on whether the rules that form program P involve only the domain state constraints or both state constraints and HCDs, respectively. We revert to this complexity in Section 5.3.5 below.

Notice that we concentrate our complexity analysis on a reasoning task that is commonly met in implementations of epistemic frameworks. In particular, we study update time reasoning for processing events as they arrive, evaluating operational conditions of cognitive agents that acquire information and react to changes at execution time. A more general complexity analysis of reasoning with the Event Calculus under different classes of expressiveness is provided in Section 5.3.6, where both query time and update time reasoning is considered, the narrative of events is not totally ordered and apart from timepoints, also the determination of time intervals is required. Our primary objective here is to compare the two basic epistemic approaches, namely possible worlds and DECKT, and reveal their main performance factors under a set of generic assumptions.

5.3.2 Classic Event Calculus Without Knowledge

We attempt a worst-case complexity analysis of reasoning with the Event Calculus following a simple, yet complete, consideration of the steps involved; typically the complexity can be significantly reduced in practice applying dedicated structures, such as caching mechanisms or indexes, still we are primarily interested here in creating a reference point for comparing the two epistemic approaches, as already mentioned. Assuming that all *HoldsAt*() predicates are stored as facts in the database that needs to be progressed or queried, the size of D_{in} for a domain of n fluents is $O(n)$ (similarly for the *ReleasedAt*() predicates). Whenever a fluent becomes released but not constrained by some state constraint, two or more different KBs need to be maintained to preserve complementary truth values (the same holds true for disjunctive fluent expressions). Thus, when non-inertial fluents are involved, in the worse case 2^n KBs need to be progressed individually, resulting in $O(n * 2^{n-1})$ as the total data size that needs to be stored (a fluent may hold only in half of the 2^n KBs). The query answering process for the Event Calculus can progress based on the following algorithm:

Algorithm 5.1: For each event e_i that occurs at t_i and for each individual KB

1. *Retrieve all effect axioms for* e_i: $\bigwedge^j[HoldsAt(f_j, t)] \Rightarrow \pi(e_i, f', t)$?, where π = *Initiates* or $Terminates$ or *Releases*

   The objective is to determine which preconditions to search for in step 2. This information is known at design time and can be modeled as a predetermined set for each action. Therefore we assume constant time, regardless of the type of action, the number of effect axioms or the size of the domain.

2. *Query the KB for the truth value of the precondition fluents of* e_i: $\bigwedge^j[HoldsAt(f_j, t_i)]$*?*

   The intension is to determine which axioms are going to be triggered, i.e., which effect fluents will change their truth value. The problem of query answering on (untyped) ground facts (without rules) reduces to the problem of unifying the query with the facts, which is $O(n)$, where n is the size of the KB. For instance, Datalog has been proved to be P-compete to the size of D_{in}, if we keep P fixed (for a variable logic program P, Datalog is DEXPTIME) [Immerman 1986]. Thus, this step takes linear time.

3. *Determine which fluents are inertial: $\neg Released(f,t)$?*
   Inertial fluents that have not been affected by e_i in step 2, i.e., they are neither released nor the event releases them, need to maintain their truth value in the successor timepoint. As before, the cost of the query is O(n), given that each fluent may either be released or not. Notice that a fluent may become released in a context-dependent manner, therefore all KBs need to be queried.

4. *Assert in the KB the new truth values of fluents.*
   As the new truth values refer to the successor timepoint, this step does not involve any update on existing facts, rather an assertion of facts to an empty KB. Therefore we assume that it does not contribute to the general complexity of reasoning with the Event Calculus, rather it takes constant time, regardless of the number of assertions, as it refers to the entire domain of n fluents.

5. *Use state constraints to infer all indirect effects and new models.*
   This step is characterized by the complexity of $SC \cup D_{in} \models F$, i.e., $O(INF^{SC})$, with the objective of determining the truth value of those atomic fluents that are released from the law of inertia (assume r in total), still are ruled by state constraints . D_{in} here refers to the $(n-r)$ fluents that are subject to the law of inertia (those that have been affected by e_i, as well as those that are transferred unaltered to the next timepoint due to inertia). Multiple models may be produced owed to the unconstrained released fluents, i.e., non-inertial fluents that are subject to no state constraint, and are added to the existing set of KBs.

Summarizing, the complexity of reasoning with the Event Calculus given a sequence of e actions is characterized by $O(e * 2^n * (2 * n + INF^{SC}))$[2]. The fact that query answering has linear complexity to the number of events has also been proved in [Paschke 2006], for a restricted class of domains that does not involve the releasing of fluents from the law of inertia. Following the intuition of Algorithm 5.1 it becomes trivial to reformulate the

[2]More accurately, as we have mention before, when all n fluents are released the overall size of the 2^n KBs (thus, the total number of accesses to them) will be $n * 2^{n-1}$ (each fluent will be true in half of the KBs). As a result, the upper bound for the complexity of progressing a potentially non-deterministic domain of n fluents given a sequence of e actions in chronological order cannot exceed $O(e*(n*2^{n-1}+n*2^n+2^n*INF^{SC}))$. There are further similar optimizations that can be identified in Algorithm 5.1, still we are mostly concerned here in the comparison of the two epistemic approaches, rather than the accurate determination of the complexity of the classical Event Calculus.

complexity when no state constraints (step 5) or non-determinism (step 3) are employed in the domain axiomatization. For the latter case in particular, no exponential number of KBs need to be maintained, as there will always be a single successor model after each action.

5.3.3 Possible Worlds Approach

According to the possible-worlds specifications, the accessibility relation K is used to determine which worlds can be considered possible from a given situation. Based on the properties of this relation, knowledge of a formula is understood as the formula having the same truth value in all possible worlds (axiom (BKT1)). Apart from the potentially different truth values of fluents, all worlds are considered to be identical from the standpoint of the reasoning agent when they are employed to model a particular domain, meaning that a state constraint is valid in all worlds, an action has always the same context-dependent effects etc. In general, this definition of knowledge introduces an exponential number of worlds in which to check truth of a formula, in the number of fluents. More precisely, the following proposition draws the detailed picture.

Proposition 1. (Model checking of possible worlds) *For a domain with n fluents, checking whether a conjunction of m of them is known requires, at worse,* 2^{n-m} *worlds to consider if the conjunction turns out to be known and* $2^{n-m} + 1$ *worlds if it turns out to be unknown.*
Checking whether a disjunction of m fluents is known requires, at worse, $2^n - 2^{n-m}$ *worlds to consider if the disjunction turns out to be known and* $2^n - 2^{n-m} + 1$ *worlds if it turns out to be unknown.*
For a domain of n fluents, determining all known formulae requires, at worse, 2^n *worlds to consider.* □

The number of possible worlds depends on the number of unknown fluents, i.e., in a domain of n fluents, u of which being unknown, we need to store at worse 2^u possible worlds, where $u \leq n$. One reasoning task needs to be performed for each of these worlds, since the same effect axioms of a given domain theory may give rise to different conclusions in each world. As such, the total size of the KBs of fluents that needs to be maintained at each timepoint is $O(u * 2^{u-1} + (n-u) * 2^u)$ (in the worst case, an unknown fluent may hold at most in half of the total 2^u worlds, whereas known fluents hold in all of them) or equally $O((2 * n - u) * 2^{u-1})$.

Domain without state constraints and non-deterministic events: Steps 3 and 5 of

Algorithm 5.1 can be disregarded as all fluents are inertial. Given a sequence of e actions, the complexity of query answering whether a conjunctive (resp. disjunctive) formula of m fluents is known (m being among the initially unknown fluents, $m \leq u$) is $O(e * 2^u * n + 2^{u-m} * n)$ (resp. $O(e * 2^u * n + (2^u - 2^{u-m}) * n)$), as we first need to progress all possible worlds and then issue a query with cost $O(n)$ to a subset of them[3].

Domain with state constraints: The logical inference using as D_{in} the domain fluents and P the fixed set of state constraints needs to be performed for each possible world. Thus, following the same analysis as before, the complexity for conjunctive queries is $O(e * 2^u * (n + INF^{SC}) + 2^{u-m} * n)$ (resp. $O(e * 2^u * (n + INF^{SC}) + (2^u - 2^{u-m}) * n)$ for disjunctive queries). Notice that even in this simple case where no releasing of fluents may occur an exponential number of KBs need to be progressed (the same holds true for the previous case, as well). Instead, the underlying non-epistemic Event Calculus without the releasing of fluents would have complexity $O(e * (2 * n + INF^{SC}))$ as only a single KB would have to be maintained.

Non-deterministic domains: Now step 3 needs to be considered and the complexity of progression -without the query- becomes $O(e * 2^u * (2 * n + INF^{SC}))$. Additionally, notice that each fluent that becomes released from the law of inertia causes the number of possible worlds to double, i.e., u increases by one. As a result, both the size of the KB and the reasoning effort increase significantly (in the particular situation where the releasing is context-dependent, then only the worlds where the preconditions hold will be duplicated, and not the complete set of worlds). Furthermore, as we argue in the sequel, one should expect that $u \simeq n$ even in the real-world case.

5.3.4 DECKT Approach

DECKT performs one reasoning task using the new axiomatization's effect axioms and replaces every atomic domain fluent with the corresponding KP and $Knows$ epistemic fluents. The former are always subject to inertia and the latter are always released, therefore step 3 can be disregarded altogether, along with the need to preserve multiple versions of

[3]Notice that in this case the number of unknown parameters u either remains constant or decreases. It would be more accurate to capture the change in possible worlds as follows: for the sequence of actions $e_1, ..., e_i$ the progression of KBs has cost $O(n * \sum_{j=1}^{i+1} 2^{u_j})$, where u_j denotes the number of unknown fluents before action e_j. To facilitate readability, we introduce the exponential worst case in our results, which after all acts as an upper bound.

KBs for released fluents (the *Knows* fluents never fluctuate). Furthermore, disjunctive epistemic expressions are preserved in this single KB without the need to be broken apart, since all appropriate derivations are reached by means of HCDs. The size of D_{in} for the first sub-process (step 2) is equal to that of reasoning without knowledge, as we only search through those *Knows* fluents that refer to atomic domain fluents. The difference now is that each domain effect axiom is replaced by 5 new ones: 2 due to (KT3), 1 due to (KT5) and 2 due to (KT6.1). Nevertheless, as with the non-epistemic theory, all we need to query in order to progress the KB after an action has occurred are the precondition fluents (plus the effect fluent for some of the axioms). Therefore, as before, the complexity of that step is $O(n)$, since a single predefined query to a domain of n fluents suffices to provide the necessary knowledge for all DECKT's effect axioms mentioned above.

Apart from the epistemic effect axioms, DECKT also introduces axioms for handling HCDs (KT6.2-4). Since HCDs are treated as ordinary inertial fluents (they are modeled in terms of the KP fluent), they fall under the influence of traditional Event Calculus inference. For these axioms the necessary knowledge that needs to be obtained is whether some HCD that incorporates the effect fluent is among the HCDs stored in the KB, whose size increases as the agent performs actions with unknown preconditions. Let d denote the number of KP fluents that represent HCDs, then the complexity of querying the KB is $O(d)$, where $d \leq 2^n$.

Domain without state constraints and non-deterministic events: HCDs are used to infer additional knowledge about interrelated fluents and therefore it is possible to produce knowledge about atomic fluents as well, if combined (see Ex. 4.9). Still, whenever state constraints are not supported by the theory, the axiomatization allows the HCD-based inference task to be executed only at the timepoint when a query is placed, i.e., after the sequence of actions. This holds true because axiom (KT6.2.2) can be disregarded, as any fluent within HCDs will stay unknown and only direct effects may change its value. Consequently, we can transfer incomplete results in favor of efficiency (more HCDs without considering their correlation) and only study their inference at query time.

Consequently, the complexity of reasoning with DECKT without state constraints is $O(e * (n + d) + INF^{HCD})$, where $O(INF^{HCD})$ is the complexity of logical inference with P being the program of HCDs, i.e., $HCD \cup D_{in} \models F$. The input is the atomic inertial fluents as usual, reified in the *Knows* fluent. Notice that the final inference task can refer only to the desirable query, therefore the above complexity is also the complexity of query answering;

no extra query to the KB is needed.

Domain with state constraints: When state constraints are incorporated, axiom (KT6.2.2) cannot be disregarded as before. Due to its dependence on indirect knowledge effects, the inference task cannot be transferred to the end of the reasoning process, instead it needs to be executed after every single action. Moreover, it can no longer refer to the query formula only, but rather to all atomic fluents as they may play the role of preconditions for the successor action. As a result, at the end of the reasoning process we need to issue a query to the KB of atomic fluents. The complexity of query answering with state constraints is $O(e * (n + d + INF^{HCD}) + n)$.

We should also remark that the complexity $O(INF^{HCD})$ is related to the number and length of HCDs, as we explain later on, that may change from timepoint to timepoint as events create new HCDs and destroy others. For matters of clarity, instead of introducing the sum of each different inference task in the previous complexity result for a sequence of e actions, we implicitly take the most demanding complexity, which would dominate the general complexity, whatsoever.

Non-deterministic domains: The treatment of non-determinism introduces no additional complexity to the axiomatization. The KP fluent is always subject to inertia and whenever a domain fluent is released, its corresponding KP fluents become false according to axiom (KT3.5). This also means that even when the domain requires reasoning with the full Event Calculus axiomatization that incorporates the *Releases* predicate, epistemic reasoning with DECKT still requires the reduced version without the releasing of fluents. This computational profit is also reflected on the complexity results between DECKT and the non-epistemic Event Calculus, as no exponential number of KBs are maintained.

5.3.5 Discussion of Results

We can see that the dominant factor in the complexity of reasoning with possible worlds is the number u of unknown world aspects. In the worse case $u = n$ resulting in exponential complexity to the size of the domain; unfortunately, even in real-world implementations $u \simeq n$, as we should expect that in large-scale dynamic domains much more world aspects would be unknown to the reasoning agent than known at any given time instant. Furthermore, since in practice the query formula that needs to be evaluated is often orders of

magnitude smaller in size than the domain itself [Cervesato 2000], i.e., ($n \gg m$), query answering of either conjunctive or disjunctive formulae is impractical (see Proposition 1).

With DECKT, on the other hand, it is the number of extra fluents capturing HCDs that dominates the complexity. In fact, although at worst case it can be that $d = 2^n$ this is a less likely contingency to expect in practice: it would mean that the agent has interacted with all world aspects having no knowledge about all preconditions or that state constraints that capture interrelated fluents embody the entire domain (so called *dominos domains* [Son 2005] which lead to chaotic environments are not commonly met in practice). Moreover, HCDs fall under the agent's control; even for long-lived agents that perform hundreds of actions, the HCDs constitute a guide as to which sense actions can provide knowledge about the largest set of interrelated fluents, thus enabling the agent to manage the size of HCDs according to available resources.

Apparently, the number and length of HCDs also affects the inference task. Still, the transition from $O(INF^{SC})$ to $O(INF^{HCD})$ has polynomial cost; the complexity of most inference procedures, such as resolution, is linearly affected when increasing the number of implication rules, given that the size of the domain remains constant (in Appendix A.3 we construct a custom algorithm to substantiates this result). Finally, one should notice that even in the worst case one reasoning task needs to be performed for each action. Specifically, the dominant factor d does not influence the entire reasoning process, as is the case of 2^u possible worlds, significantly reducing the overall complexity. This stems from the fact that only a single KB -with potentially more fluents- is progressed for each action, rather than a variable number of KBs that are required when reasoning with possible worlds.

To summarize, from the computational standpoint the previous analysis creates a flavor of how demanding model checking of knowledge formulae can be using the possible worlds specifications and the DECKT theory:

- In the worst case, possible worlds require an exponential number of reasoning tasks in the size of the domain (n) applying the full Event Calculus, whereas DECKT performs a single reasoning task on exponentially many fluents applying a reduced version of the Event Calculus that does not involve the *Releases*() predicate, thus permitting only a single KB to be maintained at all timepoints.

- The main source of computational effort in possible worlds resides in lack of knowledge; the most demanding task is to check formulae that turn out to be known (since all worlds need to be evaluated) in a relatively unknown environment. Unluckily, this is also the most common case.

  DECKT, on the other hand, is mainly affected by state constraints and HCDs. The most demanding task is to check the truth value of a query when all fluents of a domain turn out to be interrelated. Still, this is not the common case in real-world domains, where in order to answer a query we might have to check $(m + k)$ atomic fluents, where m is the length of the query, k the fluents involved in related state constraints with $(n \gg m, k)$.

  Even the number of HCDs cannot grow indefinitely. In Appendix A.4 we show that the number of distinct state constraints, i.e., the constraints that do not interact with each other to produce explicit truth about the involved fluents, is bounded by a number that is exponential in the worse case to the size of the domain.

- Based on our assumptions about realistic domains, model checking with DECKT can exhibit relatively low complexity, proportional to that of reasoning without epistemic notions, still maintaining full expressive power of a knowledge theory. Most alternative approaches, on the other hand, restrict expressiveness and require that knowledge about disjunctions be decomposed into the individual components. Yet, even if the theory were restricted to domains where all decompositions are sound, it can deal with much more expressive commonsense phenomena compared to alternative approaches, such as non-deterministic effects and trigger events, non-context-complete theories and can also model time-dependent, potentially delayed, knowledge effects.

5.3.6 General Complexity Results for the Event Calculus

It has already become clear that a substantial computational effort when reasoning with DECKT is owed to the expressiveness of the underlying Event Calculus formalism. Considering the multitude of different dialects and extensions of the Event Calculus that have been proposed in the past we briefly review in this subsection relevant results keeping track of the different variants. In contrast to the aforementioned analysis, these studies assume the number of fluents constant and focus on the number of events for non-epistemic domains, motivated by the need to investigate the applicability of a domain theory to prac-

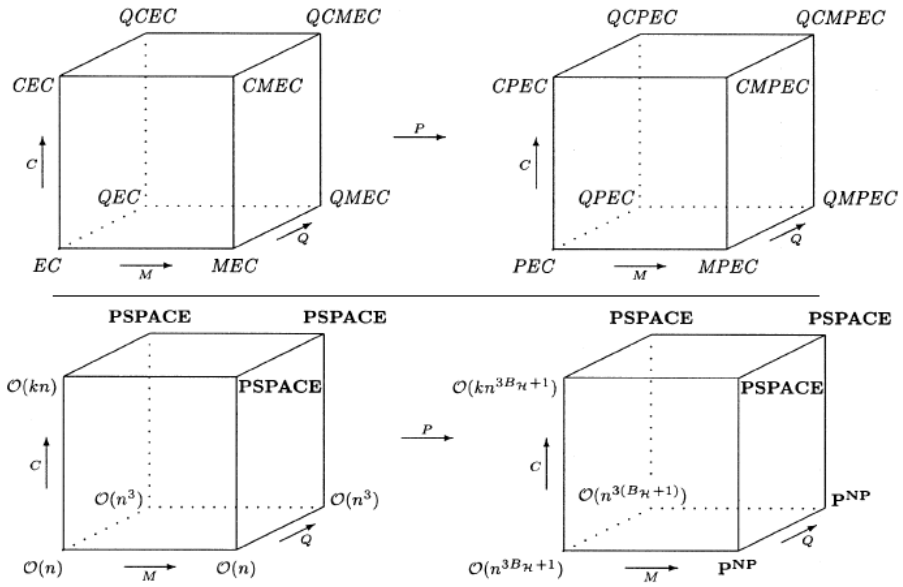

Figure 5.5: Event Calculus languages of different expressiveness and their complexity classes (source: [Cervesato 2000]).

tical implementations, where the set of event occurrences can grow arbitrarily, but the set of relevant properties that provide a characterization of the application domain remains fixed.

Focusing on the basic Event Calculus principles, as given in [Kowalski 1986], initial considerations evaluated reasoning with a set of events, whose occurrences have the effect of initiating and terminating the validity of determined fluents. The works of [Chittaro 1996] and [Cervesato 2000] evaluated the task of deriving *maximal validity intervals* (MVI) over which a fluent holds uninterruptedly, given a possibly incomplete description of when events take place and which fluents they affect. This can reasonably be considered as a model checking problem of establishing whether a formula ϕ holds or not given an initial knowledge state, where data complexity is characterized by the number n of occurring events and query complexity is characterized by the size k of the input formula, i.e., the number of logical operators occurring in ϕ[4].

Classes of different expressiveness for the Event Calculus have been studied for query time reasoning and are presented in Figure 5.5. Specifically, the figure visualizes the results for the different sublanguages based on whether model checking is trackable or not, where

[4]Notice that now letters n and k denote different sorts with respect to the previous subsections; we preserve the original notation of the the work of Cervasato et al.

in the positive case the polynomial upper bound is also provided, otherwise the complexity class of where each sublanguage belongs to is given. The prefixing of the string "EC" with any subsequence of the letters Q, C, M and P stands for the inclusion of quantifiers, connectives, modalities and preconditions, respectively. Modalities in particular are not commonly used in recent literature, therefore are not further explained here. Finally, $B_{\mathcal{H}}$ denotes the length of the longest path in the (acyclic) dependency graph of fluents for any sublanguage with preconditions, where edges denote precondition-effect relations among fluents. The two cubes in the upper part of the figure display the relation between the different sublanguages, while the ones in the bottom relate them isomorphically to their complexity classes. Summarizing the findings, we can see that:

- Model checking in the basic Event Calculus (EC), as well as in its extensions with logical connectives for boolean combinations (CEC), with quantification over events (QEC) or with preconditions on the effects of events (PEC) are polynomial-time bound. The same holds true for the extensions that combine preconditions with logical connectives (CPEC) and preconditions with quantifiers (QPEC).

- Model checking in each of the more expressive extensions that result from their combinations is **PSPACE**-complete.

- The (deterministic) time complexity of the model checking procedure is exponential in the query complexity (length of query formula) for the more expressive sublanguages QCEC and QCPEC.

Apart from the classical approach of performing query time temporal reasoning, where the set of MVIs for a given property are computed based on expensive generate-and-test strategies, a cached implementation of the Event Calculus (CEC) has also been proposed in [Chittaro 1996] that extends the classical calculus with a MVI generation and storage mechanism. According to this approach, MVIs are cached for later use in query processing and possibly updated when new events are entered in the KB. It is shown that the cost of querying about sets of MVIs for a given property is linear in the number of events for that property and more importantly it does not change when context dependency is added. Specifically, the complexity of query in CEC is linear in the number of n of events, whereas update processing is shown to be $O(n^{k+3})$, where k denotes the length of the longest path in the graph of context-dependent properties.

More recently, Paschke studied an optimized and much simpler formulation of the basic Event Calculus without the releasing of fluents, as well as an interval-based Event Calculus variation and showed linear worst-case complexity in the number of events in the case of absolute times and total ordering [Paschke 2006]. No preconditions were incorporated, though. Furthermore, the work in [Dimopoulos 2004] considers ramifications. The complexity analysis presented in this study is carried out within the language $\mathcal{E}$, which is based on the Event Calculus ontology, and describes its relation to the logic programming approach of Answer Set Programming (ASP) [Lifschitz 1999]. Dimopoulos et al. confirm the polynomial complexity of a deterministic theory without ramifications, but also show that deciding whether an $\mathcal{E}$ theory with a complete initial state and ramifications has a model is **NP**-hard. This is due to the fact that ramifications may cause a theory to have more than one model. Nevertheless, one should also note the key observation of the authors that in many classes of theories "reasoning is easier than in the general case"; indeed, in many domains ramifications can be completely replaced by effect axioms (with the price of sacrificing entirely the elaboration tolerant nature of the domain axiomatization). The theories developed by Paschke are an example of such implementations.

To conclude, we have to highlight the prominent result that reasoning with the Event Calculus can be polynomial in the number of events for many domains that present realistic formulations. In Section 7.3 we discuss efficient implementations of the Event Calculus, as well as its relation to Answer Set Programming, which is a recent essential achievement. It enables research in the field of reasoning with the Event Calculus to take advantage of the highly efficient ASP solvers that provide both faster and more expressive reasoning with respect to existing tools.

5.4 A Note on Decidability Issues

We conclude this chapter with a brief elaboration on decidability issues related to reasoning about actions, as they present an important concern when aiming at real-world implementations. This is also an interesting research field as well, since the most widely used action formalisms, such as the Situation, the Fluent and the Event Calculus, are first-order logics (or even with some higher-order variants) and apparently do not admit decidable reasoning in the general case.

Most emphasis in relevant literature has been placed on identifying decidable frag-

ments for the Situation and the Fluent Calculus. The appropriate restriction of a theory to propositional logic for describing the application domain is obviously a reasonable first step, therefore Ternovskaia [Ternovskaia 1999] presented a reduction of the problem of decidability for the Situation Calculus with propositional fluents to the emptiness problem for a finite tree automaton, a problem that is provably decidable. Such a restriction to a finite set of propositional fluents does not work for the Fluent Calculus, though [Lehmann 2000]. Instead, to identify the boundaries of decidability of the Fluent Calculus and establish decidable fragments, relevant research has considered reductions to two-counter machines based on automata theory or decidable solutions for the *monadic entailment problem* (i.e., entailment of queries without free variables) using labeled trees, as in [Hölldobler 2000] or even studied the correspondence with Petri-nets [Lehmann 2000]. In addition, restricted first-order fragments have been identified, e.g., if fluents occur at most once in a state term and there are finitely many fluent constants.

Recently, another and much more interesting direction of research has been adopted that aims at integrating action theories and description logics. A decidable fragment of Situation Calculus, as expressive as DL $\mathcal{ALCQIO}$, has been established in [Baader 2005], later extended to allow general (cyclic) TBoxes and ramifications [Liu 2006]. In fact, even the plan existence problem has been shown to be decidable in action formalisms based on this fragment of DL (but with increased complexity) [Milicic 2007]. A different approach, having strong connections with DL, is presented in [Gu 2007] that restricts the syntax of the Situation Calculus to a two-variable fragment of the first-order language leading to decidable solutions for the projection and executability problems even with incomplete initial knowledge bases.

Similar results for the Event Calculus have not been presented in relevant literature. One can primarily stay on the abductive proof procedure presented in [Russo 2002] that reduces the time structure of the Event Calculus to two timepoints guarantying decidability of deterministic domains relying on an incomplete initial state description. Results of this work are highly relevant for the DECKT axiomatization as they do not place serious restrictions in the expressiveness of the formalism (state constraints are applicable), while the technique can be used for reasoning tasks that are particularly suitable for DECKT's objectives, such as planning or knowledge updates.

To conclude, the aforementioned analysis demonstrates the difficulty in establishing decidable, while sufficiently expressive, fragments of action theories in general. This ex-

plains the reason why many implementations adopt strong assumptions, such as the closed world assumption, or are constructed considering finite domains. On the other hand, in order to apply such theories to practice, many programming languages for reasoning about action either permit undecidable reasoning, such as Golog and Flux, or cast the problem to a propositional form in order to apply for instance SAT solvers, as is the case of DECReasoner.

CHAPTER 6

Theory Extensions

Contents

One motivation for choosing the Event Calculus to build our knowledge theory has been its ability to model a wide variety of commonsense phenomena in its core ontology. The formalism's axiomatization, as described in section 4.4, provides a rich and highly flexible repertoire of tools for commonsense reasoning in domains with diverse characteristics. In this section we exploit this advantage of the Event Calculus to extend our basic axiomatization in a number of ways. First, we provide a consideration for sensing world aspects that may be subject to continuous change in a known or unknown fashion; the investigation results in a framework for building agents that can remember and forget, an important cognitive skill that goes beyond standard possible-worlds semantics. Next, we also study physical actions, actions that are automatically triggered when the world is at a particular

state. The introduction of *potential actions* with different epistemic effects becomes a necessity in order to cope with physical actions when the world state is only partially known and neither precludes nor justifies their triggering. Finally, having an expressive epistemic framework for determining the knowledge of an agent under the effect of a multitude of commonsense phenomena, we concentrate our attention on characterizing what it means for an agent to be able to achieve an objective, based on its available knowledge and its reasoning potential.

6.1 Sensing Inertial and Continuously-Changing World Features

In previous chapters we presented a unified epistemic framework for reasoning about knowledge, action and time that only considers sensing about inertial world parameters, i.e., parameters that are subject to the law of inertia at the time of sensing. This is reflected by the use of the KP fluent in axiom (KT4). In fact, most studies in relevant literature, either based on possible worlds or adopting alternative representations for knowledge, handle exclusively such types of sense actions for an agent, as the knowledge obtained is more easily handled. Nevertheless, such a choice severely limits the reasoning potential of intelligent agents in real-world domains. In the current section we extend the framework with an account of knowledge-producing actions designated for both inertial and continuously changing world features, still preserving a uniform reasoning style regardless of the type of the parameter being sensed.

To distinguish the different cases that may emerge, imagine a moving robot able to sense world aspects, such as the state of doors, the number of persons around it or its current position. Knowledge about door states can be preserved in its memory until the robot becomes aware of some relevant event that changes it, while knowledge about the other features should be considered invalid after a few moments or even at the next time instant. Still, it is important for the act of sensing to handle the different contingencies in a transparent to the robot fashion. The approach that we develop allows for sensing inertial and continuously changing properties of dynamic and uncertain domains in a uniform style, providing a level of abstraction to the design of an agent's cognitive behavior. Moreover, it augments agents not only with the ability to remember, but also to forget information, either for the purpose of preserving consistency between the actual state of the world and

the view they maintain in their KB or due to restrictions, such as limited resources, which pose critical constraints when considering real-world scenarios.

6.1.1 Inertial Fluents - Remembering and Forgetting

In general, a robot's descriptions of world states involve a large number of components that are assumed stable between action occurrences maintaining their properties for as long as occurring events do not affect them. A door remains open until some $Close(door)$ event happens, while an object's color persists even when someone lifts it and moves it around. This intuition is related to the well-known frame problem of action theories, which is concerned with specifying the non-effects of actions (see Section 2.1.1). In the Event Calculus, the commonsense law of inertia expresses that certain objects tend to stay in the same state, unless an event happens that changes this state. We refer to fluents that are subject to this law as inertial defined as follows:

Definition 6.1 (Inertial fluents) *A fluent is called* inertial *if it tends to maintain its truth value, unless affected by some event.* □

An inertial fluent, denoted in the Event Calculus by the expression $\neg ReleasedAt(f,t)$, is always subject to inertia. One can easily observe that whenever an agent senses an inertial fluent, the knowledge gained can be stored and preserved in its KB for as long as no event causes its invalidation. Sensing inertial fluents is a well-formalized task in most related epistemic action theories. In order to model real-world agents with limited resources though, the assumption of permanent knowledge preservation is too strong to accept. Apart from issues related to memory storage capacity, the rate of change in dynamic worlds quickly renders information out-of-date, forcing the agent to reconsider its knowledge about the state of certain objects that ideally could have remained unaltered.

In order to represent the "fading" validity of knowledge-producing actions, we introduce a new sense action that extends the one presented in Chapter 4. This action is captured by the following three axioms that substitute axiom (KT4) modeled before:

(KT4.1) $Happens(sense(f),t) \Rightarrow$
$Happens(remember(f),t) \wedge Happens(forget(f),t+T(f))$

(KT4.2) $Initiates(remember(f),KPw(f),t)$

(KT4.3) $\neg Happens(sense(f),t) \Rightarrow Terminates(forget(f),KPw(f),t)$

where $T(f)$ denotes a function that introduces a time delay dependent on f's properties (certain fluents tend to change more often that others). The axiomatization expresses that whenever an agent senses a fluent, two internal to the agent actions occur that cause knowledge about the state of the fluent to be kept in the memory for a specific time frame. The *remember* action produces the traditional sensing effect, while the $forget$ action's effect is canceled if a *sense* action occurs concurrently (axiom (KT4.3) is a type of *negative canceling effect* axiom for that reason, as defined in Definition 4.2).

The above axiomatization provides two alternatives for modeling knowledge-producing actions for inertial fluents. For the purpose of constructing a theoretical framework the desirable side-effect of unlimited memory persistence of fluents is achieved by retracting the instance of the $forget$ action from (KT4.1). Alternatively, an agent may also be equipped with the mental ability to forget, an essential cognitive skill for practical commonsense reasoning, particularly suited for real-world implementations. Furthermore, one may also use the same axiomatization as a means to sense continuously changing fluents, as explained next.

6.1.2 Non-Inertial and Functional Fluents

Most current logic-based approaches that study the interaction of knowledge and time focus on sensing and obtaining knowledge about inertial fluents. Still, this is hardly the case when reasoning in dynamically changing worlds. Next, we show how the aforementioned approach can also be applied to a broader class of situations. First, we elaborate on the characteristics of such situations.

In addition to inertial fluents there are also fluents that change their truth value in an arbitrary fashion at each time instant. The number of persons entering a building or the mails arriving daily at a mailbox are typical examples. Such fluents introduce a degree of uncertainty, as they give rise to several possible models, and can be defined as follows:

Definition 6.2 (Non-inertial fluents) *A fluent is called* non-inertial *if its truth value may change at each timepoint, regardless of occurring events.* □

A non-inertial fluent is always released from inertia and is represented in the Event Calculus by the predicate $ReleasedAt(f,t)$. A particular use for non-inertial fluents has been proposed by Shanahan as random value generators in problems, such as tossing

a coin, rolling a dice etc, naming them *determining fluents*, as they determine non-deterministically the value of other world aspects [Shanahan 1999b].

For the purposes of epistemic reasoning, sensing non-inertial fluents provides temporal knowledge that is only valid for one time unit, i.e., it only reflects what is known at the time of sensing, but not what will be true afterwards. Whenever a robot needs to reason about the number of persons around it, it must necessarily perform a new sense action to acquire this information; any previously obtained knowledge may not reflect the current situation. Still, there is a class of non-inertial fluents that is far more interesting, because it expresses continuous change that follows a well-defined pattern. Such fluents are utilized to denote gradual change (or *processes*, according to [Thielscher 2001b]), for instance to represent the height of a falling object, the position of a moving robot, the patience of a waiting person etc. We call this class of fluents functional non-inertial fluents:

Definition 6.3 (Functional fluents) *A non-inertial fluent is called* functional *if its value changes gradually over time, following a predefined function.* □

In order to represent gradual change in the Event Calculus, we first need to release the involved fluent from inertia, thus allowing its value to fluctuate, and then we apply a state constraint to restrain the fluctuation, so that the fluent can exhibit a functional behavior. For example, to express the change in a robot's location (on a single axis) while moving with constant velocity v, we apply the following state constraint concerning the $Position(robot, pos)$ fluent:

(SC) $HoldsAt(Position(Rob, pos), t_1) \wedge t > 0 \Rightarrow$
$HoldsAt(Position(Rob, pos + (v * t)), t_1 + t)$

It is easy to observe how axioms (KT4.1-3) can accommodate sensing non-inertial fluents (both ordinary and functional). One just needs to set T equal to one time unit, causing a $forget$ action to occur at the next timepoint and forbidding the newly acquired value to also refer to subsequent timepoints due to inertia. Nevertheless, as regards to functional fluents in particular, because of the fact that the value of the sensed fluent is subject to change that the agent is aware of, knowledge about future values can still be derived. The application of the distribution axiom (K) along with DEC knowledge theory axioms combines the narrative of actions and observations with the agent's cognitive ability.

Example 6.1. The previous discussion illustrates how the problem of sensing the two non-inertial fluents $PersonsNear(robot, num)$ and $Position(robot, pos)$ can be addressed.

Imagine that a robot named Rob performs $Happens(sense(PersonsNear(Rob, num)), 0)$ and $Happens(sense(Position(Rob, pos)), 0)$ at timepoint 0. By forming the parallel circumscription of the example's domain theory (no initial knowledge and the two event occurrences) along with Event Calculus, Knowledge Theory and uniqueness-of-names axioms, we can prove several propositions. First, two, internal to the robot, events will be triggered for each fluent; a *remember* event at timepoint 0 and a $forget$ event at timepoint 1. For the *PersonsNear* fluent it can also be proved that

(6.1.1) $\models \exists x HoldsAt(Knows(PersonsNear(Rob, x)), 1) \wedge$
$\neg\exists x HoldsAt(Kw(PersonsNear(Rob, x)), 2)$

due to (KT4.2) and (KT2) at timepoint 0 and (KT4.3), (KT2) and (KT7) at timepoint 1. The case is different for the robot's position:

(6.1.2) $\models \exists p HoldsAt(Knows(Position(Rob, p)), t)$

for all $t > 0$. This holds true, because, although the $forget$ action results in $\neg HoldsAt(KPw(Position(Rob, pos)), t)$ for $t \geq 1$, axiom (K) transforms (SC) into

(6.1.3) $HoldsAt(Knows(Position(Rob, pos)), t_1) \wedge t > 0 \Rightarrow$
$HoldsAt(Knows(Position(Rob, pos + (v * t))), t_1 + t)$

Consequently, once Rob senses his position at some timepoint, it can infer future positions, without the need to perform further sense actions. The state constraint provides all future derivations, affecting knowledge through (KT7). □

6.1.3 Context-dependent Inertia

We can now formalize complex domains that capture our commonsense knowledge about changing worlds, where fluents behave in an inertial or non-inertial manner according to context. For instance, a robot's location is regarded as a continuously changing entity only while the robot is moving; when it stands still, the location is subject to inertia. As a result only while the robot *knows* that it is not moving can knowledge about its location be stored persistently in its KB in the style described in Section 6.1.1. In general,

for any fluent that presents such dual behavior, there usually is some other fluent (or conjunction of fluents) that regulates its compliance to the law of inertia at each time instant. For the $Position(robot, pos)$ fluent, for instance, there can be a $Moving(robot)$ fluent that determines the robot's motion state. Such *regulatory* fluents appear in the body of state constraints to ensure that inconsistency does not arise when inertia is restored. According to their truth state, the fluent that they regulate can either be subject to inertia and maintain its value or released from it in order to be subject to a state constraint. To integrate regulatory fluents in the theory, (KT4.3) must be extended to ensure that the agent does not forget a fluent when it knows that the latter is inertial and should be kept in the KB:

(KT4.3+) $\neg Happens(sense(f), t) \wedge \neg HoldsAt(Knows(\neg f_{rglr}), t) \Rightarrow$
$Terminates(forget(f), KPw(f), t)$

where f_{rglr} is f's regulatory fluent. Notice that even when the agent is not aware of f's inertia state, i.e., $\neg HoldsAt(Kw(f_{rglr}), t)$, the axiom fires and knowledge about f is lost, to avoid preserving knowledge that does not reflect the actual state.

Example 6.2. Imagine that Rob's movement is controlled by actions $Start(robot)$ and $Stop(robot)$ with effect axioms:

(6.2.1) $Initiates(Start(robot), Moving(robot), t)$
(6.2.2) $Terminates(Stop(robot), Moving(robot), t)$

While the robot is on the move, its position must no longer be subject to inertia (it will regain inertia when stopped):

(6.2.3) $Releases(Start(robot), Position(robot, pos), t)$
(6.2.4) $HoldsAt(Position(robot, pos), t) \Rightarrow$
$Initiates(Stop(robot), Position(robot), t)$

In addition, the state constraint that determines the location as the robot is moving, with the *Moving* fluent playing the regulatory role, is axiomatized as follows:

(6.2.5) $HoldsAt(Moving(robot), t_1) \wedge HoldsAt(Position(robot, pos), t_1) \wedge$
$t > 0 \wedge \neg\exists t_2(Happens(Stop(robot), t_2) \wedge t_1 < t_2 < t_1 + t) \Rightarrow$
$HoldsAt(Position(robot, pos + (v * t)), t_1 + t)$

As a result, axiom (KT4.3+) will be instantiated as:

(6.2.6) $\neg Happens(sense(Position(Rob, pos)), t) \wedge$
$\neg HoldsAt(Knows(\neg Moving(Rob)), t) \Rightarrow$
$Terminates(forget(Position(Rob, pos)), KPw(Position(Rob, pos)), t)$

In brief, (6.2.6) states that the position should be stored if Rob knows that it is not moving. If, on the other hand, the robot does not possess such knowledge (even if it is unaware of its current mobility state, due to a potential malfunction), the information acquired will be retracted one time instant after the sense action. In this case, future knowledge can be inferred only if some state constraint is available.

Moreover, it can be proved that if Rob knows initially *whether* it is moving, a single sense action is sufficient to provide knowledge about all future locations, regardless of any narrative of $Start$ and $Stop$ actions before or after sensing. And, most importantly, programming Rob does not need to also involve programming different procedures depending on the state of fluents sensed; the knowledge theory abstracts the knowledge evolution reasoning process to account for both inertial or continuously changing world aspects. □

To conclude, the framework developed in this section aims at (a) offering a level of abstraction in the handling of dynamic world aspects by agents, (b) augmenting their mental skills with the ability to remember and forget and (c) formally treating the temporal aspect of knowledge, particularly suitable for planning tasks in real-world implementations. The solution extends previous logic-based approaches, as it investigates a broad range of fluent types, facilitating the development of the mental layer of cognitive agents (eg. BDI). In order to convert desires into intentions, an agent must place objectives that are realistic and appropriate to commit to. Sensing and handling sensed information becomes crucial in this setting, as sense actions need to be placed at specific time instants during planning, considering relevant preconditions, the duration of acquired information and the available contextual knowledge (e.g., see Section 6.3).

6.2 Context-Dependent and Potential Actions

Our analysis so far concentrated on context-dependent agent-driven actions with potentially unknown preconditions. Nevertheless, in real-world dynamic domains, system evolution is

also due to natural actions, *actions that occur at predicted times, provided that no earlier actions (natural or agent initiated) prevent them from occurring* [Reiter 2001a]. In the Event Calculus a trigger axiom is applied in order to allow for an event to occur as soon as the world is in a particular state.

The situation becomes complicated in partially known domains. When the agent has incomplete knowledge about the world it inhabits, it may find itself unable to determine whether such an action will actually occur or not. Moreover, each contingency may give rise to significantly different ramifications concerning the state of other fluents. In order to address this issue, we introduce the notion of *potential actions*.

6.2.1 Trigger Axioms, e_{pot} and Knowledge

To model the agent's mental state when reasoning with trigger axioms in the presence of incomplete knowledge, the original domain trigger axioms must be replaced with epistemic meta-axioms stating that a context-dependent action unambiguously occurs only when all preconditions are *known* to hold:

(TR1) $\bigwedge^{i}[HoldsAt(Knows(f_i), t)] \Rightarrow Happens(e, t)$

On the other hand, to accommodate the situation when there is uncertainty about the preconditions, we introduce a new hypothetical action, called e_{pot}, for each action e.

According to the DEC axiomatization each time the conditions of a trigger axiom are satisfied the action occurs. To avoid repeated triggering though, it is standard practice to specify as one of the action's effects to be the invalidation of some of the preconditions, thus blocking the axiom's future execution until the conditions change again. Apparently, the assumption of at least one precondition being unknown to the agent is a necessary condition for the triggering of e_{pot} instead of e, but not a sufficient one; a change must occur in the state of precondition fluents in order for the triggering of e_{pot} to be justified. Specifically, an action needs to happen that causes some of the preconditions which was not known true (i.e., it was known false or unknown) to become either known true or unknown (in the latter case, the fluent may have become true, thus the trigger axiom may be triggered). All other occasions, such as a precondition known to hold becoming unknown, do not justify the triggering of the axiom.

Notice that we also consider the situation where an unknown precondition remains unknown after an occurring action, which may only lead to a potential triggering if the action's effect is of a proper type. If, for instance, the action may terminate a fluent but the trigger axiom requires it to hold, then even if that fluent remains unknown after the action's occurrence the trigger axiom will not fire (it will either become false or remain unaltered).

As a result, for a trigger axiom of the form $\bigwedge^{i} HoldsAt(f_i, t) \Rightarrow Happens(e, t)$, a potential action is triggered under the following condition:

(TR2) $(t_1 = t_2 - 1) \wedge \neg Happens(e, t_2) \wedge \neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t_2) \wedge$
$\bigvee^{f_i \in C} [\neg HoldsAt(Knows(f_i), t_1) \wedge KmInverted(\neg f_i, t_1)] \Rightarrow$
$Happens(e_{pot}, t_2)$

In brief, a potential action occurs whenever the actual action does not happen, none of the preconditions is known not to hold (i.e., the triggering is possible) and some change in the preconditions occurs that justifies the axiom's triggering. Notice that we require for the negation of some fluent precondition to become inverted, not the precondition itself (predicate *KmInverted* has been defined in Section 4.3.2.3). For instance, if we have that $\neg HoldsAt(f_1, t) \Rightarrow Happens(e, t)$, we need $\neg HoldsAt(Knows(\neg f_1), t)$ (either f_1 is unknown or known true), as well as some event e' to occur such that $KmTerminate(e', f_1, t)$, i.e., $KmInverted(f_1, t)$.

Finally, to correlate an event e with its potential counterpart e_{pot} and specifically, to capture the effect's uncertainty that results due to the occurrence of a potential action we declare that whenever e_{pot} happens instead of e all direct effects of e become unknown to the agent. That is, for each positive or negative effect axiom about f we include in the axiomatization a new effect axiom of the form:

(TR3) $\neg HoldsAt(Knows(\bigvee^{f_i \in C} \neg f_i), t) \Rightarrow Terminates(e_{pot}, KPw(f), t)$

where $\vec{f_i}$ are f's preconditions. Moreover, for each event e such that $Initiates(e, f, t)$ we also introduce in the axiomatization $KmInitiate(e_{pot}, f, t)$ (without considering the preconditions) and similarly, for each event e' such that $Terminates(e', f, t)$ we also add $KmTerminate(e'_{pot}, f, t)$. Keep in mind that the triggering of a potential action, which is always caused by unknown preconditions, will lead to the creation of HCDs following the

axiomatization described in Section 4.3. An illustrative example that demonstrates the applicability of potential actions and epistemic reasoning in the presence of triggered events is given in Section 7.1.2, where the benchmark domain of Shanahan's circuit is extended for the case of partial knowledge.

6.3 Defining Ability

Building on the properties of a formal framework sufficiently expressive to model knowledge effects of actions, sensing and implicit knowledge, we now focus on planning tasks involving knowledge goals, which ultimately lead us to characterize the notion of ability for an agent to achieve certain objectives. In the presence of incomplete states and sensing, planning cannot be considered as finding a linear sequence of actions; it may be necessary to supplement what is known at plan time by information that can only be obtained at run time via sensing [Sardina 2004]. Thus, instead of looking for a legal sequence of actions achieving some goal, Levesque argues that the planner's task is to return a *general program* that the agent can follow and always know how to execute by virtue of its initial knowledge and the subsequent readings of its sensors [Levesque 1996]. The well-known example of an agent wishing to get on a flight at the airport is characteristic, as the agent can acquire the information of which gate to go to only after it has arrived at the airport. The idea is to provide not just the goal but also general -still understandable by the executor- instructions on how the goal is to be achieved and leave subtasks to be handled by automatic planners. As such, a planning problem with incomplete states and sensing actions is the problem of finding a conditional plan which can be proved to be executable and to achieve the goal under any circumstances, for different outcomes of sensing [Thielscher 2001a].

Our intention in this section is to study representational issues and provide a formal characterization of the notions of ability and feasibility of plan execution for expressive domains, rather than deal with algorithms for generating such plans. High-level programs that act as plan skeletons can be produced by deliberators (e.g., in FLUX [Thielscher 2005a], IndiGolog [Giacomo 1999] or MetateM [Fisher 2005]) that seek for a *strategy* to reach a final state and may involve branching, iteration, sensing etc. The semantics that we develop aim at ensuring that for a potentially non-deterministic domain specification such a program will always have enough information to continue the execution in a deterministic fashion and achieve the goal, under certain restrictions. More succinctly, prior to using a

program in a plan, we must consider whether we have the knowledge to actually execute that program.

6.3.1 Problem Characterization

In the domains we investigate, agents are assumed to acquire information about their environment and expand their KBs as they operate. A good plan for such an agent is the one that not only achieves the goal, but is also executable, i.e., ensures that the agent has enough information at every step to know what to do next; even if it can be shown that a plan must achieve the goal (and terminate), the agent may not have enough knowledge to execute it [Lespérance 2000].

Imagine the following commonsense scheme devised by Thielscher [Thielscher 2000d]: suppose that a door is closed but that its state can be altered by pressing a button next to it. Nonetheless, an agent, call it Blindie, who is unable to sense the state of the door will not be able, without assistance, to achieve the goal of entering the next room. For it can never know whether if should press the button or not. Likewise, unable to achieve the goal will be an agent, call it Dumbie, who can see but does not know how the button is causally related to the door. For this agent cannot conclude that it simply must press the button. One can even extend Thielscher's scheme with a robot Lazie that, although gifted with the previous capabilities, delays (or hurries excessively) to perform the actions needed. If the door automatically closes after 10 seconds, a mere sequential execution of the actions specified by a plan does not suffice for the agent to achieve its goal; important time constraints are placed as to when to act.

To be more precise, there are three issues that need to be considered when reasoning about the ability of an agent to achieve certain goals:

- The account of knowledge of an agent must allow for distinguishing between the actual effects of actions and what the agent knows about these effects. This is a first and fundamental step in order for an agent to come up with a plan that knows it will achieve the goal (the case of the Dumbie robot).
- The formal account of knowledge must allow for the agent to reason about what it currently knows and does not know, and what it will know after sensing (the case of the Blindie robot). Employing a theory of sensing is an essential advancement

that leads to the emergence of the concepts of knowledge-producing and conditional actions. At the very least, an agent must be able to condition its course of actions on the result of a sensing action.

- A formal account of knowledge must be coupled with an account of time. Issues such as knowing when to sense, for how long knowledge is up-to-date, when to perform a particular action and others, are critical in order to determine the ability to achieve plans in real world situations.

Most existing accounts of knowledge in action formalisms do not cover all aspects, especially the third one. Lesperance et al. [Lespérance 2000] deal with the second issue within the Situation Calculus by appealing to the notion of an *action selection function* σ, a mapping from situations to primitive actions, understood as prescribing which action the agent should perform in a situation. An agent can achieve a goal in a situation s if there exists a function σ such that it knows in s that it can get to a situation where the goal holds by following σ. The tree chopping example, where unbounded iteration is also involved, is characteristic to show that although the agent does not know how many chop actions are necessary to get the tree down, it is able to achieve the goal. Still, they do not consider the other two issues. Thielscher studies both lack of knowledge about the effects and crucial world knowledge using the Fluent Calculus [Thielscher 2000d]. Proving non-achievability relies on induction over situations (a property holds for all situations if it holds initially and if all actions preserve it) leading to properties that are reachable by an executable sequence of actions. In this section we develop a unified treatment of all three issues related to ability, considering in addition situations that may involve non-deterministic effects of actions. We exploit the fact that the Event Calculus is narrative-based, unlike the standard Situation and Fluent Calculus where an exact sequence of hypothetical actions is represented. A narrative is a possibly incomplete specification of a set of actual event occurrences and temporal orderings [Shanahan 1997] and is essential in the attempt to define the ability of an agent to achieve a goal in the general case.

We begin our investigation with the following rather generic characterization of ability that we incrementally refine in subsequent steps. In general, we could state that *an agent can achieve a goal f at time t, denoted as $HoldsAt(Can(f),t)$ if the agent knows that there exists a valid narrative s, such that, when executed at time t, it will undoubtedly result in a future world state where the goal is* known *to be true by the agent*. To make this statement

more precise we appeal to what it means for the narrative to be *valid* and the resulting situation to be *undoubtedly* reached, especially when non-determinism is involved.

6.3.2 Action Narrative

First we need to determine how the narrative of actions is properly executed. Most existing approaches in the literature assume that actions are atomic and are executed in isolation [Lespérance 2000, Thielscher 2000d]. Still, in many cases it is important for the agent to be able to perform concurrent actions [Zimmerbaum 2001]. For instance, to be able to take a photograph, an agent must press the button that holds the shutter open and sense concurrently. The Event Calculus allows for several events to occur at the same time, permitting also the effects of those events to differ from what the effects would have been had the events occurred at different times[1]. Furthermore, the explicit representation of time within predicates enables the application of temporal ordering formulae in order to relax the preciseness of an action occurrence or the order of execution of actions within the narrative. For instance, instead of stating that the $PassThroughDoor$ action must be performed one timepoint after the action $Sense(Door)$ has returned a positive value, we can permit for the action to occur at some time within the next 10 timepoints. A narrative may even permit interleaving of actions.

Consequently, having a narrative Δ_1 of a conjunction of event occurrence and temporal ordering formulae, as given in Definition 3 Section 4.4, we can define a narrative as being epistemically-valid as follows:

Definition 6.4 (Epistemically-valid Narrative) *A narrative Δ_1 is epistemically valid under a given epistemic axiomatization D iff it can be inferred, based on the initial KB Γ_2, that no sequential ordering of the actions involved in the narrative violates any of the constraints of D, i.e., the epistemic domain description Φ that consists of D, Δ_1 and Γ_2 is always consistent.* □

Example 6.3. Suppose $\Delta_1 = Happens(e_1, t_1) \wedge Happens(e_2, t_2) \wedge (t_1 = t_2)$ and let an axiomatization D that includes the event precondition constraint $Happens(e_1, t) \wedge$

[1] We should note, of course, that since cumulative and canceling effects of concurrent event occurrences are permitted, the domain axiomatization -and not the knowledge theory- must place restrictions in concurrency to prohibit inconsistent models to be produced. This is achieved in the Event Calculus by using appropriate state constraints and event occurrence constraints, when actions consume or provide the same resource at the same time.

$Happens(e_2, t) \Rightarrow \neg HoldsAt(f, t)$. The narrative Δ_1 is only valid under an epistemic domain description Φ where fluent f is known not to hold at t_1. On the other hand, for a Φ' where f is unknown, although sensing it before executing e_1 can provide knowledge about the fluent, the agent cannot be certain if the resulting domain description would be consistent, therefore the narrative is not epistemically valid under Φ'. □

Notice also that conditionals may as well occur in the plan skeleton provided by the narrative, in accordance to the classical notion of planning by deduction. Conditional actions, by means of the functions $if(f, a)$ and $ifnot(f, a)$, have been introduced in the Event Calculus in [Forth 2004] to represent that an action a is executed just if the conditional fluent f holds, or does not hold respectively. Depending on the initial knowledge and the appropriate inferences, conditionals can be evaluated at plan time or they may introduce a branching point.

6.3.3 Termination Condition

Second, we must elaborate on the condition that must be satisfied in order for Δ_1 to be considered a successful narrative, reflecting the ability of an agent to achieve a goal. Achievability means that the acting agent, by virtue of what it knows initially, the subsequent readings of its sensors and the known narrative Δ_1, will *undoubtedly* know that the goal holds after the occurrence of *all* actions prescribed by Δ_1, i.e., $\bigwedge^i Happens(e_i, t_i) \Rightarrow \exists t HoldsAt(Knows(f), t)$, where $t > max(t_1, ..., t_n)$. This is of particular importance given the possibly non-deterministic outcomes of the actions involved. One must also notice the highly epistemic-dependent nature of the notion of ability (in contrast to that of a plan, which when executed in any world satisfying the initial state description, will achieve the goal, regardless of the agent's knowledge at the beginning or during plan execution).

6.3.4 Non-Deterministic Actions

Finally, we must also characterize the type of actions that constitute Δ_1 so that a successful final state can unambiguously be reached. It is clear that when only deterministic actions are involved then the outcome of the execution of the narrative can be determined with absolute confidence from the agent side, even if these actions have context dependent effects. We have already shown how HCDs can be combined to produce knowledge about

fluents, enabling the agent to deliberate on the outcome of actions with partially known preconditions (Shanahan's circuit given in the next chapter is also an illustrative example).

On the other hand, non-deterministic effects of actions in general cannot guarantee a known final situation. Still, in our investigation of an agent's ability to achieve a goal we provide a characterization for a subset of actions with context-dependent non-deterministic effects that may still enable the agent to specify with confidence the resulting state. In brief, these are the actions that, given a particular world state, their execution does not consume any of their preconditions, therefore their execution can be iterated without any restrictions. Given that the preconditions remain unaffected, the repeated execution of such actions is guaranteed to eventually result in the desirable effect, after a non-deterministic period of time (non-deterministic iterations). We call the effects of such actions Iterative non-Deterministic effects (InDE) under a particular set of preconditions and define them below.

Definition 6.5 (Iterative non-Deterministic Effects) *If fluent f is a context-dependent non-deterministic effect of action e and C_f is its context (potentially containing f), then f is an Iterative non-Deterministic Effect (InDE) of e at timepoint t under C_f if e does not affect directly or indirectly any fluent in C_f except f itself in future timepoints. Formally, given a domain description Φ_e, where the only event term in formulae Δ_1 is the event e and all fluents in C_f hold at t, then f is InDE of e at t under C_f iff $\Phi_e \not\models \Gamma^-_{C_f \setminus f}$, where $\Gamma^-_{C_f \setminus f}$ is a world state where not all fluents in C_f, apart from f itself, hold.* □

Intuitively, event e has fluent f as InDE if, starting from some timepoint t where all preconditions hold, a future state where some precondition expect f does not hold can never be reached, given that the only legitimate action is e. In other worlds, the execution of action e does not affect any of f's preconditions, either directly or indirectly. Although finding such a resulting state using abduction can be a computationally expensive and semidecidable inference task to perform, Russo et al. [Russo 2002] proved that a reduction considering only two timepoints, current and next, can transform it to fully decidable and tractable, even when the state at the current timepoint is only partially specified (see also Section 7.2.2 that provides more details about such kind a of inferencing).

Example 6.4. Typical cases of actions with InDE fluents are the action of tossing a coin or the action of executing a voice command for opening a door within the context of the agent standing in front of the door. The agent can iteratively execute the same action and the effect is always subject to non-deterministic change. Another situation is the action

of chopping a tree that has the non-deterministic effect of the tree being down if the agent stands in front of the tree, holds an axe and the tree is not already down. Notice that now the InDE fluent is part of the precondition set, but according to the definition it does not influence the iteration.
On the other hand, the action of firing a gun does not have as iterative non-deterministic effect the fluent of shooting down a turkey; after a (predetermined in this case) number of shots the gun runs out of bullets, still the agent cannot be certain that the gunshot was accurate. Note that Definition 6.5 can trivially be extended to sequences of actions, rather than a single action, that recycle their resources. For example, assuming unlimited ammunition, a gun can be fired and reloaded repeatedly, thus converting the latter fluent an InDE one. □

What is important in establishing actions with InDE is the fact that, although the truth value of the effect is non-deterministically determined, we can be certain that after an -indefinite- number of iterations the effect will have the desired truth value. That is, if an agent repeatedly tosses a coin, it can be certain that eventually heads will come up (for otherwise, non-determinism is violated). Of course, in real-world implementations an upper bound of iterations can be specified or even percentages can be determined as to how probable for the desirable effect to turn up is.

Moreover, an effect remains InDE even if any other action that occurs afterwards does not affect the preconditions (either directly or indirectly). Unfortunately, such actions are difficult to trace beforehand, as they are domain-dependent and context-sensitive. A type of action, though, that is domain-independent is any sense action, which by definition does not affect the state of any fluent.

Proposition 2 *If a fluent f is InDE of an action e under C_f, it is also InDE of the action sequence $[e; sense(f')]$, where f' arbitrary fluent, i.e., $\Phi_e \not\models \Gamma^-_{C_f \setminus f}$ iff $\Phi_{e;sense} \not\models \Gamma^-_{C_f \setminus f}$.* □

Having this property in mind, we can now define a sequence of action occurrences as InDE-valid as follows:

Definition 6.6 (InDE-valid action sequence) *A sequence of actions $e_1, .., e_n$ is InDE-valid iff*

1. *whenever an effect f of e_i $(1 \leq i < n)$ is among the context of an e_j (where $j > i$ and no other action happens between e_i and e_j that affects f), then f is either deterministic or InDE under some, potentially empty, context set C and all fluents that constitute C are known upon e_i's execution, and*

2. *after an e_i with InDE f, the next action is a sense action that evaluates the truth value of the non-deterministic effect. The sequence $[e_i; sense(f)]$ is executed iteratively for as long as the effect does not obtain the value required for the context of e_j.* □

6.3.5 Establishing Ability

Now that we have established the specifications of valid narratives, we can define ability to achieve a goal as:

Definition 6.7 (Ability) *An agent is able to achieve a state where fluent f holds, denoted as $HoldsAt(Can(f), t)$ iff there exists an InDE- and epistemically-valid narrative Δ_1 of events $e_1, ..., e_n$, such that*

1. *the agent will undoubtedly know that the goal f holds after the occurrence of all actions prescribed by Δ_1, i.e., $\bigwedge^i Happens(e_i, t_i) \Rightarrow \exists t HoldsAt(Knows(f), t)$, where $t > max(t_1, ..., t_n)$*

2. *the agent at every time instant knows what the next event prescribed by the sequence Δ_1 is, whatever the sensing outcomes may be.* □

Definition 6.7 is in compliance to previous studies that address the problem of knowing how to execute a plan, as for instance in [Baier 2006] and [Sardina 2004] that appeal to the notion of *epistemic feasibility*. Item 2 describes what Sardina et al. refer to as *epistemically feasible deterministic programs* to characterize and formalize the notion that an agent always knows what the next step to be performed is, regardless of whether the program terminates or not. The practical significance of the aforementioned account of ability is its potential to handle challenging aspects of cognitive problems. Once the conditions under which a narrative is epistemically- and InDE-valid have been defined, they can be used as constraints on a planner to build plans that are consistent to the semantics of the high-level programs.

Previous accounts have underscored certain issues that influence the ability of an agent to reach a goal state; the agent must be able to reason about how sensing will expand its current knowledge state [Lespérance 2000]; it must reason about its knowledge on the effects of actions and distinguish between the actual effects and what it knows about them [Thielscher 2000d]; it must employ a closed coupling of knowledge and time in order to

know when to sense, for how long knowledge is up-to-date, when to perform a particular action etc. [Zimmerbaum 2001]; and, also, it must address correctly problems that involve unbounded iteration [Sardina 2004, Lespérance 2000]. In the aforementioned account of ability we integrated these issues in a subclass of particularly expressive non-deterministic domains.

Apparently, finding a way to execute high level programs and planning with incomplete states, sensing and conditionals usually involves a considerable search space, where the class of potential plans is very general. For that reason, based on the high-level programming language Flux, Thielscher uses non-deterministic heuristics for planning, where only those plans are searched which match a given skeleton [Thielscher 2001a]. Backtracking over sensing actions that lead to dead ends, the proposed approach can exploit the characterization of ability to search for suitable sensing actions, in order to solve planning problems with knowledge goals. Moreover, Sardina et al. [Sardina 2004] consider two restricted classes of programs that correspond to conformant plans and plans without cycles, respectively. For such classes, the search for epistemically feasible programs can be limited to programs of a simple tree-like form.

6.4 Summary

Although investigated in isolation, the previous extensions of the main theory glue together in a harmonious fashion structuring different layers of intelligent agents with potent reasoning skills. The DECKT axiomatization augmented with (KT4.1,2,3+) and (TR1,2,3) axioms is a complete and unified knowledge framework that can confront challenging and pragmatic commonsense problems of real-world domains. The next chapter illustrates the broadness of its applicability.

CHAPTER 7

Use Cases and Implementation Issues

Contents

This chapter serves a dual purpose. First, it exemplifies the reasoning mechanism of the theory and the way the different features and extensions can be integrated to model complex domains. A significant contribution is the modeling of a challenging benchmark problem in relevant literature, namely Shanahan's circuit, which involves ramifications with vicious cycles and delayed effects. Second, this chapter intends to demonstrate the potential of the knowledge theory in solving a wide range of commonsense phenomena that are met in practice. Specifically, the second part discusses how the epistemic reasoning capabilities are related to the challenging issues posed by Ambient Intelligence domains. We describe

different solutions that causality-based approaches can provide within the context of a running Ambient Intelligence project that takes into account the dynamic and uncertain context in which agent interactions take place. As also indicated specifically in the text, most scenarios described in the sequel present variations of well-known benchmark problems of automated logic-based commonsense reasoning. The final part raises implementation issues and shares the experience gained in adapting the theory to existing reasoners. It also describes how the framework can be implemented in terms of a custom Event Calculus reasoner that can support epistemic notions.

7.1 Shanahan's Circuit and Complex Knowledge Ramifications

The DECKT axiomatization, coupled with HCDs and potential actions, provides significant cognitive skills to intelligent software agents. To illustrate the potent assets of the theory we axiomatize a highly demanding ramification domain, extending it with a treatment of knowledge.

7.1.1 The Ramification Problem in Action Theories

The ramification problem in action theories has an interesting historical development with a multitude of benchmark scenarios being devised to test and usually surpass the limits of existing approaches. It concerns the problem of inferring indirect effects of actions, beyond those explicitly described by their associated effect axioms, derived by some general knowledge of dependencies among fluents. Capturing efficiently the potentially unbounded number of consequential effects of actions defines the essence for a solution to the ramification problem. A straightforward, yet powerful for many domains, approach to represent indirect effects is to use *state constraints* [Lin 1994, McIlraith 2000], based on minimal change. These express logical relationships of fluents that hold at all times. For instance, if a user located in a room holds an object one can infer that the object is also in the same room. It is interesting to note that state constraints can be used to model certain challenging domains that some of the more advanced techniques have trouble handling, such as the gear wheels example [Denecker 1998] that involves mutually dependent fluents.

On the other hand, this method fails to capture causal relations; if a fluent is dependent on more than one other fluents, state constraints are insufficient to determine the

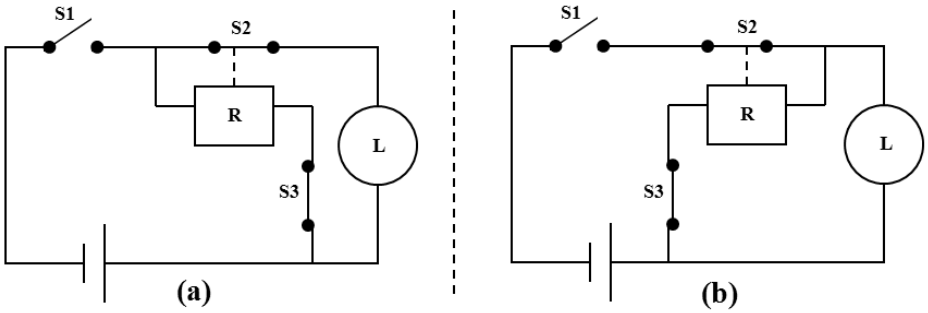

Figure 7.1: (a) Thielscher's circuit, (b) Shanahan's circuit.

dependent one. More important, in many other domains they may even give rise to unintended models. As an early effort to eliminate such problems, a distinction between primary and derived fluents was proposed. Thielscher [Thielscher 2000b] elaborated on this approach, but also indicated serious impediments by devising a particular configuration of an electronic circuit, first found in [Thielscher 1997] (Figure 7.1(a)). He suggested the use of *causal constraints* in order to work around them, i.e., causal relations that characterize the circumstances (context and triggering effect) under which the occurrence of an indirect effect is to be expected. Causality has been studied in different logics (e.g. [Giunchiglia 2004, McCain 1995]) and has even been used in temporal databases, where actions may have durations or delayed effects (e.g. when consuming alcohol an agent becomes drunk after a reasonable amount of time [Papadakis 2002]). In the Event Calculus in particular, causal constraints have been introduced by means of a new set of predicates, namely $Started(f,t)$ and $Stopped(f,t)$ to denote that the fluent f has or is about to get the intended value [Shanahan 1999a].

Nevertheless, Shanahan [Shanahan 1999a] proved that with a small modification in Thielscher's circuit, causal constraints may produce inconsistency. Shanahan's modified version of the circuit is shown in Figure 7.1(b) and involves delayed effects and cyclic fluent dependency: if initially switch $S1$ is open, but $S2$ and $S3$ closed, closing $S1$ leads to cycling ramification effects, ought to relay R, that cause light L to repeatedly become lit and unlit every 2 time points[1]. Such unstable configurations are usually deployed in practice to implement oscillating behavior, such as a flashing light or a buzzing sound. Mueller ([Mueller 2006], p. 120) presented a proper behavior for this circuit using Event Calculus

[1]For simplicity, we assume a special type of relay that closes $S2$ if no current flows through it. One could easily achieve such a behavior by properly connecting a second relay to $S2$ that is activated when $S2$ opens.

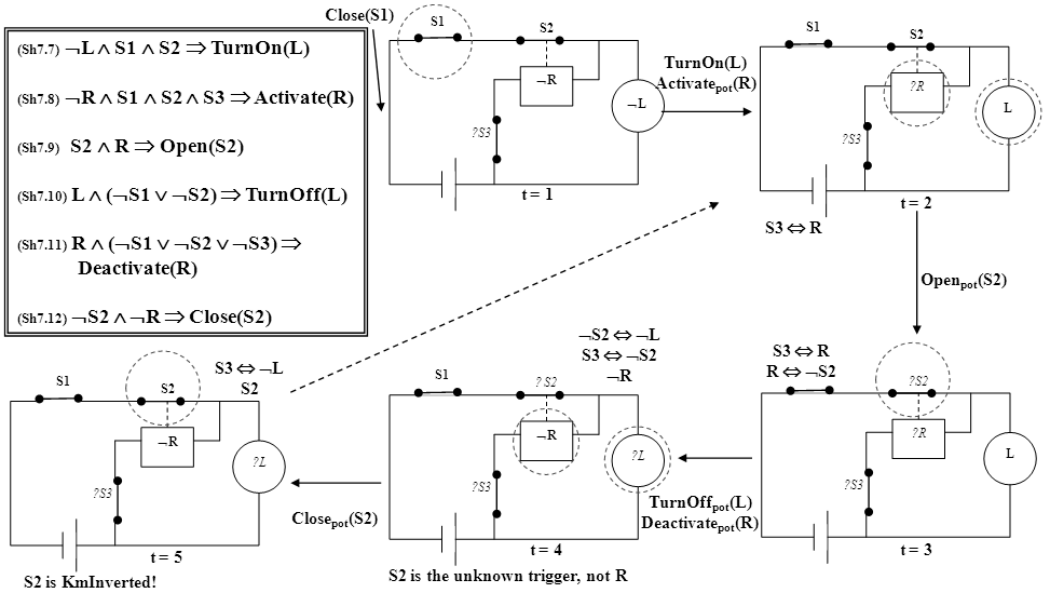

Figure 7.2: Knowledge evolution within Shanahan's circuit with vicious cycles and delayed effects.

trigger axioms. As already mentioned, these axioms specify the conditions under which actions are triggered by other actions or by the state of the world (axioms for triggered events in action theories were first formalized in [Pinto 1998] within the context of the Situation Calculus). Recently, the Event Calculus has been extended to handle ramifications in the class of instantaneously propagated mutually interacting effect domains [Forth 2007], based on a provably sound stratified theory where fluents are ordered in strata and predicate completion is performed using a prioritized minimization policy. Yet, a question raised by Shanahan in 1999 concerning his challenging benchmark problem remains still an open issue: suppose that the initial state of $S3$ is unknown, what inferences can be made?

7.1.2 Partially Observable Shanahan's Circuit

Figure 7.2 displays the *conditionally stable negative-cycling* domain [Forth 2007] of Shanahan's circuit at successive timepoints and augmented with notations representing epistemic notions. Specifically, fluents that are known to the agent are represented in bold, unknown fluents in italics with a question mark, while dashed circles mark those fluents whose epistemic state changes from one timepoint to the next.

The domain axiomatization comprises a number of effect axioms

$Initiates(Close(s), Closed(s), t)$ (Sh7.1)

$Terminates(Open(s), Closed(s), t)$ (Sh7.2)

$Initiates(Activate(r), Activated(r), t)$ (Sh7.3)

$Terminates(Dectivate(r), Activated(r), t)$ (Sh7.4)

$Initiates(TurnOn(l), Lit(l), t)$ (Sh7.5)

$Terminates(TurnOff(l), Lit(l), t)$ (Sh7.6)

and a number of trigger axioms (the latter are succinctly shown in the box on the upper left corner of Figure 7.2), such as

$\neg HoldsAt(Lit(L), t) \wedge HoldsAt(Closed(S1), t) \wedge HoldsAt(Closed(S2), t) \Rightarrow$

$Happens(TurnOn(L), t)$ (Sh7.7)

The remaining five trigger axioms (Sh7.8-12) are similarly modeled. As we are interested in representing the epistemic state of an agent having incomplete initial knowledge we apply the epistemic meta-axioms presented in previous sections, as described below.

t=0. Initially, the state of all fluents is stationary and known, apart from the state of switch $S3$.

$HoldsAt(KP(\neg Closed(S1)), 0) \wedge HoldsAt(KP(Closed(S2)), 0) \wedge$

$HoldsAt(KP(\neg Lit(L)), 0) \wedge HoldsAt(KP(\neg Activated(R)), 0) \wedge$ (Sh7.13)

$\neg HoldsAt(Kw(Closed(S3)), 0)$

(in fact, the last component is only given here for emphasis; even if it were omitted it would still be derived due to (KT7)). Now, let an agent close switch $S1$:

$Happens(Close(S1), 0)$ (Sh7.14)

t=1. From (Sh7.14) and (Sh7.1) axioms (KT3.1,2) are triggered causing the state of fluent $Closed(S1)$ to become known true, i.e.,

$HoldsAt(KP(Closed(S1)), 1)$ or, from (KT2), $HoldsAt(Knows(Closed(S1)), 1)$ (in the sequel, we will only refer to the *Knows* fluent assuming that (KT2) has been applied). But now the epistemic state of the preconditions of the first two trigger axioms has changed. In particular, all preconditions of (Sh7.7) have become known to hold and axiom (TR1) dictates that action $TurnOn(L)$ is unambiguously executed. On the other hand, preconditions in the second trigger axiom *may* have become true and, due to (TR2), instead of $Activate(R)$ the potential counterpart is executed:

$Happens(TurnOn(L), 1)$ (Sh7.15)

$Happens(Activate_{pot}(R), 1)$ (Sh7.16)

***t*=2**. While (Sh7.15) produces knowledge about L due to (Sh7.5) and (KT3.1,2), axiom (Sh7.16) has as a result the state of relay R to become unknown, due to (TR3). Still, two HCDs are created because of (Sh7.3), generated from (KT6.1.1) and (KT6.1.2): action $Activate(R)$ has as an effect to initiate R with only unknown precondition fluent $S3$ (recall from subsection 4.3.1 that the unknown preconditions of a trigger axiom for an event are also considered as effect preconditions for all the effect axioms of that event). That is, for (Sh7.3) $C = \{Activated(R), Closed(S1), Closed(S2), Closed(S3)\}$, while $C(1)^{-} = \{Closed(S3)\}$. Therefore, we have that
$HoldsAt(Knows(\neg S3 \vee R), 2) \wedge HoldsAt(Knows(\neg R \vee S3), 2)$
These HCDs are denoted by the biimplication relation $(S3 \Leftrightarrow R)$ at the bottom of the circuit at *t*=2, expressing the epistemic inferences that the agent knows.

At this point, the potential change in the truth state of R justifies a potential triggering of trigger axiom (Sh7.9), due to (TR2), because R from known false may have become initiated.

$$Happens(Open_{pot}(S2), 2) \quad \text{(Sh7.17)}$$

***t*=3**. As before, two new HCDs are created between R and $S2$, due to (KT6.1.3) and (KT6.1.4), since action $Open(S2)$ terminates $S2$ and the only unknown precondition is R. This latter potential change in the truth value of $S2$ may trigger both the fourth and fifth axioms.

$$Happens(TurnOff_{pot}(L), 3) \quad \text{(Sh7.18)}$$

$$Happens(Deactivate_{pot}(R), 3) \quad \text{(Sh7.19)}$$

***t*=4**. Before studying the resulting state let us concentrate first on the consequences of the new potential actions according to the HCD axiomatization. As before, action $TurnOff_{pot}(L)$ combined with $S2$ being unknown and L known produces $HoldsAt(Knows(\neg S2 \Leftrightarrow \neg L), 4)$, due to (KT6.1.3) and (KT6.1.4). The potential deactivation of R is more involved. Since R is potentially affected, all existing HCDs that contain R must expire, due to (KT6.2.1), but the transitivity between $S3$ and $\neg S2$ is transferred to the next timepoint, according to (KT6.2.4). The tricky part is that R's deactivation is contingent on $\neg S2$. Thus, not only a new HCD is created, due to (KT6.1.3), resulting in $HoldsAt(Knows(\neg S2 \Rightarrow \neg R), 4)$, but also the previous HCDs become dependent on $S2$ according to (KT6.2.3). $HoldsAt(Knows(R \Leftrightarrow \neg S2), 3)$ for instance is transformed into $HoldsAt(Knows(S2 \wedge R \Rightarrow \neg S2), 4)$ and $HoldsAt(Knows(S2 \wedge \neg S2 \Rightarrow R), 4)$. The latter is always true, but the former, in conjunction with the newly created HCD, is equivalent

to $HoldsAt(Knows(\neg R), 4)$! Indeed, at t=4 R is open in all possible models, regardless of whether $S3$ were initially open or not. Therefore this information becomes part of the agent's unambiguous knowledge. Finally, the potential change in the truth value of R may also trigger axiom (Sh7.12).

t=5. According to (KT6.1.1), where only the precondition fluent $S2$ is unknown, $HoldsAt(Knows(\neg S2 \Rightarrow S2), 5)$ is derived, or else $HoldsAt(Knows(S2), 5)$. But this time S2 is also *KmInverted* (see (INV) definition in section 5.2.3) therefore all previous HCDs expire and no new ones are created with (KT6.2.3). The only new HCD is due to (KT6.2.4) that preserves the transitivity relation of the involved fluents. From then on, the procedure repeats itself.

Summarizing: The example is characteristic, providing insight into the reasoning mechanism of the theory, illustrating also certain important features. The agent, for instance, is able to make epistemic derivations about the state of specific fluents by interleaving HCDs as time progresses. This is the case of inferring that R is deactivated at t=4 or $S2$ closed at t=5, transferring the knowledge from previously created fluent dependencies and expanding them according to (KT6.2.3). The evolution of interconnected HCDs reduces to unambiguous knowledge about a fluent without any additional reasoning effort. Furthermore, the existence of HCDs constitutes the agent smarter in dealing with real-world restrictions and challenges that occur at run-time, as it provides a clearer view of the causal consequences of its actions and the dependencies that have already been created. Notice the epistemic state of the circuit at t=3; a single sense action on any of the unknown fluents will allow the agent to perceive the formulation of the entire circuit. In fact, the agent knows that it is able to obtain complete knowledge even if two out of the three unknown fluents are out of reach of its sensors. The significance of developing smart plan executors, capable of performing efficient sensing at run-time, has been acknowledged in many relevant studies [Levesque 1996, Lespérance 2000, Forth 2004]. Finally, even though it is possible for a possible worlds-based theory to perform similar reasoning with less efficient manner, the commonsense phenomena involved in this domain are beyond the scope of most existing possible worlds-based or alternative epistemic theories.

7.2 Reasoning in Ambient Intelligence Environments

The trick to building effective and usable applications for practical problems is in large part in choosing the right tools. Early in the course of our research within Ambient Intelligence environments we recognized that the novel challenges introduced require techniques different than applied in traditional IS engineering. The prospect of integrating Artificial Intelligence methods was evident from the beginning, still we identified that action theories had a substantial role to play, accentuated by the fact that their targeted application domains resemble the demands raised in Ambient Intelligence. In this section we describe a running project in the field of Ambient Intelligence where causality-based reasoning has or will be used to transcend traditional rule-based reasoning capabilities, concentrating on the demands for epistemic inferencing. After briefly introducing the project's rational and the way the general solution has been approached, we elaborate on the different aspects where a causality-based knowledge theory can contribute.

7.2.1 A Reasoning Framework for Ambient Intelligence

During the last two years a multi-disciplinary project is running in our institute under the general context of Ambient Intelligence that combines expertise from different laboratories and research groups, related for instance to vision and perception, human-computer interaction, speech analysis and others [Dimitris Grammenos 2009]. The Information Systems laboratory has been assigned the task, among others, to design a centralized reasoning framework for application in the AmI system, but also to investigate opportunities and requirements of reasoning tasks emerging in the ambient environment on mobile and resource-constraint devices.

The design goals for the centralized reasoning framework have been the efficient representation, monitoring and dissemination of any low- or high-level contextual information in the Ambient Intelligence infrastructure, as well as the support for a number of general-purpose and domain-specific inferencing tasks. As regards to the task of context management in particular, ontology-based models for capturing the meaning and relation of the basic domain concepts have been combined with rule-based reasoning tools for inferring high-level contextual knowledge and supporting sensor fusion and context disambiguation functionalities. Yet, based on the experience gained we have identified limitations

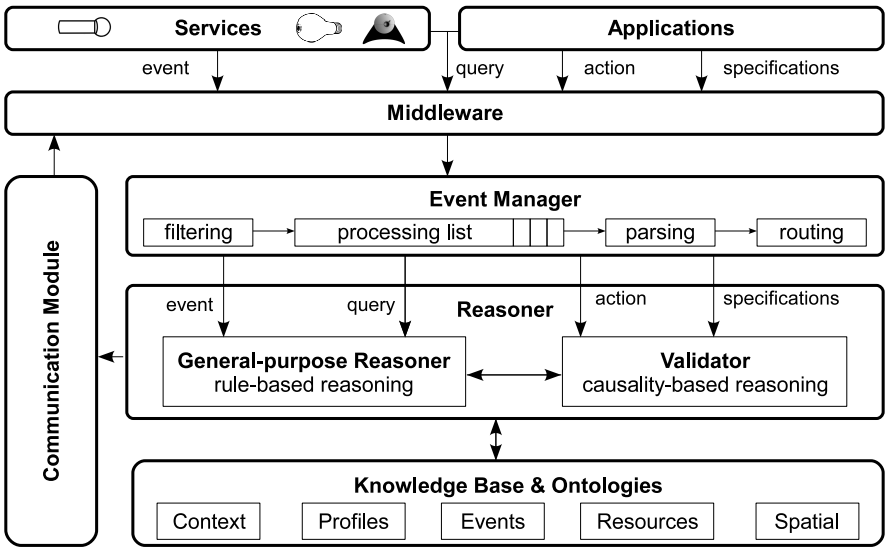

Figure 7.3: The event-based Ambient Intelligence reasoning framework architecture.

of the rule-based reasoning approach to address more challenging and sophisticated issues that emerge in such domains. We currently study the application of causality-based reasoning and action theories to provide a complete reasoning framework for Ambient Intelligence. For instance, we consider the Event Calculus for activity recognition purposes, in order to identify event patterns that describe the structure of compound events built from atomic or other compound event instances. The calculus enables us to explicitly model and reason about the intervals of composite actions, in order to alleviate semantic errors that traditional rule-based models introduce, as highlighted by recent studies [Paschke 2006, Adaikkalavan 2006]. Moreover, causality-based reasoning is also employed to support design-time application verification, as well as run-time action validation that will be detailed in the subsequent section.

The hybrid event-based architecture is shown in Figure 7.3 and comprises four main components; the Event Manager that receives and processes incoming events from the ambient infrastructure, the Reasoner that can perform both rule-based and causality-based reasoning, the Knowledge Base that stores semantic information represented using ontology-based languages, and the Communication Module that forwards Reasoner requests for action execution to appropriate services. A middleware layer undertakes the role of connecting applications and services implemented by different research groups and with different technologies. *Services* denote standalone entities that implement specific functional-

ities about world aspects, such as voice recognition, localization, light management etc., whereas *applications* group together service instances to provide an Ambient Intelligence experience in smart rooms.

The centralized reasoning component must perform inferencing tasks given the most up-to-date knowledge about various world aspects. In a highly dynamic domain such as the one designed for AmI applications, the reasoner should by no means be considered to be omniscient of all information flowing within the system. Apart from efficiency matters, there may also be physical restrictions; world aspects may be inaccessible, devices or services may be unreachable when needed or they may provide information that is not always reliable or sufficient, e.g. concerning users' location or activities. Furthermore, the devices that users operate in an AmI environment, such as PDAs and laptops, or even autonomous devices, such as robots and agents, may as well perform inferencing tasks exploiting the information acquired from the system and based on privacy restrictions. Such devices cannot store or access all context information and often experience poor communications with other ambient components. It becomes evident that in order for an AmI system to work in real-world conditions many of the simplifying assumptions must be lifted and any reasoning component must rely on its own current knowledge and reasoning potential. A powerful as well as efficient knowledge theory in particular may provide significant leverage. In the rest, we present illustrative examples of the type of deliberations an agent, either being a robot, a PDA or the large-scale centralized reasoner, can perform.

7.2.2 Run-time Action Validation and Constraint Handling

During our involvement in the project we have acknowledged how important to the management of an AmI system it is to separate the rules that govern its behavior from the domain-specific functionalities in order to enable efficient and dynamic adaptation to changes during development. This stems for the fact that the collaborators that contribute new facilities possess different backgrounds, potentially different motivations and priorities. In order to preserve robustness and efficiency, we propose a modular approach that distinguishes the rules that express *system policies and restrictions* that guarantee a consistent and error-free overall execution at all times, from *service specifications* that change in frequent time periods and by a multitude of users, as well as from *application specifications* that are usually under the responsibility of non-experts who only posses partial knowledge

Table 7.1: Defined specification axioms for application verification.

Specifications	Sample Event Calculus Axiomatization
Service Specifications	// Sorts retrieved from the *resource* ontology *sort light: object* *light Light01_MainRoom* // Events retrieved from the *event* ontology *event TurnOnLight(application,light)* *event TurnOffLight(application,light)* // Fluents retrieved from the *context* ontology *fluent LightOn(light)* // Effect Axioms based on the *event* and *context* ontology connection $\forall$*l,t Initiates(TurnOnLight(l),LightOn(l),t).* $\forall$*l,t Terminates(TurnOffLight(l),LightOn(l),t).*
System Restrictions	$\forall$*serv,light,t HoldsAt(LocalizationRunning(serv),t)* $\wedge$ *HoldsAt(LocatedIn(light,MainRoom),t)* $\Rightarrow$ *HoldsAt(LightOn(light),t).*
Application Specifications	$\forall$*chair,l,t Happens(SitOn(chair),t)* $\Rightarrow$ *Happens(TurnOffLight(l),t)* $\wedge$ *Happens(StartPresentation(<file>,TV01),t).* $\forall$*chair,l,t Happens(StandUpFrom(chair),t)* $\wedge$ $\neg$*HoldsAt(LightOn(l),t)* $\Rightarrow$ *Happens(TurnOnLight(l),t)* $\wedge$ *Happens(StopPresentation(<file>,TV01),t).*

about which the system restrictions are. Table 7.1 shows samples of the type of information that these specifications contain. The specifications of services, for instance, retrieved from the different ontologies, describe the domain and express inheritance relations, instantiations of entities, and potentially context-dependent effect properties. Application descriptions express primarily the intended behavior of a developed application as a narrative of context-dependent action occurrences. Finally, system restrictions capture assertion about attributes of system states that must hold for every possible system execution (sometimes also called *safety properties* [Russo 2002]). Such a restriction is, for instance, that for as long as the localization service is running inside a room no change in the level of lighting should be made, as it relies on analyzing camera images.

Having this knowledge description at hand an essential task is to perform application verification in order to verify that the specifications of AmI applications are in compliance with the overall system restrictions and detect errors early in the development phase. This *a priori* analysis is performed at design-time and can formally be defined as an abductive reasoning process that tries to find a set P of permissible actions that lead a consistent system to a state where some of its constraints are violated, given a domain description D, an application description AP_i for application i and a set of system constraints C:

$$D \wedge AP_i \wedge P \models \exists t \neg C(t) \text{ where } D \wedge AP_i \wedge P \text{ is consistent}$$

In fact, if such a plan is found it acts as a counterexample providing diagnostic information about violated safety properties. Apparently, such inferences are computationally expen-

sive and most importantly semidecidable. Nevertheless, Russo et al. [Russo 2002] proved that a reduction considering only two timepoints, current (t_c) and next (t_n), can transform such an abductive framework to fully decidable and tractable under certain conditions (no nesting temporal quantifiers):

$$D(T) \wedge AP_i(T) \wedge C(t_c) \wedge P \models \neg C(t_n) \text{ given a 2-timepoint structure T}$$

This way, we do not need to fully specify the state at time t_c; the generated plan P is a mixture of *HoldsAt* and *Happens* predicates without requiring a complete description of the initial system state, in contrast to similar model-checking techniques.

Example 7.1. A developer uploads an application description file to the system containing, among others, the two axioms shown in Table 7.1. The new application must first be examined for consistency with respect to the set of restrictions already stored in the system by service engineers. The developer executes the *ApplicationCheck* functionality of the Validator accessible through the middleware, which identifies a potential restriction violation whenever a user sits on a chair; the event causes the *TurnOffLight* action to occur that conflicts with the Localization being at a *Running* state (any substantial change in lighting destabilizes the localization process). As a result, the developer needs to revise the application, pausing for instance the Localizer before turning off the lights. □

Although application analysis can be accomplished at design-time and in isolation, action validation must be performed at run-time considering the available knowledge about the state of the system, as well as potential conflicts with other applications that might share the same resources. Causality-based reasoning can contribute to

(a) High-level resource management.

A reasoner for Ambient Intelligence needs to resolve high-level conflicts raised by applications that request access to the same resource. We introduce axioms to capture integrity constraints, as below:

$\forall app1, app2, t\ HoldsAt(InUseBy(Speaker01, app1), t) \wedge$
$HoldsAt(InUseBy(Speaker01, app2), t) \Rightarrow (app1 = app2)$

(b) Ramifications and priorities.

Apart from direct conflicts between applications, certain actions may cause indirect side-effects to the execution of others. Terminating a service may affect applications that do not use it directly, instead invoke services that depended on it. Since the reasoner is the only module aware of the current state of the system as a whole it can detect such unsafe effect

ramifications and take measures to prevent unintended situations to emerge, either by denying the initial actions or by reconfiguring certain system aspects. Towards this direction, actions are executed in terms of prioritization policies.

(c) Uncertainty handling.

In many situations important world parameters may only be partially observable, still any action taken must guarantee consistent and safe behavior for the system as a whole. Uncertainty may rise at any time and more often than not there is no easy way to resolve it; the reasoner must make decisions based on partial knowledge. The knowledge theory we develop in this thesis is planned to play an essential role for such type of epistemic reasoning tasks. Consider the following example.

Example 7.2 Imagine a system constraint requiring for the main room door to be in locked state iff no user is located inside it.

(7.2.1) $HoldsAt(UserInRoom(user, MainRoom), t) \Leftrightarrow$
$\neg HoldsAt(DoorLocked(MainDoor), t)$

Information about the user's presence is obtained only in the main room through a multi-camera vision localization component. Therefore, one way to infer that a user has exited the main room is when her last known location was near the door and the door is open:

(7.2.2) $HoldsAt(UserNearDoor(user, MainDoor), t) \wedge$
$HoldsAt(DoorOpen(MainDoor), t) \wedge Happens(UserLost(user), t) \Rightarrow$
$\neg HoldsAt(UserInRoom(user, MainRoom), t + 1)$

Otherwise, a $SenseUserLocation$ procedure is invoked to search using alternative means, such as RFIDs, location of personal devices etc:

(7.2.3) $(\neg HoldsAt(UserNearDoor(user, MainDoor), t) \vee$
$\neg HoldsAt(DoorOpen(MainDoor), t)) \wedge Happens(UserLost(user), t) \Rightarrow$
$Happens(SenseUserLocation(user), t)$

Consider the contingency where $Happens(UserLost(user), T)$ occurs at some timepoint T, still the door is not open and the $SenseUserLocation$ procedure cannot track the user inside the room, potentially because she is standing behind some obstacle. In such a case

where the reasoner is not aware of whether the user has left the room or not, it needs to perform reasoning based on partial knowledge. Apparently, under this contingency constraint (7.1.1) should not be applied as is; it must be reformulated otherwise we run the risk of resulting in an unintended situation where the user gets temporarily locked in the main room due to lack of knowledge about the user's presence. A more appropriate constraint that takes into consideration partial information would be:

(7.2.4) $HoldsAt(Knows(\neg UserInRoom(user, MainRoom)), t) \Leftrightarrow$
$HoldsAt(Knows(DoorLocked(MainDoor)), t)$

□

Situations of ambiguous knowledge are very common in Ambient Intelligence systems. Applying some consideration of knowledge treatment, as provided by DECKT, is crucial. Notice that the act of sensing can denote just an abstraction for procedures that can provide relevant information, such as $SenseUserLocation$ or communication between components, that provide new knowledge.

7.2.3 Uncertainty and Temporary Knowledge Example

Building on the centralized infrastructure already available, current efforts concentrate on transferring reasoning capabilities to mobile and resource-constraint devices that operate in a smart space. Such autonomous devices, capable of executing commonsense tasks, constitute the core of the Ambient Intelligence vision, as they take leading role in domains, such as ambient assisted living for supporting individuals with cognitive or physical impairments. For such devices reasoning under partial knowledge is a critical factor, as they experience substantial restrictions not only in storing relevant contextual information, but even in accessing it. In this and the next section we provide different aspects where DECKT can be applied to promote the existing AmI scheme, following the personal assistant paradigm. The following example axiomatizes in more detail a typical situation where the effects of actions are not completely at the disposal of the agent, rather some sort of sensing is required, while their temporal validity is limited. The scenario goes beyond the implemented infrastructure of our project, based on next-generation Ambient Intelligence facilities that are under development.

Example 7.3 In a smart space every user is assumed to be equipped with a virtual personal assistant, an intelligent agent living in stationary or mobile devices that the user

interacts with, such as her PDA, cellphone etc, and utilizes relevant information in order to assist the user. This assistant can be assigned different tasks: it can be instructed to proactively or reactively invent ways to accomplish user objectives that comply with her needs, given the specifications of the smart space (planning); it may assist the user while performing certain tasks and explain the behavior of the system in response to her actions (postdiction and model finding); it can foresee the result of actions by predicting the user's intentions and provide relevant assistance, e.g., suggestions or warnings (projection). The agent can take advantage of the facilities of the smart space based on certain privacy policies, such as sensors, services and devices, still we should expect that its knowledge about the environment is neither complete nor constantly updated, for reasons that may be related to limited resources, network reliability etc.

First, we start by axiomatizing a portion of the world, without considering any epistemic notions. Imagine that for a user to gain administrator privileges for the AmI space she must first apply an RFID card to the corresponding reader and then speak a valid code into a microphone located nearby. A safety mechanism provides a time window of 5 seconds for the user to present the correct code after the RFID tag has been recognized:

(7.3.1) $Happens(RFCard(ID), t) \Rightarrow$
$Happens(StartRec(Mic), t) \wedge Happens(StopRec(Mic), t + 5)$
(7.3.2) $Initiates(StartRec(Mic), Recording(Mic), t)$
(7.3.3) $Terminates(StopRec(Mic), Recording(Mic), t)$

If the spoken code is correct the procedure is successful and the user obtains full access to the room, e.g. all doors open automatically when the user approaches. Still, due to noisy conditions, even a correct pronunciation of the combination does not always guarantee that it will be successfully recognized. We can represent this uncertainty by releasing the result of the procedure from the law of inertia, allowing its truth value to fluctuate in future timepoints:

(7.3.4) $HoldsAt(Recording(Mic), t) \Rightarrow$
$Releases(Speak(\text{"3241"}), AdminMode(MainRoom), t)$
(7.3.5) $HoldsAt(AdminMode(MainRoom), t) \Rightarrow$
$Initiates(ApproachDoor(door), Open(door), t)$

Thus, after the occurrence of the *S peak* action the truth value of the *AdminMode* fluent will fluctuate in future timepoints, resulting in different models, one where the AmI space enters in Administrator mode and one where no change has occurred. In the former case, all doors that were initially locked, become unlocked. Notice how the effect of the $ApproachDoor$ action in (7.3.5) is dependent on the released *AdminMode* fluent; if the latter does not hold, the action may still occur but the user runs the risk of encountering a closed door.

Given this domain axiomatization, we can use DECKT to represent inferencing tasks that the user's personal assistant, stored on her personal PDA, can perform based on the available knowledge and its awareness of occurring events. The agent initially knows that the microphone is switched off and the doors locked:

(7.3.6) $HoldsAt(KP(\neg Recording(Mic)), 0) \wedge HoldsAt(KP(\neg Open(MainDoor)), 0)$

We reasonably assume that all non-epistemic fluent terms are subject to inertia in the initial state. Since no state constraint is applicable, apart from (K2), axiom (KT7) is compiled as:

(7.3.7) $HoldsAt(KPw(f), 0) \Leftrightarrow HoldsAt(Kw(f), 0)$

Then, a three-action narrative is generated; the user places her RFID-enabled PDA on the reader at time 0, speaks into the microphone and approaches the main-room door at time 3.

(7.3.8) $Happens(RFCard(ID), 0) \wedge Happens(Speak(\text{"3241"}), 1) \wedge$
$Happens(ApproachDoor(MainDoor), 3)$

We can now prove a number of sentences by forming the parallel circumscriptions

$CIRC[(7.3.2-5) \wedge (KT3.1-4) \wedge (KT5.1-3); Initiates, Terminates, Releases]$
$CIRC[(7.3.1) \wedge (7.3.8); Happens]$

in conjunction with (7.3.6,7), uniqueness-of-names axioms, Event Calculus axioms (DEC) and the remaining axioms of our knowledge theory. For instance, at timepoints >3 the user's agent does not know whether the door is open and also at timepoints >5 the agent knows that the microphone is not recording any more:

(7.3.9) $\models \neg HoldsAt(Kw(AdminMode(MainRoom)), 2) \wedge$
$HoldsAt(Knows(\neg Recording(Mic)), 6) \wedge$
$\neg HoldsAt(Kw(Open(MainDoor)), 4)$

This result was caused by the triggering of (KT3.1-2) at timepoint 0 and (KT3.3-4) at timepoint 5 for the *Recording*() fluent, as well as of (KT5.3) at timepoint 1 for the *AdminMode*() fluent and (KT5.1) at timepoint 3 for the *Open*() fluent. In fact, at timepoints >1 there are two possible models, one in which the room has changed mode and another where no change has occurred. In both models, though, (7.3.9) reflects part of the agent's state of knowledge.

Still, this is not all that the agent can infer. Although for the release axioms (7.3.4) the preconditions are known, this is not true for the preconditions of (7.3.5), leading to the creation of a HCD according to (KT6.1.1,2):

(7.3.10) $\models HoldsAt(Knows(AdminMode(MainRoom) \Leftrightarrow Open(MainDoor)), 4)$

This HCD may provide useful leverage to the agent. Imagine for instance that the agent can query the appropriate component of the AmI system about the current state of the main-room door, much like performing a sense action as defined by the DECKT theory. By performing this action at timepoint 4, i.e.,

(7.3.11) $Happens(Sense(Open(MainRoom)), 4)$

and following the same procedure as before now the agent will have knowledge of the actual state, reflected by the formula:

(7.3.12) $\models HoldsAt(Kw(AdminMode(MainRoom), 5) \wedge$
$HoldsAt(Kw(Open(MainDoor)), 5)$

That is, the agent can keep track of the evolution of the system in response to the user's actions and inform her, for instance, if asked, why the door is not open, as well as what to do next, e.g., to position the PDA at the reader once again (if t>5) or just to restate the correct code (if there is enough time remaining). □

7.2.4 Other Examples

The aforementioned example is a variation of "the Russian Turkey Scenario", as presented in [Mueller 2006], that deals with non-deterministic effects, in combination with other challenging scenarios discussed in [Zimmerbaum 2001] where the effects are only valid for specific time intervals. The idea is to extend such benchmark problems of cognitive robotics with a treatment of knowledge in partially observable domains and present how epistemic reasoning can progress. The theory has also been applied to a multitude of other scenarios that are briefly described next.

1. Released fluents are not the only way to treat non-determinism in the Event Calculus. Another method is to use *determining fluents*, as in [Shanahan 1999b]. These fluents, introduced in Section 6.1.2, are never subject to inertia and play the role of a random value generator for other fluents. We have used as an example the case of a user selecting randomly one of the available choices presented on a screen, with an agent reasoning about the knowledge it can have concerning that choice. Other examples of this type are the well-known "toss a coin", "roll a dice", "spin the wheel" problems etc.

2. Considering the dynamics of a domain, an agent can not only derive knowledge from direct effects of actions, but also indirectly through appropriate ramifications. Imagine a user picking up her PDA and moving around the premises of a building. Locating the PDA using the building's wireless infrastructure might not be possible at all times, due to insufficient network coverage. Still, its location varies with the location of the user holding it; knowing the location of the user, the reasoner can also infer the location of the PDA. Such types of ramification can be accommodated by the theory both with the technique of primitive and derived fluents or with the use of state constraints, such as the following:
$HoldsAt(Knows(Holding(user, object)), t) \wedge$
$HoldsAt(Knows(InRoom(user, room)), t) \Rightarrow$
$HoldsAt(Knows(InRoom(object, room)), t)$

3. Reasoning about only knowing, i.e., what an agent does not know, in the style of [Lakemeyer 1998], is an important aspect when modeling partially known environments. Consider an AmI system where two agents are responsible for supporting

visitors with their interaction. Suppose that a visitor knows that at time t at least one of the agent is online, but does not know which. The theory can be used to prove both $\neg\exists x HoldsAt(Knows(Online(x)),t)$ and $HoldsAt(Knows(\exists x Online(x)),t)$. Usually, the first sentence is called the *de dicto* reading, while the second the *de re* reading. Lack of knowledge is inferred only by explicitly representing an agent's knowledge, allowing also subtle variations in meaning, dependent on the structuring of the quantifiers of the sentence, to be captured.

4. An interesting example that the Event Calculus handles more efficiently as compared to most other formalisms, is the representation of continuous change. For instance, it can be proved that if the distance covered by a user at a particular time instant is known, then the reasoner can also infer the distance at some future timepoint, provided that it is aware of changes in her *Walking* state and, of course, the average walking speed (variation from various "falling object"-related domains). The non-monotonic nature of the Event Calculus permits accurate proactive reasoning for such domains, e.g., determining where the user might be located in the near future and updating the expectations based on newly acquired knowledge. An example of such setting has been discussed in Section 6.1

5. Furthermore, we have considered another scenario that involves the handling of temporary knowledge, i.e., knowing something for a given time interval only, based on a variation of the domains discussed in [Zimmerbaum 2001]. Imagine a light that is lit for 30 seconds, whenever a button is pressed. The light turns off afterwards, except if the button is pressed again in the meanwhile. We can axiomatize this domain either using the flexible time representation machinery of the Event Calculus (as shown in the example of Section 7.2.3) or by applying a more straightforward method of simulating time counters.

In addition to being able to cope with a multitude of diverse commonsense phenomena, the epistemic reasoning theory that implements an agent's cognitive skills needs to support long lived autonomous operation; agents may need to execute long action sequences under partial information about the environment, whereas access to required world aspects through sensing may be limited or not available when needed. AmI domains in particular require from the reasoning agents to lift most simplifying assumptions and confront the realistic restrictions posed by this challenging field of research. DECKT enables an agent

to generate and maintain long chains of dependencies among unknown fluents by creating and manipulating HCDs. These dependencies are handled as ordinary fluents minimizing the computational complexity when reasoning and at the same time providing valuable ramifications when seeking for knowledge about unobservable fluents.

Example 7.4. Among a personal assistant agent's tasks is to monitor the user's everyday activities and provide instructions or activate alerts when exceptional situations are detected with an as less intrusive manner as possible. Let fluent f_1 denote the situation when the user is cooking, which is derived by fusing relevant contextual information and user activity in the kitchen. The end of this task, based on the user's habits, can be detected from a number of activities that follow a pattern, such as turning off the hot plate, placing dishes in the sink, opening the tap water etc. Let e_1 be the event of entering the living room. If the user leaves the kitchen while cooking the assistant agent should identify a potential exceptional behavior, but should not disturb the user yet. Then, if the user picks up the phone, denoted as e_2, an alert should be activated, triggering appropriate actions to inform the user that there is, for instance, unfinished kitchen activity. If the potential exceptional situation is captured by fluent f_2 and the alter by f_3, the previous behavior can be plainly axiomatized as follows:

(7.4.1) $HoldsAt(f_1, t) \Rightarrow Initiates(e_1, f_2, t)$
(7.4.2) $HoldsAt(f_2, t) \Rightarrow Initiates(e_2, f_3, t)$

Assume now that, while at noon, the contextual information coming from the kitchen is contradicting not allowing the agent to infer either that $HoldsAt(f_1, T)$ or $\neg HoldsAt(f_1, T)$, i.e., $\neg HoldsAt(Kw(f_1), T)$ and then the user leaves the kitchen and starts making a phone call. Based on the DECKT axiomatization, the following HCDs are created for timepoints $T' > T$:

(7.4.3) $\models HoldsAt(Knows(f_1 \Rightarrow f_2), T') \wedge$
$HoldsAt(Knows(f_2 \Rightarrow f_3), T')$

Specifically, the agent does not know if an alert should be triggered, still it knows that if the user was in fact cooking the exceptional situation is a reality. Still, f_3 should be a known fact before disturbing the user. Although there is no direct means of sensing f_3, by determining whether f_1 holds the agent can also derive f_3 based on the interrelation of

HCDs. In conjecturing that there might be work in progress in the kitchen with potential hazard (e.g., a turned on apparatus), the act of sensing f_1 is translated as a check procedure of all sensitive aspects. □

Notice how HCDs are different from domain state constraints. Rather than being necessarily true at all times, the dependencies they represent are temporary, created only based on the current knowledge of the agent. They augment its mental state with valuable inference rules according to context and may as well become invalidated by other actions, as dictated by DECKT. Alternative approaches that do not deal explicitly with the notions of actions and causality, such as rule-based approaches, apart from being restricted to less expressive commonsense phenomena, require the designer to model beforehand all possible conditions that call for some action to be performed. Instead, as shown in the previous example HCDs are created and destroyed automatically, thus providing a more efficient and intuitive solution for such types of reasoning tasks. Compared to pure statistical approaches logic inference is highly expressive and computationally more potent in dealing with first-order representations, which are much richer than propositional ones that characterize most probabilistic approaches.

7.3 Implementation Issues

The account of knowledge and change we develop provides a theoretical framework for building rational agents, but also aims at programming reasoning agents for practical implementations. In order to balance theory and application the formal representation of domain dynamics need to easily be transferred to implementations that require long periods of execution with a large number of actions and efficient knowledge updating in the light of new information. To that aspect the progress that is being made on developing efficient reasoners for the Event Calculus is very important. In this section we report our experiences on implementing commonsense domains with a highly usable reasoning tool and identify limitations in applying the full potential of DECKT. We also present the ongoing effort to develop a reasoner that raises these limitations and meets all desirable features needed to pursue the goal of bridging theory and practice.

7.3.1 Requirements and Desirable Features

Faced with the task of implementing the mental state of reasoning agents using logic-based programming languages, two features are most desirable, in order to exploit DECKT framework's full potential:

1. The reasoner must permit nested reification in order to support the modeling of epistemic fluents inside Event Calculus predicates. The intension is to allow, for instance, the epistemic proposition $Knows(Open(S\,1))$ to be treated as a term of a first-order logic and not as an atom, so that $HoldsAt(Knows(Open(S\,1)), 0)$ can be regarded as a well-formed formula of first-order logic. Although Event Calculus reasoners permit reification of ordinary relational fluents inside the predicates of the calculus, most of them do not provide support for deeper levels of reification.

2. Machinery should be available for reasoning to progress incrementally to allow for run-time execution of knowledge-based programs, so that each time a program interpreter adds a new action to its agenda, the reasoner can update its current knowledge base. This is an essential feature for our targeted implementations, since some of the agent's actions will be sense actions that can fully exploit the non-monotonic nature of the Event Calculus. Although offline reasoning can be used to study the properties of different domains and simulate the outcome of sensing, a knowledge theory is most suited for online reasoning execution, where an agent can benefit from planning with the knowledge at hand, interleaving sensing with ordinary actions and acquiring the actual outcomes of its actions.

In addition to these desirable features there are also requirements raised by the DECKT axiomatization that need to be carefully handled when targeting for efficiency. In particular, axiom (KT7), which rules the behavior of all released *Knows* fluents, requires special handling. We propose two approaches for implementing DECKT, both of which can provide efficient solutions.

A. Negation-as-failure. In order to infer all epistemic derivations, dictated by a particular axiomatic system of modal logic the reasoning procedure must include inference rules, such as the distribution axiom (K) and the (P1), (P2) properties of knowledge introduced in Section 4.2. Within DECKT, axiom (KT7) defines the necessary and sufficient conditions for a formula to be known: there must be some evidence for this knowledge to be produced,

that is, either the formula must be known directly by means of the KP fluent or indirectly through some state constraint known to be triggered. By employing negation-as-failure as a means to model (KT7), the implementation of dynamic closed world assumption in DECKT can be straightforwardly implemented. Intuitively, failure to derive either that a formula is known to be true or known to be false using (KT1-6) and the available state constraints implies that the formula is unknown. This approach can undoubtedly offer benefits to the developer of epistemic-enabled agents simplifying the design process and it is adopted in our Jess-based Event Calculus reasoner that we describe in Section 7.3.3, which exploits certain properties of Jess in order to simulate negation-as-failure. Still, most existing Event Calculus reasoners are not equipped with negation-as-failure facilities, as this feature is not among the calculus basic tenets[2].

B. KT7 Minimization. The second alternative provides a way to ground axiom (KT7) to the available state constraints, replacing completely axioms (K), (P1) and (P2). In particular, (KT7) can be instantiated based on the available domain knowledge, according to the following proposition:

Proposition 3. (KT7 Minimization) *A conjunctive formula ϕ is known to hold iff all individual components of ϕ are known to hold. A disjunctive formula ϕ is known to hold iff*

i. for all state constraints that contain ϕ or a part of it, the disjunction of the complements of the subformulae that result after retracting ϕ is known, or

ii. the KP fluent for ϕ or any of its disjunctive subformulae is known to hold.

Proposition 3 provides the guidelines to instantiate (KT7) by considering only those state constraints that may provide knowledge about the formula and essentially reveals the main source of computation effort in model checking with DECKT. Consider the following example:

Example 7.5. Let a domain consisting of six fluents and the state constraints

(7.5.1) $\neg f_4 \vee \neg f_5 \vee f_3$

(7.5.2) $f_6 \vee f_1 \vee f_2$

In order to determine if clause $(f_1 \vee f_2 \vee f_3)$ is known, according to Proposition 3, the reasoner can construct (KT7) as follows:

(7.5.3) $HoldsAt(Knows(f_1 \vee f_2 \vee f_3), t) \Leftrightarrow$

[2]We should note, though, that there are a number of research attempts underway that we are aware of that aim at implementing an Event Calculus reasoner using Prolog.

$$HoldsAt(Knows((f_4 \wedge f_5) \vee \neg f_6), t) \vee$$
$$HoldsAt(KP(f_1 \vee f_2 \vee f_3), t) \vee$$
$$HoldsAt(KP(f_1 \vee f_2), t) \vee HoldsAt(KP(f_1 \vee f_3), t) \vee HoldsAt(KP(f_2 \vee f_3), t) \vee$$
$$HoldsAt(KP(f_1), t) \vee HoldsAt(KP(f_2), t) \vee HoldsAt(KP(f_3), t)$$

Assuming that all available events may only initiate or terminate (unconditionally) atomic fluents, then, based on previous discussions about KP, (KT7) for this formula can be simplified even further:

(7.5.4) $$HoldsAt(Knows(f_1 \vee f_2 \vee f_3), t) \Leftrightarrow$$
$$(HoldsAt(Knows(f_4), t) \wedge HoldsAt(Knows(f_5), t)) \vee$$
$$HoldsAt(Knows(\neg f_6), t) \vee$$
$$HoldsAt(KP(f_1), t) \vee HoldsAt(KP(f_2), t) \vee HoldsAt(KP(f_3), t)$$

If, in addition, the reasoner were given as initial knowledge $HoldsAt(KP(f_1 \vee f_2), 0)$, then this disjunction (and only this) should be included in (7.5.4). The same holds true for any disjunction created as HCD; it should be considered in (KT7), but only for as long as the HCD remains valid. Notice how this axiom completely substitutes knowledge derivations due to (K). Moreover, see that in order to determine the values of f_4, f_5, f_6 we may need to search through the state constraints of these fluents as well (it is the $Knows$ fluent that is used, not KP). Finally, it can be seen that by checking the fluents required for the initial disjunction, the reasoner produced knowledge about all subformulae of this disjunction as well, since they too are reduced to these atomic fluents. □

The previous example points out certain interesting features, such as that in order to determine the truth value of the query formula the reasoning process may need to search through all fluents of a given domain. This holds true, because DECKT lays more emphasis on state constraints among fluents. If all fluents are interrelated, as in the previous example, it is probable that in order to answer even an atomic query we may have to search through an exponential number of formulae. Still, as already argued, this is hardly the case in realistic domains. The properties that are included in state constraints form sets of interrelated fluents whose size is significantly smaller than that of the domain.

To conclude the requirement analysis, it is also appropriate to comment on a general limitation of the Event Calculus. In contrast to other calculi, such as the Situation and the Fluent Calculus, this formalism does not support functional fluents. Of course, one can

represent functional properties in terms of relational ones extended with an extra argument, along with appropriate state constraints to ensure that the fluent is functional, i.e., that the argument denoting its value will always exist and be unique. For example, the functional fluent $TossDice(d) = x$ can be represented by the relational fluent $TossDice(d, x)$, enforcing restrictions on the permissible truth values for argument x. Nevertheless, this syntactic treatment constitutes less intuitive the modeling of sense actions about functional fluents. Within the Situation Calculus, a special action $read_\tau$ is introduced to determine the value of a term τ. Instead, for DECKT a sense action's denotation would refer to evaluating all instantiations of the sensed fluent since grounding the extra argument would be of no practical value. Although notationally less elegant, this approach introduces no further inconveniences if we rest the theory on the assumption of finite domains of constants.

7.3.2 SAT-based DECReasoner

Reducing reasoning tasks into satisfiability problems has proven to be a decisive leverage for the implementation of efficient reasoners for the Event Calculus, enabling the integration of fast SAT solvers. The Discrete Event Calculus Reasoner[3] is a representative example, as well as a highly usable tool, supporting deduction, abduction, postdiction and model finding for a wide range of problems. For that reason, this reasoner has been extensively used during the course of this thesis to model use cases for DECKT employing subsets of its axiomatization. Unfortunately, neither of the previously mentioned desirable features are supported by DECReasoner, complicating the knowledge-based programming process. Regarding the first requirement in particular, we have adopted two solutions.

Syntactical treatment of epistemic fluents. Since nested reification is not directly supported in DECReasoner, we have followed a syntactical approach to express epistemic fluents. For instance, in order to represent the knowledge that an agent may have about properties, such as $Knows(InRoom(object, room))$, we introduced two new fluents for each epistemic one, namely $Knows\_InRoom(object, room)$ and $Knows\_NotInRoom(object, room)$. This way, $HoldsAt(Knows\_InRoom(object, room), time)$ can be properly handled with the classical Event Calculus axiomatization. In Appendix B.1 we provide sample code for the representation of a non-deterministic domain using this approach, which produces two models[4]. Apparently, this approach introduces serious restrictions in terms of practical

[3] DEC Reasoner *http://decreasoner.sourceforge.net* (last accessed: August 2010)

[4] Further examples can be found in *http://www.csd.uoc.gr/~patkos/deckt.htm*.

value. Although the full DECKT can be represented this way, the programming process becomes cumbersome when many fluents are involved, especially for the representation of knowledge about formulae. Even with an automatic parser that translates domain descriptions to epistemic theories, the reasoner would be flooded with far too many fluents to reason with, thus wasting one of the design objectives of the theory. For smaller domains, though, the approach is tolerable.

Extending DECReasoner's Ontology. As an alternative technique to tackle the reification problem without syntactically enhancing the language's propositions, we have extended the core ontology of DECReasoner with epistemic sorts. In sort, we have defined an auxiliary set of epistemic predicates, namely $KHoldsAt$, $KInitiates$, $KTerminates$, and reformulated the foundational Event Calculus axioms to account for these predicates as well. By exploiting the reasoner's syntax we have defined a new sort of reified epistemic fluents that each of the epistemic predicate could treat as ordinary ones. This auxiliary axiomatization is presented in Appendix B.2, along with an example implementation. Although this approach facilitates the design process of knowledge programs, it is restricted to the modeling of knowledge about atomic fluents only, thus limiting the expressive power of the theory. Furthermore, an additional parser is needed to reshape the output returned by the reasoner, as the epistemic predicates cannot be displayed in the user friendly format originally designed for ordinary Event Calculus predicates.

Both the aforementioned techniques present limitations concerning the first requirement, while there is no solution for the second requirement either. Moreover, a reduction of Event Calculus theories to satisfiability problems is neither the most efficient nor the most expressive technique any more. Recently, a reasoner that casts first-order circumscription into the general stable model semantics has been released [Kim 2009]. Taking advantage of the progress in answer set programming and the availability of very fast ASP solvers the reasoner is shown to outperform both SAT-based solvers and traditional abductive planners both in terms of breadth of domains that it can be represent and in terms of efficiency as regards to execution time and number of atoms produced. The result is very promising as the restrictions imposed by circumscription concerning predicate completion are tackled and more expressive reasoning tasks exploiting the full Event Calculus can be computed. Unfortunately, both this reasoner and DECReasoner cannot be used for online program execution.

7.3.3 Custom Jess-based Event Calculus Reasoner

Ultimately, we have concluded that in order to model the expressiveness of a theory of knowledge and exploit the full potential of HCDs the reasoner must offer both high-level modeling and low-level programming flexibility with respect to the aforementioned features. For instance, HCDs require efficient handling of list constructs to be provided by the syntax of the underlying logic programming language. For that purpose, our current effort concentrates on building an Event Calculus reasoner that can support reification of epistemic predicates on different levels (thus enabling introspection in the future) and incremental progression of execution. The reasoner is based on Jess[5], a very efficient rule engine that implements the Rete algorithm for rule matching.

Specifically, we are developing an Event Calculus reasoner that can perform deduction tasks both for epistemic and for non-epistemic theories. Concerning the former many innovative features have been embedded. Predicates are asserted as facts in the reasoner's agenda, specified by the following template definition:

```
(deftemplate EC (slot predicate)
                (slot event (default nil))
                (slot epistemic (default no))
                (multislot posLtrs )
                (multislot negLtrs )
                (slot time (default 0))
                (slot reactivation (default 1)))
```

The "predicate" slot is instantiated based on the names defined in the Event Calculus' basic ontology, such as *HoldsAt*, *Happens*, etc., whereas the "epistemic" slot is instantiated as *Knows* or *KP* when needed. Multislots represent lists denoting disjunction of fluents (conjunctions are decomposable into their components according to the definition for knowledge), so that for instance knowledge about formula $(f_1 \vee f_2 \vee \neg f_3)$ at time 1 to be captured by the fact:

```
(EC (predicate HoldsAt)
    (epistemic Knows)
```

[5] *http://www.jessrules.com/* (last accessed: August 2010)

```
    (posLtrs f_1 f_2)
    (negLtrs f_3)
    (time 1))
```

Given that this is a *HoldsAt* predicate, the remaining slots are left undefined (the reactivation slot is used for internal to the reasoner purposes). The exploitation of lists for maintaining positive and negative literals of formulae enables the representation of HCDs in a syntax-independent manner, so that all meta-axioms of DECKT can be translated into appropriately defined rules. This way, the reasoning process can be fully automated, despite the fact that the (KT6) set of meta-axioms are time-dependent: the meta-axioms are adapted at every reasoning cycle (per timepoint) based on the facts that exist in the reasoner's agenda, as explained in the following example[6].

Example 7.6.1. As an example of the way meta-axioms are constructed given a particular domain axiomatization, consider the (Sh7.12) trigger axiom of the Shanahan's circuit example:

$$\neg HoldsAt(Closed(S2)) \wedge \neg HoldsAt(Activated(R), t) \Rightarrow$$
$$Happens(Close(S2), t) \qquad \text{(Sh7.12)}$$

The precondition set is $C = \{\neg S2, \neg R\}$, while the effect of event $Close(S2)$ is $Closed(S2)$. The following rule, under the Jess syntax of our implemented reasoner, captures the instantiation of both (KT5.1) and (KT6.1) axioms:

```
; Epistemic Axioms for Switch S2 and Potential Actions
;Precondition set C={~S2,~R}
(defrule MAIN::Close_PotKT5+6.1Combined
(Time (tpoint ?t))
; Assignment of facts to variables ?prec1, ?prec2
    ?prec1 <- (fluent (name Closed) (arg S2))
    ?prec2 <- (fluent (name Activated) (arg R))
; No precondition is known not be false (triggering is possible)
    (not    (EC (predicate HoldsAt)
                (epistemic Knows)
                (negLtrs $?nf1&:(= 0 (length$ $?nf1)))
```

[6]Full code of the Jess-based Event Calculus reasoner along with examples and output, including the complete Shanahan's circuit epistemic axiomatization, are available at *http://www.csd.uoc.gr/~patkos/deckt.htm*

```
                (posLtrs ?prec1)
                (time ?t)))
    (not    (EC (predicate HoldsAt)
                (epistemic Knows)
                (posLtrs ?prec2)
                (negLtrs $?pf1&:(= 0 (length$ $?pf1)))
                (time ?t)))
    ?event <- (event (name Close_pot) (arg S2))
 =>
    ;KT6.1
(assert (EC (predicate Initiates)
            (epistemic KP)
            (event ?event)
            (posLtrs ?prec1 ?prec2)
            (time ?t)))
    ;KT5.1
(assert (EC (predicate Terminates)
            (epistemic KP)
            (event ?event)
            (negLtrs ?prec1)
            (time ?t))))
```

These facts are asserted at every timepoint, provided the rule's preconditions hold; still, before reasoning proceeds to the processing of occurring events another rule retracts from all HCDs those fluents that are known, so that only the $C(t)^{-}$ set of fluents will eventually remain, according to DECKT's principles. This way the meta-axioms are adaptable at runtime to the agent's running knowledge. □

Furthermore, Jess provides the functionality for facts to be asserted or retracted to its agenda on-the-fly based on information acquired at runtime, implementing online execution. Appendix B.3 includes samples of how the Event Calculus axiomatization has been modeled using rules within the Jess environment both for epistemic and for non-epistemic domains. It also shows one more example of an axiom that require HCD treatment, namely (KT6.2.4).

An important design characteristic of our approach is related to the way memory man-

agement is accomplished. First, we only need to instantiate at start-up all ground objects (all possible fluents and events) so that the predicates can create references to them. On the other hand, predicates are asserted only when needed. More specifically, we preserve in memory only the facts that hold at each timepoint. Second, by exploiting a feature of Jess by which rules are triggered only based on the knowledge stored in its memory, we managed not only to dismiss (DEC2,6,8) axioms from the axiomatization, but also to implement negation-as-failure, reducing significantly the design and reasoning effort. Third, during the process of generating a model the reasoner identifies timepoints where alternative models should be considered (e.g., when a fluent is release and not constrained by any state constraint). Thus, after completely constructing a model the process backtracks to these timepoints retracting unnecessary future facts from memory and maintaining only the set of clauses needed for each model. We have successfully modeled different benchmark problems included in DECReasoner involving ramifications, non-determinism, as well as trigger axioms. Furthermore, the epistemic treatment of Shanahan's circuit has also been modeled. Initial results achieve fast reasoning generating smaller set of clauses.

A useful characteristic of Jess is the fact that it enables its rule agenda to be triggered at run-time as newly produced events are detected. Our objective is to integrate the reasoner with the already implemented Ambient Intelligence infrastructure described in this chapter, in order to support temporal reasoning tasks at run-time. Another research group has recently announced an online reasoner for the Event Calculus [Federico Chesani 2009], where the agent must determine its course of actions based on information acquired at run-time. It is based on the Cached Event Calculus [Chittaro 1996] that computes and stores the maximal validity intervals of fluents, and extends or revises them as events occur. This intense research in the field creates high prospects for the future of creating practical applications with Event Calculus-enhanced intelligent agents.

CHAPTER 8

Conclusions

Contents

After presenting our work and the relevant results we now revisit the main outcome and technical contributions from a more high-level standpoint and also remark prominent future research directions.

8.1 Synopsis of Contributions

Research in the broader field of Artificial Intelligence is oriented now more than ever towards the challenges of pragmatic and real-world domains and focuses on application areas that can support humans in dealing with everyday activities and problems. Frequently encountered difficulties and simplifying assumptions are not overlooked any more when deploying systems that incorporate new, innovative AI technology. The impact of the present study lies in this line of research, acknowledging the dynamic, highly complex and uncertain nature of the real world, as well as the restrictions of rational agents in obtaining information and reacting in a commonsense manner.

Reasoning about action and knowledge in dynamic environments has recently adopted alternative and more efficient representations of the epistemic notions of an agent. Towards this direction, this study significantly progresses the field with respect to the range of phenomena that such alternative representations can model. DECKT is a highly expressive formal framework for describing agents' knowledge under partial observability by taking advantage of both the way knowledge change is axiomatized and the particular characteristics of the underlying Event Calculus formalism. The result is a theory that combines the

expressiveness of possible-worlds specifications with the efficiency of reasoning without the accessibility relation to model an agent's mental state and dynamics. To balance theory and application both characteristics are imperative: even a simple formulation such as Shanahan's circuit requires a rich repertoire of reasoning skills on behalf of the agent to reason about the commonsense phenomena that emerge.

One further contribution of the present study is the introduction of the notion of HCDs. The axiomatization of the dependencies among unknown preconditions and effects is completely independent of the underlying formalism, as it builds on the understanding of the properties of possible worlds and the way they affect knowledge change. The intuition can be translated to other knowledge theories, in order to augment them with the intended epistemic notions of causality. A multi-agent system can reap benefits not only in terms of computational efficiency, but also by enabling agents to perform well-targeted sense actions in order to acquire knowledge about other, potentially inaccessible to their sensors, world aspects. In a real-world system the act of sensing may also mean communicating with other agents; having an explicit representation of the dependencies before deciding which information to ask for, contributes to policies for managing communication load.

The impact of the proposed epistemic theory has been illustrated both within the field of cognitive robotics and for different application areas of Ambient Intelligence. The former has formulated interesting and challenging problems that arise in different domains where autonomous agents need to exhibit commonsense behavior. A broad range of benchmark use cases can be treated with DECKT, especially when partial observability is a determinant factor. Ambient Intelligence, on the other hand, embraces several scientific areas and dictates in large the objectives and directions of current research activities and implementations. We have highlighted opportunities for epistemic reasoning from different perspectives; from inference tasks of resource-constraint mobile devices to the management of a large-scale Ambient Intelligence infrastructure where context-dependent service coordination needs to be attained at design- and at run-time.

Finally, we have developed a methodology and a tool for automating commonsense reasoning in practice. Existing reasoning tools for the Event Calculus introduce syntactical restrictions or execution limitations when encountering epistemic reasoning tasks. Our Jess-based Event Calculus reasoner supports progression of epistemic KBs where reasoning can be executed in an online fashion.

8.2 Directions for Future Research

This dissertation opens interesting future research directions. The theory we developed focuses exclusively on the formalization of the mental state of an agent in terms of knowledge. However, an adequate theory of agents should include both knowledge and belief [Halpern 1985]. In contrast to knowledge, an agent may have erroneous beliefs and can adopt different strategies to handle contradictory information concerning aspects of the environment. DECKT's core ontology requires only the property of consistency to be satisfied by its knowledge base, thus constituting the transition to belief a plausible future step. Building on the basic tenets of DECKT the axiomatization of belief revision and the study of related issues is a reasonable next step with expected results both in terms of efficiency and in expressiveness, considering the broadness of the Event Calculus.

Moreover, a fundamental question in the area of cognitive robotics (that Reiter had begun to examine) is the relationship between pure logical representations of incomplete knowledge and the more numerical measures of uncertainty. Initial attempts that assume noisy sensors and effectors incorporate probabilities in actions theories or implement Markov Logic Networks. As the emphasis of this study is on real-world domains, a relaxation of certain assumptions and the treatment of ambiguity in the agent's knowledge will enable a much tighter coupling of the high-level control program and other parts of a robot's software, like mapping and localization, or even vision. The objective will be to narrow the gap between cognitive and traditional robotics.

A third significant future direction would be to depart from the single agent case and consider multi-agent systems. Highly interesting issues emerge, such as reasoning about the knowledge of other agents, common knowledge and group knowledge, or about introspection matters. Finally, we already consider extensions of the basic axiomatization to support even more expressive domains, e.g., we investigate temporal indeterminacy of event occurrences (events occurring sometime over an interval rather than at specific timepoints), where we expect that HCDs can provide significant leverage.

APPENDIX A

Proofs of Theorems and Propositions

A.1 DECKT Correctness Property

For proving the equivalence result of Chapter 5 we will use some lemmas. The intention is to show that starting from DECKT we can construct BDECKT axioms, essentially proving completeness (and the reverse, to show soundness). Nevertheless, during the course of the proof we come up with the conclusion that in order to construct BDECKT axioms, DECKT axiomatization must not only be complete with respect to the possible-worlds semantics but also sound (for the actions that we consider here, i.e., ordinary and sensing actions). This becomes more clear by understanding the intuition of the proof procedure that follows.

A.1.1 Preliminaries

The ultimate objective is to determine which worlds should be K-related to the world that results after the occurrence of any event -either ordinary or sensing- according to the DECKT treatment of knowledge. Considering DECKT as given, we start with the knowledge that DECKT produces at the resulting state (after the occurrence). This knowledge needs to be supported by the possible worlds theory as well, since knowledge is understood as universal according to (M1). Therefore, we then investigate which of the initial worlds (before the occurrence) can justify the conclusions derived from DECKT. The successors of these worlds and only them can be K-related to the successor of the initial world. In certain cases all the initial worlds can be considered as *candidates* (e.g., when ordinary actions occur as in Lemma 1), while in others only a subset of the initial worlds can explain the given knowledge (e.g., for knowledge producing actions as in Lemma 2). Finally, we show that from those candidates, all must be considered as necessary initial worlds, in order not to violate any of the underlying Event Calculus principles. The worlds that are K-related

to the successor of the initial world characterize exactly how knowledge should evolve in BDECKT, i.e., axioms (BKT5) and (BKT6). The reader should also keep in mind that according to our assumption about lack of non-determinism, the set of K-related worlds can only diminish or stay the same as events occur, due to (M2).

As we also accentuate throughout the proving procedure, had some DECKT axiom been omitted we would have had less knowledge in the resulting state (more possible worlds) that could not be justified by any of the initials. Correspondingly, if some DECKT axiom gave more knowledge than appropriate (less possible worlds and unsound conclusions) then fewer initial worlds would be required again violating some Event Calculus principles. Lemma 1 and the remark that follows it exemplify such situations with examples. Because of this fact the reverse procedure, constructing DECKT axioms for ordinary and sense actions from BDECKT, becomes less intriguing as it serves mainly verification purposes. Still, there is an interesting remark about this statement after Theorem 1 below.

To follow the proving process, the reader should understand that we proceed by distinguishing a number of situations depending on the type of event and the initial knowledge. The set of situations investigate whether initially the preconditions are known or not and, respectively, whether the effect is known or not (18 situations in total, most of which similarly treated). For each situation we study what happens to the effect fluent, as well as to those fluents that the effect fluent is related to (through HCDs), or more specifically, what knowledge is gained and what is lost. All other fluents that remain unaffected maintain their truth value due to the law of inertia.

A.1.2 Proofs

Lemma 1. $DECKT \wedge (BKT1-4) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (BKT5)$

Proof: Let e_i denote arbitrary non-sensing events, f_i arbitrary fluents and s_i arbitrary situations. We need to show

$Happens_B(e, s_1, s_2) \Rightarrow$

$(K(s_2', s_2) \Leftrightarrow \exists s_1'(S(s_1', s_2') \wedge K(s_1', s_1)))$ (BKT5)

Suppose $Happens_B(e, s_1, s_2)$ (1)

From (1), (L2) we have $Happens(e, s_1) \wedge S_L(s_1) = s_2$ (2)

Applying (M2) to correlate S_L with K, we get

$\forall s_2'(K(s_2', s_2) \Rightarrow \exists s_1'(S(s_1', s_2') \wedge K(s_1', s_1)))$

That is, the worlds accessible from s_2 cannot appear out of nowhere; they must have an ancestor which, in addition, must be K-related to the ancestor of s_2, namely s_1. The first part of the proof aims at determining which of s_1's K-related worlds may have successors that are among s'_2.

Part A

We need to distinguish the different situations, depending on the effects of e and context knowledge. We start with positive effect axioms $HoldsAt(f_{pr}, s_1) \Rightarrow Initiates_B(e, f, s_1, s_2)$:

Case (i): $HoldsAt(Knows_B(f_{pr}), s_1)$ and $\neg HoldsAt(Kw_B(f), s_1)$
Since the precondition fluent is known true, the knowledge axiom (T) declares that f_{pr} must hold, therefore the positive effect axiom is triggered and we can conclude that $Initiates_B(e, f, s_1, s_2)$ or, due to (L3), $Initiates(e, f, s_1)$. But then, (KT3.1) and (KT3.2) are triggered because of the above, (2) and the fact that $HoldsAt(Knows_B(f_{pr}), s_1) \Leftrightarrow HoldsAt(Knows(f_{pr}), s_1)$ from (M1). As a result, we get $Initiates(e, KP(f), s_1)$ and by application of (DEC1) $HoldsAt(KP(f), s_2)$, which (KT2) transforms into $HoldsAt(Knows(f), s_2)$ or equally $HoldsAt(Knows_B(f), s_2)$.

This means that, due to (BKT1), the definition of knowledge in possible worlds, $\forall s'_2(K(s'_2, s_2) \Rightarrow HoldsAt(f, s'_2)$. That is, f must hold in all worlds K-related to s_2. As already mentioned, due to (M2) these worlds cannot appear out of nowhere; they must have an ancestor which, in addition, must be K-related to the ancestor of s_2, namely s_1. No restriction is placed on the truth value of f in the ancestor worlds of s'_2, other than f being unknown from the hypothesis. That is, assuming that s'_4 is the ancestor of s'_2, $S(s'_4, s'_2)$, we are unaware whether $HoldsAt(f, s'_4)$ or $\neg HoldsAt(f, s'_4)$ (we only know that f is not released at all states and that it cannot have the same truth value, otherwise it would be known). As a result, from (BDEC5) due to contraposition, we have that $HoldsAt(f, s'_4) \vee \exists e'(Happens(e', s'_4, s'_2) \wedge Initiates(e', f, s'_4, s'_2))$. As mentioned before, the first component of the disjunction is not necessarily true, and moreover we already know that an event has occurred initiating fluent f (BKT2,3). Hence
$\forall s'_2 K(s'_2, s_2) \Rightarrow \exists e', s'_4(Happens(e', s'_4, s'_2) \wedge Initiates(e', f, s'_4, s'_2))$ (3)
Due to (BKT2), when an event occurs in s_1, it also happens in all other K-related worlds, therefore $e' = e$ since it is the only event occurrence we assume to be aware of.

The important conclusion from (3) is that s_2 is only K-related to worlds that have resulted from s_1's K-related worlds where some event (e in particular) initiated fluent f

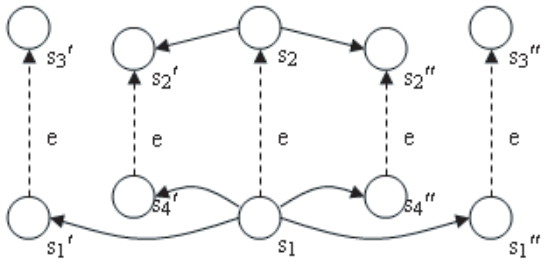

Figure A.1: Accessible worlds before and after the occurrence of event e; intermediate stage for proving Lemma 1.

(one should see how this becomes a critical factor in the proof of (BKT6)). But in this case, due to (BKT2) and (BKT3), all of s_1's K-related worlds qualify for this condition, therefore all of them are candidates. Schematically, this situation can be depicted as the graph of Figure A.1, where nodes denote worlds (or states), solid arcs denote accessibility relations and dashed arcs denote successor worlds due to event occurrences.

Indirect effects: DECKT creates no new HCD involving f, since the preconditions are known. In fact, no HCD containing f can be present after the action, even if it existed before the action, because (KT6.2.1) would instruct its expiration. The only new HCDs could be due to the transitivity axiom (KT6.2.4), if applicable, between fluents involved in HCD with f and $\neg f$. Still, the knowledge reflected by these HCDs must also exist in all initial worlds, because the distribution axiom (K) creates perfect reasoners. As a result, the conclusion that all initial worlds are candidates remains valid for indirectly affected fluents as well.

Case (ii): $HoldsAt(Knows_B(f_{pr}), s_1)$ and $HoldsAt(Knows_B(f), s_1)$
It progresses exactly as before, since the precondition fluent is known and the effect axiom is triggered in all initial worlds. Again, no initial world is excluded and all are considered as candidates whose successor can be K-related to s_2.

Case (iii): $HoldsAt(Knows_B(f_{pr}), s_1)$ and $HoldsAt(Knows_B(\neg f), s_1)$
As in case (i). All initial worlds are considered candidates.

Cases (iv-vi): $HoldsAt(Knows_B(\neg f_{pr}), s_1)$
Since the precondition is known not to hold, the effect axiom will not get triggered and the event e will have no effect in any of the initial worlds. Due to inertia inflicted by (DEC5,6), all fluents will preserve their truth value in the successor states. Thus, regardless of whether

f is known or unknown, all initial worlds are candidates, following the process developed in case (i).

Case (vii): $\neg HoldsAt(Kw_B(f_{pr}), s_1)$ and $\neg HoldsAt(Kw_B(f), s_1)$
In the same style as before, we can see that (KT5.1) is triggered leading now, after application of (KT2) and (M1), to $\neg HoldsAt(Kw_B(f), s_2)$. This means that, using (BKT1), the last statement is decomposed into

$\exists s_2'' K(s_2'', s_2) \wedge \neg HoldsAt(f, s_2'')$ (4)

$\exists s_2''' K(s_2''', s_2) \wedge HoldsAt(f, s_2''')$ (5)

(4), (5) are the consequence of lack of knowledge about f; since we are in a state where f is unknown, there must be both worlds where f is true and worlds where it is not.

Formulae (4) and (5) tell us what should be known after the occurrence of the action (HCD are studied later), now we see what this knowledge requires for the initial states. As before, (M2) dictates that these worlds cannot appear out of nowhere, they must have an ancestor which, in addition, must be K-related to the ancestor of s_2, namely s_1. From (5) and contraposition of (BDEC5) we get that

$\exists s_2''' K(s_2''', s_2) \wedge$

$(HoldsAt(f, s_4''') \vee \exists e'(Happens(e', s_4''', s_2''') \wedge Initiates(e', f, s_4''', s_2')))$ (6)

where s_4''' is the ancestor of s_2''', $S(s_4''', s_2''')$.
From (4) and contraposition of (BDEC4) we get that

$\exists s_2'' K(s_2'', s_2) \wedge$

$(\neg HoldsAt(f, s_4'') \vee \exists e'(Happens(e', s_4'', s_2'') \wedge Terminates(e', f, s_4'', s_2'')))$

where s_4'' is the ancestor of s_2'', $S(s_4'', s_2'')$. According to the hypothesis, no such e' that terminates f exists, therefore we must assume that

$\exists s_2'' K(s_2'', s_2) \wedge \neg HoldsAt(f, s_4'')$. (7)

Of course, we also assume $e = e'$ as in case (i) without loss of generality, since this is the only occurring event.

Formulae (6) and (7) state that s_2 is only K-related to worlds that have resulted from s_1's K-related worlds where f did not hold or f did hold or f became initiated. Apparently, no initial world can be excluded from this statement and all are candidates. The hypothesis also requires for both the initial and successor states to be K-related to worlds where f is true and f is false, but we will see at the second part which of the candidate worlds must be considered.

Indirect effects: According to DECKT, also (KT6.1.1) is triggered leading, after appli-

cation of (KT2) and (M1), to $HoldsAt(Knows_B(\neg f_{pr} \vee f), s_2)$. Or equally, using (BKT1):

$$\forall s_2'(K(s_2', s_2) \Rightarrow \neg HoldsAt(f_{pr}, s_2') \vee HoldsAt(f, s_2') \quad (8)$$

(8) is transformed into the following formula if we apply contraposition of (BDEC4) to the first component of the disjunction and (BDEC5) to the second:

$$\forall s_2'(K(s_2', s_2) \Rightarrow$$
$$\neg HoldsAt(f_{pr}, s_4') \vee \exists e'(Happens(e', s_4', s_2') \wedge Initiates(e', f, s_4', s_2')) \quad (9)$$

where s_4' is the ancestor of s_2', $S(s_4', s_2')$. Again we assume $e = e'$ as before. Formula (9) introduces one more restriction to the initial worlds, either the precondition is false initially or the effect becomes initiated. Again, this holds true in all worlds, otherwise the hypothesis would be violated (specifically, axiom (DEC1)) . Furthermore, DECKT also makes certain arrangements for the relations where f participated (KT6.2.2,3): they remain valid only if the precondition does not hold in the successor (and consequently, in the initial) state. This again holds true in all initial situations, as all the worlds where the precondition is false remain unaltered at the resulting state, thus transferring their relations.

Case (viii): $\neg HoldsAt(Kw_B(f_{pr}), s_1)$ and $HoldsAt(Knows_B(\neg f), s_1)$

Exactly as before, with the only difference that (KT6.1.2) is also triggered. Therefore, in addition to (9), we also have

$$\forall s_2'(K(s_2', s_2) \Rightarrow$$
$$HoldsAt(f_{pr}, s_4') \vee \neg HoldsAt(f, s_4') \quad (10)$$

where s_4' is the ancestor of s_2', $S(s_4', s_2')$. To obtain (10) after the appropriate contrapositions, we have also considered the fact that there is no event terminating fluent f. Intuitively, from (9) and (10) we have that initially either f_{pr} holds and f becomes initiated or f_{pr} does not hold and neither does f. According to the hypothesis, in all worlds f does not hold and f becomes initiated whenever f_{pr} is true, therefore again all worlds are candidates.

Case (ix): $\neg HoldsAt(Kw_B(f_{pr}), s_1)$ and $HoldsAt(Knows_B(f), s_1)$

This case is easier than the previous, because neither (KT5.1) nor any (KT6) axioms is triggered according to DECKT. In fact, according to DECKT no change in knowledge would be caused from execution of action e. As a result, no initial world can be eliminated.

Cases (x-xviii) The procedure is identical for negative effect axioms $HoldsAt(f_{pr}, s_1) \Rightarrow Initiates_B(e, f, s_1, s_2)$.

What cases (i) to (xviii) have shown is that regardless of the effect that an ordinary action produces on f or the knowledge about its context, all worlds K-related to s_1 should be considered as candidates for producing worlds that are K-related to s_2, as shown in

Figure A.1 (notice that this is not the case if the action is a sensing action). In particular, what we have concluded up until this point is that some subset of worlds accessible from s_1 produces successor worlds accessible from s_2, regardless of which these are. Formally:

$Happens_B(e, s_1, s_2) \Rightarrow$

$(K(s'_2, s_2) \Rightarrow \exists s'_1(S(s'_1, s'_2) \wedge K(s'_1, s_1)))$ (11)

Part B

It remains to be shown that this set of s'_1 worlds is not an open subset of the worlds that are K-related to s_1, but rather they are all those worlds, i.e., there cannot be s'_4 that is K-related to s_1, whose successor is not K-related to s_1's successor.

But there could not be any other way. To see why, recall that in DECKT knowledge is treated as an ordinary fluent. Therefore, according to (DEC5) and (DEC6) that force inertia to the *HoldsAt* predicate, if a fluent is known (resp. unknown) and not affected by some event, it must remain known (resp. unknown) at the next time instant. It is trivially proved that by omitting some of the worlds in the resulting situation there is a possibility to obtain knowledge about fluents other than f, thus violating inertia. Imagine, for instance, there is a fluent f' that holds in all K-related to s_1 worlds except s'_4 (thus f' is unknown in s_1). If an event happens that does not affect f' and the successor of s'_4 is not K-related to the successor of s_1 then f', by definition, must be considered known after the occurrence of the event, which violates axiom (DEC6): neither $HoldsAt(Knows(f'), s_1)$ nor is there any event that initiates the $Knows(f')$ fluent.

As a result, every fluent not directly or indirectly affected by e that holds in s'_4 for instance must also hold in some world accessibly related to s_2, say s'_2 (more specifically, not only the truth value of the fluents, but also the accessibility relation of all K-related to s_1 worlds must also be "transferred" to the K-related to s_2 worlds; imagine for instance what would be the case if K was not considered an equivalence relation! Worlds whose fluents have the same truth value may affect differently the world they are K-related to). Consequently, all the successors of the K-related to s_1 worlds must be K-related to s_2.

Therefore it has been proved that

$Happens_B(e, s_1, s_2) \Rightarrow$

$(K(s'_2, s_2) \Leftrightarrow \exists s'_1(S(s'_1, s'_2) \wedge K(s'_1, s_1)))$ (BKT5)

□

The proofs for Lemma 2 proceed in a similar and less complex fashion.

Lemma 2. $DECKT \wedge (BKT1-4) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (BKT6)$

Proof sketch: The process is very similar to that of Lemma 1. We need to show that

$Happens_B(sense(f), s_1, s_2) \Rightarrow$

$(K(s'_2, s_2) \Leftrightarrow \exists s'_1(S(s'_1, s'_2) \wedge K(s'_1, s_1) \wedge (HoldsAt(f, s_1) \Leftrightarrow HoldsAt(f, s'_1))))$

Transforming $Happens_B(sense(f), s_1, s_2)$ into the corresponding linear DEC $Happens(sense(f), s_1)$ we apply axiom (KT4) of DECKT that produces $HoldsAt(KPw(f), s_2)$ or equally $HoldsAt(Knows(f), s_2) \vee HoldsAt(Knows(\neg f), s_2)$. Therefore, we distinguish two different cases, one where the effect fluent is known to be true and another where it is known to be false.

In each of these cases, f has the same truth value in the resulting situation after the *sense* action, therefore according to (DEC5) either it must have had that value before the action or some event happened that initiated (resp. terminated) f in all initial worlds, i.e.,

$HoldsAt(f, s'_4) \vee \exists e'(Happens_B(e', s'_4, s'_2) \wedge Initiates_B(e', f, s'_4, s'_2))$ (1)

where, as in Lemma 1, s'_4 are ancestors of s_2's K-accessible worlds s'_2. The main difference with Lemma 1 is that now not all initial worlds justify f having the same truth value, and this originates from the definition of the sense action. Specifically, according to the hypothesis the only action that occurs is $sense(f)$. But by definition, knowledge producing actions cannot change the state of the world, they only change the mental state of the agent. Therefore, since no ordinary fluent can be influenced by $sense(f)$ there cannot be any such e' in (1), thus the first component of the disjunction must hold. That is, only the initial worlds where f has the same truth value as in the resulting worlds, can be considered as candidates for justifying the situation given by the hypothesis. Similarly for the case where the sensed fluent is false. The rest of the proof is as in Lemma 1. An interesting remark is that by having fewer worlds in the resulting state, apart from the sensed fluent we may also obtain knowledge about other fluents indirectly. This knowledge is captured in HCDs that are transferred without change from the initial state to the next. □

Important remark about the proving process: We now elaborate a bit more on why the previous process requires for DECKT to also provide sound conclusions and what the effects would be had the axiomatization been constructed differently. Let us concentrate on case (vii) of Lemma 1. If DECKT did not include axiom (KT6.1.1) formula (8) would not be valid and therefore we would have had less knowledge (more worlds) after the action. Specifically, there would be no reason to exclude as possible a world s'_2 where $HoldsAt(f_{pr}, s'_2) \wedge \neg HoldsAt(f, s'_2)$, since the only knowledge from DECKT is

due to (KT5.1) and the hypothesis which declares that both f and f_{pr} are unknown. Yet, following the same methodology as in Lemma 1, the ancestor of s'_2, say s'_4 must have $HoldsAt(f_{pr}, s'_4)$, since no event happens that affects f_{pr}, as well as $\neg HoldsAt(f, s'_4)$, since no event happens that terminates f. Even worse, the transition from s'_4 to s'_2 must not initiate f. But then this would violate a fundamental axiom of the Event Calculus, namely (DEC1). The same conclusion is reached if we omit any of DECKT axioms that get triggered in the cases that have been presented: a successor world is permitted that is not justified to have occurred from any of the initials. Recall that axiom (M2) prohibits ancestors not to be K-related to the initial.

Let us now consider the situation where some DECKT axiom provides different knowledge than the one given in our axiomatization. Imagine for instance that (KT5.1), instead of lack of knowledge, derived knowledge about the effect being true. Then, for case (vii) of Lemma 1, instead of $\neg HoldsAt(Kw_b(f), s_2)$ we should have $HoldsAt(Knows_B(f), s_2)$. The problem is that now we end up with more knowledge (less worlds) in the successor state than we should, knowledge that, although permitted by all initial worlds, can be justified by only some of them. As a result, we are unable to construct (BKT5), since not all initial candidate worlds must be considered necessarily. Similarly, if (KT3.1) for case (i) did not provide knowledge about f, a world s'_2 would be permitted such that $HoldsAt(f_{pr}, s'_2) \wedge \neg HoldsAt(f, s'_2)$. This would violate axiom (DEC1) for the reason explained before[1].

Theorem 1. (Completeness) *The DECKT axiomatization produces all BDECKT epistemic derivations.*

Proof: Follows from Lemmas 1,2. □

Proving the remaining lemmas that consider BDECKT as given using the process already described for Lemmas 1 and 2 is a straightforward and often less complex procedure as the knowledge that emerges after action occurrences is trivially obtained by the definition and properties of knowledge. Moreover, as already mentioned, the soundness of these axioms has indirectly been proved by Lemmas 1,2. This of course does not mean that proving Theorem 2 is an unnecessary step. To the contrary, although soundness has already been proved for ordinary and sense actions, one can define any type of other ac-

[1]Indeed, the way that all DECKT axioms were constructed in the first place involved an in depth consideration of the potential violations that would result or the knowledge that should exist in order for the derivations to comply with the possible worlds specifications.

tions with custom properties that do not comply with the possible worlds specifications. A *forget* actions is a typical example or a particular $\overline{sense}$ action that instead of returning that a fluent is true if it is true, it returns false (for some eerie reason). Such actions can be formally axiomatized in a knowledge theory if necessary, still they will violate logical equivalence with a possible worlds-based theory. Theorem 2 proves that no such actions exist in the fundamental DECKT axiomatization.

Lemma 3. $BDECKT \wedge (KT1, 2, 7) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (KT3)$
Lemma 4. $BDECKT \wedge (KT1, 2, 7) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (KT4)$
Lemma 5. $BDECKT \wedge (KT1, 2, 7) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (KT5)$
Lemma 6. $BDECKT \wedge (KT1, 2, 7) \wedge BDEC \wedge LDEC \wedge L \wedge M \vdash (KT6)$

Theorem 2. (Soundness) *All epistemic derivations produced by the DECKT axiomatization are also produced by BDECKT.*
Proof: Follows from Lemmas 3 to 6. □

A.2 Propositions

Proposition 1. (Model checking with possible worlds) *For a domain with n fluents, checking whether a conjunction of m of them is known requires, at worse,* 2^{n-m} *worlds to consider if the conjunction turns out to be known and* $2^{n-m} + 1$ *worlds if it turns out to be unknown. Checking whether a disjunction of m fluents is known requires, at worse,* $2^n - 2^{n-m}$ *worlds to consider if the disjunction turns out to be known and* $2^n - 2^{n-m} + 1$ *worlds if it turns out to be unknown.*
For a domain of n fluents, determining all known formulae requires, at worse, 2^n *worlds to consider.*
Proof: As we are interested in the worst case scenario, we assume that the reasoner will have to search through the maximum number of worlds before reaching to any conclusion regarding the queried formula. These worlds are created from those aspects of a domain that are unknown, which in the worse case (complete lack of knowledge) require for the potential worlds to consider all possible combinations, i.e., 2^n.

As such, if a conjunction of m fluents is known this will mean that these fluents should have the same truth value in all possible worlds, therefore only for the remaining $n - m$

fluents need we form the possible combinations, which are 2^{n-m}. All of them must be checked, as only one possible world where the queried formula is not satisfied is enough to violate the definition for knowledge. On the other hand, if the formula is unknown then all possible worlds are 2^n, still for some of them the formula will not hold. In suffice to find the one where the formula does not hold. In the worst case, the reasoner may have to search through all worlds where the formula holds, i.e., 2^{n-m}, before finding one where it does not, therefore in this case we need $2^{n-m} + 1$ worlds to consider.

For a disjunctive formula the situation is similar. Notice that a disjunctive formula of m fluents does not hold in 2^{n-m} worlds. In particular, these are the worlds where the conjunction of the negated fluents holds, based on the previous analysis. Therefore, the disjunction may be satisfied in the worst case in the remaining $2^n - 2^{n-m}$ worlds. All of them need to be checked if the formula turns out to be known and one additional will be checked if it turns out to be unknown.

For determining the truth value of all formulae, the worst case is to have complete lack of knowledge, therefore 2^n worlds are needed to check truth in. □

Proposition 2 *If a fluent f is InDE of an action e under C_f, it is also InDE of the action sequence* $[e; sense(f')]$*, where f' arbitrary fluent, i.e., $\Phi_e \not\models \Gamma^-_{C_f \setminus f}$ iff $\Phi_{e;sense} \not\models \Gamma^-_{C_f \setminus f}$.*
Proof sketch: Follows directly from the definition of sense actions, according to which a sense action does not cause any effect to the state of the world, but only to the mental state of the agent. □

Proposition 3. (KT7 Minimization) *A conjunctive formula ϕ is known to hold iff all individual components of ϕ are known to hold. A disjunctive formula ϕ is known to hold iff*

i. for all state constraints that contain ϕ or a part of it, the disjunction of the complements of the subformulae that result after retracting ϕ is known, or

ii. the KP fluent for ϕ or any of its disjunctive subformulae is known to hold.

Proof sketch: The decomposition property of conjunctive formulae is directly obtained by application of axiom (P1). The minimization of (potentially atomic) disjunctive formulae is directly obtained from the (KT7) axiom. According to this axiom, a formula is known iff there is some state constraint with head the original formula whose body is known or else if the KP fluent referring to the original formula holds.

Since we adopt a convention of representing state constraints as disjunctions, "knowing

the body" for a formula $f_1 \vee .. \vee f_n$ is translated into knowing the conjunction of those fluents that are not included in the original formula, in a negated form. Notice that due to the distribution axiom (K) knowledge about the original formula can also be obtained even from knowledge about its constituent parts, therefore state constraints that refer to them should also be considered. The same holds true for the (KT2) axioms where the KP fluent refers to sub-disjunctions of the original formula; knowledge about the latter can be inferred. All the aforementioned constraints need to be considered in order to ground the minimized (KT7) axiom to the domain constraint and to completely dismiss the distribution axiom from the axiomatization. □

A.3 An Algorithm for Efficient Inference with HCDs

The following algorithm has been depicted to show that the inference task $SC \cup HCD \cup D_{in} \models F$, i.e., INF^{HCD}, can be performed with complexity $O(2^{n-1} * \sum_{i=1}^{d}(|d_i| - 1))$, where n is the size of the domain, d is the number of HCDs and state constraint rules that constitute program P, and $|d_i|$ the length of rule d_i, i.e., the number of fluents comprising it. The objective is not to build an optimal algorithm in terms of efficiency, rather to verify that for a given domain the complexity of executing inference tasks is affected linearly as the number of HCDs increase (or, more precisely, the transition from INF^{SC} to INF^{HCD} has polynomial cost).

The intuition of the algorithm is motivated by the way possible worlds are structured to infer conclusions. According to this approach, as worlds are removed if a fluent (or fluent formula) ends up having the same truth value in all remaining worlds then it can be concluded that this is the fluent's actual truth value. But as we have pointed out, by removing worlds, relations between certain fluents are created. As a result, it is straightforward to conclude (in fact, it has been proved in Theorem 1) that the fluents that have the same truth value in all worlds may also be inferred from the relations that characterize uniquely this set of worlds.

In the case of inference with implication rules, we assume that we have as input a set of rules and facts and we are interested in what conclusions can be made about other fluents or fluent formulae. Therefore, the algorithm starts with the set of all possible worlds and for each rule removes those worlds that do not satisfy it (the reverse direction than followed

with possible worlds). After considering all rules, if a fluent is found to belong to all remaining worlds then it is can be considered a valid logical inference.

The structure that we use to implement efficiently the algorithm preserves for each fluent the set of worlds to which it belongs (rather that storing for each world the fluents that hold in it). For each rule we intersect the sets of the involved fluents to determine which of them should be disregarded. It is interesting to note that this set of worlds that have to be removed need only be applied at query time for the fluent that we are querying, instead of passing it through all fluents. If we find that the fluent we are querying does not belong to any world, then its negation can be assumed to be a valid logical inference.

Our basic structure is as follows. Each world is characterized by a serial number (for n fluents we have 2^n worlds, but each fluent may belong only to half of them, i.e., 2^{n-1}). We keep separately a fluent literal f and its complement $\bar{f}$. For instance, for a domain of three fluents f_1, f_2, f_3, we initialize their structures as follows (see Fig. 4.5 at timepoint 0 for verification):
$f_1 : \{1, 2, 3, 4\}$, $\bar{f_1} : \{5, 6, 7, 8\}$
$f_2 : \{1, 2, 5, 6\}$, $\bar{f_2} : \{3, 4, 7, 8\}$
$f_3 : \{1, 3, 5, 7\}$, $\bar{f_3} : \{2, 4, 6, 8\}$

The size of the KB is $O(u * 2^u)$, where u denotes the unknown fluents. For the rule $f_1 \wedge f_2 \Rightarrow f_3$ for instance, we seek for the set returned by the intersection $f_1 \cap f_2 \cap \bar{f_3} = \{2\}$, which means that world 2 should be removed when query answering, since in this world the rule is violated.

Algorithm A.1:

1. *Initialization.*
   For each fluent initialize its corresponding list. Apply sets of (2^{u-1}) successor numbers for the first fluent and its negation, sets of (2^{u-2}) for the next 2 etc. Notice that the resulting lists are sorted, simplifying the intersection.

2. *For each state constraint determine which worlds should be disregarded.*
   These can be obtained by intersecting the sets of the fluents appearing in each constraint (the head negated). For example, for $f_1 \wedge f_2 \wedge \neg f_3 \Rightarrow f_4$ we take $f_1 \cap f_2 \cap \bar{f_3} \cap \bar{f_4}$. As the complexity of the intersection of two sorted lists is proportional to the length of the longest list, with d rules of length $|d_i|$, this step takes $O(\sum_{i=1}^{d}(|d_i| - 1) * 2^{u-1})$,

where the sum captures the number of intersections that need to be evaluated and the exponential assumes the maximum number of fluents for each intersection. Consequently, the complexity is linear to the number of HCDs d, as well as to their length $|d_i|$.

3. *Answer query.*
   In order to answer if a fluent is inferred from the rules, we just need to check if that fluent or its negation has an empty set of worlds, after we retract the set found in step 2. Thus, the complexity of query answering for a single fluent is the complexity of retraction of two sets, which can be as bad as $O(2^{u-1})$.

As mentioned before, the important conclusion obtained from this algorithm is that between $O(INF^{SC})$ and $O(INF^{HCD})$ the addition of HCD rules has a linear cost (given that the number of domain fluents u remain constant). In a nutshell, if we extend a given domain axiomatization with epistemic notions, the transition from possible worlds to DECKT introduces a linear increase in computational effort to the number of HCD for each reasoning task, but at the same time an exponential decrease of reasoning tasks.

A.4 Computing the Number of State Constraints

Let n be the number of fluents of a domain. In case of complete lack of knowledge all possible worlds will be 2^n. Let us now investigate how the number of state constraints is related to the number of fluents. Attempting a naive approach that will help us understand the procedure, we can start by enumerating the number of disjunctions that can be created with n fluents with only constraint that a fluent and its negation cannot appear in the same negation. Hence, having, for instance, fluents f_1, f_2 we can create an exponential number of disjunctions: $(f_1 \vee f_2), (f_1 \vee \neg f_2), (\neg f_1 \vee f_2), (\neg f_1 \vee \neg f_2)$ and no other, from which we can construct all possible state constraints. In other words, with two fluents we can create four disjunctions. Nevertheless, with more fluents we should consider not only the set of disjunctions made by all fluents, but also by any combination of their subsets: four disjunctions for each 2-fluent combination, eight disjunctions for each 3-fluent combination, etc. following the exponential creation of disjunctions with the number of fluents available. This leads us to the formula $\sum_{r=2}^{n}(\frac{n!}{r!(n-r)!} * 2^r)$, where variable r denotes the number of fluents that are used to create disjunctions in each step (which explains why r is initiated

as 2). Intuitively, the above formula declares that for each combination of fluents involved in the initial formula, we get an exponential number of state constraints to the number of fluents considered in the combination. Each 2-fluent combination gives 2^2 potential state constraints and there are $(\frac{n!}{2!(n-2)!})$ such combinations, each 3-fluent combination gives 2^3 potential state constraints and there are $(\frac{n!}{3!(n-3)!})$ such combinations, etc.

Apparently, all these combinations create a vast set of state constraints. Yet, not all of them can be considered as distinct in a domain axiomatization; the majority of them are redundant and reduce to smaller constraints, based on two observations. First, any two disjunctions containing the one a fluent and the other the negation of that fluent are merged into a new disjunction having all but that fluent as its components. Second, a disjunction having both a term and its negation is trivial truth and can be eliminated. Considering this analysis, it can be seen that the exponential number of disjunctions for each combination of fluents mentioned before can no longer hold. In fact, in each set only one disjunction can be accounted for at a time. For the disjunctions created in the previous example using fluents f_1, f_2, for instance, we can only include one in a particular domain, otherwise they will be reduced into a single fluent or trivial truths; apart from one, all interactions between larger disjunctions degrade into smaller that successively interact with those already existent and get eliminated. Therefore, we conclude that having n fluents we can construct, in the worst case, a number of state constraints given by the formula:

$$\text{(A4.1)} \sum_{r=2}^{n-1}\left(\frac{n!}{r!(n-r)!}\right) + 1 = 2^n - n - 1$$

given that it can be shown $\sum_{r=1}^{n} \frac{n!}{r!(n-r)!} = 2^n - 1$. The aforementioned result states that, as in possible worlds, the number of state constraints possible in the worst case is exponential to the number of fluents.

Appendix B

Source Code

Appendix B presents sample code of examples designed for execution with DECReasoner, as well as for the custom Event Calculus reasoner we have implemented for JESS. It provides evidence as to how the two different approaches discussed in Section 7.3 have been implemented and how similar epistemic notions have been modeled within JESS.

B.1 Syntactically Extended Epistemic Fluents within DECReasoner

Below is the code for a non-deterministic domain that provides two different models as output. The axiomatization follows the principles already discussed in previous sections; for each fluent we introduce (syntactically) new fluents that capture the intuition of the Knowledge Theory. Certain trivial modifications have been applied to reduce unnecessary models or to comply with the reasoner's requirements. For instance, we do not allow for the $Opened(door)$ fluent to be released for more that 1 timepoints.

Detailed information about the coding can be found in ([Mueller 2006] ch.13); as a brief guide, note that variables inside square brackets ([,]) within a formula denote universal quantification, whereas plain brackets ({,}) denote existential quantification, the "!" character denotes negation and lines starting with ";" indicate commends.

```
; Pass Through Door Example with Nondeterministic actions
load foundations/Root.e
load foundations/EC.e

sort door sort room
```

```
;--------------------------------
fluent Opened(door)
fluent Knows_Opened(door)
fluent Knows_NotOpened(door)

;Kw Definition
fluent Kw_Opened(door)
noninertial Kw_Opened
[door,time] HoldsAt(Kw_Opened(door),time) <->
HoldsAt(Knows_Opened(door),time) |
HoldsAt(Knows_NotOpened(door),time).

fluent KP_Opened(door)
fluent KP_NotOpened(door)

;Knowledge Axiom K
[door,time] HoldsAt(Knows_Opened(door),time) ->
HoldsAt(Opened(door),time).
[door,time]HoldsAt(Knows_NotOpened(door),time) ->
!HoldsAt(Opened(door),time).

;KT1
noninertial Knows_Opened
noninertial Knows_NotOpened

;KT2.1
[door,time] HoldsAt(KP_Opened(door),time) ->
HoldsAt(Knows_Opened(door),time).
;KT2.2
[door,time] HoldsAt(KP_NotOpened(door),time) ->
HoldsAt(Knows_NotOpened(door),time).

;KT7
[door,time] !HoldsAt(KP_Opened(door),time) &
```

```
!HoldsAt(KP_NotOpened(door),time) -> !HoldsAt(Kw_Opened(door),time).

;KT4
event Sense_Opened(door)
event Sense_OpenedT(door)
event Sense_OpenedF(door)
[door,time] Happens(Sense_Opened(door),time) ->
Happens(Sense_OpenedT(door),time) |
Happens(Sense_OpenedF(door),time).

[door,time] Initiates(Sense_OpenedT(door),KP_Opened(door),time).
[door,time] Initiates(Sense_OpenedF(door),KP_NotOpened(door),time).

;---------------------------------------------------------
fluent InRoom(room)
fluent Knows_InRoom(room)
fluent Knows_NotInRoom(room)

;Kw Definition
fluent Kw_InRoom(room)
noninertial Kw_InRoom
[room,time] HoldsAt(Kw_InRoom(room),time) <->
HoldsAt(Knows_InRoom(room),time) |
HoldsAt(Knows_NotInRoom(room),time).

fluent KP_InRoom(room)
fluent KP_NotInRoom(room)

;Knowledge Axiom K
[room,time] HoldsAt(Knows_InRoom(room),time) ->
HoldsAt(InRoom(room),time).
[room,time] HoldsAt(Knows_NotInRoom(room),time) ->
!HoldsAt(InRoom(room),time).
```

```
;KT1
noninertial Knows_InRoom
noninertial Knows_NotInRoom

;KT2.1
[room,time] HoldsAt(KP_InRoom(room),time) ->
HoldsAt(Knows_InRoom(room),time).
;KT2.2
[room,time]
HoldsAt(KP_NotInRoom(room),time) ->
HoldsAt(Knows_NotInRoom(room),time).

;KT7
[room,time] !HoldsAt(KP_InRoom(room),time) &
!HoldsAt(KP_NotInRoom(room),time) -> !HoldsAt(Kw_InRoom(room),time).

;-----------------------------------------------------------
event Speak()
[door,time] Releases(Speak(),Opened(door),time).

;KT5.3
[door,time] Happens(Speak(),time) ->
Terminates(Speak(),KP_Opened(door),time).
[door,time] Happens(Speak(),time) ->
Terminates(Speak(),KP_NotOpened(door),time).

;-----------------------------------------------------------
event PassThrough(door,room)
[door,room,time] HoldsAt(Opened(door),time) ->
Initiates(PassThrough(door,room),InRoom(room),time).

;KT3.1
[door,room,time] HoldsAt(Knows_Opened(door),time) &
Happens(PassThrough(door,room),time)->
```

```
Initiates(PassThrough(door,room),KP_InRoom(room),time).

;KT3.2
[door,room,time] HoldsAt(Knows_Opened(door),time) &
Happens(PassThrough(door,room),time)->
Terminates(PassThrough(door,room),KP_NotInRoom(room),time).

;KT5.1
[door,room,time] !HoldsAt(Kw_Opened(door),time) &
!HoldsAt(Knows_NotOpened(door),time) &
!HoldsAt(Knows_InRoom(room),time) &
Happens(PassThrough(door,room),time)->
Terminates(PassThrough(door,room),KP_InRoom(room),time).

[door,room,time] !HoldsAt(Kw_Opened(door),time) &
!HoldsAt(Knows_NotOpened(door),time) &
!HoldsAt(Knows_InRoom(room),time) &
Happens(PassThrough(door,room),time)->
Terminates(PassThrough(door,room),KP_NotInRoom(room),time).

;-------------------------------------------------------
;Observations and Narrative

door D1
room R1
HoldsAt(KP_NotOpened(D1),0).
HoldsAt(KP_NotInRoom(R1),0).

;To reduce unnecessary models, we initiate/terminate fluent Opened
;from being released.
event X(door)
[door,time] HoldsAt(Opened(door),time) ->
Initiates(X(door),Opened(door),time).
[door,time] !HoldsAt(Opened(door),time) ->
```

```
Terminates(X(door),Opened(door),time).

;If we cannot use completion, an alternative approach is the
following
Happens(Speak(),0).
Happens(Sense_Opened(D1),1).
Happens(PassThrough(D1,R1),2).

[event] event!= Speak() -> !Happens(event,0).
[event] event!= X(D1) & event!= Sense_Opened(D1) & event!=
Sense_OpenedT(D1) & event!= Sense_OpenedF(D1) -> !Happens(event,1).
[event] event!= PassThrough(D1,R1) -> !Happens(event,2).

range time 0 3
range offset 1 1
```

The output of the program's execution is shown below, where 2 models are produced, due to the non-deterministic effect of the *Speak*() action at timepoint 0. Initially, only the fluents that are true are presented. Moreover, in all future timepoints only the fluents that change from one timepoint to the next are shown; fluents that are added start with the "+" character and fluents that are retracted start with the "-" character. Some irrelevant information has been omitted.

```
Copyright (c) 2005 IBM Corporation and others. All rights reserved.
This program and the accompanying materials are made available under
the terms of the Common Public License v1.0 which accompanies this
distribution, and is available at
http://www.eclipse.org/legal/cpl-v10.html
Contributors: IBM - Initial implementation

Discrete Event Calculus Reasoner 1.0
loading deckt/PassThroughDoor.e
loading foundations/Root.e
loading foundations/EC.e
```

```
114 variables and 352 clauses
relsat solver
2 models
---
model 1:
0
KP_NotInRoom(R1).
KP_NotOpened(D1).
Knows_NotInRoom(R1).
Knows_NotOpened(D1).
Kw_InRoom(R1).
Kw_Opened(D1).
Happens(Speak(), 0).
1
-KP_NotOpened(D1).
-Knows_NotOpened(D1).
-Kw_Opened(D1).
Happens(Sense_Opened(D1), 1).
Happens(Sense_OpenedF(D1), 1).
Happens(X(D1), 1).
2
+KP_NotOpened(D1).
+Knows_NotOpened(D1).
+Kw_Opened(D1).
Happens(PassThrough(D1, R1), 2).
3
---
model 2:
0
KP_NotInRoom(R1).
KP_NotOpened(D1).
Knows_NotInRoom(R1).
Knows_NotOpened(D1).
Kw_InRoom(R1).
```

```
Kw_Opened(D1).
Happens(Speak(), 0).
1
-KP_NotOpened(D1).
-Knows_NotOpened(D1).
-Kw_Opened(D1).
+Opened(D1).
Happens(Sense_Opened(D1), 1).
Happens(Sense_OpenedT(D1), 1).
Happens(X(D1), 1).
2
+KP_Opened(D1).
+Knows_Opened(D1).
+Kw_Opened(D1).
Happens(PassThrough(D1, R1), 2).
3
-KP_NotInRoom(R1).
-Knows_NotInRoom(R1).
+InRoom(R1).
+KP_InRoom(R1).
+Knows_InRoom(R1).
EC: 7 predicates, 0 functions, 0 fluents, 0 events, 0 axioms
PassThroughDoor: 0 predicates, 0 functions, 12 fluents, 6 events, 34
axioms
Root: 0 predicates, 0 functions, 0 fluents, 0 events, 0 axioms
encoding 0.1s solution 0.0s total 0.4s
```

B.2 Extending DECReasoner's Ontology

Below is the code augmented to DECReasoner that extends the foundational axioms with a treatment of epistemic fluents. Examples can be found online at *http://www.csd.uoc.gr/~patkos/deckt.htm*.

```
reified sort kfluent
```

```
kfluent Knows(fluent,time)
kfluent KnowsNot(fluent,time)
kfluent Kw(fluent,time)

[fluent,time] Kw(fluent,time) <->
Knows(fluent,time) | KnowsNot(fluent,time).

reified sort epfluent
epfluent KP(fluent)
epfluent KPNot(fluent)

predicate KHoldsAt(epfluent,time)
predicate KInitiates(event,epfluent,time)
predicate KTerminates(event,epfluent,time)

; --------------------  Inertia of KHoldsAt  --------------------
[epfluent,time] (KHoldsAt(epfluent,time) &
 !({event} Happens(event,time) & KTerminates(event,epfluent,time))) ->
KHoldsAt(epfluent,time+1).

[epfluent,time] (!KHoldsAt(epfluent,time) &
 !({event} Happens(event,time) & KInitiates(event,epfluent,time))) ->
!KHoldsAt(epfluent,time+1).

; -------------- Influence of Events on Fluents ----------------
[event,epfluent,time] (Happens(event,time) &
KInitiates(event,epfluent,time)) -> KHoldsAt(epfluent,time+1).

[event,epfluent,time] (Happens(event,time) &
KTerminates(event,epfluent,time)) -> !KHoldsAt(epfluent,time+1).

; --------------------  (KT2) Axiom  --------------------
[fluent,time] KHoldsAt(KP(fluent),time) -> Knows(fluent,time).
[fluent,time] KHoldsAt(KPNot(fluent),time) -> KnowsNot(fluent,time).
```

```
; --------------------  (KT7) Axiom  --------------------
[fluent,time] Kw(fluent,time) <-> KHoldsAt(KP(fluent),time) |
KHoldsAt(KPNot(fluent),time).

; --------------------  (T) Knowledge Axiom  --------------------
[fluent,time] Knows(fluent,time) -> HoldsAt(fluent,time).
[fluent,time] KnowsNot(fluent,time) -> !HoldsAt(fluent,time).
```

B.3 Jess-based Event Calculus Reasoner

Below we show how (DEC5) is modeled as a rule within the Jess environment of the newly developed reasoner. The complete source code can be found in *http://www.csd.uoc.gr/~patkos/deckt.htm*. Variables start with a question mark (?). Inside the body of the rule, all variables are assigned values based on the facts stored in the memory of Jess. For instance, variable *t* is assigned the current timepoint of the reasoning cycle, as declared by the fact *Time* stored in the KB, which is increased after each cycle.

```
; DEC Inertia of HoldsAt Axiom

(defrule EVENTCALC::InertiaOfHoldsAtAxiom
(Time (tpoint ?t))
?holdsAtAxiom <- (EC    (predicate HoldsAt)
                        (fluent ?fluent)
                        (time ?t&:(< ?t ?*Endtime*)))
(not (and   (EC (predicate Happens)
                (event ?event)
                (time ?t))
        (or (EC (predicate Terminates)
                (event ?event)
                (fluent ?fluent)
                (time ?t)
            )
```

```
            (EC (predicate Releases)
                (event ?event)
                (fluent ?fluent)
                (time ?t)
            )
        )
    )
) (not (EC (predicate ReleasedAt)
            (fluent ?fluent)
            (time ?t)))
=> (duplicate ?holdsAtAxiom (time (+ 1 ?t))))
```

Next, is a rule encoding the trigger axiom for the *Activated* fluent of Shanahan's circuit. In brief, the rule states that if the fluent $Activated(R)$, captured by variable $prec1$, does not hold, whereas fluents $Closed(S1)$, $Closed(S2)$, $Closed(S3)$, captured by variables $prec2$, $prec3$, $prec4$ respectively, do hold, then predicate $Happens(Activate(R), t)$ should be included in the knowledge base, where t denotes the current timepoint.

```
(defrule MAIN::HappActivated
(declare (salience 570))
(Time (tpoint ?t))
    ?prec1 <- (fluent (name Activated) (arg R))
    (not (EC (predicate HoldsAt) (fluent ?prec1) (time ?t)))
    ?prec2 <- (fluent (name Closed) (arg S1))
    (EC (predicate HoldsAt) (fluent ?prec2) (time ?t))
    ?prec3 <- (fluent (name Closed) (arg S2))
    (EC (predicate HoldsAt) (fluent ?prec3) (time ?t))
    ?prec4 <- (fluent (name Closed) (arg S3))
    (EC (predicate HoldsAt) (fluent ?prec4) (time ?t))
    ?event <- (event (name Activate) (arg R))
(not (EC (predicate Happens) (event ?event) (time ?t)))    =>
(assert (EC (predicate Happens) (event ?event) (time ?t))))
```

The epistemic treatment extents the basic ontology with epistemic predicates. The reasoner does not make any distinction in the treatment of fluents, still the new axioms of

DECKT need to be modeled. Below, for instance, is the (KT6.2.4) axiom. If there are two HCDs that are related with the transitive relation and they may be destroyed, a new HCD is asserted that unifies their fluents, apart from the one that may have changed.

```
(defrule KT6.2.4 (declare (salience 550)) (Time (tpoint?t))
    (EC (predicate Happens)
        (event ?e)
        (time ?t) )
    (EC (predicate KmAffect)
        (event ?e)
        (posLtrs ?effect)
        (time ?t) )
    (EC (predicate HoldsAt)
        (epistemic KP)
        (posLtrs $?pf1&:(neq FALSE (member$ ?effect $?pf1)))
        (negLtrs $?nf1)
        (time ?t) )
    (test (< 1 (+ (length$ $?pf1) (length$ $?nf1))))
    (EC (predicate HoldsAt)
        (epistemic KP)
        (posLtrs $?pf2)
        (negLtrs $?nf2&:(neq FALSE (member$ ?effect $?nf2)))
        (time ?t) )
    (test (< 1 (+ (length$ $?pf2) (length$ $?nf2))))
    (test (neq $?nf1 $?pf2)) ;otherwise we have a trivially true disjunction
=>
    (bind ?del_f_place1 (member$ ?effect $?pf1))
    (bind ?del_f_place2 (member$ ?effect $?nf2))
(assert (EC (predicate Initiates)
            (epistemic KP)
            (event ?e)
            (posLtrs (union$ (delete$ $?pf1 ?del_f_place1 ?del_f_place1) $?pf2))
            (negLtrs (union$ $?nf1 (delete$ $?nf2 ?del_f_place2 ?del_f_place2)))
            (time ?t) )))
```

Bibliography

[Abowd 1999] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith and Pete Steggles. *Towards a Better Understanding of Context and Context-Awareness*. In HUC'99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 304–307, London, UK, 1999. Springer-Verlag.

[Adaikkalavan 2006] Raman Adaikkalavan and Sharma Chakravarthy. *SnoopIB: Interval-based Event Specification and Detection for Active Databases*. Data and Knowledge Engineering (DKE), vol. 59, no. 1, pages 139–165, 2006.

[Amir 2003] Eyal Amir and Stuart J. Russell. *Logical Filtering*. In IJCAI'03: International Joint Conferences on Artificial Intelligence, pages 75–82, 2003.

[Baader 2005] Franz Baader, Carsten Lutz, Maja Miličic, Ulrike Sattler and Frank Wolter. *Integrating Description Logics and Action Formalisms: First Results*. In AAAI'05: Proceedings of the 20th national conference on Artificial intelligence, pages 572–577. AAAI Press, 2005.

[Baier 2006] Jorge A. Baier and Sheila A. McIlraith. *On Planning with Programs that Sense*. In KR'06: Tenth International Conference on Principles of Knowledge Representation and Reasoning, pages 492–502, 2006.

[Baldoni 2001] Matteo Baldoni, Laura Giordano, Alberto Martelli and Viviana Patti. *Reasoning about Complex Actions with Incomplete Knowledge: A Modal Approach*. In ICTCS '01: Proceedings of the 7th Italian Conference on Theoretical Computer Science, pages 405–425, London, UK, 2001. Springer-Verlag.

[Baldoni 2004] Matteo Baldoni, Alberto Martelli, Viviana Patti and Laura Giordano. *Programming Rational Agents in a Modal Action Logic*. Annals of Mathematics and Artificial Intelligence, vol. 41, no. 2-4, pages 207–257, 2004.

[Blythe 1999] Jim Blythe. *Decision-Theoretic Planning*. The AI Magazine, vol. 20, no. 2, pages 37–54, 1999.

[Boutilier 1999] Craig Boutilier, Thomas Dean and Steve Hanks. *Decision-Theoretic Planning: Structural Assumptions and Computational Leverage*. Journal of Artificial Intelligence Research, vol. 11, pages 1–94, 1999.

[Bresina 2002] John Bresina, Richard Dearden, Nicolas Meuleau, Sailesh Ramkrishnan, David Smith and Rich Washington. *Planning under Continuous Time and Resource Uncertainty: A Challenge for AI*. In UAI-02: Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence, pages 77–84, San Francisco, CA, 2002. Morgan Kaufmann.

[Cervesato 2000] Iliano Cervesato, Massimo Franceschet and Angelo Montanari. *A Guided Tour through Some Extensions of the Event Calculus*. Computational Intelligence, vol. 16, no. 2, pages 307–347, 2000.

[Chittaro 1996] Luca Chittaro and Angelo Montanari. *Efficient Temporal Reasoning in the Cached Event Calculus*. Computational Intelligence, vol. 12, pages 359–382, 1996.

[Chittaro 2000] Luca Chittaro and Angelo Montanari. *Temporal Representation and Reasoning in Artificial Intelligence: Issues and Approaches*. Annals of Mathematics and AI, vol. 28, no. 1-4, pages 47–106, 2000.

[Classen 2006] Jens Classen and Gerhard Lakemeyer. *A Semantics for ADL as Progression in the Situation Calculus*. In Proceedings of the 11th Workshop on Nonmonotonic Reasoning, 2006.

[Claßen 2009] Jens Claßen and Gerhard Lakemeyer. *Tractable First-Order Golog with Disjunctive Knowledge Bases*. In Ninth International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense 2009), Toronto, Canada, 2009.

[De Giacomo 2000] Giuseppe De Giacomo, Yves Lespérance and Hector J. Levesque. *ConGolog, a Concurrent Programming Language based on the Situation Calculus*. Artificial Intelligence, vol. 121, no. 1-2, pages 109–169, 2000.

[de Weerdt 2005] Mathijs de Weerdt, Adriaan ter Mors and Cees Witteveen. *Multi-agent Planning: An Introduction to Planning and Coordination*. In Handouts of the European Agent Summer School, pages 1–32, 2005.

[Demolombe 2000] R. Demolombe and MP Pozos-Parra. *A Simple and Tractable Extension of Situation Calculus to Epistemic Logic*. ISMIS-00: Twelfth International Symposium on Methodologies for Intelligent Systems, pages 515–524, 2000.

[Denecker 1998] Marc Denecker, Daniele Theseider Dupré and Kristof Van Belleghem. *An Inductive Definition Approach to Ramifications*. Electronic Transaction on Artificial Intelligence, vol. 2, pages 25–67, 1998.

[Dimitris Grammenos 2009] Antonis A. Argyros Constantine Stephanidis Dimitris Grammenos Xenophon Zabulis. *FORTH-ICS Internal RTD Programme Ambient Intelligence and Smart Environments*. In AMI'09: Proceedings of the 3rd European Conference on Ambient Intelligence , 2009.

[Dimopoulos 2004] Yannis Dimopoulos, Antonis C. Kakas and Loizos Michael. *Reasoning About Actions and Change in Answer Set Programming*. In LPNMR'04: 7th International Conference on Logic Programming and Nonmonotonic Reasoning, pages 61–73, 2004.

[Eiter 2003a] Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer and Axel Polleres. *A Logic Programming Approach to Knowledge-State Planning, II: the DLVk System*. Artificial Intelligence, vol. 144, no. 1-2, pages 157–211, 2003.

[Eiter 2003b] Thomas Eiter and Thomas Lukasiewicz. *Probabilistic Reasoning about Actions in Nonmonotonic Causal Theories*. In Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, pages 192–199, 2003.

[Eiter 2004] Thomas Eiter, Wolfgang Faber, Gerald Pfeifer and Axel Polleres. *Declarative Planning and Knowledge Representation in an Action Language*. 2004.

[Fagin 2003] Ronald Fagin, Joseph Y. Halpern, Yoram Moses and Moshe Y. Vardi. Reasoning About Knowledge. MIT Press, Cambridge, MA, USA, 2003.

[Federico Chesani 2009] Marco Montali Paolo Torroni Federico Chesani Paola Mello. *Commitment Tracking via the Reactive Event Calculus*. In IJCAI-09: Twenty-first International Joint Conference on Artificial Intelligence, pages 91–96, 2009.

[Ferraris 2000] Paolo Ferraris and Enrico Giunchiglia. *Planning as Satisfiability in Nondeterministic Domains*. In AAAI'00: Proceedings of the Seventeenth National

Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pages 748–753. AAAI Press / The MIT Press, 2000.

[Fikes 1971] Richard Fikes and Nils J. Nilsson. *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving*. In IJCAI'71: International Joint Conferences on Artificial Intelligence, pages 608–620, 1971.

[Finzi 2005] Alberto Finzi and Thomas Lukasiewicz. *Game-Theoretic Reasoning About Actions in Nonmonotonic Causal Theories*. In LPNMR'05: 8th International Conference on Logic Programming and Nonmonotonic Reasoning, pages 185–197, 2005.

[Fisher 2005] Michael Fisher. *MetateM: The Story so Far*. In PROMAS'05: Third International Workshop on Programming Multi-Agent Systems, pages 3–22, 2005.

[Forth 2004] Jeremy Forth and Murray Shanahan. *Indirect and Conditional Sensing in the Event Calculus*. In Ramon Lo'pez de Ma'ntaras and Lorenza Saitta, editeurs, ECAI, pages 900–904. IOS Press, 2004.

[Forth 2007] Jeremy Forth and Rob Miller. *Ramifications: An Extension and Correspondence Result for the Event Calculus*. Journal of Logic and Computation, vol. 17, no. 4, pages 639–685, 2007.

[Funge 1999] John Funge. *Representing Knowledge within the Situation Calculus using Interval-valued Epitemic Fluents*. Journal of Reliable Computing, vol. 5, no. 1, pages 35–61, 1999.

[Gandon 2004] Fabien L. Gandon and Norman M. Sadeh. *Semantic Web Technologies to Reconcile Privacy and Context Awareness*. Journal of Web Semantics, vol. 1, no. 3, pages 241–260, 2004.

[Gelfond 1998] Michael Gelfond and Vladimir Lifschitz. *Action Languages*. Electronic Transactions on AI, vol. 3, 1998.

[Ghallab 2004] M. Ghallab, D. Nau and P. Traverso. Automated Planning: Theory and Practice. Morgan Kaufmann, 2004.

[Giacomo 1999] Giuseppe De Giacomo and Hector J. Levesque. *An Incremental Interpreter for High-Level Programs with Sensing*. pages 86–102, 1999.

[Giunchiglia 1998] Enrico Giunchiglia and Vladimir Lifschitz. *An Action Language based on Causal Explanation: Preliminary Report*. In AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, pages 623–630. American Association for Artificial Intelligence, 1998.

[Giunchiglia 2004] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain and Hudson Turner. *Nonmonotonic Causal Theories*. Artificial Intelligence, vol. 153, no. 1-2, pages 49–104, 2004.

[Greenfield 2006] Adam Greenfield. Everyware: The Dawning Age of Ubiquitous Computing. Peachpit Press, Berkeley, CA, USA, 2006.

[Grosskreutz 2000] Henrik Grosskreutz and Gerhard Lakemeyer. *Turning High-Level Plans into Robot Programs in Uncertain Domains*. In ECAI'00: European Conference on Artificial Intelligence, pages 548–552, 2000.

[Gu 2007] Yilan Gu and Mikhail Soutchanski. *Decidable Reasoning in a Modified Situation Calculus*. In IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence, pages 1891–1897, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[Halpern 1985] J. Halpern and Y. Moses. *Towards a Theory of Knowledge and Ignorance: Preliminary Report*. pages 459–476, 1985.

[Halpern 2007] J.Y. Halpern and R. Pucella. *Dealing with Logical Omniscience*. Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge, pages 169–176, 2007.

[Halpern 2008] J.Y. Halpern and L.C. Rêgo. *Interactive Unawareness Revisited*. Games and Economic Behavior, vol. 62, no. 1, pages 232–262, 2008.

[Helal 2009] Sumi Helal and Chao Chen. *The Gator Tech Smart House: Enabling Technologies and Lessons Learned*. In i-CREATe '09: Proceedings of the 3rd International Convention on Rehabilitation Engineering & Assistive Technology, pages 1–4, New York, NY, USA, 2009. ACM.

[Hintikka 1962] Jaakko Hintikka. Knowledge and Belief: An Introduction to the Logic of the Two Notions. Cornell University Press, Ithaca, N. Y., 1962.

[Hofer 2003] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann and Werner Retschitzegger. *Context-Awareness on Mobile Devices - the Hydrogen Approach*. In HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9, pages 292–301, Washington, DC, USA, 2003. IEEE Computer Society.

[Hölldobler 1990] Steffen Hölldobler and Josef Schneeberger. *A New Deductive Approach to Planning*. New Generation Computing, vol. 8, no. 3, pages 225–244, 1990.

[Hölldobler 2000] Steffen Hölldobler and Dietrich Kuske. *The Boundary between Decidable and Undecidable Fragments of the Fluent Calculus*. In LPAR'00: Logic for Programming and Automated Reasoning, 7th International Conference, pages 436–450, 2000.

[Immerman 1986] Neil Immerman. *Relational Queries Computable in Polynomial Time*. Information and Control, vol. 68, no. 1-3, pages 86–104, 1986.

[Jin 2004] Yi Jin and Michael Thielscher. *Representing Beliefs in the Fluent Calculus*. In ECAI'04: European Conference on Artificial Intelligence, pages 823–827, 2004.

[Kakas 1997] Antonis C. Kakas and Rob Miller. *A Simple Declarative Language for Describing Narratives With Actions*. The Journal of Logic Programming, vol. 31, no. 1-3, pages 157–200, 1997.

[Kakas 2000] Antonis Kakas, Rob Miller and Francesca Toni. *E-RES - A System for Reasoning about Actions, Events and Observations*. In NMR'00: 8th International Symposium on Nonmonotonic Reasoning, 2000.

[Kakas 2002] Antonis C. Kakas and Loizos Michael. *Modeling Complex Domains of Actions and Change*. In NMR'02: 9th International Workshop on Non-Monotonic Reasoning, pages 380–390, 2002.

[Kelly 2008] Ryan F. Kelly and Adrian R. Pearce. *Complex Epistemic Modalities in the Situation Calculus*. In KR'08: International Conference on Principles of Knowledge Representation and Reasoning, pages 611–620, 2008.

[Kim 2009] Tae-Won Kim, Joohyung Lee and Ravi Palla. *Circumscriptive Event Calculus as Answer Set Programming*. In IJCAI-09: Twenty-first International Joint Conference on Artificial Intelligence, pages 823–829, 2009.

[Konolige 1986] K. Konolige. A Deduction Model of Belief. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1986.

[Kowalski 1986] R Kowalski and M Sergot. *A Logic-based Calculus of Events*. New Generation Computing, vol. 4, no. 1, pages 67–95, 1986.

[Kowalski 1997] Robert A. Kowalski and Fariba Sadri. *Reconciling the Event Calculus With the Situation Calculus*. Journal of Logic Programming, vol. 31, no. 1-3, pages 39–58, 1997.

[Kristof Van Belleghem and Marc Denecker and Danny De Schreye 1995] Kristof Van Belleghem and Marc Denecker and Danny De Schreye. *Combining Situation Calculus and Event Calculus*. In International Conference on Logic Programming, pages 83–97, 1995.

[Lakemeyer 1998] Gerhard Lakemeyer and Hector J. Levesque. *AOL: A Logic of Acting, Sensing, Knowing, and Only Knowing*. In Principles of Knowledge Representation and Reasoning, pages 316–329, 1998.

[Lehmann 2000] Helko Lehmann and Michael Leuschel. *Decidability Results for the Propositional Fluent Calculus*. In CL '00: Proceedings of the First International Conference on Computational Logic, pages 762–776, London, UK, 2000. Springer-Verlag.

[Lespérance 2000] Yves Lespérance, Hector J. Levesque, Fangzhen Lin and Richard B. Scherl. *Ability and Knowing How in the Situation Calculus*. Studia Logica, vol. 66, no. 1, pages 165–186, 2000.

[Levesque 1996] Hector J. Levesque. *What Is Planning in the Presence of Sensing?* In AAAI'96: Proceedings of 13th National Conference on Artificial Intelligence, volume 2, pages 1139–1146, 1996.

[Levesque 1997] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin and Richard B. Scherl. *Golog: A Logic Programming Language for Dynamic Domains*. Journal of Logic Programming, vol. 31, no. 1-3, pages 59–83, 1997.

[Levesque 1998] H. Levesque, F. Pirri and R. Reiter. *Foundations for the Situation Calculus*. In Linkoping Electronic Articles in Computer and Information Science, volume 3, 1998.

[Levy 1998] F. Levy and J.J. Quantz. *Representing Beliefs in a Situated Event Calculus*. Proceedings of the Thirteenth European Conference on Artificial Intelligence, pages 547–551, 1998.

[Lifschitz 1986] Vladimir Lifschitz. *On the Semantics of STRIPS*. In Michael P. Georgeff and Amy L. Lansky, editeurs, Reasoning about Actions and Plans: Proceedings of the 1986 Workshop, pages 1–9, Timberline, Oregon, June-July 1986. Morgan Kaufmann.

[Lifschitz 1994] V. Lifschitz. *Circumscription*. Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, pages 297–352, 1994.

[Lifschitz 1999] Vladimir Lifschitz. *Action Languages, Answer Sets and Planning*. The Logic Programming Paradigm: a 25 year perspective, vol. 4, no. 1, pages 357–373, 1999.

[Lin 1994] Fangzhen Lin and Ray Reiter. *State Constraints Revisited*. Journal of Logic and Computation, vol. 4, no. 5, pages 655–677, 1994.

[Liu 2004] Yongmei Liu, Gerhard Lakemeyer and Hector J. Levesque. *A Logic of Limited Belief for Reasoning with Disjunctive Information*. In 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR'04), pages 587–597, 2004.

[Liu 2005] Yongmei Liu and Hector J. Levesque. *Tractable Reasoning with Incomplete First-Order Knowledge in Dynamic Systems with Context-Dependent Actions*. In IJCAI'05: International Joint Conferences on Artificial Intelligence, pages 522–527, 2005.

[Liu 2006] Hongkai Liu, Carsten Lutz, Maja Milicic and Frank Wolter. *Reasoning about Actions using Description Logics with general TBoxes*. In Michael Fisher, Wiebe van der Hoek, Boris Konev and Alexei Lisitsa, editeurs, JELIA'06: 10th European Conference on Logics in Artificial Intelligence, Lecture Notes in Artificial Intelligence, pages 266–279. Springer-Verlag, 2006.

[Lobo 2001] Jorge Lobo, Gisela Mendez and Stuart R. Taylor. *Knowledge and the Action Description Language A*. Theory Pract. Log. Program., vol. 1, no. 2, pages 129–184, 2001.

[Martin 2003] Yves Martin. *The Concurrent, Continuous FLUX*. In IJCAI'03: International Joint Conferences on Artificial Intelligence, pages 1085–1090, 2003.

[Marzano 2003] S. Marzano and E. Aarts. The New Everyday View on Ambient Intelligence. Uitgeverij 010 Publishers, Rotterdam, The Netherlands, 2003.

[Mateus 2001] Paulo Mateus, António Pacheco, Javier Pinto, Amílcar Sernadas and Cristina Sernadas. *Probabilistic Situation Calculus*. Annals of Mathematics and Artificial Intelligence, vol. 32, no. 1-4, pages 393–431, 2001.

[McArthur 1988] G.L. McArthur. *Reasoning about Knowledge and Belief: A Survey*. Computational Intelligence, vol. 4, no. 3, pages 223–243, 1988.

[McCain 1995] Norman McCain and Hudson Turner. *A Causal Theory of Ramifications and Qualifications*. In IJCAI'95: International Joint Conferences on Artificial Intelligence, pages 1978–1984, 1995.

[McCain 1997] Norman McCain and Hudson Turner. *Causal Theories of Action and Change*. In AAAI'97: National Conference on Artificial intelligence, pages 460–465, 1997.

[McCarthy 1968] J. McCarthy. *Situations, Actions and Causal Laws*. In Stanford University. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968.

[McCarthy 1977] John L. McCarthy. *Epistemological Problems of Artificial Intelligence*. In IJCAI'77: International Joint Conferences on Artificial Intelligence, pages 1038–1044, 1977.

[McCarthy 1987] J. McCarthy and P. J. Hayes. *Some Philosophical Problems from the Standpoint of Artificial Intelligence*. pages 26–45, 1987.

[McIlraith 2000] S. McIlraith. *Integrating Actions and State Constraints: A Closed-Form Solution to the Ramification Problem (Sometimes)*. Artificial Intelligence, vol. 116, no. 1-2, pages 87–121, January 2000.

[Milicic 2007] Maja Milicic. *Planning in Action Formalisms based on DLs: First Results*. In DL'07: International Workshop on Description Logics, 2007.

[Miller 2002] Rob Miller and Murray Shanahan. *Some Alternative Formulations of the Event Calculus*. In Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II, pages 452–490, London, UK, 2002. Springer-Verlag.

[Moore 1985] R. C. Moore. *A Formal Theory of Knowledge and Action*. In Formal Theories of the Commonsense World, pages 319–358. J. Hobbs, R. Moore (Eds.), 1985.

[Moreno 1998] Antonio Moreno. *Avoiding Logical Omniscience and Perfect Reasoning: A Survey*. AI Communications, vol. 11, no. 2, pages 101–122, 1998.

[Mueller 2004] Erik T. Mueller. *Event Calculus Reasoning Through Satisfiability*. Journal of Logic and Computation, vol. 14, no. 5, pages 703–730, 2004.

[Mueller 2006] Erik Mueller. Commonsense Reasoning. Morgan Kaufmann, 1st édition, 2006.

[Mueller 2007a] Erik Mueller. *Automating Commonsense Reasoning Using the Event Calculus*. Communications of the ACM, no. (In Press), 2007.

[Mueller 2007b] Erik Mueller. *Discrete Event Calculus with Branching Time*. In 8th International Symposium on Logical Formalizations of Commonsense Reasoning, pages 126–131, 2007.

[National Research Council Staff 2001] National Research Council Staff. Embedded Everywhere: A Research Agenda for Networked Systems of Embedded Computers. National Academy Press, Washington, DC, USA, 2001.

[Papadakis 2002] Nikos Papadakis and Dimitris Plexousakis. *Actions with Duration and Constraints: the Ramification Problem in Temporal Databases*. In ICTAI '02: Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence, page 83, Washington, DC, USA, 2002. IEEE Computer Society.

[Paschke 2006] Adrian Paschke. *ECA-RuleML: An Approach Combining ECA Rules with Temporal Interval-based KR Event/Action Logics and Transactional Update Logics*. Computer Research Repository, vol. abs/cs/0610167, 2006.

[Patkos 2007a] Theodore Patkos, Antonis Bikakis, Grigoris Antoniou, Maria Papadopouli and Dimitris Plexousakis. *A Semantics-Based Framework for Context-Aware Ser-*

vices: Lessons Learned and Challenges. In UIC'07: 4th International Conference on Ubiquitous Intelligence and Computing, pages 839–848, 2007.

[Patkos 2007b] Theodore Patkos, Antonis Bikakis, Grigoris Antoniou, Maria Papadopouli and Dimitris Plexousakis. *Distributed AI for Ambient Intelligence: Issues and Approaches*. In AmI'07: Ambient Intelligence, European Conference, pages 159–176, 2007.

[Patkos 2008] Theodore Patkos and Dimitris Plexousakis. *A Theory of Action, Knowledge and Time in the Event Calculus*. In SETN '08: 5th Hellenic Conference on Artificial Intelligence, pages 226–238, Greece, 2008.

[Patkos 2009a] Theodore Patkos and Dimitris Plexousakis. *Reasoning with Knowledge, Action and Time in Dynamic and Uncertain Domains*. In IJCAI'09: Twenty-first International Joint Conference on Artificial Intelligence, pages 885–890, 2009.

[Patkos 2009b] Theodore Patkos and Dimitris Plexousakis. *Sensing Inertial and Continuously-Changing World Features*. In AIAI'09: Proceedings of the 5TH IFIP Conference on Artificial Intelligence Applications and Innovations, pages 379–388, 2009.

[Patkos 2010a] Theodore Patkos, Ioannis Chrysakis, Antonis Bikakis, Dimitris Plexousakis and Grigoris Antoniou. *A Reasoning Framework for Ambient Intelligence*. In SETN'10: 6th Hellenic Conference on AI, pages 213–222, 2010.

[Patkos 2010b] Theodore Patkos and Dimitris Plexousakis. *Reasoning about Potential Actions and Indirect Effects in Partially Observable and Uncertain Domains*. In under review for the Journal of Automated Reasoning, 2010.

[Patkos 2011] Theodore Patkos and Dimitris Plexousakis. *Epistemic Reasoning for Ambient Intelligence*. In under review for the 3rd International Conference on Agents and Artificial Intelligence, 2011.

[Pednault 1989] Edwin P. D. Pednault. *ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus*. In Proceedings of the first international conference on Principles of knowledge representation and reasoning, pages 324–332, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[Petrick 2002a] Ronald P. A. Petrick and Fahiem Bacchus. *A Knowledge-Based Approach to Planning with Incomplete Information and Sensing*. In Malik Ghallab, Joachim Hertzberg and Paolo Traverso, editeurs, AIPS-2002: Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling, pages 212–221, Menlo Park, CA, April 2002. AAAI Press.

[Petrick 2002b] Ronald P. A. Petrick and Hector J. Levesque. *Knowledge Equivalence in Combined Action Theories*. In KR'02: International Conference on Principles of Knowledge Representation and Reasoning, pages 303–314, 2002.

[Petrick 2004] Ronald P. A. Petrick and Fahiem Bacchus. *Extending the Knowledge-Based Approach to Planning with Incomplete Information and Sensing*. In Didier Dubois, Christopher Welty and Mary-Anne Williams, editeurs, KR'04: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, pages 613–622, Menlo Park, CA, June 2004. AAAI Press.

[Petrick 2008] Ronald P. A. Petrick. *Cartesian Situations and Knowledge Decomposition in the Situation Calculus*. In KR'08: International Conference on Principles of Knowledge Representation and Reasoning, pages 629–639, 2008.

[Pinto 1994] Javier Andre's Pinto. *Temporal Reasoning in the Situation Calculus*, 1994.

[Pinto 1995] Javier Pinto and Raymond Reiter. *Reasoning About Time in the Situation Calculus*. Annals of Mathematics and Artificial Intelligence, vol. 14, no. 2-4, pages 251–268, 1995.

[Pinto 1998] Javier Pinto. *Occurrences and Narratives as Constraints in the Branching Structure of the Situation Calculus*. Journal of Logic and Computation, vol. 8, no. 6, pages 777–808, 1998.

[Pinto 2000] Javier Pinto, Amílcar Sernadas, Cristina Sernadas and Paulo Mateus. *Non-Determinism and Uncertainty in the Situation Calculus*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 8, no. 2, pages 127–149, 2000.

[Pollack 1999] Martha Pollack and John F. Horty. *There's More to Life than Making Plans*. The AI Magazine, vol. 20, no. 4, pages 71–84, 1999.

[Prekop 2003] Paul Prekop and Mark Burnett. *Activities, Context and Ubiquitous Computing*. Computer Communications, vol. 26, no. 11, pages 1168 – 1176, 2003.

[Ramos 2008] Carlos Ramos, Juan Carlos Augusto and Daniel Shapiro. *Ambient Intelligence–the Next Step for Artificial Intelligence*. IEEE Intelligent Systems, vol. 23, no. 2, pages 15–18, 2008.

[Reiter 1991] Raymond Reiter. *The Frame Problem in the Situation Calculus: a Simple Solution (Sometimes) and a Completeness Result for Goal Regression*. pages 359–380, 1991.

[Reiter 1993] Raymond Reiter. *Proving Properties of States in the Situation Calculus*. Artificial Intelligence, vol. 64, no. 2, pages 337–351, 1993.

[Reiter 1996] R. Reiter. *Natural Actions, Concurrency and Continuous Time in the Situation Calculus*. In KR'96: Proceedings of the Fifth International Conference Principles of Knowledge Representation and Reasoning, pages 2–13, Cambridge, Massachusetts, U.S.A., November 1996.

[Reiter 2001a] R. Reiter. Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, 2001.

[Reiter 2001b] Ray Reiter. *On Knowledge-Based Programming with Sensing in the Situation Calculus*. ACM Transactions on Computational Logic, vol. 2, no. 4, pages 433–457, 2001.

[Rogers 2006] Yvonne Rogers. *Moving on from Weiser's Vision of Calm Computing: Engaging UbiComp Experiences*. In Ubicomp'06: International Conference on Ubiquitous Computing, pages 404–421, 2006.

[Russo 2002] Alessandra Russo, Rob Miller, Bashar Nuseibeh and Jeff Kramer. *An Abductive Approach for Analysing Event-Based Requirements Specifications*. In ICLP '02: Proceedings of the 18th International Conference on Logic Programming, pages 22–37, London, UK, 2002. Springer-Verlag.

[Sadeh 2006] N.M. Sadeh, F. Gandon and Oh Byung Kwon. *Ambient Intelligence: The MyCampus Experience*. In Book Chapter in Ambient Intelligence and Pervasive Computing. T. Vasilakos and W. Pedrycz, ArTech House, 2006.

[Sardina 2004] Sebastian Sardina, Giuseppe De Giacomo, Yves Lespérance and Hector J. Levesque. *On the Semantics of Deliberation in Indigolog—from Theory to Implementation*. Annals of Mathematics and Artificial Intelligence, vol. 41, no. 2-4, pages 259–299, 2004.

[Scherl 1993] Richard B. Scherl and Hector J. Levesque. *The Frame Problem and Knowledge-Producing Actions*. In AAAI'93: Proceedings of the Eleventh National Conference on Artificial Intelligence, pages 689–697, Washington, D.C., USA, 1993. AAAI Press/MIT Press.

[Scherl 2003] Richard B. Scherl and Hector J. Levesque. *Knowledge, Action, and the Frame Problem*. Artificial Intelligence, vol. 144, no. 1-2, pages 1–39, 2003.

[Scherl 2005] Richard B. Scherl. *Action, Belief Change and the Frame Problem: A Fluent Calculus Approach*. In Proceedings of the Sixth workshop on Nonmonotonic Reasoning, Action, and Change, August 2005.

[Schiffel 2006] Stephan Schiffel and Michael Thielscher. *Reconciling Situation Calculus and Fluent Calculus*. In AAAI'06: Proceedings of the Fourteenth National Conference on Artificial Intelligence, pages 287–292, Boston, MA, July 2006. AAAI Press.

[Schwind 1999] Camilla Schwind. *Causality in Action Theories*. Electronic Transactions on Artificial Intelligence, vol. 3, no. A, pages 27–50, 1999.

[Shanahan 1997] M. Shanahan. Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. MIT Press, 1997.

[Shanahan 1999a] M. Shanahan. *The Ramification Problem in the Event Calculus*. IJCAI'99: International Joint Conference on Artificial Intelligence, vol. 16, pages 140–146, 1999.

[Shanahan 1999b] Murray Shanahan. *The Event Calculus Explained*. Artificial Intelligence Today, vol. 1600, pages 409–431, 1999.

[Shanahan 2001] Murray Shanahan and Mark Witkowski. *High-Level Robot Control through Logic*. In ATAL '00: Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages, pages 104–121, London, UK, 2001. Springer-Verlag.

[Shapiro 2000] S. Shapiro, M. Pagnucco, Y. Lespénce and H. J. Levesque. *Iterated Belief Change in the Situation Calculus*. In A. G. Cohn, F. Giunchiglia and B. Selman, editeurs, KR'00: Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference, pages 527–538, San Francisco, CA, 2000. Morgan Kaufmann.

[Shapiro 2002] Steven Shapiro, Yves Lespérance and Hector J. Levesque. *The Cognitive Agents Specification Language and Verification Environment for Multiagent Systems*. In AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, pages 19–26, New York, NY, USA, 2002. ACM.

[Shirazi 2005] Afsaneh Shirazi and Eyal Amir. *First-Order Logical Filtering*. In IJCAI'05: Proceedings of the 19th International Joint Conference on Artificial Intelligence, pages 589–595, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[Sim 1997] K.M. Sim. *Epistemic Logic and Logical Omniscience: A Survey*. International Journal of Intelligent Systems, vol. 12, no. 1, pages 57–81, 1997.

[Son 2001] Tran Cao Son and Chitta Baral. *Formalizing Sensing Actions—A Transition Function based Approach*. Artificial Intelligence, vol. 125, no. 1-2, pages 19–91, 2001.

[Son 2005] Tran Cao Son, Phan Huy Tu, Michael Gelfond and A. Ricardo Morales. *Conformant Planning for Domains with Constraints: a New Approach*. In AAAI'05: Proceedings of the 20th National Conference on Artificial Intelligence, pages 1211–1216. AAAI Press, 2005.

[Ternovskaia 1999] Eugenia Ternovskaia. *Automata Theory for Reasoning about Actions*. In IJCAI'99: Proceedings of the 16th international joint conference on Artifical intelligence, pages 153–158, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[Thielscher 1997] Michael Thielscher. *Ramification and Causality*. Artificial Intelligence Journal, vol. 89, no. 1–2, pages 317–364, 1997.

[Thielscher 1998] Michael Thielscher. *Introduction to the Fluent Calculus*. Electronic Transactions on Artificial Intelligence, vol. 2, no. 3–4, pages 179–192, 1998.

[Thielscher 1999a] M. Thielscher. *Fluent Calculus Planning with Continuous Change*. Linköping University Electronic Articles in Computer and Information Science, vol. 4, no. 11, 1999.

[Thielscher 1999b] Michael Thielscher. *From Situation Calculus to Fluent Calculus: State Update Axioms as a Solution to the Inferential Frame Problem*. Artificial Intelligence, vol. 111, no. 1-2, pages 277–299, 1999.

[Thielscher 2000a] M. Thielscher. *The Fluent Calculus - A Specification Language for Robots with Sensors in Nondeterministic, Concurrent, and Ramifying Environments*. In Technical Report CL-2000-01, Artificial Intelligence Institute, Department of Computer Science, Dresden University of Technology, 2000.

[Thielscher 2000b] Michael Thielscher. Challenges for Action Theories: Solving the Ramification and Qualification Problem, volume 1775 of *LNAI*. Springer, 2000.

[Thielscher 2000c] Michael Thielscher. *Modeling Actions with Ramifications in Nondeterministic, Concurrent, and Continuous Domains - and a Case Study*. In AAAI'00: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pages 497–502. AAAI Press / The MIT Press, 2000.

[Thielscher 2000d] Michael Thielscher. *Representing the Knowledge of a Robot*. In A. Cohn, F. Giunchiglia and B. Selman, editeurs, KR'00: 7th International Conference on Principles of Knowledge Representation and Reasoning, pages 109–120, Breckenridge, CO, 2000.

[Thielscher 2001a] M. Thielscher. *Inferring Implicit State Knowledge and Plans with Sensing Actions*. Ki 2001: Advances in Artificial Intelligence: Joint German/Austrian Conference on AI, Vienna, Austria, September 19-21, 2001: Proceedings, 2001.

[Thielscher 2001b] M. Thielscher. *The Concurrent, Continuous Fluent Calculus*. Studia Logica, vol. 67, no. 3, pages 315–331, 2001.

[Thielscher 2005a] Michael Thielscher. *FLUX: A Logic Programming Method for Reasoning Agents*. Theory and Practice of Logic Programming, vol. 5, no. 4–5, pages 533–565, 2005.

[Thielscher 2005b] Michael Thielscher. *Handling Implication and Universal Quantification Constraints in FLUX*. In CP'05: 11th International Conference on Principles and Practice of Constraint Programming, pages 667–681, 2005.

[Thielscher 2010] Michael Thielscher. *A Unifying Action Calculus*. Artificial Intelligence Journal, To appear, 2010.

[Van Belleghem 1995] K. Van Belleghem, M. Denecker and D. De Schreye. *Combining Situation Calculus and Event Calculus*. Logic Programming: Proceedings of the Twelfth International Conference on Logic Programming, 1995.

[Van Belleghem 1997] K. Van Belleghem, M. Denecker and D. De Schreye. *On the Relation Between Situation Calculus and Event Calculus*. The Journal of Logic Programming, vol. 31, no. 1-3, pages 3–37, 1997.

[van Harmelen 2007] Frank van Harmelen, Vladimir Lifschitz and Bruce Porter. Handbook of Knowledge Representation. Elsevier Science, San Diego, USA, 2007.

[Vassos 2007] Stavros Vassos and Hector Levesque. *Progression of Situation Calculus Action Theories with Incomplete Information*. In IJCAI'07: International Joint Conferences on Artificial Intelligence, pages 2024–2029, 2007.

[Vassos 2008] Stavros Vassos and Hector J. Levesque. *On the Progression of Situation Calculus Basic Action Theories: Resolving a 10-year-old Conjecture*. In AAAI'08: Proceedings of the 23rd National Conference on Artificial Intelligence, pages 1004–1009. AAAI Press, 2008.

[Vassos 2009] Stavros Vassos, Stavros Sardina and Hector Levesque. *Progressing Basic Action Theories with non-Local Effect Actions*. In Gerhard Lakemeyer, Leora Morgenstern and Mary-Anne Williams, editeurs, CS'09: Proceedings of the Ninth International Symposium on Logical Formalizations of Commonsense Reasoning, pages 135Ű–140, Toronto, Canada, 2009.

[Vo 2005] Quoc Bao Vo and Norman Y. Foo. *Reasoning About Action: an Argumentation-Theoretic Approach*. Journal of Artificial Intelligence Research, vol. 24, no. 1, pages 465–518, 2005.

[Weiser 1991] M. Weiser. *The Computer for the 21st Century*. Scientific American, vol. 265, pages 94–104, 1991.

[Witkowski 2006] Thomas Witkowski and Michael Thielscher. *The Features-and-Fluents Semantics for the Fluent Calculus*. In P. Doherty, J. Mylopoulos and C. Welty, editeurs, Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR), pages 362–370, Lake District, UK, June 2006.

[Zimmerbaum 2001] Stephen Zimmerbaum and Richard B. Scherl. *Sensing Actions, Time, and Concurrency in the Situation Calculus*. In ATAL '00: Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages, pages 31–45, London, UK, 2001.