EPISTEMIC REASONING FOR AMBIENT INTELLIGENCE

Theodore Patkos and Dimitris Plexousakis
Institute of Computer Science, FO.R.T.H., Heraklion Crete, Greece

Keywords: Ambient intelligence, Epistemic reasoning, Reasoning about action and change.

Abstract: Ambient Intelligence is an emerging discipline that requires the integration of expertise from a multitude of scientific fields. The role of Artificial Intelligence is crucial not only for bringing intelligence to everyday environments, but also for providing the means for the different disciplines to collaborate. In this paper we highlight the importance of reasoning under partial observability in such dynamic and context-rich domains and illustrate the integration of an epistemic theory to an operational Ambient Intelligence infrastructure.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Ambient Intelligence (AmI) is an emerging discipline that introduces novel and highly demanding challenges. Within AmI environments a multitude of smart entities operate autonomously and perform actions, motivated by their own view of the user's current needs or intentions. Intelligent software agents operating on smart devices or rational robots that are assigned specific tasks are expected to perform long sequences of context dependent actions; still, context knowledge is not always available or accessible in AmI spaces:

- AmI domains are highly dynamic and adaptive to changes. Contextual information changes rapidly and in a fashion that is not easily prefigured,
- contextual knowledge may not be readily available. Continuously sensing all desirable information is impractical when executing long action sequences with resource constraint devices,
- context data often needs to be filtered and aggregated (fused). As this task is typically assigned to specialized components, mobile devices can utilize context under certain accessibility, connectivity or even privacy restrictions,
- many context parameters that may act as preconditions for actions, especially for composite ones, are not directly acquired through sensing, rather are fragments of high-level knowledge that requires some form of reasoning to be performed.

As a result, agents inhabiting real-world AmI systems need to incorporate both epistemic and temporal reasoning skills; whenever sensing is not available or profitable, context knowledge needs to be derived through commonsense inference, considering the most recent information about the world and the effects of observed actions. A formal model is imperative to ensure that the behavior of all entities is properly regulated. In this paper we demonstrate how a recently proposed epistemic action theory from the field of cognitive robotics can contribute in facilitating different aspects of AmI. We report on our experiences in participating in an ongoing AmI-related project at the FORTH institute (Grammenos et al., 2009), having encountered restrictions by other commonly used approaches, such as rule-based reasoning, to handle certain complex tasks. Our objective is to signify the importance of epistemic reasoning for AmI and to exhibit the ways that it contributes to our infrastructure.

The basic properties of the epistemic action theory are discussed in Section 2. Sections 3 and 4 demonstrate different aspects where epistemic reasoning can be applied in an AmI domain, from the standpoint of a centralized infrastructure and that of mobile devices, respectively. Section 5 describes implementation issues and the paper concludes in Section 6.

2 EPISTEMIC THEORY

The Discrete Event Calculus Knowledge Theory (DECKT), thoroughly described in (Patkos and Plexousakis, 2009), is a formal theory for reasoning about knowledge, actions and causality. It builds on the Event Calculus (Kowalski and Sergot, 1986) and ex-

Patkos T. and Plexousakis D..
EPISTEMIC REASONING FOR AMBIENT INTELLIGENCE.
DOI: 10.5220/0003149902420248
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 242-248
ISBN: 978-989-8425-40-9
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

tends it with epistemic capabilities enabling reasoning about a wide range of commonsense phenomena, such as temporal and delayed knowledge effects, knowledge ramifications, concurrency, nondeterminism and others. It is a many-sorted first-order language, where *events* indicate changes in the environment, *fluents* denote time-varying properties and a *timepoint* sort implements a linear time structure. The Event Calculus defines predicates for expressing, among others, which fluents hold when (*HoldsAt*), what events happen (*Happens*) and which their effects are (*Initiates*, *Terminates*).

The theory employs the discrete time Event Calculus axiomatization described in (Mueller, 2006). It assumes agents acting in dynamic environments, having accurate but potentially incomplete knowledge and able to perform knowledge-producing actions and actions with context-dependent effects. Knowledge is treated as a fluent, namely the *Knows* fluent, which expresses knowledge about fluents and fluent formulae, obtained either from direct effects of actions or indirectly through ramifications modeled as state constraints. For technical reasons, for direct effects the auxiliary *KP* fluent (for "knows persistently") is used that is related with the *Knows* fluent by the axiom[1]:

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

DECKT augments a domain axiomatization with a set of meta-axioms describing epistemic derivations. For instance, for positive effect axioms that specify under what conditions action e initiates fluent f, i.e., $\bigwedge^i HoldsAt(f_i,t) \Rightarrow Initiates(e,f,t)$, DECKT introduces the (KT3) set of axioms expressing that if the conjunction of preconditions $C = \{\vec{f_i}\}$ is known then after e the effect will be known to be true, such as:

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

In a similar fashion, the (KT5) axiom set captures the fact that if some precondition is unknown while none is known to be false, then after e knowledge about the effect is lost. The approach proceeds analogously for negative effect axioms $\bigwedge^i HoldsAt(f_i,t) \Rightarrow Terminates(e,f,t)$ and release axioms $\bigwedge^i HoldsAt(f_i,t) \Rightarrow Releases(e,f,t)$. The latter model non-deterministic effects, therefore they result in loss of knowledge about the effect.

Knowledge-producing (sense) actions provide information about the truth value of fluents and, by definition, cause no effect to the environment, instead only affect the mental state of the agent:

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

[1]Free variables are implicitly universally quantified.

Kw is an abbreviation for whether a fluent is known (similarly for *KPw*): $HoldsAt(Kw(f),t) \equiv HoldsAt(Knows(f),t) \vee HoldsAt(Knows(\neg f),t)$ Furthermore, the theory also axiomatizes so called *hidden causal dependencies* (HCDs). HCDs are created when executing actions with unknown preconditions and capture the fact that in certain cases, although knowledge about the effect is lost, it becomes contingent on the preconditions; obtaining knowledge about the latter through sensing can provide information about whether the effect has actually occurred. Consider the positive effect axiom $HoldsAt(f',t) \Rightarrow Initiates(e,f,t)$, where fluent f' is unknown to the agent and f known to be false at T (f may denote that a door is open, f' that a robot stands in front of that door and e the action of pushing forward gently). If e happens at T, f becomes unknown at $T+1$, as dictated by (KT5), still a dependency between f' and f must be created to declare that if we later sense any of them we can infer information about the other, as long as no event alters them in the meantime (either the robot was standing in front of the door and opened it or the door remained closed).

DECKT introduces the (KT6) set of axioms that specify when HCDs are created or destroyed and what knowledge is preserved when an HCD is destroyed. For instance, for positive effect axioms (KT6.1.1) below creates an appropriate implication relation:

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

where $C(t)^-$ denotes those precondition fluents that are unknown to the agent at timepoint t. In a nutshell, DECKT augments the mental state of an agent with a disjunctive knowledge formula, equivalent to $HoldsAt(Knows(\bigwedge^{f_j \in C(t)^-} f_j \Rightarrow f),t+1)$, that encodes a notion of epistemic causality in the sense that if future knowledge brings about $(\bigwedge^{f_j \in C(t)^-} f_j)$ it also brings about f.

DECKT adopts an alternative representation for knowledge that does not employ the possible worlds semantics, which are computationally intensive and not appropriate for practical implementations. Instead, its efficiency stems from the fact that HCDs, which are based on a provably sound and complete translation of possible worlds into implication rules, are treated as ordinary fluents.

3 AMI FRAMEWORK

In the next sections we demonstrate the benefits of epistemic reasoning for AmI, by describing how DECKT is employed within a large-scale AmI-related project that is being conducted in our institute over the last 2 years. The project integrates expertise from different laboratories that contribute a multitude of hardware and software technologies; our work package is assigned the task of designing and implementing a reasoning framework for the AmI infrastructure.

3.1 Reasoning Framework Architecture

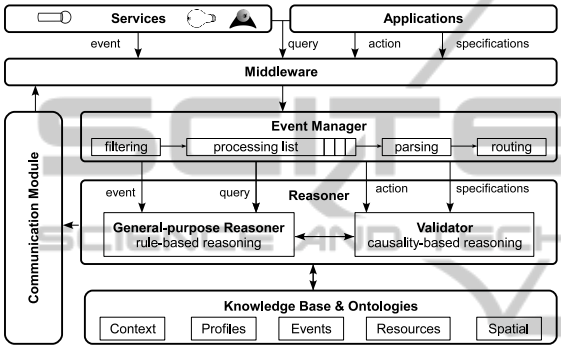

Figure 1: Event management framework architecture.

The design goals of our reasoning framework have been the efficient representation, monitoring and dissemination of any low- or high-level contextual information in the AmI infrastructure, as well as the support for a number of general-purpose and domain-specific inference tasks. The need for hybrid approaches for such systems has already been recognized (Bettini et al., 2009), still is mostly restricted on specific aspects, such as for context inference (Bulfoni et al., 2008). In our system we couple methodologies from the Semantic Web and the action theories domains and deploy the hybrid event-based reasoning architecture shown in Fig. 1, that comprises three 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 inferencing, and the *Knowledge Base* that stores semantic information represented using ontology-based languages. A middleware layer undertakes the role of connecting applications and services developed by different research parties and with diverse technologies. *Services* denote standalone entities that implement specific functionalities, such as voice recognition, localization, light management etc., whereas *applications* group together service instances to provide an AmI experience in smart spaces.

A number of ontologies capture the meaning and relations of concepts regarding low-level context acquired from sensors, high-level context inferred through rule-based reasoning, user and devices profiles and spatial features. High-level context, in particular, may concern a user's emotional state, identity, intentions, location and others. Within our system rule-based reasoning contributes to the design of enhanced applications and also provides feedback to services for sensor fusion purposes, in order to resolve conflicts of ambiguous or imprecise context. On the other hand, rule-based languages offer only limited expressiveness to describe certain complex features, such as compound actions. We utilize the Event Calculus to formalize event operators for the construction of composite events, characterized by their initiation and termination occurrence times, as well as to declare their effects and potential preconditions.

3.2 Partial Knowledge

During our involvement in the first phases of the project, we recognized how crucial it is for the management of an AmI system to distinguish between the rules that govern its behavior from the domain-specific functionalities and restrictions, in order to enable efficient and dynamic adaptation to changes during development. This need was primarily driven by the fact that multiple research groups participate and collaborate in parallel tasks, having different degrees of contribution and therefore comprehension of the general objectives and the system restrictions introduced from other groups.

Towards that end, we implement 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 by developers, as well as from *application specifications* that are usually under the responsibility of non-experts who only possess partial knowledge about the system restrictions. The specifications of services, for instance, retrieved from the KB, are axioms that describe the domain, such as:

$$Terminates(TurnOffLight(l), LightOn(l), t)$$

Application descriptions express the intended behavior of an application as a narrative of context-dependent action occurrences:

$$Happens(SitOn(chair), t) \wedge$$
$$HoldsAt(LightOn(l), t) \Rightarrow$$
$$Happens(TurnOffLight(l), t) \wedge$$
$$Happens(StartSlideshow(< file >, TV01), t)$$

Finally, system restrictions (or safety properties) capture assertions about attributes of system states that must hold for every possible system execution. For instance, the Localizer's specifications require that no change in lighting conditions should occur when in *Running* mode, otherwise the camera-based localization process may be destabilized:

$$HoldsAt(LocalizaterRunning(ROOM\_23),t) \wedge HoldsAt(LocatedIn(light,ROOM\_23),t) \Rightarrow HoldsAt(LightOn(light),t)$$

A core task of our framework is to verify that the specifications of AmI applications are in compliance with the overall system restrictions and detect errors early at the development phase. This analysis is performed at design-time and can formally be defined as an abductive reasoning process 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:

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

Russo et al. (2002) provided a fully decidable reduction of this abductive task. The absence of available knowledge about certain context parameters was found to be a less likely contingency to consider by developers and usually arises late at the evaluation phase or even after deployment.

Example 1. Imagine that a developer uploads an application description file to the system containing axioms such as the one mentioned above. After executing the *ApplicationCheck* functionality of the Validator to check for consistency with respect to system restrictions already stored by service engineers, potential violations are raised: whenever a user sits on the chair the $TurnOffLight$ action is triggered conflicting with the Localization service being at a *Running* state. Moreover, the developer does not consider the situation where the state of lights are unknown due to some other application's intervention, i.e., when neither $HoldsAt(LightOn(l),t)$ nor $\neg HoldsAt(LightOn(l),t)$ are true. For the latter case in particular, the appropriate epistemic counterpart should be used, i.e., $HoldsAt(Knows(LightOn(l),t))$ and also a new axiom should be introduced to account for lack of knowledge. DECKT enables the formalization of such processes for deriving epistemic conclusions regarding the effects of actions. □

Application verification is only one dimension of the system management process and can be executed at design-time and in isolation. Still, run-time action validation is equally important and must be performed considering the current state of the system, as well as potential conflicts with other applications that might request access to the same resources (e.g., speakers). Causality-based reasoning is applied for high-level resource management, as well as for uncertainty handling due to ambiguous context or for determining harmful ramifications of actions, which may cause side-effects to the execution of other applications.

4 EPISTEMIC REASONING FOR MOBILE DEVICES

Building on the centralized infrastructure already available, our 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 AmI 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. We provide different aspects where DECKT can contribute, adopting the personal assistant paradigm.

4.1 The Personal Assistant Paradigm

In a smart space every user is assumed to be equipped with a virtual personal assistant, an intelligent agent that can live in stationary or mobile devices that the user interacts with, such as her PDA, cellphone etc, and utilize relevant information in order to assist the user. This assistant can be assigned different tasks: it can be instructed to proactively or reactively devise 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.

Uncertainty Handling. Apart from actions with unknown preconditions that lead to lack of knowledge about the effect, uncertainty is evident in differ-

ent forms within an AmI environment. For instance, given noisy conditions, it is common practice to axiomatize the action of speaking voice commands into a microphone as having non-deterministic effects

$$HoldsAt(Rec(mic),t) \wedge HoldsAt(Admin(u),t) \wedge HoldsAt(InFrontOf(u,door),t) \Rightarrow Releases(Speak(u,mic,"Open"),Open(door),t)$$

Even if the preconditions are known when the command is given, axioms (KT5) will result in $\neg HoldsAt(Kw(Open(door)),t+1)$.

Temporal Knowledge. The consideration of the temporal aspect of knowledge, such as knowing facts for specified time intervals only, is a crucial aspect in systems where temporal constraints are ubiquitous. The very nature of most AmI environments involves dynamically changing context, information becoming outdated as time passes and events occurring at specified instants, more often than not concurrently with other events. Such issues, also discussed in (Scherl, 2003), are efficiently handled with the Event Calculus and DECKT, where the dimension of time is inherently modeled within the basic ontology, e.g.,:

$$Happens(PlaceRFCard(id),t) \Rightarrow Happens(StartRec(Mic),t) \wedge Happens(StopRec(Mic),t+5)$$

Ramifications. Considering the dynamics of a domain, an agent can not only derive knowledge from direct effects of actions, but also indirectly through appropriate ramifications. In the simplest case, these can be represented as state constraints of a domain axiomatization and are accommodated by DECKT with axioms, such as the following:

$$HoldsAt(Knows(Holding(user,object)),t) \wedge HoldsAt(Knows(InRoom(user,room)),t) \Rightarrow HoldsAt(Knows(InRoom(object,room)),t)$$

Reasoning about only Knowing. Deliberating about what an agent does not know, in the style of (Lakemeyer and Levesque, 1998), is an important aspect when modeling partially known environments. It enables the representation of subtle differences in the meaning of propositions, distinguishing, for instance, between the knowledge that some microphone is currently recording, i.e., $HoldsAt(Knows(\exists mic\ Rec(mic)),t)$, without explicitly knowing which, i.e., $\neg\exists mic\ HoldsAt(Knows(Rec(mic)),t)$.

4.2 Hidden Causal Dependencies

In addition to being able to cope with a multitude of diverse commonsense phenomena, an epistemic theory implementing an agent's cognitive skills in AmI 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. DECKT enables an agent to generate and maintain long chains of dependencies among unknown fluents by managing HCDs. These dependencies, handled as ordinary fluents, minimize computational complexity and at the same time facilitate valuable ramifications when seeking for knowledge about unobservable fluents.

Example 2. Among the personal assistant'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 can be detected from a number of activities that follow a pattern, such as turning off the hot plate, placing dishes in the sink and opening the tap water. If the user leaves the kitchen (event e_1) while cooking, the assistant agent should identify a potential exceptional behavior (fluent f_2), but not disturb the user just yet. Only if the user picks up the phone (event e_2) an alert (fluent f_3) should be activated, initiating appropriate actions to notify the user about unfinished kitchen activity. The previous behavior can be plainly axiomatized as follows:

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

Assume now that at timepoint T the contextual information coming from the kitchen is ambiguous, i.e., $\neg HoldsAt(Kw(f_1),T)$ and then the user leaves the kitchen and starts making a phone call. Based on DECKT, the following HCDs are created for $T' > T$:

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

Specifically, the agent does not know if an alert should be triggered and, before disturbing the user, this contingency needs to be verified. Although there is no direct means of sensing f_3, by evaluating f_1 the agent can also derive f_3 based on the interrelation of HCDs. In conjecturing that there is work in progress in the kitchen with potential hazard (e.g., a turned on apparatus), the act of sensing f_1 is implemented as a check

procedure of 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 or destroyed based only on the agent's current KB. Possible worlds-based epistemic theories, which maintain all alternative worlds rather than HCDs, are less suitable in practice, as they require considerable storage and computational resources (Petrick and Levesque, 2002). Alternative approaches that do not deal explicitly with the notions of causality, such as rule-based approaches, apart from being restricted to less expressive commonsense phenomena, require for the designer to model beforehand all possible conditions that call for some action to be performed.

5 IMPLEMENTATION

The AmI infrastructure that is being implemented at FORTH institute expands in a three-room set up, where a multitude of hardware and software services have been developed, such as 3D person localization, speech recognition etc. The middleware responsible for creating, connecting and consuming the services is CORBA-based. The rule-based component of the reasoner module uses Jess[2] as its reasoning engine that deploys the efficient Rete algorithm for rule matching, while the Validator component uses both Jess and DEC Reasoner[3], a SAT-based Event Calculus reasoner. All available knowledge is encoded in OWL using the Prot*égé* platform for editing and querying the corresponding models.

As far as the DECKT epistemic axiomatization is concerned, we are currently constructing an Event Calculus reasoner on top of Jess, that can support, among others, two important features: (a) it enables reasoning to progress incrementally to allow for runtime execution of knowledge-based programs (online reasoning) and (b) it permits nested reification of epistemic fluents in predicates, to allow epistemic propositions, such as $Knows(Open(S1))$, to be treated as terms of a first-order logic rather than atoms.

Predicates are asserted as facts in the reasoner's agenda, specified by the template definition:

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

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

Treating positive and negative literals of formulae with lists enables us to represent and dynamically manipulate HCDs according to DECKT axioms[4].

6 CONCLUSIONS

In this paper we described different aspects of research in AmI to which reasoning under partial knowledge can contribute, providing also examples from an implemented system that couples rule- and causality-based techniques. The presented application of an epistemic action theory is an enabling direction for the realization of the AmI vision. Our current emphasis is on evaluating the solution on different scenarios, as well as on investigating even more challenging aspects, such as distributed knowledge.

ACKNOWLEDGEMENTS

This work has been supported by the FORTH-ICS internal RTD Programme "Ambient Intelligence and Smart Environments".

REFERENCES

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D. (2009). A Survey of Context Modelling and Reasoning Techniques. *Pervasive and Mobile Computing*.

Bulfoni, A., Coppola, P., Della Mea, V., Di Gaspero, L., Mischis, D., Mizzaro, S., Scagnetto, I., and Vassena, L. (2008). AI on the Move: Exploiting AI Techniques for Context Inference on Mobile Devices. In *18th European Conference on AI*, pages 668–672.

Grammenos, D., Zabulis, X., Argyros, A., and Stephanidis, C. (2009). FORTH-ICS Internal RTD Programme Ambient Intelligence and Smart Environments. In *AMI'09: Proceedings of the 3rd European Conference on Ambient Intelligence* .

[2]Jess, http://www.jessrules.com/

[3]DECReasoner, http://decreasoner.sourceforge.net/

[4]The Jess-based Event Calculus reasoner and examples are available at *http://www.csd.uoc.gr/˜patkos/deckt.htm*.

Kowalski, R. and Sergot, M. (1986). A Logic-based Calculus of Events. *New Generation Computing*, 4(1):67–95.

Lakemeyer, G. and Levesque, H. J. (1998). AOL: A Logic of Acting, Sensing, Knowing, and Only Knowing. In *Principles of Knowledge Representation and Reasoning*, pages 316–329.

Mueller, E. (2006). *Commonsense Reasoning*. Morgan Kaufmann, 1st edition.

Patkos, T. and Plexousakis, D. (2009). Reasoning with Knowledge, Action and Time in Dynamic and Uncertain Domains. In *IJCAI'09*, pages 885–890.

Petrick, R. P. A. and Levesque, H. J. (2002). Knowledge Equivalence in Combined Action Theories. In *KR'02: International Conference on Principles of Knowledge Representation and Reasoning*, pages 303–314.

Scherl, R. (2003). Reasoning about the Interaction of Knowledge, Time and Concurrent Actions in the Situation Calculus. In *IJCAI'03*, pages 1091–1098.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS