Indirect and Conditional Sensing in the Event Calculus

Jeremy Forth and **Murray Shanahan** [1]

Abstract.
Controlling the sensing of an environment by an agent has been accepted as necessary for effective operation within most practical domains. Usually, however, agents operate in partially observable domains where not all parameters of interest are accessible to direct sensing. In such circumstances, sensing actions must be chosen for what they will reveal indirectly, through an axiomatized model of the domain causal structure, including ramifications. This article shows how sensing can be chosen so as to acquire and use indirectly obtained information to meet goals not otherwise possible. Classical logic Event Calculus is extended with both a knowledge formalism and causal ramifications, and is used to show how inferring unknown information about a domain leads to conditional sensing actions.

1 Introduction

To perform effectively in most practical domains, a rational agent must possess the ability to reason about, and create plans for sensing, conditional, and knowledge-producing actions [8]. This requires the agent not only reason about the state of objects in the domain, but also about the agent's own knowledge about the state of the domain. Such acquisition of knowledge may be achieved by direct sensing of the environment, or alternatively, indirectly through inference using existing or newly acquired knowledge.

This paper extends Classical Logic Event Calculus [11] to reason about knowledge for the purposes of describing and controlling sensing actions in a partially observable causal domain. Partial observability is usual in most practical circumstances, and restricts an agent in its range of directly sensed parameters of the environment. In cases where the state of an unobservable domain fluent is required, this information must be inferred using the causal domain model along with other known fluents. Determining which fluents to sense, and when, is the problem to be solved at plan time.

Unknown fluents frequently need to be inferred through the modeling of ramifications in the environment. This is achieved through a new causal ramification theory presented later. At plan time, it may not be possible to find an effective fixed set of sensing actions that will yield the desired result. While it is always possible to plan to sense more fluents than are strictly needed, and thereby guarantee sufficient information has been acquired for all possible cases, a real agent would never be expected to function in this manner. Instead, a behavior of sensing solely what is necessary is much to be preferred. This conditional sensing is part of a later ramification example.

Previous work in the literature such as [10], [5], [6], and [15] have addressed different aspects of knowledge producing actions, adopting Possible-World state-based approaches, different to this account. The present work is based on a meta-language representation of knowledge, more similar in style to [9], though this used a state-based action representation unlike Event Calculus.

Knowledge producing actions were first introduced into an executable logic programming version of the Event Calculus by [14] which focused on a map building task. We contribute further by extending the expressiveness of the Event Calculus formalism and addressing the inferential aspect of sensing. We present a solution for selective and conditional sensing of an environment with ramifications in order to infer desired knowledge.

2 Integrating Knowledge

Providing an agent with the capability to reason about its knowledge of the environment requires that we provide a theory that can represent and reason about that part of itself devoted to describing the environment. In particular, knowledge producing actions modify an agent's theory, as well as possibly modifying the environment. To represent the effects of such actions properly, the theory must contain a representation of what will be derivable from future versions of itself. To do this, we augment the domain fluents with meta-level knowledge fluents which at any given time point are semantically attached to the corresponding domain fluent using the knowledge axioms.

Such a theory introduces logical complexity because of the inherent requirement for formulas representing objects in the domain to be presented as arguments to other predicates. Moreover, permitting inference in such a general system introduces well known inconsistency as exhibited by paradoxes such as the Knower. It therefore becomes necessary to use a logical system that will allow useful reasoning to take place whilst avoiding inconsistency.

The language used to represent knowledge here is a self-referential (amalgamated) language permitting a predicate to take a formula as an argument through the use of a naming relation to convert the formula to a unique term. For simplicity, we adopt the convention of sentences naming themselves. Using such a language, very general expressions are possible, such as those describing incomplete knowledge, and it may be comfortably used to represent what will become known after executing a knowledge producing action.

Previous work by Gupta and Herzberger, further refined by [16] and [2], provides a system of logic with a semantic definition of truth based on an iterative revision process termed "Taski-revision". This operates on a traditional two-valued truth model obeying the law of excluded middle. Based upon classical semantics, truth is defined by a non-monotonic revision process that proceeds by determining the truth models at each stage of iteration. The process finishes when no further stabilization occurs with successive revision steps. The

[1] Imperial College, London, UK. email: jforth,m.shanahan@imperial.ac.uk

stable models then provide a semantic definition of truth for the self-referential system. In this paper, we will recast such a system to reason about knowledge, where knowledge is defined semantically by stable models.

Shown below are axioms employed in a first-order self-referential theory of knowledge, somewhat similar to modal logic's S4 system. To guarantee freedom from inconsistency, the necessitation rule K5 has been weakened over S4 so that just all classical truths are known. Such flexibility avoids the logical omniscience problem of Possible World semantics, and makes it possible to model inference power of varied proof procedures. Full necessitation of S4 is particularly problematic when reasoning about other agents knowledge, due to it forcing an assumption that all agents know all axioms. Since Event Calculus uses the distinguished predicate $HoldsAt$ to represent the state of all time varying fluents, including knowledge fluents, all knowledge axioms must be defined within this predicate.

$$HoldsAt([k(f_1 \rightarrow f_2) \rightarrow [k(f_1) \rightarrow k(f_2)]], t) \quad \text{(K1)}$$

$$HoldsAt([\forall x k(f) \rightarrow k(\forall x\ f)], t) \quad \text{(K2)}$$

$$HoldsAt([k(f) \rightarrow f], t) \quad \text{(K3)}$$

$$HoldsAt([k(f) \rightarrow k(k(f))], t) \quad \text{(K4)}$$

$$if\ \text{PC} \vdash HoldsAt(f, t)\ then\ HoldsAt(k(f), t) \quad \text{(K5)}$$

$$HoldsAt([kw(f) \stackrel{\text{def}}{\equiv} [k(f) \vee k(\neg f)]], t) \quad \text{(K6)}$$

We use a single temporal structure to represent both domain knowledge, and also the agent's knowledge. In order to reason about agent knowledge within the temporal structure, we must define inference rules KEC1, KEC2 that apply to fluent formulas contained within. There must be a way to couple a negated derived fluent, for instance $HoldsAt(\neg On, t)$ with a negated predicate taking the non-negated fluent as its argument: $\neg HoldsAt(On, t)$. NEG accomplishes this.

$$HoldsAt(f_1, \rightarrow f_2, t) \rightarrow [HoldsAt(f_1, t) \rightarrow HoldsAt(f_2, t)] \quad \text{(KEC1)}$$

$$\forall x\ HoldsAt(f_1, t) \rightarrow HoldsAt(\forall x\ f_1, t) \quad \text{(KEC2)}$$

$$HoldsAt(\neg f, t) \equiv \neg HoldsAt(f, t) \quad \text{(NEG)}$$

3 Classical Logic Event Calculus

The Event Calculus formalism employed here is a positive-time only version based on classical logic, described in [11], with further discussion available in [7] and [12].

Definitions of the predicates *Happens*, *Initiates*, *Terminates* and *Possible* are given in the domain-dependent part of the theory, along with an appropriate axiomatization of the sort time (e.g. as natural numbers). Action variables are represented by a,a_1 while time variables are represented by t, t_1, t_2. Since the knowledge theory allows us to derive new fluent formulas at each time point, fluents are divided into sorts for primary (frame) fluents f_p as specified in the domain axiomatization, and also derived (non-frame) fluents. Generalized fluents f, comprise both sorts. Knowledge fluents are subject to default persistence.

A fluent's positive (respectively negative) persistence between two timepoints is compromised if an action occurs within this interval that terminates (respectively initiates) that fluent, or if any action occurs that is not "possible" at any earlier timepoint:

$$Clipped(t_1, f_p, t_2) \stackrel{\text{def}}{\equiv} [\exists a, t[Happens(a, t) \wedge t < t_2 \wedge \neg Possible(a, t)] \vee \exists a, t[Happens(a, t) \wedge t_1 \leq t < t_2] \wedge Terminates(a, f_p, t) \wedge Releases(a, f_p, t)]] \quad \text{(EC1)}$$

$$Declipped(t_1, f_p, t_2) \stackrel{\text{def}}{\equiv} [\exists a, t[Happens(a, t) \wedge t < t_2 \wedge \neg Possible(a, t)] \vee \exists a, t[Happens(a, t) \wedge t_1 \leq t < t_2] \wedge Initiates(a, f_p, t) \wedge Releases(a, f_p, t)]] \quad \text{(EC2)}$$

Prior to a non "possible" action, fluents which have been initiated by an occurrence of an action continue to hold until an occurrence of an action which terminates them:

$$HoldsAt(f_p, t_2) \leftarrow [Happens(a, t_1) \wedge Initiates(a, f_p, t_1) \wedge t_1 < t_2 \wedge \neg Clipped(t_1, f_p, t_2)] \quad \text{(EC3)}$$

Fluents which have been terminated by an occurrence of an action continue not to hold until an occurrence of an action which initiates them:

$$\neg HoldsAt(f_p, t_2) \leftarrow [Happens(a, t_1) \wedge Terminates(a, f_p, t_1) \wedge t_1 < t_2 \wedge \neg Declipped(t_1, f_p, t_2)] \quad \text{(EC4)}$$

Prior to a non "possible" action, fluents change their truth values only via the occurrence of initiating and terminating actions:

$$HoldsAt(f_p, t_2) \leftarrow [InitallyP(f_p, t_1) \wedge t_1 < t_2 \wedge \neg Clipped(t_1, f_p, t_2)] \quad \text{(EC5)}$$

$$\neg HoldsAt(f_p, t_2) \leftarrow [InitallyN(f_p, t_1) \wedge t_1 < t_2 \wedge \neg Declipped(t_1, f_p, t_2)] \quad \text{(EC6)}$$

$$InitallyP(f_p) \vee InitallyN(f_p) \quad \text{(EC7)}$$

4 Acting in Causal Domains

Sensing actions can be axiomatized in a domain independent way by assuming all fluents can be sensed directly. The effect of a pure sensing action is limited, by definition, to changing the agent's knowledge base, with no effect on the domain. Clearly, in many practical situations, sensing must impact the environment, but this will be handled as a refinement to the pure sensing action defined below. A function $sense$ is introduced, mapping a fluent to an action:

$$Initiates(sense(f_p), kw(f_p), t) \quad \text{(SA1)}$$

$$Possible(sense(f_p), t) \quad \text{(SP1)}$$

Many actions have effects on the domain as well as on the agent's knowledge of the domain's state. An agent may come to know the status of a door by sensing, or alternatively by performing an $OpenDoor$ action. It is desirable to free the axiomatizer from the requirement of capturing all the knowledge effects of actions separately from physical effects on the domain. We may have a

domain dependent axiom $Initiates(OpenDoor, Open, t)$ which initiates fluent $Open$ upon the occurrence of action $OpenDoor$. Accordingly, we need to be able to initiate the fluent $k(Open)$ whenever action $OpenDoor$ is planned to be performed. This is accomplished through axioms KE1 and KE2. To address the special case of non-deterministic actions, we exclude knowledge effects where an action both initiates and terminates the same fluent, as in the case of a coin-toss action.

$$\textit{Initiates}(a, k(f_p), t) \leftarrow \quad \text{(KE1)}$$
$$[\textit{Initiates}(a, f_p, t) \wedge \neg \textit{Terminates}(a, f_p, t)]$$

$$\textit{Terminates}(a, k(f_p), t) \leftarrow \quad \text{(KE2)}$$
$$[\textit{Terminates}(a, f_p, t) \wedge \neg \textit{Initiates}(a, f_p, t)]$$

So as to avoid possible inconsistency between the representation of the meta-knowledge and domain object knowledge, persistence must not occur on a knowledge fluent after an action has taken place affecting that fluent, either by direct action, or indirectly through ramifications.

$$\textit{Releases}(a, kw(f_p), t) \leftarrow \quad \text{(KP1)}$$
$$[\textit{Initiates}(a, f_p, t) \vee \textit{Terminates}(a, f_p, t)]$$

Action preconditions are usually specified in terms of physical domain prerequisites for an action to take place. An action precondition for an open-door action may be door-closed. Conditional actions however require preconditions to refer to the agent's knowledge. For instance, switching a light on just if it's off, has a precondition of knowing whether the light is on prior to executing the conditional action. Handling knowledge preconditions correctly requires that we specify when an agent knows enough to execute an action. If an action definition takes the form of conjunction of ground *HoldsAt* literals,

$$\textit{Initiates}(a, f_p, t) \leftarrow \quad \text{(DE1)}$$
$$[(\neg)\textit{HoldsAt}(f_1, t)_1 ... \wedge ... (\neg)\textit{HoldsAt}(f_n, t)_n]$$

$$\textit{Terminates}(a, f_p, t) \leftarrow \quad \text{(DE2)}$$
$$[(\neg)\textit{HoldsAt}(f_1, t)_1 ... \wedge ... (\neg)\textit{HoldsAt}(f_n, t)_n]$$

we say we have enough knowledge to perform an action dependent on binary conditions when we know whether any fluent defined as a condition for the action's effect is true or not at the time of execution. This principle is captured in the following axiom:

$$\textit{Possible}(a, t) \leftarrow \quad \text{(KP)}$$
$$[\textit{HoldsAt}(kw(f_1), t)_1 ... \wedge ... \textit{HoldsAt}(kw(f_n), t)_n]$$

Although much of the purpose of a rational agent planning to execute conditional actions is to increase knowledge and certainty about the environment, non-deterministic actions actually reduce it. Accordingly, this must be allowed for in the theory. After executing a non-deterministic action, we know we don't know the resulting fluent state:

$$\textit{Terminates}(a, kw(f_p), t) \leftarrow \quad \text{(KP2)}$$
$$[\textit{Initiates}(a, f_p, t) \wedge \textit{Terminates}(a, f_p, t)]$$

Finally, we introduce the functions if and $ifnot$ (from fluent and action pairs, to action) to represent conditional actions, executed just if the conditional fluent holds, or does not hold respectively.

$$\textit{Initiates}(if(f_1, a), f_{2p}, t) \leftarrow \quad \text{(CA1)}$$
$$[\textit{Initiates}(a, f_{2p}, t) \wedge \textit{HoldsAt}(f_1, t)]$$

$$\textit{Terminates}(if(f_1, a), f_{2p}, t) \leftarrow \quad \text{(CA2)}$$
$$[\textit{Terminates}(a, f_{2p}, t) \wedge \textit{HoldsAt}(f_1, t)]$$

$$\textit{Possible}(if(f, a), t) \leftarrow \textit{HoldsAt}(kw(f), t) \quad \text{(CP1)}$$

Similarly for action ifnot:

$$\textit{Initiates}(ifnot(f_1, a), f_{2p}, t) \leftarrow \quad \text{(CA3)}$$
$$[\textit{Initiates}(a, f_{2p}, t) \wedge \neg \textit{HoldsAt}(f_1, t)]$$

$$\textit{Terminates}(ifnot(f_1, a), f_{2p}, t) \leftarrow \quad \text{(CA4)}$$
$$[\textit{Terminates}(a, f_{2p}, t) \wedge \neg \textit{HoldsAt}(f_1, t)]$$

$$\textit{Possible}(ifnot(f, a), t) \leftarrow \textit{HoldsAt}(kw(f), t) \quad \text{(CP2)}$$

5 Ramifications

If a domain has fluents with interdependencies, then these can frequently be represented as ramifications. In such cases, it is clear that gaining knowledge about one fluent may provide knowledge about another. We show how reasoning about knowledge can yield useful information about which sensing actions to take in an environment. Closely connected to reasoning about knowledge in the presence of ramifications, is the ability of an agent to function effectively in a partially observable environment, where not all fluents can be sensed directly. Planning in such domains, with the capability to consider the epistemic state, results in the generation of tests or experiments in the course of meeting a goal; a type of active perception.

Solutions to different classes of ramification problem are detailed in [13] for the Event Calculus. Two distinct methods presented use state constraints to represent static relationships, and effect constraints for dynamic (triggered) changes. Here we will present a combination of the two approaches, chiefly because such a combination aligns very well with the semantic definition of causal ramifications in the language $\mathcal{E}$, based on inductive fixed point definitions. A version of language $\mathcal{E}$ without ramifications is translated into Event Calculus in [7]. We will provide the result of a mapping of a version of $\mathcal{E}$ with ramifications defined semantically in [3] into Event Calculus.

Language $\mathcal{E}$ has two types of causal construct, one for direct actions (**initiates**, **terminates**) and another for expressing ramification relationships between fluents (**whenever**). If the causal description of a domain is expressed in terms of members A of the set of actions and members F of the set of fluents, and C comprised of literals $\{L_1, ..., L_n\}$, then the direct action constructs A **initiates** F **when** C, and A **terminates** F **when** C, can be mapped into classical logic straightforwardly.

$$\textit{Initiates}_0(A, F, t) \leftarrow \quad \text{(R1)}$$
$$\bigwedge_{F_n \in C} \textit{HoldsAt}(F_n, t) \wedge \bigwedge_{\neg F_n \in C} \neg \textit{HoldsAt}(F_n, t)$$

$$\textit{Terminates}_0(A, F, t) \leftarrow \quad \text{(R2)}$$
$$\bigwedge_{F_n \in C} \textit{HoldsAt}(F_n, t) \wedge \bigwedge_{\neg F_n \in C} \neg \textit{HoldsAt}(F_n, t)$$

For domains described using ramification construct **whenever** of language $\mathcal{E}$, there arises a natural question over which class of domain is to be represented. As an example, consider a domain description with two ramifications: F_1 **whenever** $\neg F_2$, and F_2 **whenever** F_1.

This may arise in the description of a domain containing two logic gates, one inverting the other not, connected back to back. In practice this would create an oscillating circuit having no steady state. We wish to specify a class of domain that explicitly excludes domains with such negative cycles. A suitable class of domains for this purpose are those in which the fluents can be ordered in stratified layers, so that each stratum of fluents only depends on effects in layers strictly lower. Although stratification assignment is not unique, any correct choice of valid stratification is equivalent to any other correct choice from a semantic standpoint.

A domain theory containing ramification statements F **whenever** C, where C is comprised of literals $\{L_1, ..., L_n\}$, is stratified in terms of fluent effect arrangement, with direct actions at stratum level 0, and progressively propagated indirect effects at lower priorities (higher indices). A stratum function $s : fluent \mapsto stratum$ is defined to apply such an index to each fluent so as to make each stratum dependent solely on fluents in the layer below.

Indirect triggering of fluents by actions is then expressed by looking at the conditions under which **whenever** is triggered. Let $\Sigma(\varphi)$ and $\Omega(\varphi)$ be the following sub-formulae expressed in terms of a single fluent variable φ.

$$\Sigma_i(\varphi) \ := \ \mathit{Initiates}_{s(\varphi)}(a, \varphi, t) \tag{RM1}$$

$$\Sigma_t(\varphi) \ := \ \mathit{Terminates}_{s(\varphi)}(a, \varphi, t) \tag{RM2}$$

$$\Omega_{hp}(\varphi) \ := \mathit{HoldsAt}(\varphi, t) \wedge \neg\exists a_1(\mathit{Happens}(a_1, t) \ \wedge \ \mathit{Terminates}_{s(\varphi)}(a_1, \varphi, t)) \tag{RM3}$$

$$\Omega_{hn}(\varphi) \ := \neg\mathit{HoldsAt}(\varphi, t) \wedge \neg\exists a_1(\mathit{Happens}(a_1, t) \ \wedge \ \mathit{Initiates}_{s(\varphi)}(a_1, \varphi, t)) \tag{RM4}$$

Propagated effects between fluents due to a **whenever** statement will then occur under the trigger conditions expressed compactly in RM5, where $\{C_1, C_2\}$ is a partition of C.

$$\Psi = \bigvee_{C_1 \in \mathcal{P}(C)} \Big(\Big(\bigwedge_{F \in C_1} \Sigma_i(F) \ \wedge \bigwedge_{F \in C_2} \Omega_{hp}(F) \Big) \wedge \Big(\bigwedge_{\neg F \in C_1} \Sigma_t(F) \ \wedge \bigwedge_{\neg F \in C_2} \Omega_{hn}(F) \Big) \Big) \tag{RM5}$$

Given the above specification of the triggering conditions for an arbitrary **whenever** statement, we must now just specify the effect to be propagated upon triggering. The case of effect literal L being atomic fluent F corresponds to RM6, while L being $\neg F$, to RM7.

$$\mathit{Initiates}_{s(F)}(a, F, t) \ \leftarrow \ \Psi \tag{RM6}$$

$$\mathit{Terminates}_{s(F)}(a, F, t) \ \leftarrow \ \Psi \tag{RM7}$$

Causal domain closure must now be preformed by prioritized parallel circumscription using the assigned stratification to express a preference for minimizing firstly the effects of direct actions, and subsequently the effects of successive levels of indirect actions.

$$\bigwedge_{i=0}^{n} \text{CIRC[R1-2, RM6-7; } \mathit{Initiates}_i, \mathit{Terminates}_i; \ \mathit{Initiates}_{i+1}, \mathit{Terminates}_{i+1}, ..., \mathit{Initiates}_n, \mathit{Terminates}_n \text{]}$$

All that now remains is to combine each stratum's effect axiom into a single pair of minimized causal predicates containing the complete description of domain causality:

$$\mathit{Initiates}(a, f_p, t) \ \equiv \ [\mathit{Initiates}_0(a, f_p, t) \vee \mathit{Initiates}_1(a, f_p, t) \vee , ..., \vee \mathit{Initiates}_n(a, f_p, t)] \tag{RM8}$$

$$\mathit{Terminates}(a, f_p, t) \ \equiv \ [\mathit{Terminates}_0(a, f_p, t) \vee \mathit{Terminates}_1(a, f_p, t) \vee , ..., \vee \mathit{Terminates}_n(a, f_p, t)] \tag{RM9}$$

With the triggering of change of state now axiomatized, the static component must be dealt with in terms of a constraint. For the case where L corresponds to atom F, SC2 applies, while for its negation $\neg F$, SC3 applies:

$$\mathit{HoldsAt}(F, t) \ \leftarrow \bigwedge_{F_n \in C} \mathit{HoldsAt}(F_n, t) \wedge \bigwedge_{\neg F_n \in C} \neg\mathit{HoldsAt}(F_n, t) \tag{SC2}$$

$$\neg\mathit{HoldsAt}(F, t) \ \leftarrow \bigwedge_{F_n \in C} \mathit{HoldsAt}(F_n, t) \wedge \bigwedge_{\neg F_n \in C} \neg\mathit{HoldsAt}(F_n, t) \tag{SC3}$$

The above described mapping from language $\mathcal{E}$ specifying ramifications, into classical logic can be proved to be sound and complete with respect to $\mathcal{E}$'s semantics. A full proof has been undertaken as part of other work, consisting of a direct mapping into Least Fixed Point logic with an inductive model-theoretic proof to show correspondence with a circumscribed classical-logic Event Calculus theory similar to that described here.

6 Indirect Sensing through Ramifications

Unobservable fluents may still be important for their effect on the environment. Consequently there are occasions when we must use the state of such unobservable fluents to determine further actions. When this information is needed, it is possible to infer the state of the hidden fluent using the theory of ramifications after we know sufficiently about the state of the associated fluent(s). This is the subject of the next example.

First, the sensing action definition must be extended to only permit sensing of fluents which are accessible in the environment. We introduce a notion of time-varying observability for fluents:

$$\mathit{Possible}(sense(f_p), t) \ \leftarrow \ \mathit{HoldsAt}(observable(f_p), t) \tag{SP2}$$

6.1 Example 1

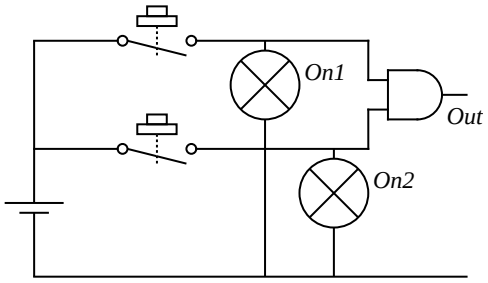

Figure 1: Unobservable output.

The purpose of Figure 1 is to illustrate the use of conditional sensing actions chosen to allow an agent to infer the value of Out after sensing just one of the two inputs, as long as this sensed input is found to

be inactive. If found to be active, then a second sensing action must take place. Either way, the goal of knowing whether the output is active is met.

There are two primary fluents: $On1$ and $On2$, both of which can be directly observed. The output is a derived fluent, which, for the sake of example is specified to be unobservable. The goal is a knowledge goal stating that the robot knows if the output is active or not: $HoldsAt(kw(Out), 3)$. Due to sensing being the only action in this example, it is necessary to use just the state constraint component of the ramification description. The power of the knowledge axioms becomes immediately apparent, with knowledge constraint KSC2 being generated automatically from the domain constraint SC2 given by the domain axiomatization.

$$HoldsAt(Out, t) \equiv \quad \text{(SC2)}$$
$$[HoldsAt(On1, t) \wedge HoldsAt(On2, t)]$$

$$HoldsAt(kw(Out), t) \leftarrow \quad \text{(KSC2)}$$
$$[HoldsAt(kw(On1), t) \wedge HoldsAt(kw(On2), t)]$$

Stating that the fluents $On1$ and $On2$ are observable:

$$HoldsAt(observable(On1), t) \quad \text{(FD1)}$$

$$HoldsAt(observable(On2), t) \quad \text{(FD2)}$$

A two-action narrative is generated at plan time; a sensing action of fluent $On1$ followed by a conditional sensing action of fluent $On2$, contingent upon $On1$ having been found active:

$$Happens(sense(On1), 1) \quad \text{(N3)}$$

$$Happens(if(On1, sense(On2)), 2) \quad \text{(N4)}$$

Uniqueness-of-names axioms are:

$$\text{UNA}[On1], \text{UNA}[On2] \quad \text{(UN2)}$$

We can prove the knowledge goal by forming the parallel circumscriptions,
CIRC[SA1,CA1-2; $Initiates, Terminates$],
CIRC[N3-4; $Happens$], and
CIRC[CP1,SP2; $Possible$],

in conjunction with FD1, FD2, KSC2, UN2, knowledge axioms and Event Calculus axioms already presented:

$$HoldsAt(kw(Out), 3) \quad \text{(P10)}$$

This result demonstrates that the conditional sensing plan for knowledge acquisition in the domain of Figure 1 will always yield information about the state of the unobservable output. However, depending on the state of the first sensed fluent $On1$, the agent may not have to sense $On2$ at all.

7 Discussion and Conclusion

We have introduced a knowledge formalism and accompanied causal ramifications into Event Calculus to allow an agent to represent its knowledge of an environment sufficiently for reasoning about sensing actions, conditional actions, and inference of environmental unknowns. While increasing the expressivity of the EC formalism does allow for a wider range of problems to be represented, it also makes the mapping into an efficient executable form more challenging. In addition to [14], a promising approach is an amalgamated language logic programming system permitting self-reference as described by [1] and [4]. Future research plans also include augmenting language $\mathcal{E}$ with knowledge, which will provide an alternative semantic description from which efficient executables can be generated.

ACKNOWLEDGEMENTS

Thanks to Rob Miller for many helpful discussions, and to the referees whose comments improved this paper.

REFERENCES

[1] Kenneth A. Bowen and Robert A. Kowalski, 'Amalgamating language and metalanguage in logic programming', in *Logic Programming*, eds., Keith L. Clark and Sten-Åke Tårnlund, Academic Press, New York, (1982).
[2] N. J. Davies, 'A first order theory of knowledge, belief and action', in *Proceedings of the 10th European Conference on Artificial Intelligence*, ed., Bernd Neumann, pp. 408–412, Vienna, Austria, (August 1992). John Wiley Sons.
[3] Antonios C. Kakas and Rob Miller, 'Reasoning about actions, narratives and ramifications', volume 1(4). Linkping University Electronic Press, (1997).
[4] Robert Kowalski and Jin-Sang Kim, 'A metalogic programming approach to multi-agent knowledge and belief', 231–246, (1991).
[5] Hector J. Levesque, 'What is planning in the presence of sensing?', in *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pp. 1139–1146, Menlo Park, (August 4–8 1996). AAAI Press / MIT Press.
[6] Sheila A. McIlraith and Richard Scherl, 'What sensing tells us: Towards a formal theory of testing for dynamical systems', in *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pp. 483–490, Menlo Park, CA, (July 30– 3 2000). AAAI Press.
[7] R. Miller and M. Shanahan, 'The event calculus in classical logic - alternative axiomatisations', *Linkping Electronic Articles in Computer and Information Science, 4(16), 1999.*, (1999).
[8] Robert C. Moore, 'A formal theory of knowledge and action', in *Formal Theories of the Commonsense World*, Ablex Publishing Corp., Norwood, New Jersey, (1984).
[9] Leora Morgenstern, 'Knowledge preconditions for actions and plans', in *Proceedings of the National Conference on Artificial Intelligence*, pp. 867–874, Seattle, WA, (August 1987). (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 151–158, Morgan Kaufmann, 1988.).
[10] Richard B. Scherl and Hector J. Levesque, 'The frame problem and knowledge-producing actions', in *Proceedings of the Eleventh National Conference on Artificial Intelligence*, eds., Richard Fikes and Wendy Lehnert, pp. 698–695, Menlo Park, California, (1993). American Association for Artificial Intelligence, AAAI Press.
[11] Murray Shanahan, 'The event calculus explained', *Lecture Notes in Computer Science*.
[12] Murray Shanahan, *Solving the frame problem: a mathematical investigation of the common sense law of inertia*, MIT Press, 1997.
[13] Murray Shanahan, 'The ramification problem in the event calculus', in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99-Vol1)*, ed., Dean Thomas, pp. 140–146, S.F., (July 31–August 6 1999). Morgan Kaufmann Publishers.
[14] Murray Shanahan and Mark Witkowski, 'High-level robot control through logic', *Lecture Notes in Computer Science*.
[15] Michael Thielscher, 'Inferring implicit state knowledge and plans with sensing actions', in *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, eds., F. Baader, G. Brewka, and T. Eiter, volume 2174 of *LNAI*, Vienna, Austria, (September 2001). Springer.
[16] R. Turner, *Truth and Modality for Knowledge Representation*, Pitman Publishing: London, 1990.