Reasoning with Incomplete Initial Information and Nondeterminism in Situation Calculus

Lars Karlsson
Department of Computer and Information Science
Linkoping University, S-581 83 Linkoping, Sweden
larka@ida.liu.se

Abstract

Situation Calculus is arguably the most widely studied and used formalism for reasoning about action and change. The main reason for its popularity is the ability to reason about different action sequences as explicit objects. In particular, planning can be formulated as an existence problem. This paper shows how these properties break down when incomplete information about the initial state and nondeterministic action effects are introduced, basically due to the fact that this incompleteness is not adequately manifested on the object level. A version of Situation Calculus is presented which adequately models the alternative ways the world can develop relative to a choice of actions.

1 Introduction

The contribution of this paper is to highlight some problems that arise in Situation Calculus when incomplete initial information or nondeterministic actions are considered, in particular when reasoning about plans. A new version of Situation Calculus is proposed that avoids these problems by containing incompleteness and nondeterminism within individual logical models just like different choices of actions are.

Situation Calculus [McCarthy and Hayes, 1969; Reiter, 1991] is a formalism for reasoning about action and change that has been widely studied and applied to a broad range of problems. Its major strength relative to other formalisms is the possibility to do a large extent of reasoning on the object level. Object-level reasoning occurs completely *within* the object language and within one single theory as opposed to meta-level reasoning which involves reasoning *about* theories. The advantage of object-level reasoning is that it is completely supported by the deductive system of the logic in use. Problem solving is equivalent to finding logical proofs.

There are variations between different formalisms for action and change in how much reasoning can be done on the object level. On one extreme, there are STRIPS-style formalisms [Fikes and Nilsson, 1971], where even the state transition resulting from the application of an action is specified on the meta-level, namely as adding and deleting sentences to/from a state description. In the middle there are narrative-based formalisms such as Event Calculus [Kowalski and Sergot, 1986] and Sandewall's logics [Sandewall, 1994], where each theory *T* represents a specific choice of actions. At the other end of the scale there are branching formalisms such as Situation Calculus. The use of a branching time structure implies that different courses of actions can be contained within the same theory.

In Situation Calculus, henceforth referred to as SC, the time-structure is built up by successive applications of actions, starting from the initial situation S_0. Each action application results in a new situation. Situations have names specifying via what sequence of actions they are reached, for instance $do(Load, S_0)$, $do(Fire, S_0)$ and $do(Fire, do(Load, S_0))$. The fact that a fluent / holds in a situation *s* is expressed as $Holds(f,s)$, for instance $\neg Holds(alive, do(Fire, do(Load, S_0)))$. The effects of actions are specified using axioms such as the following.[1]

$$Poss(Fire, s) \land Holds(loaded, s) \Rightarrow (\neg Holds(loaded, do(Fire, s)) \land \neg Holds(alive, do(Fire, s))). \quad (i)$$

The predicate $Poss(a,s)$ denotes the conditions under which *a* is possible to execute, for instance $Poss(Fire,s) \equiv Holds(has\text{-}gun, s)$. The strength of this representation is that given a SC theory *T* with action effect axioms, all possible different action sequences and their effects are contained within T Situations are first-order objects, and thus reasoning about different situations (i.e. different action sequences) can be done on the object level, deductively. In particular, a planning problem can be straight-forwardly represented as an existence problem, whereas in narrative-based formalisms planning has to be formulated as an abductive, and thus metalogical, problem. Let T be a SC theory after the appropriate nonmonotonic preprocessing for minimizing change (for instance use of the explanation closure tech-

[1]For notational convenience, any variable occurring free in a formula is assumed to be universally quantified. For instance, in the axiom for *Fire* there is an implicit quantifier $\forall s$ prefixing the formula.

niques in [Reiter, 1991]), and let *G(s)* be a goal. In addition, let *Exec(s)* represent the fact that the situation s is reachable from the initial situation via a sequence of executable actions. Formally expressed, Exec(s) $S_0 \vee \exists a, s'. s = do(a, s') \wedge Poss(a, s') \wedge Exec(s')]$.*[s* => „ Then

the problem of finding a plan that achieves the goal is equivalent to proving the following.

$$\Gamma \vdash \exists s. [G(s) \wedge Exec(s)] \tag{2}$$

The binding of the *s* that satisfies $G(s) \wedge Exec(s)$ is the plan. For instance, if one assumes a proper theory *T* for loading and shooting, a solution to the problem

$$\begin{aligned}&\Gamma \wedge \neg Holds(loaded, S_0) \wedge Holds(alive, S_0) \vdash \\ &\quad \exists s. [\neg Holds(alive, s) \wedge Exec(s)]\end{aligned} \tag{3}$$

is found in the binding *a = do(Fire, do(Load,* S_0)).

The fact that action sequences (situations) are objects in SC, which makes it possible to perform reasoning tasks like planning purely deductively, has been emphasized as the major advantage of SC relative to narrative-based formalisms (see for instance [Reiter, 1996]). This has motivated a substantial amount of work attempting to incorporate reasoning about metric time and temporal relations into SC (see [Reiter, 1996] again), which is one of the strong points of narrative-based formalisms. Unfortunately, the advantage of SC breaks down when the implicit assumptions of information completeness and determinism are abandoned. Section 2 demonstrates this problem. Section 3 presents a version of SC which deals correctly with incomplete information and nondeterminism on the level of individual actions, and section 4 introduces constructs for action composition. Section 5 presents some conclusions and comparisons.

2 Problems with incomplete information

There is a strong assumption regarding complete information of the initial situation in the traditional SC formulation of the planning problem. To see why this is the case, consider the following scenario where an agent has to choose between two doors. One of the doors leads to freedom *(free),* whereas one leads to death *(-alive)* in the shape of a vicious cobra. Let // denote that the left door leads to freedom, and assume that *Holds(alive,So)AHolds(free,So)* but // is unspecified at S_0. There are two actions *GoLeft* and *GoRight* defined as follows:

$$\begin{aligned}&Poss(GoLeft, s) \equiv Holds(alive, s), \\ &Poss(GoLeft, s) \Rightarrow \\ &(Holds(lf, s) \Rightarrow Holds(free, do(GoLeft, s)) \wedge \\ &\neg Holds(lf, s) \Rightarrow \neg Holds(alive, do(GoLeft, s)))\end{aligned} \tag{4}$$

$$\begin{aligned}&Poss(GoRight, s) \equiv Holds(alive, s), \\ &Poss(GoRight, s) \Rightarrow \\ &(\neg Holds(lf, s) \Rightarrow Holds(free, do(GoRight, s)) \\ &\wedge \\ &Holds(lf, s) \Rightarrow \neg Holds(alive, do(GoRight, s)))\end{aligned} \tag{5}$$

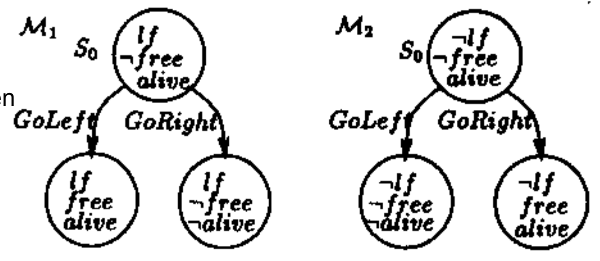

Figure 1: In model $\mathcal{M}_1$, $do(GoLeft, S_0)$ passes as a plan, and in $\mathcal{M}_2$, $do(GoRight, S_0)$ passes as a plan.

The following inference is correct (again, Γ is the theory after the appropriate nonmonotonic preprocessing)

$$\begin{aligned}&\Gamma \wedge Holds(alive, S_0) \wedge \neg Holds(free, S_0) \vdash \\ &\quad \exists s. [Holds(free, s) \wedge Exec(s)]\end{aligned} \tag{6}$$

In the case where $Holds(lf, S_0)$, the binding $s = do(GoLeft, S_0)$ proves the existential, and in the case where $\neg Holds(lf, S_0)$, $s = do(GoRight, S_0)$ proves the existential (Fig. 1). However, neither of these can be considered a plan. The problem is that different initial situations are realised in different models, whereas the statement $\exists s. [Holds(free, s) \wedge Exec(s)]$ should be true in each individual model. Thus, it suffices to find one plan for each model; the plans can be different in different models. Eq. (2) states that "for each possible initial situation one should find a plan that leads to the goal". The correct formulation should be "find a plan that leads to the goal for each possible initial situation". Unfortunately, the latter is not possible to express in SC. Even quantifying over initial situations like in

$$\begin{aligned}&\forall s_0 . [Holds(alive, s_0) \wedge \neg Holds(free, s_0) \Rightarrow \\ &\quad \exists s. [Holds(free, s) \wedge Exec(s_0, s)]],\end{aligned} \tag{7}$$

just repeats the same erroneous formulation as above *(Exec(so,s)* denotes that there is a sequence of executable actions from so to s). The dual problem of negative plan existence gives rise to similar difficulties. $\neg\exists s.[Holds(free, s) \wedge Exec(s)]$ is clearly an inadequate formulation of negative plan existence for the two-doors scenario. Notice that it contradicts (6). Similar counterexamples can be constructed where incompleteness is due to an action with nondeterministic effects. As the action, due to the *do* function, just has one resulting situation in each model, it suffices to find one plan for each model.

In summary, when the implicit assumptions of complete initial information and determinism are abandoned, severe problems arise for SC. The reasons are the identification of plans with situations and, in the case of nondeterminism, the use of a do function. Thus, the critique presented here applies to all versions of SC that have any of these two features. As a consequence, it is not possible to express properties such as positive and negative plan existence on the object level, and this severely limits the usefulness of SC as a tool for analysis and specification of higher-level reasoning.

3 Actions and their effects

In this and the following sections, we will introduce a new version of SC. It is a state-based approach, where the frame problem is solved in a way similar to the PMON logic by Sandewall [Sandewall, 1994; Doherty, 1994]. PMON is a narrative-based logic with integer time and has been formally assessed correct for sequential worlds with actions with context-dependent and nondeterministic effects.

The following types are used: *S* for situations, F for fluents, *O* for objects in the domain and *A* for primitive actions. The two predicates *Holds(f, a)* and *Occl(f,* a, *a)* respectively denote that fluent / is true in situation s_1 and that if action a is executed in s, then / might be changed in the next state. *Occl* stands for "occluded" and denotes an explicit exception to the general principle of no-change. Notice that *Occl* represents a potential to change, not a necessity.

An essential feature of the logic is that the *do(a,8)* function representing the result of doing *a* is replaced by a predicate *Rea(s,a,$').* This makes it possible to represent nondeterminism within individual models; if *a* behaves nondeterministically in situation s then there are more than one *a'* such that *Res(s,a,s').* In previous approaches to nondeterminism in SC (e.g. [Lin, 1996]), *do(a, a)* has denoted different situations in different models.

3.1 Elements of a theory

A theory consists of a set of axioms for unique names of objects, fluents and actions (UNA), and the following axioms.

Situation existence (SE). A second-order situation existence axiom (SE) defines the space of situations. In Baker's state-based approach [Baker, 1989], a situation existence axiom was used to obtain a correct minimization of change. In the approach presented here, there are no *do* terms denoting situations. Thus, without SE there might be models where some relevant situations are missing or even where the situation space is completely empty (an example of the former is given later in Fig. 2). SE states that for each possible state, that is to say each truth assignment φ over the set of fluents, there is a corresponding situation *a.* In addition, all situations are unique.

$$\forall\Phi.\exists s.\forall f.\Phi(f) \equiv Holds(f,s) \quad (8)$$

$$\forall s,s'.\forall f.(Holds(f,s) \equiv Holds(f,s')) \Rightarrow s = s' \quad (9)$$

If the set of fluents is finite, a first-order SE is possible [Baker, 1989].

Action effects (LAW). TO get a compact representation of action effects, a *Cauae* macro is used. *Cauae* involves both a statement that a fluent is allowed to change, and a statement of the result of this change.

$$Cause(s,a,s',f,T) \stackrel{\text{def}}{=} Occl(f,a,s) \wedge (Res(s,a,s') \Rightarrow Holds(f,s')) \quad (10)$$

$$Cause(s,a,s',f,F) \stackrel{\text{def}}{=} Occl(f,a,s) \wedge (Res(s,a,s') \Rightarrow \neg Holds(f,s')) \quad (11)$$

The effects of actions are specified in a set LAW. The normal form of an effect law for an action *a* is as follows.

$$\forall s,s'.\bigwedge_i(\ \alpha_i(s) \Rightarrow \bigvee_j(\bigwedge_k Cause(s,a,s',f_{ik},V_{ijk}))) \quad (12)$$

The formulae $\alpha_i(s)$ should only contain *Holds* expressions relating to *s*, and $V_{ijk} \in \{F,T\}$. The consequents are disjunctions of conjunctions, where each conjunction represents one possible nondeterministic outcome of the action. In each such disjunction, all conjunctions should contain the same set of fluents.

The following is an example of an action with a nondeterministic outcome.

$$\begin{aligned} &T \Rightarrow \\ &[(Cause(s,FlipCoin,s',has\text{-}coin,F) \wedge \\ &\quad Cause(s,FlipCoin,s',heads\text{-}up,F)) \vee \\ &\ (Cause(s,FlipCoin,s',has\text{-}coin,F) \wedge \\ &\quad Cause(s,FlipCoin,s',heads\text{-}up,T))] \end{aligned} \quad (13)$$

No-change (NCH). If a fluent is not explicitly influenced by an action (that is appears in a *Cause* statement) it is assumed not to change. This is realised with the no-change axiom (NCH), which only allows fluents that are occluded to change. NCH states that if *s'* is a situation resulting from doing *a* in *s* then only those fluents explicitly influenced by the action *a* (that is to say *Occl(f, a, s)*) may change. The fluents that are not occluded must have the same truth values after *a* as they had before *a*. NCH is as follows.

$$Res(s,a,s') \Rightarrow \forall f.[\neg Occl(f,a,s) \Rightarrow Holds(f,s) \equiv Holds(f,s')] \quad (14)$$

Possibility to execute actions (POSS and EXE). Rules for when actions are possible to execute (POSS) are written on the form $Poss(a,s) \Rightarrow \alpha(s)$. There is also a condition that *s'* is a resulting situation only if *a* is possible to execute in *s* (EXE).

$$Res(s,a,s') \Rightarrow Poss(a,s) \quad (15)$$

3.2 Reasoning with a theory

In order to obtain a theory suitable for reasoning, a step of nonmonotonic processing is required. In the two-doors scenario, the possibility conditions (POSS) and effect laws (LAW) for the two actions of going through the left door and the right door can be defined as follows.

$$Poss(GoLeft,s) \Rightarrow Holds(alive,s) \wedge \neg Holds(free,s) \quad (16)$$

$$\begin{aligned} &(Holds(lf,s) \Rightarrow \\ &\quad Cause(s,GoLeft,s',free,T)) \wedge \\ &(\neg Holds(lf,s) \Rightarrow \\ &\quad Cause(s,GoLeft,s',alive,F)) \end{aligned} \quad (17)$$

$$Poss(GoRight,s) \Rightarrow Holds(alive,s) \wedge \neg Holds(free,s) \quad (18)$$

$$\begin{array}{l}(Holds(lf,s) \Rightarrow \\ \quad Cause(s, GoRight, s', alive, F)) \wedge \\ (\neg Holds(lf,s) \Rightarrow \\ \quad Cause(s, GoRight, s', free, T))\end{array} \tag{19}$$

Using the definition of *Cause* in (10), (11) on the statements (17) and (19) results in (20) and (21) respectively.

$$\begin{array}{l}Holds(lf,s) \Rightarrow [Occl(free, GoLeft, s) \wedge \\ \quad (Res(s, GoLeft, s') \Rightarrow Holds(free, s'))] \wedge \\ \neg Holds(lf,s) \Rightarrow [Occl(alive, GoLeft, s) \wedge \\ \quad (Res(s, GoLeft, s') \Rightarrow \neg Holds(alive, s'))]\end{array} \tag{20}$$

$$\begin{array}{l}Holds(lf,s) \Rightarrow [Occl(alive, GoRight, s) \wedge \\ \quad (Res(s, GoRight, s') \Rightarrow \neg Holds(alive, s'))] \wedge \\ \neg Holds(lf,s) \Rightarrow [Occl(free, GoRight, s) \wedge \\ \quad (Res(s, GoRight, s') \Rightarrow Holds(free, s'))]\end{array} \tag{21}$$

This in turn can be rewritten as follows, separating the *Occl* (22) and *Res* (23) parts.

$$\begin{array}{l}[(Holds(lf,s) \wedge \langle f,a\rangle \in \{\langle free, GoLeft\rangle, \\ \quad \langle alive, GoRight\rangle\}) \vee \\ (\neg Holds(lf,s) \wedge \langle f,a\rangle \in \{\langle alive, GoLeft\rangle, \\ \quad \langle free, GoRight\rangle\})] \Rightarrow Occl(f,a,s)\end{array} \tag{22}$$

$$\begin{array}{l}Res(s,a,s') \Rightarrow \\ \ [a = GoLeft \Rightarrow \\ \quad (Holds(lf,s) \Rightarrow Holds(free, s') \wedge \\ \quad\ \neg Holds(lf,s) \Rightarrow \neg Holds(alive, s'))] \wedge \\ \ [a = GoRight \Rightarrow \\ \quad (Holds(lf,s) \Rightarrow \neg Holds(alive, s') \wedge \\ \quad\ \neg Holds(lf,s) \Rightarrow Holds(free, s'))]\end{array} \tag{23}$$

The nonmonotonic step consists of turning the respective sums of conditions for *Poss*, *Occl* and *Res* into biconditionals. The conditions for *Poss* are all contained in POSS and give the following sentence, which characterizes exactly when specific actions are possible to execute.

$$\begin{array}{l}Poss(a,s) \equiv \\ \ ((a = GoLeft \vee a = GoRight) \Rightarrow \\ \quad Holds(alive, s) \wedge \neg Holds(free, s))\end{array} \tag{24}$$

The *Occl* predicate should hold only when explicitly stated in LAW. Thus, the conditions in LAW for *Occl* (22) give the following sentence, which characterizes exactly under what conditions specific fluents may change.

$$\begin{array}{l}Occl(f,a,s) \equiv \\ (Holds(lf,s) \wedge \\ \langle f,a\rangle \in \{\langle free, GoLeft\rangle, \langle alive, GoRight\rangle\}) \vee \\ (\neg Holds(lf,s) \wedge \\ \langle f,a\rangle \in \{\langle alive, GoLeft\rangle, \langle free, GoRight\rangle\})\end{array} \tag{25}$$

Conditions for *Res* are contained in NCH (14), EXE (15) and LAW (23). Together they give the following completion of *Res*.

$$\begin{array}{l}Res(s,a,s') \equiv \\ \ [\ (\forall f. \neg Occl(f,a,s) \Rightarrow \\ \quad (Holds(f,s) \equiv Holds(f,s'))) \wedge \\ \quad Poss(a,s) \wedge \\ \quad [a = GoLeft \Rightarrow \\ \quad\ (Holds(lf,s) \Rightarrow Holds(free, s') \wedge \\ \quad\ \ \neg Holds(lf,s) \Rightarrow \neg Holds(alive, s'))] \wedge \\ \quad [a = GoRight \Rightarrow \\ \quad\ (Holds(lf,s) \Rightarrow \neg Holds(alive, s') \wedge \\ \quad\ \ \neg Holds(lf,s) \Rightarrow Holds(free, s'))]\ ]\end{array} \tag{26}$$

Observe that *Res* is determined by *Poss* and *Occl*. Informally, this means that first one decides when action are possible to execute and in what circumstances fluents are allowed to change. After that, and with these constraints in mind, the result relation is generated.

The completion of *Res* implies that the extension of *Res* is maximized. Why this is important becomes clear if one considers a nondeterministic action, such as the coin flipping action (13). From a situation s such that *Holds*(*has-coin*, s), *FlipCoin* should be able to lead both to a situation s'_1 such that $\neg$*Holds*(*heads-up*, s'_1) and to another situation s'_2 such that *Holds*(*heads-up*, s'_2). That is to say, both *Res*(s, *FlipCoin*, s'_1) and *Res*(s, *FlipCoin*, s'_2) should hold. In order to guarantee this, *Res* is maximized. If *Res* was not maximized, there would be models where either $\neg$*Res*(s, *FlipCoin*, s'_1) or $\neg$*Res*(s, *FlipCoin*, s'_2) is true, and in those models *FlipCoin* would not be nondeterministic. Even worse, there might be models where both $\neg$*Res*(s, *FlipCoin*, s'_1) and $\neg$*Res*(s, *FlipCoin*, s'_2) are true, and thus *FlipCoin* would not be possible to execute. In short, maximizing *Res* implies that there is one resulting situation for each possible outcome of an action in a given situation.

The completions of the three predicates, that is (24), (25) and (26), contain all relevant information regarding the effects of actions, and can be used for theorem proving together with SE and UNA. Figure 2 shows one model. One can notice that there are only arcs starting from the upper two situations, as these are the only ones that satisfy *Poss* (24). Furthermore, observe that all arcs result in fluent changes that are sanctioned by *Occl* (25).

To summarize, the process of generating the completions of *Occl*, *Poss* and *Res* is as follows. (a) The extension of *Occl* is minimized relative to LAW, where it only occurs positively. (b) The extension of *Poss* is maximized relative to POSS, where it only occurs negatively. (c) The extension of *Res* is maximized relative to NCH, EXE and LAW, where it only occurs negatively. This process can be automated with second-order circumscription [Lifschitz, 1985], as follows.

$$\begin{array}{l}\text{UNA} \wedge \text{SE} \wedge Circ[\text{LAW}; Occl] \wedge \\ Circ[\text{POSS}; \neg Poss] \wedge \\ Circ[\text{NCH} \wedge \text{EXE} \wedge \text{LAW}; \neg Res]\end{array} \tag{27}$$

The expression $Circ[\Gamma; \neg P]$ denotes that the extension of the predicate P is maximized. Although the result

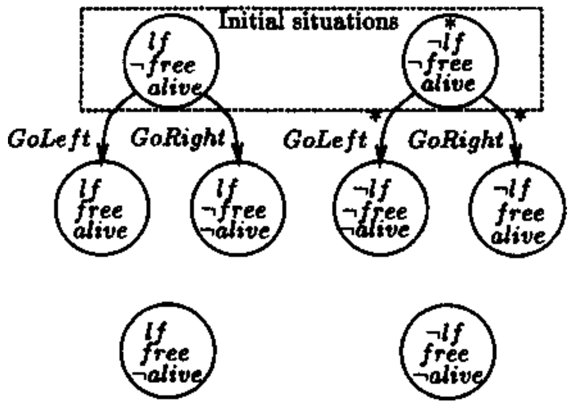

Figure 2: A model of the two-doors scenario (all other models are isomorphic). Neither *GoLeft* nor *GoRight* leads to the goal *free* from all initial situations satisfying $\neg free \wedge alive$. However, if SE were excluded and the situation and arcs marked with asterisks were missing, *GoLeft* would have appeared to lead to *free.*

of this circumscription is a second-order formula, two results by Lifschitz [Lifschitz, 1988] and the fact that *Occl, -Poss* and *-Res* only occur positively within the theories relative to which they are circumscribed imply that it is equivalent to a first-order formula. Thus, the circumscribed parts of the theory are completely first-order. However, this is not (always) the case for the situation existence axiom, or for the induction axiom mentioned in the next section.

4 Action composition

With the use of a *Res* predicate above, a sequence of actions is no longer represented as a term of nested do's. Instead, a type N for composite actions is introduced together with a predicate $Res^*(s, n, s')$. It is assumed that this type is supported by a (second-order) induction axiom (IND). Constructs for primitive actions and sequences are defined as follows.

$$Res^*(s, do(a), s') \equiv Res(s, a, s') \tag{28}$$

$$Res^*(s, seq(n_1, n_2), s') \equiv \exists s''.Res^*(s, n_1, s'') \wedge Res^*(s'', n_2, s') \tag{29}$$

A $Poss^0$ predicate for composite actions can be defined in terms of *Res** and *Poss.*

$$Poss^*(do(a), s) \equiv Poss(a, s) \tag{30}$$

$$Poss^*(seq(n_1, n_2), s) \equiv [Poss^*(n_1, s) \wedge \forall s'.Res^*(s, n_1, s') \Rightarrow Poss^*(n_2, s')] \tag{31}$$

Notice that this definition implies that all the primitive actions along any path described by $seq(n_1, n_2)$ must be possible to execute.

Now, the positive (negative) plan existence problem can be formulated as follows (let *I(s)* describe the initial conditions).

$$(\neg)\exists n.\forall s.[I(s) \Rightarrow [Poss^*(n, s) \wedge \forall s'.Res^*(s, n, s') \Rightarrow G(s')]] \tag{32}$$

For instance, consider the two-doors scenario again. It is obvious that for neither n = *do(GoLeft)* nor n = *do(GoRight)* (Fig. 2), it holds that

$$\forall s.[Holds(alive, s) \wedge \neg Holds(free, s) \Rightarrow [Poss^*(n, s) \wedge \forall s'.Res^*(s, n, s') \Rightarrow Holds(free, s')]]. \tag{33}$$

Also observe the importance of the situation existence (SE) axiom. If there are models where situations are missing, one might fail to conclude that there is no plan (Fig 2).

When planning with an incompletely specified initial situation and nondeterminism, the concept of a plan as a sequence appears to be insufficient. What one would like is the possibility to take different courses of actions depending on the circumstances. This is the approach taken in contingency planning or conditional planning [Peot and Smith, 1992]. In order to demonstrate the feasibility of representing plans as complex terms, it is important to address the problem of conditional constructs. The following is a definition for *Res** and *Poss** for a conditional construct (the predicate *Holds'* denotes that a condition holds).

$$Res^*(s, cond(c, n_1, n_2), s') \equiv (Holds'(c, s) \wedge Res^*(s, n_1, s')) \vee (\neg Holds'(c, s) \wedge Res^*(s, n_2, s')) \tag{34}$$

$$Poss^*(cond(c, n_1, n_2), s) \equiv (Holds'(c, s) \wedge Poss^*(n_1, s)) \vee (\neg Holds'(c, s) \wedge Poss^*(n_2, s)) \tag{35}$$

In order to let one of the conditional branches be empty, a *noop* action is introduced.

$$Res^*(s, noop, s') \equiv s = s' \tag{36}$$

$$Poss^*(noop, s) \equiv T \tag{37}$$

The definition of *cond* raises a question regarding the status of the condition component c. One possibility is that c is a fluent, like in *cond(loaded, noop, do(Load))*. In [Levesque, 1996], c is a sensing action that returns either 0 (for false) or 1 (for true): *cond(check_loaded, noop, do(Load))*.

A more general approach, following a proposal by McCarthy [McCarthy, 1979] (also see [Moore, 1985]), is to represent complex conditions as terms as follows. First, there is a type $\mathcal{D}$ for object designators, which are either used as variables x_i (short for $x(i)$ where $i \in \mathbb{N}$), or as constants; the latter are formed with the function $@ : \mathcal{O} \rightarrow \mathcal{D}$. A function $V : \mathcal{D} \rightarrow \mathcal{O}$ maps object designators to objects. In particular, $V(@(x)) = x$. For instance, $@(fred)$ (or $@fred$) denotes a name that via V refers to the object $fred$. Then there is a type $\mathcal{C}$ for complex conditions. It includes references to the different fluents in the domain, like alive or in($@robot, @room1$), and constructs equivalent to the standard logical connectives and quantifiers and equality, like $\nabla : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ (or or to retain a term-like syntax). A predicate $Holds' : \mathcal{C} \times \mathcal{S}$ is used for complex conditions like $Holds$ is used for single fluents. $Holds'$ is defined in terms of *Holds*. For instance, $Holds'(\mathrm{loaded}\overline{\vee}\mathrm{alive}, s) \equiv Holds(loaded, s) \vee Holds(alive, s)$.

The following axioms define the *Holds'* predicate. For each fluent designator symbol f_i corresponding to a fluent symbol f_i, there is an axiom as follows.

$$Holds'(\mathrm{f}_i(d_1, \ldots, d_n), s) \equiv \\ Holds(f_i(V(d_1), \ldots, V(d_n)), s) \tag{38}$$

The reified equality relation $\overline{=}$ and the reified connectives $\overline{\neg}$, $\overline{\mathrm{C}} \in \{\overline{\wedge}, \overline{\vee}, \overline{\Rightarrow}, \overline{\equiv}\}$ and quantifiers $\overline{\mathrm{Q}} \in \{\overline{\exists}, \overline{\forall}\}$ are defined as follows.

$$\begin{aligned} &Holds'(d_1 \overline{=} d_2, s) \equiv V(d_1) = V(d_2) \\ &Holds'(\overline{\neg}\mathrm{c}, s) \equiv \neg Holds'(c, s) \\ &Holds'(\mathrm{c}_1 \overline{\mathrm{C}} c_2, s) \equiv \\ &\quad Holds'(\mathrm{c}_1, s)\, \mathrm{C}\, Holds'(c_2, s) \\ &Holds'(\overline{\mathrm{Q}} v.c, s) \equiv \\ &\quad \mathrm{Q}x.\, Holds'(\mathrm{sb}(c, v, \mathbb{Q}(x)), s) \end{aligned} \tag{39}$$

The substitution functions *sb : D x D x D —> D* and sb : *C* x *V x V* -► *C* are defined in the obvious way. Finally, a set of unique name axioms are supplied for the conditions. Essentially, two distinct terms of types D or *C* always denote two distinct objects, even if they refer to the same object via the *V* function or have the same truth value via the *Holds'* predicate. Conditions and designators are considered syntactical entities, and equality is based on syntactical identity, not reference.

Now it is possible to construct conditional plans such as

$$n = cond(\ \mathrm{lf}, do(GoLeft), do(GoRight)) \tag{40}$$

which is a solution to the two-doors-problem. It is straight-forward to show that n satisfies (33).

A final remark: the condition in the plan above refers to the actual state of the world. Preferably, such decisions should be based on the knowledge of the agent. That requires a theory that explicitly represents the agent's knowledge or beliefs [Moore, 1985; Levesque, 1996]. This is compatible with the technique presented in this section.

5 Conclusions

Situation Calculus is a formalism for reasoning about action and change that supports reasoning about alternative courses of action on the object level. However, it fails to provide adequate object-level support for reasoning about the different ways the world can develop relative to a specific choice of actions. This results in anomalies when Situation Calculus is used for higher-level reasoning such as planning, as illustrated by the two-doors scenario. This paper has presented a modified version of SC that deals with this problem. It allows multiple initial situations and multiple alternative results of actions within individual models of an SC theory. In fact, the approach presented here can be considered a "modalization" of SC — it is possible to reason about possible and necessary results of a course of actions — although no explicit modal operators have been introduced. Three important technical considerations are the use of a *Res* relation instead of a do function (for nondeterminism, which is central to the paper), a situation existence axiom (as there are no terms denoting individual situations) and a type for composite actions, more suitable than individual situations for representing plans. Also observe the simple circumscription policy employed (27), which does not involve any prioritizing or varying of predicates. This policy is a variant of Sandewall's PMON policy [Sandewall, 1994], originally for an integer time structure, which has been altered to fit the new temporal structure.

The motivating example of this paper is analogous to an example from [Manna and Waldinger, 1987] with a monkey and two boxes, one of which contains a banana and one of which contains a bomb. In their SC-based approach, the synthesis of plans is supported by a deductive-tableau inference system. The point the authors make with the example is that the proofs for generating a plan should be constructive. Manna's and Waldinger's plan theory introduces plans as explicit objects. The application of an action or a plan to a state is modeled as a function yielding a new state, which implies determinism. Although their motives are similar, the approach of Manna and Waldinger is substantially different from the work presented in this paper; in particular, the authors are not concerned with axiomatic solutions to the frame problem.

There are also similarities to work on planning with sensory actions by (among others) Levesque [Levesque, 1996] and Davis [Davis, 1994], which involve explicit theories of knowledge. Levesque introduces complex action terms with sequential and conditional composition. Furthermore, a plan is required to lead to the goal from all situations that are compatible with the agent's knowledge of the initial situation, and not just the initial situation. However, the purpose is not to deal with the problems addressed in this paper. In particular, actions are assumed to be deterministic. Actually, as Levesque uses a *do* function, the agent will "know" that actions are deterministic, even if they are specified not to be. Also Davis provides a type for plans, and in addition a *Result* relation. The purpose with the relation is to let plans have unspecified results, and not to model nondeterminism. Davis mentions nondeterminism as a future problem.

Dynamic logic [Pratt, 1976] is a polymodal formalism that has been applied to reasoning about action and change and to planning [Rosenschein, 1981]. In dynamic logic semantics, a relation defines the possible next states when an action is executed in a state, in a manner similar to the *Res(s,a,s')* relation in this paper.

To conclude, the essential novelty of this paper is that it combines plans as explicit objects with not only incomplete initial information but also nondeterministic effects. This is done in classical logic, with the aid of a relatively simple circumscriptive policy based on a compact yet powerful solution to the frame problem. Regarding future work, an interesting project would be to combine nondeterminism and knowledge operators.

Acknowledgments

I would like to thank Silvia Coradeschi, Patrick Doherty, Marcus Bjareland and the anonymous referees for their comments on previous versions of this paper. This research has been supported by the Swedish Research Council for Engineering Sciences (TFR) grant 96-728.

References

[AAAI96, 1996] *Proceedings of the ThiHeenth National Conference on Artificial Intelligence.* AAAI Press, Menlo Park, California, 1996.

[Baker, 1989] Andrew B. Baker. A simple solution to the Yale shooting problem. In *Principles of Knowledge Representation and Reasoning: Proceedings of the First International Conference.* Morgan Kaufmann, 1989.

[Davis, 1994] Ernest Davis. Knowledge preconditions for plans. *Journal of Logic and Computation,* 4(5): 721-766, October 1994.

[Doherty, 1994] Patrick Doherty. Reasoning about action and change using occlusion. In *Proceedings of the Eleventh European Conference on Artificial Intelligence.* John Wiley & Sons, 1994.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence,* 2:189-208,1971.

[Kowalski and Sergot, 1986] R. A. Kowalskiand M. Sergot. A logic-based calculus of events. *New Generation Computing,* 4:67-95, 1986.

[Levesque, 1996] Hector J. Levesque. What is planning in the presence of sensing? In AAAI96 [1996].

[Lifschitz, 1985] Vladimir Lifschitz. Computing circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* pages 121-127. Morgan Kaufmann, 1985.

[Lifschitz, 1988] Vladimir Lifschitz. Pointwise circumscription. In M. Ginsberg, editor, *Readings in Nonmonotonic Reasoning.* Morgan-Kauffmaim, 1988.

[Lin, 1996] Fangzhen Lin. Embracing causality in specifying the indeterminate effects of actions. In AAAI96 [1996].

[Manna and Waldinger, 1987] Zohar Manna and Richard Waldinger. A theory of plans. In M.P. Georgeff and A.L. Lansky, editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop.* Morgan Kaufmann, 1987.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence,* 4:463-502, 1969.

[McCarthy, 1979] John McCarthy. First order theories of individual concepts and propositions. *Machine Intelligence,* 9, 1979.

[Moore, 1985] Robert C. Moore. A formal theory of knowledge and action. In J.R. Hobbs and R.C. Moore, editors, *Forma/ Theories of the Commonsense World,* chapter 9. Ablex, Norwood, New Jersey, 1985.

[Peot and Smith, 1992] M. Peot and D. Smith. Conditional nonlinear planning. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems.* Morgan Kaufmann, 1992.

[Pratt, 1976] V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings of the 11th IEEE Symposium on Foundations of Computer Science,* 1976.

[Reiter, 1991] Ray Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy,* pages 359-360. Academic Press, 1991.

[Reiter, 1996] Ray Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference.* Morgan Kaufmann, 1996.

[Rosenschein, 1981] Stanley J. Rosenschein. Plan synthesis: A logical perspective. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* pages 331-37. Morgan Kaufmann, 1981.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents.* Oxford Press, 1994.