ACTIONS AND OTHER EVENTS IN SITUATION CALCULUS

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

2002 Jan 27, 7:49 p.m.

DRAFT DRAFT DRAFT DRAFT DRAFT

Abstract

This article presents a situation calculus formalism featuring events as primary and the usual actions as a special case. Events that are not actions are called *internal events* and actions are called *external events*. The effects of both kinds of events are given by effect axioms of the usual kind. The actions are assumed to be performed by an agent as is usual in situation calculus. An internal event e occurs in situations satisfying the *occurrence axiom* for that event.

A formalism involving actions and internal events describes what happens in the world more naturally than the usual formulations involving only actions supplemented by domain constraints. Ours uses only ordinary logic without special causal implications.

The first example is the buzzer with only internal events and which cannot be treated at all with domain constraints, because the system never settles down.

Our second example is the stuffy room scenario. One occurrence axiom states that when both vents are blocked and the room isn't stuffy, the event $Getstuffy$ occurs. Domain constraints are unneeded.

The stuffy room formalization tolerates an elaboration asserting that when the room becomes stuffy someone unblocks a vent. If we further add that someone else then finds the room cold and blocks the vent again, we get a system that oscillates.

The third example is the blocks world.

The nonmonotonic reasoning involves circumscribing occurrences, changes, and prevention one situation at a time.

Mostly the proposals of this paper are alternatives (better I hope) to other methods of formalizing some phenomena. However, the buzzer example and the elaboration in which the room being stuffy causes someone to unblock a vent don't seem to be treatable by many of the earlier methods.

Contents

1 Introduction: Actions and other events

[McC59] proposed mathematical logic as a tool for representing facts about the consequences of actions and using logical reasoning to plan sequences of actions that would achieve goals. Situation calculus as a formalism was proposed in [McC63] and elaborated in [MH69]. The name "situation calculus" was first used in [MH69] but wasn't defined there. [McC86a] proposed to solve the frame and qualification problems by circumscription. [Sha97] and [Rei01a] describe several situation calculus formalisms and give references.

This article treats events in situation calculus with actions by agents as a special case. Besides *effect axioms* formalizing $Result(e, s)$ [$do(e, s)$ in Canada and its colonies], there are *occurrence axioms* asserting that in situations satisfying certain expressions in the fluents, an event e occurs.[1]

It may be easier to see an action a as an event if we regard the action symbol a as an abbreviation for the event expression $Does(person, a)$. Looking at it this way prepares the way for elaborations with occurrence axioms that give conditions under which $person$ will do the action a.

Some common sense phenomena that have been treated in situation calculus using domain constraints are better treated by postulating that certain *internal events* occur in situations satisfying certain fluent expressions. Thus in the well known [GS88] stuffy room scenario, our formalization says that the room *becomes* stuffy when both vents are blocked and *becomes* not stuffy when at least one vent is unblocked.

[1]I suspect I need to pound the table a little here. Actions are just a kind of event, and formalized *reasoning about actions and change* need to treat events as the general case and those events which are actions as special. This has long seemed obvious to me, but I find that many other researchers don't want to use the same formalism for events that are not actions of agents and those which are.

The consequence has been the introduction of extensions to logic for treating what are called state constraints, most of which are better treated by formalizing events.

$Occurs(e, s)$ means that event e occurs in situation s. Three of the axioms of the stuffy room phenomenon are

$$\begin{aligned}
&Holds(Blocked1, s) \wedge Holds(Blocked2, s)\\
&\qquad \wedge \neg Holds(Stuffy, s) \rightarrow Occurs(Getstuffy, s),\\
&Holds(Stuffy, Result(Getstuffy, s)), \qquad (1)\\
&\text{and}\\
&Holds(Blocked1, Result(Block1, s)).
\end{aligned}$$

$Getstuffy$ is an internal event and occurs all by itself when the vents are blocked.

The reasoning we formalize is limited to forward projection in time, i.e. we project from a situation to situations that result from the occurrence of events. This permits a simple form of nonmonotonic reasoning in which we minimize one situation at a time..

We use circumscription to minimize occurrences, to minimize change (frame problem), and to minimize the fluents that prevent actions and other events (qualification problem).

Treating internal and external events by the same formalism admits elaborations that turn external events into internal events. Thus we can elaborate the stuffy room scenario by adjoining an occurrence axiom saying that when the room becomes stuffy, someone unblocks a vent, which makes the room unstuffy. The further elaboration that when a vent is unblocked, someone blocks it again, perhaps from feeling cold, then the system oscillates, i.e. never settles down.

An external event can create a situation in which the occurrence axiom for an internal event is satisfied. This leads to a new situation in which a new internal event can occur. When no more internal events occur the process *settles down*, and we can infer a statement about the resulting stable state. Stable states are usually characterized by *domain constraints*. In physics these states often minimize potential energy.

However, we begin with a system, a buzzer, that has only internal events and which never settles down. For that reason, the buzzer's behavior cannot be characterized by a domain constraint; attempting to write one in the usual way leads to a contradiction.

2 Sequential processes, determinism and completeness

We need to distinguish among *phenomena*, *processes*, and theories of phenomena and processes. Phenomena are aspects of the world, and processes are a kind of phenomenon—the kind this article concerns. We have to distinguish between processes and theories of processes, because the same process may be described by different theories, and these theories may be more or less complete.

A process may or may not be deterministic, but this article deals with deterministic processes. In a deterministic *discrete* process, a definite event occurs in each situation. However, a theory about the process may or may not permit inferring what this event is. We'll say that the theory is *complete* if it permits inferring what occurs in any *completely described* situation, i.e. a situation in which all the necessary fluents have values asserted in the theory. A lesser form of completeness is when an *initial situation* $S0$ is completely described, and the theory allows inferring complete descriptions of all situations that follow $S0$.[2]

The buzzer theory of the next section completely describes a discrete sequential process.

3 Formalizing a buzzer

A buzzer consists of a relay connected to a battery by a switch that is opened when the relay operates. If the switch is on, the relay operates and opens the switch which turns off the relay which closes the switch. Thus the circuit oscillates and *never settles down* to a stable state.

A buzzer has only internal events—at least once it is started, and this makes its operation easy to formalize.

State constraint axioms for formalizing a buzzer analogous to those used for the stuffy room scenario would be immediately contradictory, asserting that the relay is on if and only if it is off. Our present situation calculus formalism follows human common sense reasoning directly and requires no special causal formalim or logic with implications not equivalent to their

[2]In Reiter's situation calculus formalism, the situations following from $S0$ are all the situations there are. We don't wish to assume that.

contrapositives.

There are effect axioms and occurrence axioms. The former are well known and give the effects of events. The latter assert that in situations in which certain fluents hold, certain events will occur.

We distinguish between the fluent On(Sw) asserting that the switch is on and the event Onn(Sw) that turns the switch on. The fluent holding in a situation is asserted by $Holds(\mathrm{On}(Sw), s)$. Likewise for the fluent On(R) and the event Onn(R) that concern the relay. We also have Off and Offf for the switch and the relay.

Effect axioms:

$$\begin{aligned}
&Holds(On(R), Result(Onn(R), s))\\
&\neg Holds(On(R), Result(Offf(R), s))\\
&Holds(On(Sw), Result(Onn(Sw), s)\\
&\neg Holds(On(Sw), Result(Offf(Sw), s)).
\end{aligned} \tag{2}$$

Occurrence axioms:

$$\begin{aligned}
&\neg Holds(On(Sw), s) \wedge Holds(On(R), s) \rightarrow Occurs(Offf(R), s)\\
&Holds(On(Sw), s) \wedge \neg Holds(On(R), s) \rightarrow Occurs(Onn(R), s))\\
&Holds(On(R), s) \wedge Holds(On(Sw), s) \rightarrow Occurs(Offf(Sw), s)\\
&\neg Holds(On(R), s) \wedge \neg Holds(On(Sw), s) \rightarrow Occurs(Onn(Sw), s)
\end{aligned} \tag{3}$$

Note that each of the above occurrence axioms has a second term in the precondition. They are needed to avoid unwanted concurrent events.

Frame assertions—for now axioms:

$$\begin{aligned}
&e = Onn(Sw) \vee e = Offf(Sw)\\
&\qquad \rightarrow Holds(On(R), Result(e, s)) \equiv Holds(On(R), s)).
\end{aligned} \tag{4}$$

$$\begin{aligned}
&e = Onn(R) \vee e = Offf(R)\\
&\qquad \rightarrow Holds(On(Sw), Result(e, s)) \equiv Holds(On(Sw), s)).
\end{aligned} \tag{5}$$

These frame assertions tell what doesn't change. They are few enough in this case, since there are few actions and few fluents. In general it is more efficient to say what does change. In this case we have

$$\begin{array}{l} Changes(Onn(R), On(R), s), \\ Changes(Offf(R), On(R), s), \\ Changes(Onn(Sw), On(Sw), s), \\ \text{and} \\ Changes(Offf(Sw), On(Sw), s). \end{array} \tag{6}$$

In section 6 we describe how to get the frame assertions by circumscribing $Changes(e, f, s)$.

If an event e is asserted to occur in a situation s, we can characterize a *next* situation by an axiom.

Next axiom:

$$Occurs(e, s) \rightarrow Next(s) = Result(e, s) \tag{7}$$

The initial situation is given by

$$\neg Holds(On(Sw), S0) \wedge \neg Holds(On(R), S0) \tag{8}$$

We can proceed a step at a time. We have

$$Occurs(\mathrm{Onn}(Sw), S0) \tag{9}$$

in accordance with (3). Hence

$$Next(S0) = Result(Onn(Sw), S0), \tag{10}$$

and therefore, letting

$$S1 = Next(S0), \tag{11}$$

we have

$$Holds(\mathrm{On}(R), S1) \wedge Holds(\mathrm{Off}(Sw), S1). \tag{12}$$

Some elaborations of the buzzer axioms will be worth doing.

1. Allow the action of stopping the buzzer to occur at any situation.

2. Consider the action of stopping the buzzer as a concurrent event.

3. A concurrency elaboration along the lines of [McC95b] and [MC98] might be to have two non synchronized buzzers B1 and B2 with no guaranteed temporal relation between the events involving B1 and B2.

4 The stuffy room scenario

The well-known problem arises when the stuffy room is formalized with a domain constraint that when both vents are blocked by pillows the room is stuffy. This can lead to the unintended model that when one vent is already blocked the action of blocking the other event causes the blocked vent to become unblocked in order to minimize change. Some complication of the formalism is required to deal with the phenomenon. Direct formalization in terms of actions and events avoids the difficulty and corresponds better to the way we humans think about the problem.

We use fluents Blocked1, Blocked2, and Stuffy. We have the action events Block1, Unblock1, Block2, Unblock2 and the internal events Getstuffy and Ungetstuffy.

Effect axioms:

$$\begin{array}{l} Holds(Blocked1, Result(Block1, s)) \\ Holds(Blocked2, Result(Block2, s)) \\ \neg Holds(Blocked1, Result(Unblock1, s)) \\ \neg Holds(Blocked2, Result(Unblock2, s)) \\ Holds(Stuffy, Result(Getstuffy, s)) \\ \neg Holds(Stuffy, Result(Ungetstuffy, s)) \end{array} \tag{13}$$

Occurrence axioms:

$$\begin{array}{l} Holds(Blocked1, s) \wedge Holds(Blocked2, s) \wedge \neg Holds(Stuffy, s) \\ \qquad \rightarrow Occurs(Getstuffy, s) \\ \text{and} \\ (\neg Holds(Blocked1, s) \vee \neg Holds(Blocked2, s)) \wedge Holds(Stuffy, s) \\ \qquad \rightarrow Occurs(Ungetstuffy, s) \end{array} \tag{14}$$

The frame axioms are

$$\begin{array}{l} Changes(Block1, Blocked1, s), \\ Changes(Block2, Blocked2, s), \\ Changes(Unblock1, Blocked1, s), \\ Changes(Unblock2, Blocked2, s), \\ Changes(Getstuffy, Stuffy, s), \\ \text{and} \\ Changes(Ungetstuffy, Stuffy, s). \end{array} \tag{15}$$

How they work is described in section 6.

We need to distinguish between *internal events* like $Getstuffy$ and *external events* like $Block1$. As we shall see, an external event may be an internal event of a more comprehensive narrative, e.g. one in which $Block1$ occurs when Mike is annoyed by cold air coming from $Vent1$.

We can tell a simple sequential story by by first describing $S0$, e.g. by

$$\neg Holds(Blocked1, S0) \wedge \neg Holds(Blocked2, S0) \wedge \neg Holds(Stuffy, S0). \tag{16}$$

We can now write the narrative

$$\begin{array}{l} S1 = Result^*(Block1, S0) \\ S2 = Result^*(Block2, S1) \\ S3 = Result^*(Unblock2, S2) \\ S4 = Result^*(Block2, S3) \\ \text{etc.} \end{array} \tag{17}$$

Here $Result^*(e, s)$ is like the Rr of [McC95a]. It is the result of doing a followed by the occurrence of whatever internal events occur. The assumption is that some sequence of internal events will occur after which the situation remains the same until an external event occurs. $Result^*(e, s)$ is undefined in the buzzer example in which internal events occur forever.

$Result^*$ requires an induction axiom or schema. Here's one candidate:

$$\begin{array}{l} P(s) \wedge (\forall s\ e)(P(s) \wedge Occurs(e, s) \rightarrow P(Result(e, s))) \\ \rightarrow P(Result^*(e, s)). \end{array} \tag{18}$$

$$\begin{array}{l} Result^*(e, s) = Next^*(Result(e, s)) \\ (\forall e)(\neg Occurs(e, s)) \rightarrow Next^*(s) = s \\ Occurs(e, s) \rightarrow Next^*(s) = Next^*(Result(e, s)). \end{array} \tag{19}$$

We also have

$$p(Result(Nochange, s)) \equiv p(s). \tag{20}$$

The purpose of $Nochange$ is to allow events to occur in $Result^*(e, s)$ without spoiling the settling down.

In the present case we will have $S1 = Result^*(Block1, S0)$ will be the same as $Result(Block1, S0)$, because no internal event will occur in $Result(Block1, S0)$. However, we'll have

$$S2 = Result^*(Block2, S1) = Result(Getstuffy, Result(Block2, S1)), \tag{21}$$

because now the internal event $Getstuffy$ will occur. Thus we'll have $\neg Holds(Stuffy, S1)$ but $Holds(Stuffy, S2)$, $\neg Holds(Stuffy, S3)$, and $Holds(Stuffy, S4)$.

We can write

$$\begin{aligned} S4 = {} & Result^*(Block2, Result^*(Unblock2, \\ & \qquad Result^*(Block2, Result^*(Block1, S0)))) \\ = {} & Result(Getstuffy, Result(Block2, \\ & \qquad Result(Ungetstuffy, Result(Unblock2, \\ & \qquad Result(Getstuffy, Result(Block2, Result(Block1, S0))))))), \end{aligned} \tag{22}$$

which can also be written

$$\begin{aligned} S4 &= Result^*(Block1; Block2; Unblock2; Block2, S0) \\ &= Result^*(Block1; Block2; Getstuffy; Unblock2; \\ & \qquad\qquad Ungetstuffy; Block2; Getstuffy, S0). \end{aligned} \tag{23}$$

Here we extend the meaning of *Result* to allow a sequence of events as an argument.

We regard the formulas with $Result^*$ and $Result^*$ as syntactic abbreviations for the previous formulas, so we won't try to axiomatize them.

4.1 Telling stories using $Occurs$ and $Next$

Another way of telling stories is to always use $Occurs$. An external event is axiomatized by asserting that it occurs.

The above story is then given by

$$\begin{array}{l}
Occurs(Block1, S0) \\
S1 = Next(S0) = Result(Block1, S0) \\
Occurs(Block2, S1) \\
S1' = Next(S1) = Result(Block2, S1) \\
Occurs(\text{Getstuffy}, S1'), \text{by inference} \\
S2 = Next(S1') = Result(Getstuffy, S1') \\
Occurs(Unblock2, S2) \\
S2' = Next(S2) = Result(Unblock2, S2) \\
Occurs(Ungetstuffy, S2')\text{by inference} \\
S3 = Next(S2') = Result(Ungetstuffy, S2') \\
Occurs(Block2, S3) \\
S3' = Next(S3) = Result(Block2, S3) \\
Occurs(Getstuffy, S3')\text{by inference} \\
S4 = Next(S3') = Result(Getstuffy, S3').
\end{array} \tag{24}$$

We can also write the story more briefly as

$$\begin{array}{l}
Occurs(Block1, S0) \\
S1 = Next^*(S0) = Result^*(Block1, S0) \\
Occurs(Block2, S1) \\
S2 = Next^*(S1) = Result^*(Block2, S1) \\
Occurs(Unblock2, S2) \\
S3 = Next^*(S2) = Result^*(Unblock2, S2) \\
Occurs(Block2, S3) \\
S4 = Next^*(S3) = Result^*(Block2, S3)
\end{array} \tag{25}$$

Still more briefly

$$\begin{array}{l}
S4 = Next^*(Next^*(Next^*(Next^*(S0)))) \\
\quad = Result^*(Block2, Result^*(Unblock2, \\
\qquad\qquad Result^*(Block2, Result^*(Block1, S0))))
\end{array} \tag{26}$$

4.2 Two elaborations of the stuffy room scenario

The first elaboration says that when Pat finds the room stuffy he unblocks vent2. We have

$$Holds(Stuffy, s) \rightarrow Occurs(Unblock2, s), \tag{27}$$

or, more elaborately,

$$\begin{array}{l} Holds(Stuffy, s) \rightarrow Occurs(Becomes\text{-}Ucomforable(Pat), s), \\ Holds(Uncomfortable, Pat, Result(Becomes\text{-}Ucomforable(Pat), s)) \\ Holds(Uncomfortable, Pat, s) \rightarrow Occurs(Does(Pat, Unblock\text{-}Vent2), s) \\ \neg Holds(Blocked2, Result(Does(Pat, Unblock\text{-}Vent2), s0)). \end{array} \tag{28}$$

(24) remains the same except that perhaps we should change the notation so that instead of $S3$ and $S3'$ we write $S2''$ and $S2'''$, since these are now intermediate situations. The situation $S4$ is now unstable.

Now let's add a second elaboration in which Mike finds the room cold when there is an unblocked vent and blocks vent2. It is expressed by adding

$$Holds(Unstuffy, s) \rightarrow Occurs(Block2, s). \tag{29}$$

With both of these elaborations, we get an oscillation; Pat unblocks vent2 and Mike blocks it again. $Result^*$ and $Next^*$ are no longer defined.

5 The blocks world

Assume enough unique names axioms.

The blocks world involves the frame problem in a more significant way than do the buzzer and the stuffy room scenarios.

We use the predicate $Prevents(p, e, s)$ to say that a move is prevented by there being a block on top of the block to be moved or on the destination unless the destination is the table. We thereby skip the use of the fluent $Clear(x)$ prevalent in many blocks world sitcalc theories.

Here's the effect axiom for moving a block.

$$\begin{array}{l} (\forall p)(\neg(Prevents(p, Move(x, y), s) \wedge Holds(p, s))) \\ \qquad\qquad \rightarrow \\ (Holds(On(x, y), Result(Move(x, y), s))) \\ \wedge \\ ((Holds(On(x, z), s)) \rightarrow \neg Holds(On(x, z), Result(Move(x, y), s)))), \end{array} \tag{30}$$

and here are the axioms for prevention:

$$\begin{array}{l} Prevents(On(z,x), Move(x,y), s) \\ \text{and} \\ y \neq Table \rightarrow Prevents(On(z,y), Move(x,y), s). \end{array} \tag{31}$$

We adopt the usual way of emphasizing the frame problem by introducing the action of painting a block a certain color. Thus

$$\begin{array}{l} (\forall p)(\neg(Prevents(p, Paint(x,c), s) \wedge Holds(p,s))) \\ \quad \rightarrow Holds(Color(x, color), Result(Paint(x, color), s)), \end{array} \tag{32}$$

or, using object valued, i.e. non propositional, fluents,

$$\begin{array}{l} (\forall p)(\neg(Prevents(p, Paint(x,c), s) \wedge Holds(p,s))) \\ \quad \rightarrow Value(Color(x), Result(Paint(x, color), s)) = color. \end{array} \tag{33}$$

The change axioms for the blocks world are

$$\begin{array}{l} Changes(Paint(x,c), Color(x), s), \\ Holds(On(x,z), s) \rightarrow Changes(Move(x,y), On(x,z), s) \\ \wedge Changes(Move(x,y), On(x,y), s). \end{array} \tag{34}$$

The nonmonotonic reasoning associated with the blocks world will be discussed after the section dealing with nonmonotonic reasoning in situation calculus in general.

6 Nonmonotonic reasoning—situation by situation

We use circumscription to minimize the events that occur in a situation, the fluents that might prevent an event from having its standard effect, and the changes in fluents. In contrast to the formalism of [McC86a] which minimized predicates over all the arguments, we minimize for each successive situation separately. However, in doing this minimization in s we take as fixed the $Holds(f,s)$ sentences and the $Value(exp,s) = \ldots$ sentences inferred from the effects of the previous situation.

Doing the nonmonotonic reasoning in situations successively corresponds to the way people predict the consequences of sequences of actions and events.

It seems to give the same conclusions as Yoav Shoham's chronological minimization [Sho88] but is computationally more straightforward. It also avoids the Yale shooting problem and its friends.[3]

However, we advocate this only for projection problems, i.e. reasoning about the future from information about the past. It sometimes has the same effects as chronological minimization [Sho88] but is formally quite different—and simpler. The method is not appropriate for the *stolen car scenario* in which one has to reason from an assertion (that the car is missing) about a later situation. [4]

With the present formalism, the person or agent setting up the problem must know that projection forward in time is appropriate. It would be better if this were a consequence of the formalized facts.

Now let's consider circumscribing at each situation separately. The simplest case is when we have a predicate $Foo(x, y, s)$.

We write

$$\begin{aligned}
&Foo' \leq_s Foo \equiv (\forall x\ y)(Foo'(x, y, s) \rightarrow Foo(x, y, s)),\\
&(Foo' <_s Foo) \equiv (Foo' \leq_s Foo) \wedge \neg(Foo' =_s Foo),\\
&\text{and}\\
&Foo' =_s Foo \equiv (\forall x\ y)(Foo'(x, y, s) \equiv Foo(x, y, s)).
\end{aligned} \tag{35}$$

Then the circumscription of $Foo(x, y, s)$ takes the form

$$\begin{aligned}
Axiom(Foo, vars, s) \wedge (\forall foo'\ vars')(Axiom(foo', vars')\\
\rightarrow \neg(foo' <_s Foo)).
\end{aligned} \tag{36}$$

Here $vars$ stands for a list of the entities being varied as Foo is minimized.

This spells out to

[3]The ideas of internal and external events of the preceding sections are independent of the formalism used for nonmonotonic reasoning. For example, Golog [Rei01a] or the Causal Calculator [aA01] could be used—perhaps with some modifications for the buzzer and the oscillating stuffy room.

[4]Actually part of the stolen car scenario can be treated provided we don't suppose that the car being missing is to be projected from information about the past. Certainly we can go forward from the situation in which the car is missing to further events in the future. Likewise, in the story of Junior's travels [McC95b], we can assert that Junior loses his ticket to Moscow in London and reason forward from that fact.

$$
\begin{aligned}
&Axiom(Foo, vars, s) \wedge (\forall foo'\ vars') \\
&(Axiom(foo', vars') \wedge ((\forall x\ y)(foo'(x, y, s) \rightarrow Foo(x, y, s)) \\
&\rightarrow (\forall xy)(Foo(x, y, s) \equiv foo'(x, y, s)))).
\end{aligned} \tag{37}
$$

Call this formula $Circ(Axiom; Foo; vars; s)$. This is the notation of [Lif94] with the addition of the argument s to say that s is kept fixed.

The general frame axioms are

$$\neg Changes(e, p, s) \rightarrow Holds(p, Result(e, s)) \equiv Holds(p, s) \tag{38}$$

for propositional fluents and

$$\neg Changes(e, f, s) \rightarrow Value(f, Result(e, s)) = Value(f, s). \tag{39}$$

for general fluents.

Suppose we allow complex fluents, say $p\ And\ q$ when p and q are propositional fluents. We then need an axiom

$$Changes(e, p, s) \vee Changes(e, q, s) \rightarrow Changes(e, p\ And\ q, s). \tag{40}$$

Similar axioms are required for the other propositional functions of fluents and for the compositions of non-propositional fluents.

[This leads to difficulties when we want to delimit what changes, since there are arbitrarily complex compositions of fluents. We'll confine ourselves to elementary fluents for now by not putting compositions in the language.]

In these circumscriptions we also minimize $Holds$.

This tolerates elaborations like

$$Holds(Weak, s) \rightarrow Prevents(Weak, Move(x, y), s). \tag{41}$$

If $Holds(Weak, s)$ isn't asserted, $Move(x, y)$ will not be prevented.

Lin and Shoham, [LS95] consider a theory of action to be *provably correct* if doing the nonmonotonic reasoning results in a complete nonmonotonic theory of the action.

6.1 Nested circumscriptions

We can regard the successive circumscriptions as nested, i.e. in reasoning about $Result(e, s)$, we take as an axiom the circumscriptions done in s. This would be similar to the nested abnormality theories of [Lif95]. The differences are interesting.

Lifschitz circumscribes predicates called ab, a different ab for each level of nesting. He gets rid of the abs in the result of the circumscription by writing $(\exists ab)Foo(ab)$ rather than just $Foo(ab)$. We circumscribe three predicates: $Prevents$, $Occurs$, and $Changes$. Maybe we could get rid of these by existential quantification, but it seems to me that at least $Occurs(e, S_k)$ should be part of the final story. We could accomplish this by introducing an Ab predicate and writing $\neg Ab(e, S_k) \rightarrow \neg Occurs(e, S_k)$ and varying $Occurs$. Existential quantification then gets rid of Ab while keeping $Occurs$. $Prevents$ and $Changes$ seem to be genuinely auxiliary predicates and might be removed from final formulas by existential quantification.

7 Circumscriptions in the blocks world

Let an initial situation $S0$ be described by the conjunction

$$\begin{aligned}&Holds(On(A, B), S0) \wedge Holds(On(B, C), S0)\\ &\wedge Holds(On(C, Table), S0) \wedge Holds(On(D, Table), S0)\\ &\wedge Value(Color(A), S0) = Red \wedge Value(Color(B), S0) = Blue\\ &\wedge Value(Color(C), S0) = Green \wedge Value(Color(D), S0) = Purple.\end{aligned} \tag{42}$$

The presence of other blocks than A, ...,D is excluded when we circumscribe $Holds$ in the situation $S0$. The result is

$$\begin{aligned}Holds(p, S0) \equiv p = On(A, B)&\\ \vee p = On(B, C) \vee p = On(C, Table) \vee p = On(D, Table).&\end{aligned} \tag{43}$$

Using a function for color saves us this trouble of limiting in the colors. If we used $Holds(Color(x, c), s)$ we would have to exclude block A having another color as well as red.

Call $T0$ the conjunction of the formulas (38–43) and the formulas (30–34). Circumscribing $Occurs(e, S0)$ using $T0$ lets us conclude that no event occurs in $S0$.

Now add

$$Occurs(Move(A, D), S0). \tag{44}$$

to begin a story.

Circumscribing $Occurs$ with $T0 \wedge Occurs(Move(A, D), S0)$ now shows that $Move(A, D)$ is the only event that occurs in $S0$. The circumscription formula is

$$Circ(T0 \wedge Occurs(Move(A, D), S0); Occurs; S0) \tag{45}$$

leads to

$$Occurs(e, S0) \equiv e = Move(A, D). \tag{46}$$

Circumscribing $Changes$ gives

$$Changes(Move(A, D), f, S0) \equiv f = On(A, B) \vee f = On(A, D), \tag{47}$$

and circumscribing $Prevents$ gives

$$(\forall f)\neg Prevents(f, Move(A, D), S0). \tag{48}$$

With these we can infer

$$Next^{*}(S0) = Next(S0) = Result(Move(A, D), S0) \tag{49}$$

and

$$Holds(On(A, D), Result(Move(A, D), S0)) \tag{50}$$

and

$$\begin{aligned}&Holds(f, Next(S0)) \equiv \\ &f = On(A, D) \vee f = On(B, C) \vee f = On(C, Table) \vee f = On(D, Table).\end{aligned} \tag{51}$$

The colors of the blocks remain the same.

8 Extensions and remarks

The content of this section isn't needed to project forward. However, there are some interesting issues.

Aarati Parmar pointed out that the semantics of an action theory is changed by the presence of *Occurs* axioms. Without *Occurs*, the interpretations are trees branching for the different events possible in a situation. With *Occurs* an interpretation must say which event occurs, so the interpretations are linear. The actual scenarios are sometimes mixed. The interpretations branch at the external actions and are linear where an *Occurs* axiom determines what internal action occurs.

8.1 Induction in the situation calculus

Several kinds of mathematical induction seem to be required. For example, one may want to prove a proposition $P(Next^*(s))$ by showing that it is true for s and is preserved by the events that occur between s and $Next^*(S)$. A related kind of induction is needed to prove that something is true for all situations arising in the operation of a buzzer. The simplest case of the $Next^*$ induction might be to show that a block unmoved by each of a sequence of events is in the same position in $Next^*(s)$.

The simplest situation calculus is Reiter's [Rei01b]. The formula is

$$P(S0) \wedge ((\forall a\ s)(P(s) \rightarrow P(Result(a, s)))) \rightarrow (\forall s)P(s). \tag{52}$$

Here are two formulas

$$\begin{aligned}&P(s) \wedge ((\forall e\ s)(P(s) \wedge Occurs(e, s) \rightarrow P(Nexts))))\\ &\rightarrow P(Next^*(s)).\end{aligned} \tag{53}$$

(53) is appropriate when $Next^*(s)$ is defined.

When $Next^*(s)$ is not defined, as in the buzzer case, we use $s \leq s'$ to mean that s' is a distant successor of s and have the axiom.

$$\begin{aligned}&P(s) \wedge s \leq s' \wedge ((\forall e\ s)(P(s) \wedge Occurs(e, s) \rightarrow P(Nexts))))\\ &\rightarrow P(s').\end{aligned} \tag{54}$$

8.2 Formalizing Oscillations

The buzzer oscillates, i.e. the situation repeats again and again. So does the stuffy room scenario with the two elaborations that cause Vent2 to become blocked and unblocked repeatedly. However, we don't need a complete repetition of the situation to have oscillation. Suppose, or example, we add a clock to the buzzer, a natural number valued fluent that each event increments by 1. Then although the whole situation would not repeat, we would still want to consider the system as oscillatory.

This suggests a relative notion of oscillatory, i.e. oscillatory with respect to certain fluents.

Moreover, we would like to consider the buzzer as oscillating even if we provide for it stopping its oscillation by being turned off.

8.2.1 Intervening in an oscillatory system

As we have described the buzzer, it cannot be turned off. Likewise the stuffy room process cannot be changed once we have added the elaborations about people blocking and unblocking the vent. See (27) and (29).

Here are two ways of doing it.

1. Introduce a new propositional fluent $Wait$ and add effect axioms to each of the events

$$Wait(Result(e, s)). \tag{55}$$

Add a precondition for each of the previous occurence axioms of $\neg Wait(s)$. Add a new axiom

$$[(\forall e)(\neg Occurs(e, s))] \rightarrow \neg Wait(Next(s)). \tag{56}$$

The effect of these axioms is after each event there is a wait situation in which an external event can occur, e.g. opening a switch that will turn off the buzzer.

2. Another way is with the axioms

$$\begin{aligned}&Occurs(a, s) \land External(a) \land Occurs(e, s)\\ &\rightarrow Next(s) = Result(a, Result(e, s))\end{aligned} \tag{57}$$

and

$$Occurs(e, s) \land (\forall e')(Occurs(e', s) \rightarrow e' = e) \rightarrow Next(s) = Result(e, s). \tag{58}$$

Public opinion (3 votes) prefers this solution, which is a limited kind of concurrency. Only certain kinds of interventions can be done this way.

8.3 State constraints after all

As was shown in Section 4, the condition for a room being stuffy is better formalized with effect axioms, occrrence axioms, and the events $Getstuffy$ and $Ungetstuffy$. Lin and Reiter [LR94] consider the Emperor's decree that no more than one object (block) be yellow, which may be regarded as a domain constraint. They point out that it is more efficient to encode the constraint as a precondition that a block may be painted yellow only if no block is already yellow. Their way of expressing this does not readily elaboate to require that no more than seven blocks be yellow.

I think logical AI needs a more complex treatment. It seems to me that efficiency conflicts with generality. It is bad or dangerous to have more than one yellow block, but perhaps only if one is not a special favorite of the emperor or if one is just about to die anyway. The point is that common sense (at least human level common sense) requires that such constraints tolerate elaboration. Human level common sense also allows the constraint to become an action precondition as a result of some inference. This inference should take place within the logical formalization.

Lin and Reiter include the following formula.

$$\begin{aligned}&(\forall x\ y\ s)(Poss(Paint(x,y),s)\\ &\equiv (Nearby(x,s) \wedge Haspaint(y,s)\\ &\wedge(\forall x_1)(Color(x_1,Yellow,s) \wedge y = Yellow \rightarrow x = x_1))).\end{aligned} \tag{59}$$

This formula is specialized to the emperor tolerating just one yellow block. If he tolerates 7 yellow blocks, we had better use set notation, i.e. refer to $card(\{x|Color(x,Yellow)\}) \leq 7$.[5]

There are some domain constraints that are not naturally formalized by internal actions. One is the blocks world constraint that a block may not be on top of itself. Formulas like

$$Above(Top(block), Bottom(block), s) \tag{60}$$

[5]I pound the table here because of some resistance to the idea that axiomatic set theory makes logical AI easier.

or even

$$Height(Top(block),s) - Height(Bottom(block),s) \geq 1.0cm \tag{61}$$

tell more about the world than the simple

$$\neg On(block, Top(block), s). \tag{62}$$

An important application for the direct use of state constraints is when an event starts a process that eventually leads to an equilibrium state. For example, if a drop a coin on the floor it will bounce around for a while and then settle down. It will reach equilibrium in a second or so, and I am interested in whether the coin ends up heads or tails rather than in the process of its settling down. In the case of the coin the equilibrium condition, at least what we want to know about it is easy to state, namely

$$\begin{aligned}&On(coin, floor, Result^*(Drop(coin,s))) \wedge (Heads(coin, Result^*(s))\\&\vee Tails(coin, Result^*(s))),\end{aligned} \tag{63}$$

where using $Result^*$ means that we are skipping by some internal events, in this case not formalized.

Consider tossing a coin. This sets in motion the events of the coin falling to the floor, bouncing a few times and settling down. All these processes cannot be followed in detail because of lack of knowledge of the initial conditions and the laws of motion. However, what we do know is that the coin will stop moving shortly and end up heads or tails. That's all we need to know.

While everyone understands this informally, physics offers an explanation. The kinetic energy is dissipated and the system of the tossed coin ends up in a local minimum of the potential energy of which there are two: heads and tails.

Physics aside, this is a case where state constraints, in this case for equilibrium give us the answer of how the process ends up.

Another example may be concocted from the elaborated stuffy rooom scenario. While Pat and Mike disagree in their preferences, under normal circumstances we can suppose they will come to an agreement in some short time. One will defer to the other in the matter of the blocked vents. As with the coins, the theory of eventual agreement doesn't predict what the agreement will be.

More generally, Aarati Parmar suggests that internal events arise in response to non-equilibrium situations.

8.4 Events whose occurrence depends on the past

Suppose we want George to unblock both vents when the room becomes stuffy. When he has unblocked one vent, the room becomes unstuffy, so the physical situation is as it was when he blocked the first vent, so he needs to remember that the room was previously stuffy. We can make occurrences depend on past situations by adding for each event e an additional effect axiom

$$Past(Result(e, s)) = s. \tag{64}$$

Notice that $Past(Past, s))$ is the situation two events back.

We can have George unblock Vent1 after he has unblocked Vent2 and the room has become unstuffy by introducing the occurrence axiom

$$Stuffy(Past(Past(s)) \rightarrow Occurs(Unblock1, s). \tag{65}$$

The history as just described does not say what events occurred. This information is provided by having for each event e the axiom

$$Lastevent(Result(e, s)) = e. \tag{66}$$

Notice that this formalization is noncommittal as to whether the information is in an actor's memory.

This seems neat, and maybe it will be useful.

8.5 The stolen car scenario

The stolen car scenario gives unwarranted conclusions when simple projection is applied.

$$\begin{array}{l} Stolen(car, s) \wedge Seen(car, s') \wedge s' < s \\ \rightarrow (\exists s'' thief)(s' < s'' < s \wedge Occurs(Does(thief, Steal(car)), s'') \end{array} \tag{67}$$

$$Stolen(car, s) \rightarrow \neg Findable(car, s). \tag{68}$$

Since there are several causes that may make the car not findable, the conclusion that it was stolen is normally an abduction.

8.6 Blocks world heuristics

The object is to represent heuristic information declaratively. We represent the final-position heuristic discussed in [Sie97] and [Sie99].

In general, a goal will be a predicate on situations. However, in the blocks world a goal can be like a situation assuming that the goal is to create a definite total configuration of blocks. We'll treat this conceptually easy case first and then consider how to generalize it, e.g. to making a given tower but not caring about the rest of the blocks.

We are interested in the predicate $Final(x, g, s)$ asserting that block x is in final position in situation s with respect to goal g. A block in final position need never be moved again, and it is always a good move to move a block to final position. Otherwise some block should be moved to the table, i.e. unless a block is moved to final position, it should not be moved onto another block.

The following formulas characterize what occurs when a strategy satisfying these conditions is employed.

$$\begin{aligned}
&Final(x, g, s) \equiv \\
&\textbf{if } Holds(On(x, Table), g) \\
&\qquad \textbf{then } Holds(On(x, Table), s) \\
&\qquad \textbf{else } (\exists y)(Holds(On(x, y), s) \land Holds(On(x, y), g) \land Final(y, g, s))
\end{aligned} \tag{69}$$

$$\begin{aligned}
Achieved(g, s) \equiv{} & (\forall x \in Blocks(g))(Holds(On(x, y), g) \\
& \rightarrow Holds(On(x, y), s)),
\end{aligned} \tag{70}$$

where

$$Blocks(s) = \{x | (\exists y) Holds(On(x, y), s)\}. \tag{71}$$

$$\begin{aligned}
&MovesFinal(x, y, g, s) \\
&\quad \equiv Holds(Clear(x), s) \land Holds(Clear(y), s) \\
&\qquad \land Final(y, g, s) \land Holds(On(x, y), g).
\end{aligned} \tag{72}$$

$$
\begin{aligned}
&\neg Achieved(g,s)\\
&\rightarrow\\
&((\exists x\ y)(MovesFinal(x,y,g,s) \wedge Occurs(Move(x,y),s)))\\
&\vee\\
&\neg(\exists x\ y)(MovesFinal(x,y,g,s))\\
&\wedge(\exists x)(\neg Final(x,g,s) \wedge \neg Holds(On(x,Table),s)\\
&\qquad\qquad \wedge Occurs(Move(x,Table),s))
\end{aligned}
\tag{73}
$$

This equation says that if the goal isn't yet achieved and there is a move of a block to final position, then such a move is made. If there isn't such a move, then some block is moved to the table. Since (73) doesn't say which block is moved to the table, additional formulas restricting which block is moved, e.g. to a block whose move to the table breaks a cycle inhibiting moving some block to final position, are compatible with (73).

(73) doesn't give a strategy, but it says what happens if any behavior that moves some block to final position when this is possible and otherwise moves some block to the table. From (73) it should be inferrable that the goal is eventually achieved.

A real strategy involves lookahead which should be done on an imaginary board on which moves can be made and taken back.

8.7 Change, the frame problem, persistence, and elaboration tolerance

The frame problem arises when we need to describe the effects of events that change some fluents and not others. The prototypical example is that moving a block doesn't change the colors of any blocks, and painting a block a certain color doesn't change the positions of any blocks. There are several ways of formalizing this. Monotonically we can attach the fluents to a frame - or state and use the functions $c(x,\xi)$ and $a(x, value, \xi)$. Non-monotonically, we can minimize change, assuming the only fluents are color and location.

However, painting and moving is too simple to exhibit the general problem. Suppose a, b, and c are three fluents satisfying $value(a,s) + value(b,s) + value(c,s) = 0$, and we want to describe the effect of changing a. We can have b unchanged or c unchanged but not both.

If we have a frame ξ, we can describe change in terms of the functions a and c. However, an effect axiom will sometimes have to specify changes in more than one variable.

Now for elaboration tolerance.

1. The simplest elaboration is to introduce new fluents unconnected with the old and perhaps new events involoving them.

2. The frame assertions we have used state that a fluent retains its value unless there is a reason for change. A more general persistence property would be that a process continues unless there is some reason for it not to.

   Clearly this should only apply to some processes, but the buzzer should be one of them. The buzzer will continue to buzz until something stops it. For this we propose a fluent $Buzzing$ and events $StartBuzzing$ and $StopBuzzing$. We want the general principle of minimizing change to apply to $Buzzing$. We'll need more if we want to consider interactions with specific states of the buzzer. It would seem that at least the fluent $Buzzing$ should be definable, but this will require a more general notion than an instantaneous situation.

3. Consider the buzzing elaboration of the stuffy room scenario given in section 4.2. Suppose Pete becomes annoyed at Pat and Mike repeatedly unblocking and blocking the vent2 and intervenes in some way. We would like to be able to formalize Pete's intervention in a way that does not depend on Pat and Mike having done a specific number of $Unblock2$ and $Block2$ actions.

   This isn't much, but I hope it's a start on turning oscillatory processes into fluents. Note that the matter gives people little trouble, except when we need to get specific about when an intervention occurs.

Graham White [Whi01] asks, "Does the situation calculus have a semantics?". He points out that two situation calculus theories, equivalent in their sets of models, can have different nonmonotonic consequences when change of fluents is minimized even when the fluents of each theory have definitions in terms of the fluents of the other. I draw a different conclusion from this phenomenon than White does. My conclusion is that a theory including circumscription specifies what is minimized, and when it is transformed to a different language, the circumscription formulas also need to be appropriately transformed. When a situation calculus theory is transformed, then expressions representing what fluents persist by default must also be transformed. Such a transformation will, in general, replace an atomic fluent by a complex expression. This means that when transformations are to be allowed, the circumscription formalism must allow the minimization of expressions and not just predicates.

9 Concluding remarks

As mentioned in Section 6, our formalism requires that the rules about what to circumscribe are outside the theory at first. In [McC86b] I proposed *simple abnormality theories* in which all the decision about what to circumscribe were included in axioms involving the single predicate *ab* which was the only predicate minimized. The paper was vague about what was to be varied, although varying all predicates seemed to work on the examples given, except that the formalism suffered from the unintended models of the Yale shooting problem [HM86] and similar problems discovered by Vladimir Lifschitz in the blocks world. I still held some hope that only the particular simple abnormality theories were inadequate, and that YSP and friends could be solved by different simple abnormality theories. Tom Costello [Cos98] showed that no simple abnormality theory could work under certain circumstances.

10 Acknowledgments

This work benefitted from discussions with Eyal Amir, Tom Costello, Ron Fadel, Hector Levesque, Vladimir Lifzchitz, Fangzhen Lin, Sheila McIlraith, Aarati Parmar, Raymond Reiter, and Tran Son.

This research was partly supported by SRI Subcontract No. 34-000144 under SPAWAR Prime Contract No. N66001-00-C-8018.

References

[aA01] Texas Action Group at Austin. Causal calculator home page, 2001. http://www.cs.utexas.edu/users/tag/cc.

[Cos98] Tom Costello. The expressive power of circumscription. *Artificial Intelligence*, 104(1–2):313–329, 1998.

[GS88] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35(2):165–195, 1988.

[HM86] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics and frame problem. In *Proceedings of AAAI-86*, pages 328–333. Morgan Kaufmann, 1986.

[Lif94] Vladimir Lifschitz. Circumscription. In J. A. Robinson Dov M. Gabbay, C. J. Hogger, editor, *Handbook of logic in artificial intelligence and logic programmin*, volume 3, pages 297–352. Oxford, 1994.

[Lif95] V. Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 74:351–365, 1995.

[LR94] Fangzhen Lin and Ray Reiter. State constraints revisited. *Journal of Logic and Computation*, 4:655–678, 1994.

[LS95] Fangzhen Lin and Yoav Shoham. Provably correct theories of action. *Journal of the ACM*, 42(2):293–320, March 1995.

[MC98] John McCarthy and Tom Costello. Combining narratives. In *Proceedings of Sixth Intl. Conference on Principles of Knowledge Representation and Reasoning*, pages 48–59. Morgan-Kaufman, 1998.

[McC59] John McCarthy. Programs with Common Sense[6]. In *Mechanisation of Thought Processes, Proceedings of the Symposium of the National Physics Laboratory*, pages 77–84, London, U.K., 1959. Her Majesty's Stationery Office. Reprinted in [McC90].

[McC63] John McCarthy. Situations, actions and causal laws. Technical Report Memo 2, Stanford University Artificial Intelligence Laboratory, Stanford, CA, 1963. Reprinted in [Min68].

[6]http://www-formal.stanford.edu/jmc/mcc59.html

[McC86a] John McCarthy. Applications of Circumscription to Formalizing Common Sense Knowledge[7]. *Artificial Intelligence*, 28:89–116, 1986. Reprinted in [McC90].

[McC86b] John McCarthy. Applications of Circumscription to Formalizing Common Sense Knowledge[8]. *Artificial Intelligence*, 28:89–116, 1986. Reprinted in [McC90].

[McC90] John McCarthy. *Formalizing Common Sense: Papers by John McCarthy*. Ablex Publishing Corporation, 1990.

[McC95a] John McCarthy. Situation Calculus with Concurrent Events and Narrative[9]. 1995. Web only, partly superseded by [MC98].

[McC95b] John McCarthy. Overcoming unexpected obstacles. http://www-formal.stanford.edu/jmc/glasgow.html, 1995.

[MH69] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence[10]. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. Reprinted in [McC90].

[Min68] Marvin Minsky, editor. *Semantic information processing.* MIT Press, 1968.

[Rei01a] Raymond Reiter. *Knowledge in Action.* M.I.T. Press, 2001.

[Rei01b] Raymond Reiter. *Knowledge in Action.* M.I.T. Press, 2001.

[Sha97] Murray Shanahan. *Solving the Frame Problem, a mathematical investigation of the common sense law of inertia.* M.I.T. Press, 1997.

[Sho88] Yoav Shoham. Chronological ignorance: Experiments in nonmonotonic temporal reasoning. *Artificial Intelligence*, 36(3):279–331, 1988.

[7]http://www-formal.stanford.edu/jmc/applications.html
[8]http://www-formal.stanford.edu/jmc/applications.html
[9]http://www-formal.stanford.edu/jmc/narrative.html
[10]http://www-formal.stanford.edu/jmc/mcchay69.html

[Sie97] Josefina Sierra. Blocks world heuristics manuscript, 1997.

[Sie99] J. Sierra. Declarative formalization of heuristics (taking advice in the blocks world). In *International Conference on Computational Intelligence for Modelling Control and Automation*, pages 221–228, 1999.

[Whi01] Graham White. Does the situation calculus have a semantics? 2001. manuscript for now.

/@steam.stanford.edu:/u/ftp/jmc/buzzer.tex: begun Fri Apr 6 2001, latexed January 27, 2002 at 7:49 p.m.