Beyond Minimizing Change

Tom Costello
Dept of Computer Science,
Stanford,
CA 94305
costello@cs.Stanford.EDU

Abstract

We introduce a new methodology for comparing non-monotonic treatments of change. We consider the *elaboration tolerance* of various proposals. The *elaboration tolerance* of a non-monotonic approach is defined as the elaborations, or changes, that can be made to the non-monotonic consequences, by conjoining on new information.

The standard problem, *the frame assumption*, is capturing the tendency of properties to persist over time. We show that almost all approaches allow new *effects to be added*, and *preconditions to be dropped.*

There are other ways of describing the world, and we investigate one in particular, assuming there are as few preconditions for an action as possible. This is equivalent to assuming that actions change properties as often as possible, if they ever change that property. We show that this assumption is in conflict with the usual *frame* assumption. We show that this methodology allows new *effects to be added*, and *preconditions to be added.*

We show that this *precondition assumption* is naturally opposite to the *frame assumption.* We then show that this assumption can be naturally captured in a similar way to the frame problem, using circumscription.

Introduction

Reasoning about action has been one of the favorite domains for non-monotonic reasoning. One of the problems with non-monotonic approaches to the *frame problem*, or characterizing the default that actions have as few effects as possible, has been judging the correctness, scope and appropriateness of the various solutions. This dilemma first arose in terms of extra minimal models—Lifschitz's preconditions, or the better known Yale Shooting Problem, showed that naive ideas gave weaker solutions than expected.

This was not the end however, as new solutions arose, new problems, on which the new solutions gave unintuitive results, were developed. The Stolen Car Problem, the Stanford Murder Mystery, the Ferry Boat Problem, and many other problems were posed as counter examples to proposed solutions.

This led researchers to look for ways to validate their theories. One of the first ways was proposed by Lin and Shoham (1995), where they proposed reducing a non-monotonic theory to a monotonic theory to show correctness. Lifschitz and Gelfond suggested reducing reasoning about action to well understood models like automata, and developed a language $\mathcal{A}$ well suited to describing automata as they arise in action. This work has been greatly extended, and many solutions have been shown equivalent under certain assumptions.

Kartha studied the relationship between various different proposals, and showed that under certain syntactic assumptions, that many solutions were equivalent to those solutions based on $\mathcal{A}$.

Lifschitz has considered certain properties, like stability under definitional extension, and restricted monotonicity, that he claims all approaches to the frame problem should obey. Sandewall (Sandewall 1994) proposed justifying non-monotonic treatments by reducing them to dynamical systems.

We propose another way of studying non-monotonic treatments of actions, though the methodology is not restricted to actions. We consider what elaborations, or changes, the treatment allows, in terms of conjoining information.

In this paper we shall keep to as simple a language as possible. We choose the situation calculus, which we define in a later section. Many authors have added extra predicates to the situation calculus. We avoid doing this, as we firstly are interested in what can be done without extra machinery, and secondly as we believe that many of the uses of extra predicates are unnecessary. In particular, we do not use a *causes* predicate, rather we use McCarthy and Hayes effect axioms, that do not necessitate a new predicate.

We first introduce the situation calculus, and then sketch what adding and removing effects and preconditions means. We give some examples why adding preconditions is natural. We then present two formalisms, in terms of circumscription. One, first considered in

(Costello 1997), allows adding effects and removing preconditions. The new proposal allows adding effects and adding preconditions. We show that these two proposals treat preconditions in naturally opposite ways. We show that the new proposal can capture the usual challenge problems, and new problems that allow the addition of preconditions.

The Situation Calculus

To look at the problem of reasoning about change we need a logical language to represent time, events, and the properties that hold in the world. We use the situation calculus. The paper that introduced situations (McCarthy and Hayes 1969) defined them as follows, "A situation s is the complete state of the universe at an instant of time". The basic mechanism used in the calculus to define a new situation is the *result* function,

$$s' = \text{result}(e, s).$$

In this formula s is a situation, and e is an event, and s' is a new situation that results when e occurs. A function or predicate of a situation is called a *fluent.*

Effect Axioms

Effect axioms are the usual way of representing the effects of actions in the situation calculus. If we wish to say that *moving* a clear block A to a location l, causes it to be in location l, we write,

$$\forall s\ l.\text{holds}(\text{clear}(\text{A}), s) \rightarrow \text{holds}(\text{at}(\text{A}, l), \text{result}(\text{move}(\text{A}, l), s)).$$

We note that this notation demands that we list the preconditions explicitly. If we wish to add another precondition, for instance, that the gripper arm is empty, we would need to replace the above sentence with,

$$\forall s\ l.\text{holds}(\text{clear}(\text{A}), s) \wedge \text{holds}(\text{empty}(\text{Arm}), s) \rightarrow \text{holds}(\text{at}(\text{A}, l), \text{result}(\text{move}(\text{A}, l), s)).$$

However, note that the second sentence is implied by the first. Thus, we can remove a precondition by adding the first sentence, as the two sentences together are logically equivalent to the first. We cannot add a precondition by conjoining the second sentence—adding it has no logical effect.

Another Approach

If we wish to make assumptions about adding preconditions, we need a way of expressing the effects of actions that does not assume that we explicitly list sufficient conditions.

We can state that the action A, may cause the fluent F to change from false to true, by asserting,

$$\exists s.\neg\text{holds}(F, s) \wedge \text{holds}(F, \text{result}(A, s)).$$

We note that this allows us to add more effects of action by conjoining on sentences. We can describe the effects of actions by stating that they sometimes have that effect. Consider our previous example, we would state that moving a block sometimes causes its location to change by writing,

$$\exists s.\neg\text{holds}(\text{at}(\text{A}, l), s) \wedge \text{holds}(\text{at}(\text{A}, l), \text{result}(\text{move}(\text{A}, l), s)).$$

We now have a problem, how can we now specify preconditions. We can write *frame axioms.*

$$\forall s.\neg\text{holds}(\text{clear}(A), s) \wedge \neg\text{holds}(\text{at}(\text{A}, l), s) \rightarrow \neg\text{holds}(\text{at}(\text{A}, l), \text{result}(\text{move}(\text{A}, l), s))$$

This corresponds to adding a clause to the left hand side of an effect axiom, that states that the block must be clear to be moved. In a later section we show that this style of representation can capture the usual problems in reasoning about action.

It is important to notice that if we represent preconditions in this way then the sentences describing a smaller set of preconditions do not logically imply the sentences we would write to represent a larger set of preconditions. This differs from the what occurs with effect axioms. There, we saw that writing that being clear was a *sufficient* precondition implied that being clear and the gripper arm being empty was *sufficient.* We now represent what are necessary preconditions. Thus adding the precondition that the gripper arm is empty, using the formula,

$$\forall s.\neg\text{holds}(\text{empty}(Arm), s) \wedge \neg\text{holds}(\text{at}(\text{A}, l), s) \rightarrow \neg\text{holds}(\text{at}(\text{A}, l), \text{result}(\text{move}(\text{A}, l), s))$$

now gives a logically different theory. The major change we have made is that our new notation allows us to list *necessary* conditions rather than *sufficient* conditions. This will allow us to add preconditions, in contrast to the previous method of using effect axioms that did not allow adding preconditions by conjoining on sentences.

Non-monotonic Completion

As we mentioned earlier, a certain problem arises when we formalize domains. Assume there is an axiom that states what happens to the location of a block when an action *move* is performed. For concreteness we assume that *moving* a block A, from l to l' changes its position, from l to l', if the *move* action succeeds.

If another property, *colour*, is added, the axiom which described what effect the action had on the location does not give us any information on what happens to blocks. The usual intent is that colour is unaffected.

$$\forall x\ c\ s.\text{holds}(\text{colour}(x, c), s) \rightarrow \text{holds}(\text{colour}(x, c), \text{result}(\text{move}(x, l), s))$$

To express all the fluents that do not change, an axiom would be needed for each action/fluent pair. These axioms stating that fluents do not change are called *frame axioms.* Having to specify all this lack of change seems extravagant, and more importantly it seems unnecessary to have to write these out individually. It seems

that it should be possible to generate these frame axioms automatically. The *frame problem* is the problem of characterizing these frame axioms.

We now note that in our new proposal, the problem is not characterizing frame axioms, as we list these explicitly[2], but rather adding the correct effect axioms.

Non-monotonic Logic

We cannot hope that classical logic will generate the frame axioms for us, as classical logic is monotonic and will not infer any sentence that might be denied by an additional fact. Thus we will have to use a logic without this property, a non-monotonic logic.

Non-monotonic logics are logics which can make default assumptions which can be withdrawn later. We are looking at an assumption, "the tendency of facts to endure over time" (Hanks and McDermott 1986). This seems to be exactly the sort of problem at which non-monotonic reasoning should excel. In this paper we use circumscription, a non-monotonic logic introduced by McCarthy (1986). We write $Circ(A;\Phi;\Psi)$ for the sentence of second order logic that minimizes the formulas Φ and varies the predicates Ψ. For space reasons we do not define circumscription here, (McCarthy 1986) is the definitive reference.

The Precondition Problem

Often we wish to make the assumption that there are as few preconditions to an action succeeding as possible. This has been termed the qualification problem by several researchers, (Ginsberg and Smith 1988; Lifschitz 1987). However, I consider the qualification problem to be the problem of assuming, in the absence of other information, that an action succeeds in its effects. This is to be contrasted with the assumption, which we call the *precondition problem*, that the preconditions mentioned are exhaustive of all the preconditions.

An example helps to make the difference clearer.

Example: 1
We know that to fly from Glasgow to Moscow requires an airline ticket. A solution to the precondition problem would assume that a ticket was all that was needed. A solution to the qualification problem would assume that you had a ticket.

We are going to look at the problem of assuming that there are as few preconditions as possible in the framework of the situation calculus with deterministic change. We shall use exactly the language that we introduced earlier, the situation calculus. We shall also make the assumption of deterministic change, namely, (4). As before, this states that if a change occurs after a situation, then it occurs after all situations that agree with the first situation on all fluents.

We now need a way of stating that actions have effects. We wish to say that, under some unspecified preconditions, an action a causes the fluent f to stop holding. One possible way of writing this is,

$$\exists s.\text{holds}(f,s) \wedge \neg\text{holds}(f,\text{result}(a,s)).$$

We note this is a very different way of expressing that an action has an effect than by adding an effect axiom. An effect axiom gives sufficient conditions to a change to occur. Here we merely say that the change happened once. As we are dealing with deterministic change this means that there is some condition that allows the change to happen.

We now want to state some preconditions for the action a changing the fluent f from true to false. We can write that f' holding is a precondition for a to change f, by writing:

$$\forall s.\neg\text{holds}(f',s) \rightarrow (\text{holds}(f,s) \equiv \text{holds}(f,\text{result}(a,s)))$$

This is also very different to the way we specify preconditions using effect axioms. Effect axioms specify sufficient preconditions for an action to succeed. Here we have given sufficient preconditions for the action to have no effect. Thus we are listing the complement of the set we would list with effect axioms. If the number of preconditions that makes an action fail is small, then listing these is more efficient.

Note this is recognizable as a frame axiom. Also note that every frame axiom, where the left hand side is a quantifier free formula whose only situation term is s, is equivalent to a conjunction of sentences of the following form, where g is unary predicate on fluents.

$$\forall s.(\forall f'.\text{holds}(f',s) \equiv g(f')) \rightarrow (\text{holds}(f,s) \rightarrow \text{holds}(f,\text{result}(a,s))).$$

This represents that when the fluents that hold are exactly those satisfying g, then the action a does not change the fluent f from true to false. The use of g is a technical trick to allow us to encode preconditions. Every precondition can be represented as a set of fluents, those that hold.

Our solution to the precondition problem will now be to minimize the possible changes that actions can make, and then to minimize the number of preconditions that there are.

To minimize the changes that an action can cause we minimize the formulas:

$$\begin{array}{l}\exists s.\text{holds}(f,s) \wedge \neg\text{holds}(f,\text{result}(a,s)) \\ \exists s.\neg\text{holds}(f,s) \wedge \text{holds}(f,\text{result}(a,s))\end{array} \quad (1)$$

If we choose models where these formulas are minimized, then we will prefer models where actions have fewer effects.

To minimize the preconditions we then minimize:

$$\forall s.(\forall f'.\text{holds}(f',s) \equiv g(f')) \rightarrow (\text{holds}(f,s) \equiv \text{holds}(f,\text{result}(a,s))) \quad (2)$$

[2] We actually only list those with some preconditions. If an action never changes a fluent, we assume that the unconditional frame axiom holds.

The second minimization should be done at lower priority than the first, as we do not want to introduce an effect of an action merely so that we can have fewer preconditions for that change. Thus what we do is to minimize 1, and then to minimize 2. This uses the encoding of priority as iterated circumscription introduced by Lifschitz (1985).

This is our solution to the precondition problem—minimize the changes that actions make, and then assume that these changes occur as often as possible.

We now need unique name axioms for fluents and actions and our function *sit*,

$$UNA[\overline{f}] \wedge UNA[\overline{a}] \wedge UNA[sit]. \quad (3)$$

We have introduced a function *sit* here, which maps sets of fluents to situations. This exists to guarantee that the domain of situation will be large enough. Circumscription does not compare structures that have different domains, thus in the absence of an assertion that there are more than one situation, our solution will have models with only one situation.

For simplicity we look at the case where we have deterministic change, that is, what is true at the next situation is fully described given the fluents that holds at the current situation, and the action that occurs.

$$\begin{aligned}&\forall afss'.(\forall f'.\text{holds}(f', s') \equiv \text{holds}(f', s)) \rightarrow \\ &\text{holds}(f, \text{result}(a, s)) \equiv \text{holds}(f, \text{result}(a, s'))\end{aligned} \quad (4)$$

Definition: 1
Given a theory A in the language of the situation calculus, the consequences of A under the precondition default, written $A \models_p \phi$, are the consequences of the following circumscription.

$$Circ(Circ(A \wedge 4 \wedge 3; 1; \text{holds}); 2; \text{holds})$$

The Frame Problem

We now consider a solution to the frame problem introduced by the author. We define ab_1 and ab_2 as syntactic sugar, so that conjunctions of ab's correspond to instances of change formulas, that is effect axioms, with the extra proviso that the fluent affected is changed. More correctly, every change formula is equivalent to a conjunction of ab's.

$$\begin{aligned}&ab_1(a, f, g) \stackrel{def}{\equiv} \\ &\quad \forall s.(\forall f'.g(f') \equiv \text{holds}(f', s)) \wedge \text{holds}(f, s) \rightarrow \\ &\quad\quad \neg\text{holds}(f, \text{result}(a, s)), \\ &ab_2(a, f, g) \stackrel{def}{\equiv} \\ &\quad \forall s.(\forall f'.g(f') \equiv \text{holds}(f', s)) \wedge \neg\text{holds}(f, s) \rightarrow \\ &\quad\quad \text{holds}(f, \text{result}(a, s)).\end{aligned} \quad (5)$$

Now that we have written our change formulas as instantiations of a pair of second order logic sentences we can minimize all change formulas by minimizing these sentences.

Definition: 2
The consequences of a theory A in the language of the situation calculus, under the frame assumption, written $\models_{frame}$, are the consequences of,

$$Circ(\ A \wedge 4 \wedge 3; ab_1, ab_2\ ).$$

On simple examples this agrees with the proposals of Baker (1991), Lifschitz and Kartha,(1995) Lin and Reiter(1994) , Haugh(1987) (Sandewall 1994) etc. It differs in example that include disjunctions of effect axioms with domain constraints, or observations. These issues are explored in detail in a paper by the author currently under review.

We now have two ways of describing action. We first look at simple examples to see how we use the first method. We then analyze the elaboration tolerance of the two proposals.

Examples

The standard examples against which any solution to the frame problem must be tested are the Yale Shooting Problem, (YSP) and the Stanford Murder Mystery, an explanation problem, essentially the YSP where we reason backwards in time.

The Language of the Situation Calculus

The situation calculus of order n, L^n_{sit}, is a language of n^{th} order logic with equality, with three disjoint sorts: fluent names, actions, and situations[3]. It contains exactly one predicate constant, holds, in addition to equality, one function constant result(a, s) which returns a situation, a function sit(g) from predicates of fluents to situations, and one situation constant $S0$, plus a set of fluent constants $\overline{f}$, and a set of action constants $\overline{a}$. When we wish to specify the action and fluent constants we write $L^n_{sit}(\overline{f}, \overline{a})$.

The Yale Shooting Problem

The Yale Shooting Problem (YSP) is the problem of correctly predicting what happens in the following scenario. An individual named Fred is known to be alive, and a gun is known to be loaded initially. We know that shooting causes an individual to not be alive if the gun is loaded. We then want to infer that Fred will be dead after we wait, then shoot. In some solutions to the frame problem there are multiple minimal models, some where the gun becomes unloaded during the wait action, some where the shoot action kills Fred.

We axiomatize the Yale shooting problem as follows:

$$\begin{aligned}&\exists s.\text{holds}(\text{alive}, s) \wedge \neg\text{holds}(\text{alive}, \text{result}(\text{shoot}, s)) \\ &\forall s.\neg\text{holds}(\text{loaded}, s) \rightarrow (\text{holds}(\text{alive}, s) \rightarrow \\ &\quad \text{holds}(\text{alive}, \text{result}(\text{shoot}, s)))\end{aligned} \quad (6)$$

$$\text{holds}(\text{alive}, S0) \wedge \text{holds}(\text{loaded}, S0) \quad (7)$$

[3]We use s with primes and indices to range over situations, and a and f to range over actions and fluent names, and g to range over unary predicates of fluents.

The sentence that we should infer is:

$$\neg\text{holds}(\text{alive}, \text{result}(\text{shoot}, \text{result}(\text{wait}, S0))))$$

Proposition: 1
We note that the if the theory A is the conjunction of 6, 7, over the language $L^1_{sit}(\{\text{loaded}, \text{alive}\}, \{\text{wait}, \text{shoot}\})$ then

$$A \models_p \neg\text{holds}(\text{alive}, \text{result}(\text{shoot}, \text{result}(\text{wait}, S0))))$$

Proof: When we carry out the first minimization, as per 1, we find that,

$$\begin{aligned}&\exists s.\text{holds}(f, s) \wedge \neg\text{holds}(f, \text{result}(a, s)) \equiv \\ &\quad f = \text{alive} \wedge a = \text{shoot} \\ &\neg\exists s.\neg\text{holds}(f, s) \wedge \text{holds}(f, \text{result}(a, s))\end{aligned}$$

This yields,

$$\begin{aligned}&\forall s.f \neq \text{alive} \vee a \neq \text{shoot} \rightarrow \\ &\quad \text{holds}(f, s) \rightarrow \text{holds}(f, \text{result}(a, s)) \\ &\forall s.\neg\text{holds}(f, s) \rightarrow \neg\text{holds}(f, \text{result}(a, s)).\end{aligned}$$

We now minimize the preconditions for an action succeeding, as per 2. This gives,

$$\begin{aligned}&(\forall s.(\forall f'.\text{holds}(f', s) \equiv g(f')) \rightarrow \\ &\quad (\text{holds}(f, s) \equiv \text{holds}(f, \text{result}(a, s)))) \equiv \\ &(\neg g(\text{alive}) \wedge f = \text{loaded} \wedge a = \text{shoot}) \vee \\ &a \neq \text{shoot} \vee f \neq \text{loaded}\end{aligned}$$

This is clearly minimal, if it is satisfiable, as it is entailcd by 6. It is consistent, as the intended model satisfies it.

From the above sentence we can derive that $S0$ agrees with result(wait, $S0$) for all fluents. From the initial condition that the gun is loaded and that Fred is alive we get that holds(loaded, result(wait, $S0$)) and holds(alive, result(wait, $S0$)). From this and the above sentence, instantiating g with $\lambda f.\text{holds}(f, S0)$, we get that

$$\begin{aligned}&\exists s.(\forall f'\text{holds}(f', s) \equiv \text{holds}(f', S0)) \wedge \\ &\text{holds}(\text{alive}, s) \wedge \neg\text{holds}(\text{alive}, \text{result}(\text{shoot}, s))\end{aligned}$$

Finally, using the axiom of deterministic change, 4, we can derive that all situations that agree with the existentially quantified sentence above have the required change. This includes result(wait, $S0$). Thus the change from alive to dead occurs when we shoot in this situation, as required.∎

Explanation

We now consider reasoning backwards in time. If we know that after waiting and shooting that Fred was dead, and that he was alive initially, we should derive that the gun must have been loaded initially.

$$\begin{aligned}&\text{holds}(\text{alive}, S0) \wedge \\ &\neg\text{holds}(\text{alive}, \text{result}(\text{shoot}, \text{result}(\text{wait}, S0))))\end{aligned} \quad (8)$$

Proposition: 2
We note that the if the theory A is the conjunction of 6, 8, over the language L^1_{sit} ({loaded, alive}, {wait, shoot}) then

$$A \models_p \text{holds}(\text{loaded}, S0)$$

Proof: The proof is similar to the previous proposition. We get the same results for our minimization. We get there is a change from alive to dead exactly in situations where the gun is loaded and Fred is alive. We can also derive that there is no change when the gun is not loaded. This suffices to show that the gun was loaded initially. ∎

An extra precondition

We now give an example of adding a precondition, something impossible in axiomatizations using effect axioms. We add the precondition that to kill someone by shooting, they must not have a bullet proof vest. We write this by saying that there is no change if they are wearing a vest,

$$\begin{aligned}&\forall s.\text{holds}(\text{vest}, s) \wedge \text{holds}(\text{alive}, s) \rightarrow \\ &\text{holds}(\text{alive}, \text{result}(\text{shoot}, s))\end{aligned} \quad (9)$$

Proposition: 3
We note that the if the theory A is the conjunction of 6, 7, 9, over the language $L^1_{sit}(\{\text{loaded}, \text{alive}, \text{vest}\}, \{\text{wait}, \text{shoot}\})$ then

$$A \models_p \begin{aligned}&\neg\text{holds}(\text{alive}, \text{result}(\text{shoot}, \text{result}(\text{wait}, S0)))) \equiv \\ &\neg\text{holds}(\text{vest}, S0)\end{aligned}$$

Proof: This is again similar to the proof of the first proposition. The first minimization is the same. The second adds the vest clause in the second line. We instantiate g with what held at the first situation as before. When we simplify we have a holds(vest, s) clause. We carry out the same reasoning using determinism to derive that there is a change, modulo the vest clause. ∎

We note that in previous axiomatizations using effect axioms, adding

$$\begin{aligned}&\forall s.\neg\text{holds}(\text{vest}, s) \wedge \text{holds}(\text{loaded}, s) \rightarrow \\ &\quad \neg\text{holds}(\text{alive}, \text{result}(\text{shoot}, s))\end{aligned}$$

would have no effect, as this is a consequence of the axiom,

$$\forall s.\text{holds}(\text{loaded}, s) \rightarrow \neg\text{holds}(\text{alive}, \text{result}(\text{shoot}, s))$$

Elaboration Tolerance

In this section we now define elaboration tolerance for theories of action. We show that the usual frame assumption only allows adding effects and removing preconditions, while the precondition assumption we introduce allows adding effects and adding preconditions.

We can now define adding effects, and adding and removing preconditions.

Definition: 3
A model $\mathfrak{A}$ of the language $L^1_{sit}(\overline{f}, \overline{a})$ has the possible effect of the action $A \in \overline{a}$ sometimes changing $F \in \overline{f}$ if

$$\mathfrak{A} \models \exists s.\neg(\text{holds}(F, s) \equiv \text{holds}(F, \text{result}(A, s))).$$

A model $\mathfrak{A}$ has more effects that a model $\mathfrak{B}$, if the set of possible effects of $\mathfrak{A}$ is a superset of the possible effects of $\mathfrak{B}$.

This states that an action has a possible effect if it ever changes the value of that fluent.

Definition: 4
Let P, N be two disjoint subsets of $\overline{f}$. A model $\mathfrak{A}$ of the language $L^1_{sit}(\overline{f}, \overline{a})$ has the preconditions P, N for the action $A \in \overline{a}$ changing $F \in \overline{f}$ if

$$\mathfrak{A} \models \begin{array}{l} \forall s.\neg \bigwedge_{f \in P} \text{holds}(f, s) \vee \neg \bigwedge_{f \in N} \neg\text{holds}(f, s) \rightarrow \\ (\text{holds}(F, s) \equiv \text{holds}(F, \text{result}(A, s))). \end{array}$$

A model $\mathfrak{A}$ has more preconditions that a model $\mathfrak{B}$, if the set of preconditions of $\mathfrak{A}$ is a superset of the precondition of $\mathfrak{B}$.

This states that a precondition, defined by a set of fluents that must hold, and a set of fluents whose negation must hold, is a precondition for an action having an effect, if whenever the precondition if false, then the action does not have that effect.

Theorem: 4
Let T be a theory in L_{sit}, and α a sentence in L_{sit}. Let the non-monotonic consequences of T under $\models_{frame}$ be T_1, let the non-monotonic consequences of $T \wedge \alpha$ under $\models_{frame}$ be T_2. Then, every model of T_2 has as many effects as some model of T_1, and every model of T_1 has as many preconditions as some model of T_2.

Let the non-monotonic consequences of T under $\models_p$ be T_3, let the non-monotonic consequences of $T \wedge \alpha$ under $\models_p$ be T_4. The every model of T_4 has as many effects as some model of T_3, and every model of T_4 has as many prcconditions as some model of T_3.

The elaborations that are possible under $\models_{frame}$, are adding effects, and removing preconditions. In contrast we can add both preconditions and effects using the *precondition assumption.*

Conclusion

Adding preconditions is natural, at least as natural as removing preconditions. We have shown that it is possible to axiomatize change so that both effects and preconditions can be added. We have shown that usual axiomatization allow adding effects and removing preconditions.

We suggest that this method of considering what elaborations can be made to a theory is a useful tool for investigating non-monotonic phenomena. Considering the elaboration tolerance of a non-monotonic solution gives a clear statement of the applicability of the method.

Acknowledgments

This research is supported in part by ARPA/Rome Laboratory planning initiative under grant (ONR) N00014-94-1-0775. I am grateful to John McCarthy and Anna Patterson who gave useful criticism on earlier drafts.

References

Baker, A. B. 1991. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence* 49:5–23.

Costello, T. 1997. *Non-monotonicity and Change.* PhD thesis, Stanford University.

Ginsberg, M. L., and D. E. Smith. 1988. Reasoning about action II: The qualification problem. *Artificial Intelligence* 35(3):311–342.

Hanks, S., and D. McDermott. 1986. Default reasoning, nonmonotonic logics and the frame problem. In *Proc. National Conference on Artificial Intelligence (AAAI '86)*, 328–333, Philadelphia.

Haugh, B. A. 1987. Simple causal minimization for temporal persistence and projection. In *Proc. National Conference on Artificial Intelligence (AAAI '87).*

Kartha, G. N., and V. Lifschitz. 1995. A simple formalization of actions using circumscription. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence (IJCAI '95).*

Lifschitz, V. 1985. Computing Circumscription. In *Proc. Ninth International Joint Conference on Artificial Intelligence (IJCAI '85)*, 121–127.

Lifschitz, V. 1987. Formal Theories of Action. In F. Brown (Ed.), *The Frame Problem in Artificial Intelligence*, 121–127. Morgan Kaufmann.

Lin, F., and R. Reiter. 1994. State Constraints Revisited. *Journal of Logic and Computation* 4(5):655–678.

Lin, F., and Y. Shoham. 1995. Provably Correct Theories of Action. *Journal of the ACM* 42(2):293–320.

McCarthy, J. 1986. Applications of Circumscription to Formalizing Commonsense Knowledge. *Artificial Intelligence* 28:89–116.

McCarthy, J., and P. Hayes. 1969. Some Philosophical Problems From the Standpoint of Artificial Intelligence. In D. Michie (Ed.), *Machine Intelligence 4*, 463–502. Edinburgh, UK: Edinburgh University Press.

Sandewall, E. 1994. *Features and Fluents.* Oxford University Press.