Machinery for Elaborating Action $0.2\beta 1$

Eyal Amir
Department of Computer Science,
Gates Building, 2A wing
Stanford University, Stanford, CA 94305-9020, USA
eyala@cs.stanford.edu

Abstract

We present a method for elaborating action without the need to amend the knowledge already present in the database. It does that by reifying the elaboration, and by introducing a new action for every elaboration that we add. This method is built on top of an existing action formalism that applies situation calculus and circumscription. Finally we argue that the method is applicable to a variety of action formalisms.

Introduction

Our long term program deals with the *Elaboration Tolerance* of formalisms of knowledge. *Elaboration Tolerance* is a term coined by McCarthy in (McCarthy 1993) and used ever since in describing one of the more fundamental problems that AI and Computer Science research faces.

A knowledge system is *Elaboration Tolerant* if in order to change some data in the knowledge base, there is no need to know *too many things* about the knowledge already in the knowledge base, and especially, there is no need to manually select and change/retract some formulas (i.e., any retraction is embedded in the semantics of the knowledge base). We are interested in formalisms that allow elaboration without the need for manual amendments/retraction of axioms/facts.

In this paper we focus on one proposed element in elaborating actions. The idea is based on using the reified elaboration in creating a new action out of the original one. Adding *Elaboration Axioms* then proves useful in reasoning about that new action. The set of actions is enlarged by an order of magnitude, and we apply some policies in selecting the right level of an elaborated action, while invalidating the rest. These issues are made precise in the following sections.

Most of the work was motivated by a developed formalization of the *Missionaries and Cannibals Problem* presented in an unpublished manuscript available on our web page, which we do not present here for lack of space.

Our approach can be seen in the framework of *Belief Change* (e.g., (Alchourrón, Gärdenfors, & Makinson 1985)), but the theory of Belief Change deals with different granularities, and different goals. The presentation of Belief Change focuses on forming the *right* rules for belief change, under some assumptions. In our work we use the full power of the selected action formalism (in our chosen formalism, this is the underlying situation calculus, and circumscription), and we deal with extending the theory, given that the inference operator is predetermined.

Another somewhat related topic is Reiter's theory of primitive actions (cf (Reiter 1991)). Roughly speaking, an action is primitive if it is not defined in terms of other actions. Our approach has something of that flavor of relations between actions, but in a different manner. We deal with *basic* actions as those that have no further generalization (i.e., they are not elaborations to any other action), while Reiter refers to primitive actions as actions that cannot be divided into subactions.

Our results are presented in the scope of the action formalism of (Kartha & Lifschitz 1995) but the reader may understand the main ideas of the paper without knowing the exact technicalities of that formalism. The results presented here are not bound to that formalism, and they are casted in a specific action theory for coherency and completeness only. For background material on situation calculus and theories of Action, the reader is refered to (Lifschitz 1993), (McCarthy 1990) and (Sandewall & Shoham 1994).

First - pick an Action formalism

In this section we display the action theory of (Kartha & Lifschitz 1995) in some length (alas, action theories tend to be long). The causal language is essentially the same as that described in (Lin & Reiter 1994). Unless otherwise mentioned, we use lower-case english letters to designate variables, and english Capital letters to designate constants.

Consider a first order language with object variables for situations and actions, and possibly variables of other sorts. Here s denotes a situation variable, a an action variable, and x, y tuples of distinct variables of other sorts. The nonlogical constants of the language are:

- $S0$ is the initial situation.
- $Result(a,s)$ is the resulting situation from applying a to s.
- $Poss(a,s)$ expresses that a is possible to execute in s.
- function constants (possibly with arity 0), called *action symbols*. For an action symbol A, $A(x)$ is an action term.
- predicate constants called *fluent symbols*. For a fluent symbol F, $F(x,s)$ is an atomic formula.

A language of this kind will be called a *causal language*.

Two groups of axioms are allowed in a causal theory - *effect axioms* and *ramification constraints*. A formula Φ is a *simple state formula*, if every occurence of a situation term in Φ is an occurence of the same variable s as the last argument of a fluent symbol. Clearly, $S0$,$Result$ and $Poss$ cannot occur in a simple state formula.

A set of axioms in a causal language is a *causal theory* if it consists of some *ramification constraints* that are assumed to be simple state formulas that do not contain action terms, and some *effect axioms* of the form

$$Poss(a,s) \implies [\Psi_F^+(x,a,s) \Rightarrow F(x, Result(a,s))]$$
$$Poss(a,s) \implies [\Psi_F^-(x,a,s) \Rightarrow \neg F(x, Result(a,s))]$$

where $\Psi^+(x,a,s), \Psi^-(x,a,s)$ are simple state formulas.

The "commonsense law of inertia" is expressed by a formula for each fluent F in

$$\neg Ab(Aspect(F,x),a,s) \wedge Poss(a,s) \implies (F_R(x,a,s) \equiv F(x,s)) \quad (1)$$

These are called *Inertia formulas*.

The theory of Nested Abnormality Theories (NAT, (Lifschitz 1995)) reformulates $Circ[\Phi_1 \wedge ... \wedge \Phi_n; Ab; C_1, ..., C_m]$, where Ab shows in some of the formulas Φ_i, in "blocks" of the form

$$\{C_1, ..., C_m \ : \ \Phi_1, ..., \Phi_n\}.$$

Besides using NAT, for each fluent F, add the following formula to the theory:

$$F_R(x,a,s) \equiv F(x, Result(a,s))$$

They then use the circumscription embedded in NAT to provide their solution to the Frame Problem. The additional formula provides for the same idea applied by (Lin & Reiter 1994) and (Baker 1991): Fix the original situation, and vary only the resulting one. The resulting theory is of the form

$$\begin{array}{ll} UNA, & \\ \mathbf{C}, & \\ F_R(x,a,s) \equiv F(x, Result(a,s)), & (F \in \mathbf{F}) \\ \{\mathbf{F}_R \ : \ \mathbf{I}, \ \mathbf{E}_R, \ \mathbf{C}_R\} & \end{array}$$

where $UNA, \mathbf{C}, \mathbf{F}, \mathbf{I}, \mathbf{F}_R, \mathbf{E}_R, \mathbf{C}_R$ are the Unique Names Axioms, Ramification constraints, Fluents set, Inertia formulas, Fluents in the result situation, Effect axioms, and Ramification constraints for the result situation, respectively.

What are elaborations?

The *Missionaries and Cannibals Problem* framework deals with transfering people accross a river when given some constraints. In that framework, there is only one *basic* action schema, which is crossing the river. We write it here $go(group, bank)$, and use this framework for drawing examples.

Elaboration of actions can be viewed as adding or weakening preconditions and effects (although, look at the discussion). We divide the (*Direct*) elaborations into four categories:

1. Adding (strengthening) preconditions. Example: oars are needed for the operation of the boat.
2. Adding effects/postconditions. Example: Each cross of the river makes the cannibals more hungry.
3. Weakening/removing preconditions. Example: Actually the boat does not need oars, because it is a motorboat.
4. Weakening/removing effects/postconditions. Example: The action might not succeed (undeterministic effects).

Some formalizations of action (e.g., (Lin & Reiter 1994),(Kartha & Lifschitz 1995)) support adding effects and preconditions in an easy manner (after you formulated the elaboration in terms of appended preconditions and effects, of course). On the other hand, there is no way to retract or change previously asserted effects or preconditions, without actually pulling the right part of the formula out.

Elaboration of actions

Let a be an action. The action $using(boat, a)$ is an elaboration of a that does not seem to fit easily into our previous description of elaborating action (in terms of preconditions and postconditions). On the other hand, it seems to give a hint at the true nature of elaboration tolerance. In a sense it is a new action, and thus we can say anything we want about it, preserving what was said previously about the original action a.

Reified elaborations

We take the approach of reifying the elaborations, i.e., that every elaboration has a name. We then use a function called *elaborate* to map actions and elaboration names to elaborated actions. The properties of the new action are then derived from the original action and some other assertions, using generic or special-purpose axioms. In this framework and henceforth we use set theoretic notation, but refer to the corresponding high-order logic syntax.

Definition 0.1 *The set of elaboration names of an action a (these are the only elaborations possible for it) is denoted by $Elaborations(a)$. $elaborate(a,e)$ is a mapping from an action a and an elaboration $e \in Elaborations(a)$ to the elaborated action $elaborate(a,e)$.*

We can equivalently define $elaborate(a, e)$ to be a partial function from actions and elaborations to actions, setting $Elaborations(a) = \{e \mid elaborate(a, e) \text{ is defined}\}$.

EXAMPLE The object *using-boat* is an elaboration of $go(group, bank)$. The elaborated action is then $elaborate(go(group, bank), using\text{-}boat)$.

Note that the elaborations of an action do not necessarily relate to the set of combined elaborations for that action. The set of elaborations *does not* represent the set of all possible successive elaborations to the given action (one should keep in mind that after one elaboration is applied to an action, we have a new, *different*, action).

EXAMPLE The following is a possible state of affairs:

$$Elaborations(go(Group, Lb)) = \{using(Boat)\}$$
$$Elaborations(elaborate(go(Group, Lb), using(Boat))) = \{using(One\text{-}oar)\}$$

Since elaborations are objects under our consideration, we can form terms whose values are elaborations, and terms with free variables. The latter denote the set of elaborations that are ground instances of these terms. Note that in the case of elaborating an action schema with an elaboration schema, some of the variables can be identical, leading to the natural combination.

EXAMPLE Let *boat* be a variable, and *Boat* be an object name. $using(boat)$ is an elaboration that among other things, stands for $using(Boat)$. The elaborated action $elaborate(go(group, bank), canibals(group))$ is a possible way of expressing an elaboration of an action restricted to groups of cannibals.

A good question is, where do we get this set of elaborations from? Let A have a new elaboration E. In that case, the following axiom is added

$$E \in Elaborations(A)$$

and we then assume that the only elaboration available for an action, are those that are mentioned, e.g., by applying the circumscription policy

$$\textbf{Circ } \lambda ae \; elaborate(a, e) \tag{2}$$

The benefits from reifying elaborations will be detailed in the following sections. Notice that we sometimes write $using(boat, go(group, bank))$ for $elaborate(go(group, bank), using(boat))$, but remember that we refer to the latter.

Elaboration Axioms

The following elaboration schema for an action a into an action $a' = elaborate(a, e)$ seems to give us some of the dynamics we need:

$$property(a') \equiv ((property(a) \wedge \varphi) \vee \psi) \tag{3}$$

where φ and ψ are formulas describing the change between the relative propeties of a and a'.

EXAMPLE Let $a' = elaborate(a, using(boat))$, and $\varphi = precond(using(boat))$, $\psi = False$, and $Poss(a', s) \equiv (Poss(a, s) \wedge \varphi) \vee \psi$.

It serves us in three ways:

1. Elaborating action in the manner described above (i.e., adding/weakening pre/postconditions), is done more simply, and explicitly.

2. Specifically distinguishing the first action a from its elaboration, by saying that it is an elaboration. This way we get that an action is a series of elaborations, each is adding its own changes φ, ψ.

3. We can reason about the elaboration process itself, and the possible elaborations of an action (e.g., the possible ways of transfering the group to the other bank of the river).

For each possible elaboration, we form a set of axioms that defines the new elaborated action, possibly in terms of the original action, or some other features of the system. It also lets us create *Common Sense Axioms* about the elaboration schemas (the elaborations with free variables). We describe this issue briefly, and leave most of it for later work.

The following is a list of examples of possible elaborations.

1. $using(tool)$ - such as using the boat to cross the river.

2. $when(time)$ - such as specifying the exact time for an action to commence.

3. $at(location)$ - specifying location for the action.

4. $with\text{-}stages(stages)$ - breaking the action into stages (changing granularity).

5. $while(action2)$ - concurrency statement.

Of course, some of the *Common Sense Axioms* here are difficult to get at (take for instance the concurrency of actions problem). It does, though, give us a very expressive language for granularity control, and as we hope to show in the future, a fine mean for expressing relations between actions.

Let us develop some of the common sense axioms for the elaboration of using a tool (this is the first use of the causal language mentioned above).

$$\begin{aligned}
&Poss(elaborate(action, using\text{-}tool(tool)), s) \implies \\
&\qquad accessible(tool, Actor(action), s) \\
&location(tool, s) = location(actor, s) \implies \\
&\qquad accessible(tool, actor, s) \\
&Poss(elaborate(a, using\text{-}tool(tool)), s) \implies \\
&\qquad usable(tool, s) \\
&\neg abq_u(tool, s) \implies usable(tool, s)
\end{aligned} \tag{4}$$

Circ *accessible*
Circ abq_u **var** *usable*

Selecting the right action

The above schema lets us define new actions, but we do not dispose of the old ones, and we probably should not anyway. The problem now is what action should the formalism prefer among the many possible elaborations, when reasoning about an action. At the end of this work we will argue that this process highly depends on the goal of the reasoning process, but for now we take the approach that the most elaborate action is the one we want to use, and we leave the arguments for and against it to the discussion.

Preliminary observations

We can use the fact that the actions are ordered using the *elaborate* function, to take the most elaborate as our chosen one.

Definition 0.2 *We define the relation* $<_e$.

$$\begin{array}{l} \forall e \in Elaborations(a)\ (elaborate(a,e) <_e a) \\ \exists a_3\ (a_1 <_e a_3 \wedge a_3 <_e a_2) \implies a_1 <_e a_2 \end{array} \tag{5}$$

Also, $a_1 \leq_e a_2 \iff a_1 = a_2 \vee a_1 <_e a_2$. ■

Notice that in order to complete $<_e$ we have to use the second order circumscription formula to minimize $<_e$ with

$$Circ[\Sigma; <_e] \tag{6}$$

where Σ is the conjunction of (5), which forces $<_e$ be the transitive closure of $Elaborations$. Otherwise, we will not be able to prove that the *chosen* action (below) is the most elaborate one.

Example Let $a' = elaborate(a, using(Boat))$.

$$\begin{array}{l} a' <_e a \\ elaborate(a', using(one\text{-}oar)) <_e a \end{array}$$

Figure 1 shows an example situation more clearly.

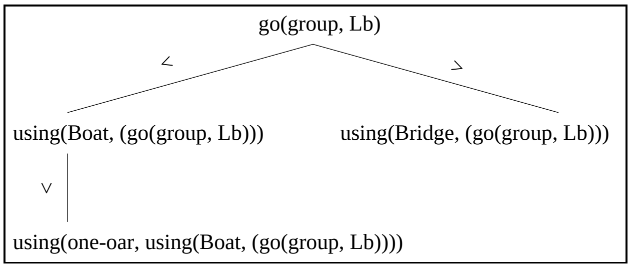

Figure 1: different levels of elaborations of an action

Some observations are straightforward from this definition. First, in the case that only a single minimal (in the sense of $<_e$) action exists, picking that one might do. Also, if we have several minimal actions, they might all be available (e.g., $elaborate(a, using(one\text{-}oar))$ and $elaborate(a, using(two\text{-}oars))$). As mentioned above, the situation is not that simple (although for now we will assume so), and we discuss that in more length later.

The most elaborate action

Let us give the choice of a minimum action, with regard to $<_e$, a formal treatment.

Definition 0.3 *Let* $a \in Actions$. a *is* Basic, *if it has no generalization (according to* $<_e$*).*

$$Basic(a) \stackrel{\text{def}}{=} a \in Actions \wedge \\ (\neg \exists a' \in Actions\ (a <_e a'))$$

a *is* Minimal, *if it has no elaboration.*

$$Minimal(a) \stackrel{\text{def}}{=} a \in Actions \wedge \\ (\forall a' \in Actions) \neg (a' <_e a)$$

■

Example Figure 1 is an example of a single *Basic* action $go(group, Lb)$, and two *minimal* actions $using(one\text{-}oar, using(Boat, go(group, Lb)))$ and $using(Bridge, go(group, Lb))$.

Finally, given an arbitrary action, we would like to say that there is a set of *chosen* elaborations of it. This set defines the applicable actions for a given situation. In this setting, we choose the set of *minimal* actions to be that *chosen* set. Notice, though, that the rest of the treatment regarding *choice* does not depend on the specific selection for its definition (we keep *chosen* dependent on s for generality in future treatment).

$$chosen(a, s) \iff Minimal(a) \tag{7}$$

Embedding *chosen* in the selected action framework

Collectors

In causal theories, $Poss(a, s)$ serves for collecting the preconditions for the application of a to s. (Lin & Reiter 1994) uses sentences of the form

$$Poss(a, s) \implies \pi(a, s)$$

which are then put together into

$$\pi_1(a, s) \wedge ... \wedge \pi_n(a, s) \iff Poss(a, s)$$

This way $Poss$ serves as a *collector* for the preconditions.[1] Notice that the way collectors work depends directly on the language exercised. If our language used sentences of the form $\pi(a, s) \implies Poss(a, s)$, then the completion of $Poss$ would lead to $Poss(a, s) \iff \pi_1(a, s) \vee ... \vee \pi_n(a, s)$, and $\pi(a, s)$ is a *sufficient* (instead of necessary) condition for $Poss(a, s)$. The same goes to all other such collectors.

For our framework, we would like to introduce other collectors to the language. For a fluent F and $\Psi_F^+(x, a, s)$ as mentioned above, we will have

$$Precond^+(F, x, a, s) \implies \Psi_F^+(x, a, s)$$

and the same for $\Psi_F^-(x, a, s)$ with $Precond^-(F, x, a, s)$. $Precond^+, Precond^-$ will be used in elaborating the effects of an action.

[1] We do not define the term *collector* exactly here, but rather turn to it intuitively. The absence of its exact definition has no influence on our results here.

Adding Elaboration Axioms

In a developed system, there will be many variants and elaborations of the same *Basic* action. In order to make sure that only *chosen* actions are actually applicable, we need to apply some changes to the action framework dealt with. In the framework of action detailed above, the following general axioms describe the relations among actions and their elaborations (α comes as a shorthand for F, x, a, s)

$$\begin{aligned}
&Poss(elaborate(a,e),s) \implies \\
&\qquad ((Poss(a,s) \wedge \varphi^{Poss}_{a,e}(a,s)) \vee \psi^{Poss}_{a,e}(a,s)) \\
&Precond^{+}(F,x,elaborate(a,e),s) \implies \\
&\qquad (Precond^{+}(\alpha) \wedge \varphi^{Pre+}_{F,a,s}(\alpha) \vee \psi^{Pre+}_{F,a,e}(\alpha)) \\
&Precond^{-}(F,x,elaborate(a,e),s) \implies \\
&\qquad (Precond^{-}(\alpha) \wedge \varphi^{Pre-}_{F,a,s}(\alpha) \vee \psi^{Pre-}_{F,a,e}(\alpha))
\end{aligned}$$

$$Applicable(a,s) \iff chosen(a,s) \wedge Poss(a,s) \quad (8)$$

$$\begin{aligned}
&Applicable(a,s) \wedge Precond^{+}(F,x,a,s) \implies \\
&\qquad F(x, result(a,s)) \\
&Applicable(a,s) \wedge Precond^{-}(F,x,a,s) \implies \\
&\qquad \neg F(x, result(a,s))
\end{aligned}$$

The motivation behind these axioms is three-fold:

- We want to let only *chosen* actions have any possible effect, this way neutralizing the others.
- We want to use *inheritance* (the simple case), or (3) (the more complicated case) in infering from an action's effects and preconditions, on its elaboration's effects and preconditions. For that we use the φ's and ψ's as *inhibitors* and *conceders* respectively.
- We want to keep the solution to the Frame Problem intact.

Notice that we changed the language to include *Applicable*, and that only *Applicable* actions have results.[2] Now, effect axioms take the form

$$\begin{aligned}
Precond^{+}(F,x,a,s) &\implies \Psi^{+}_{F}(x,a,s) \\
Precond^{-}(F,x,a,s) &\implies \Psi^{-}_{F}(x,a,s)
\end{aligned}$$

and we replace the effect axioms of the form $Poss(a,s) \wedge \Psi^{+}_{F}(x,a,s) \implies F(x, Result(a,s))$ with

$$\begin{aligned}
&Applicable(a,s) \wedge Precond^{+}(F,x,a,s) \implies \\
&\qquad F(x, Result(a,s))
\end{aligned}$$

Notice that since in this action formalism the effect axioms are centered around the fluents (and not around actions), $Preconds(a,s)$ serve as the postconditions *collector*.

[2] In this schema, we are submitted to the inherent properties of the underlying action formalism. In this framework, applying an action that is not applicable in a situation, leads to a *ghost-situation* that is equivalent (in terms of fluent values) to the original situation (much because of the frame axiom (1)).

The NAT theory has several changes, all external to the solution for the Frame Problem. Besides adding the Elaboration Axioms and framework, we add the two circumscription policies (6), (4) and (2) (in their NAT equivalent). The additional nonmonotonic change is focused at the selection of *chosen* actions, as we add new axioms to the system.

Examples

In the beginning we said that we have several types of *Direct* elaborations. Let us see how we incorporate them in our new framework. For each change we give an example and added sentences. Let $A(group, bank) = using(Boat, go(group, bank))$.

Strengthening effects We could do that even with the original formalism. Example: crossing the river makes the group happy.

$$\begin{aligned}
&Applicable(A(group,bank),s) \implies \\
&\qquad Happy(group, Result(a,s))
\end{aligned}$$

Strengthening preconditions We could do that even with the original formalism. Example: The boat needs oars for its operation.

$$\begin{aligned}
&using(Oars) \in Elaborations(A(group,bank)) \\
&Poss(using(Oars, A(group,bank)),s) \implies \\
&\qquad Poss(A(group,bank),s) \wedge in(Oars, Boat)
\end{aligned}$$

Weakening preconditions Example: The boat is actually a motorboat, and does not require oars.

$$is\text{-}motorboat \in Elaborations(A(group,bank))$$

Weakening/removing effects/postconditions Example: The boat has a leak and therefore using it does not lead to the expected consequence (of the passengers being on the other bank of the river). Let $A(group, bank) = using(Boat, go(group, bank))$.

$$broken(Boat) \in Elaborations(A(group,bank))$$

That will do, because the new action does not have the previous effect (of transfering the group to the other bank).

Discussion

For lack of space we only point at directions of interest, and do not try to solve any of them here.

First, we turn to the last paragraph. Fangzhen Lin noticed that the last example hints on a need for a *default elaboration axiom*, something in the form of (3). With such an axiom we can elaborate an action without changing any of its characteristics (we can do it right now as well, but we have to say that *elaboration axiom* explicitly). This is a new form of knowledge that we can now embody in our formalism, due to the reification of elaborations.

Lin also noticed that our Elaboration Axiom (3) is somewhat oriented towards precondition elaboration than effects. It is our view that this latter point is much dependent on the exercised action formalism.

The choice of chosen actions is dependent on the goal of our reasoning. Sometimes we don't even have to "choose" a specific action/action level, but rather change granularities dynamically (e.g., hierarchical planning).

Our motivation for selecting the most elaborate action to be *chosen* ,is that our goal is to reason about the solvability of a problem. This is why we have to make sure that *unwanted* actions are no *Applicable* (otherwise we might be able to cross the river without using the boat). This choice of ours relied on some implicit assumptions, and some quesions arise such as: What happens if there is more than one minimal? What happens in the situation where an internal node a is also a feasible solution? Should we allow such a situation? Are there elaborations that are relevant/valid only for some *situations* and some *contexts*? Do we obey *speech acts*, saying that if something is mentioned (e.g., as an elaboration), it is of importance? Is there any significance to the order of elaborations?

The original formalism was built on another framework, but for lack of space we did not draw it here. Other formalizations of action that we successfully tried our action-elaboration schema with, are (McCarthy 1986), (Sandewall 1989) (Filtered Chronological Minimization), and (Lin & Reiter 1994). All had similar conclusions, although different languages and action approaches required different formalizations of elaborations. The basic framework remained the same.

One surprising benefit from our approach is the ability to treat granularity directly. It is easy to see how the granularity of actions can be varied accross the theory without too complicated notations, using an elaboration such as $with\text{-}stages(stages)$.

A problem with our proposed formalism, is that we cannot elaborate our φ/ψ further. This seems to be equivalent to the Elaboration Tolerance problem as a whole, and we decided not deal with it here.

Future Work

Many directions for further work are evident. The first is applying it to a developed theory, and creating more common sense axioms. Applications of context, attachments to the elaboration (such as the purpose of the elaboration), and other *chosen* formula formalizations, are also immediate directions.

A lesson we drew from this work is that different action formalisms realize elaborations differently, and we expect to use this lesson in devising new action formalisms that will be more elaboration tolerant. Our hope is that this direction of research will lead to better understanding of Elaboration Tolerance, and to development of its theory. This issue is our main goal.

Acknowledgements

John McCarthy, Tom Costello and Sasa Buvac came with the many examples of elaborations. McCarthy also came up with the first formalization of the missionaries and cannibals that we used as a basis for our work. We owe special thanks to Vladimir Lifschitz, Avi Pfeffer and Fangzhen Lin for reading and making some important comments on drafts of this paper.

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic* 50:510–530.

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

Kartha, G. N., and Lifschitz, V. 1995. A simple formalization of actions using circumscription. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence.*

Lifschitz, V. 1993. Circumscription. In D.M. Gabbai, C.J.Hogger, J., ed., *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning.* Oxford University Press.

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

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes.*

McCarthy, J. 1986. Applications of Circumscription to Formalizing Common Sense Knowledge. *Artificial Intelligence* 28:89–116. Reprinted in (McCarthy 1990).

McCarthy, J. 1990. *Formalization of common sense, papers by John McCarthy edited by V. Lifschitz.* Ablex.

McCarthy, J. 1993. History of circumscription. *Artificial Intelligence* 59(1):23–26.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation.* Academic Press. 359–380.

Sandewall, E., and Shoham, Y. 1994. Nonmonotonic Temporal Reasoning. In Gabbai, D.; C.J.Hogger; and J.A.Robinson., eds., *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 4: Epistemic and Temporal Reasoning.* Oxford University Press.

Sandewall, E. 1989. Filter preferential entailment for the logic of action in almost continuous worlds. In *IJCAI-89.*

This is /u/eyal/ongo/elaboration/aaai/machinery-ea.tex.