Machinery for Elaborating Action – Preliminary Report

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

April 15, 1997

Abstract

We present the problem of Elaborating Action as a sub-problem of *Elaboration Tolerance*, and demonstrate a proposed solution. Knowledge Representations have the *Elaboration Tolerance Problem*, i.e., that slight variations of the knowledge result in significant changes to the original representation. *Action Elaboration* is a proposed direction in the theories of action scope. The problem of *Elaborating Action* is that of modifying properties of actions (such as preconditions and effects). *Action Elaboration* reifies the elaboration (the change), and introduces a new action for every elaboration that we add. The reasoning about the properties of the new action is then done using Elaboration Axioms.

1 Introduction

Our long term program deals with the *Elaboration Tolerance* of formalisms of knowledge. *Elaboration Tolerance* is a term coined by McCarthy in [McC93a] and used ever since in describing one of the more fundamental problems that AI and Computer Science research faces.

The motivation of *Elaboration Tolerance* is that small changes in one's knowledge/problem, should result in small changes to the representation of that knowledge/problem. One direction that we consider promising is the pursuit of formalisms that allow elaboration without the need for manual amendments/retraction of axioms/facts. The notion of *Elaboration* in this context is admittedly vague, but we found the current work to contribute to clearing it a bit.

An *Elaboration of Action* is any change in properties of a given action, that we can put explicitly in the system. Such properties are not necessarily exhibited as values of functions over that action, but rather as generic axioms including that action. In this paper, whenever we refer to adding/dropping preconditions/effects of an action we turn to the possibility of doing so using only the addition of formulas (no retraction/modification is allowed). We will use the nonmonotonic property of circumscription ([McC86]) to allow for changes in such fasion.

The current work is motivated by a developed formalization of the *Missionaries and Cannibals Problem* (MCP) presented in [MA96a], [MA96b], and [Ami97]. We begin with an exerpt from these works, and describe what kinds of elaborations we require. We then show how existing techniques let us absorb some elaborations but not others, and then proceed to describe our proposed formalism.

In our proposed formalism, instead of adding axioms regarding the action on which we wish to elaborate, we create a new action that inherits properties from the original action. We can then specify any change that we wish with regard to that new action. The technique we apply uses *Reified Elaborations* and *Elaboration Axioms* to reason about the properties of the new action. The last complication we consider is related to our initial motivation: In the MCP, we wish to prove that under some conditions (namely four missionaries and four cannibals), the problem is not solvable. For that we need to ignore the original action (the one before the elaboration), and we deal with that problem in some length.

Recently, [Cos97] argued that there is a tradeoff between the ability to add preconditions and to drop them, and a similar tradeoff with regard to adding and dropping effects.

Another somewhat related topic is Reiter's theory of primitive actions (cf [Rei91]). 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 sub-actions.

We take the convension that whenever we refer to objects in our model, upper case english letters denote constant object symbols, and lower case english letters denote object variables. For background material on situation calculus and theories of Action, the reader is refered to [Lif93], [McC90] and [SS94].

2 The Missionaries and Cannibals Problem

The missionaries and cannibals problem (abbreviated MCP) is stated as follows:

> **Three missionaries and three cannibals come to a river and find a boat that holds two. If the cannibals ever outnumber the missionaries on either bank, the missionaries will be eaten.**
>
> **How shall they cross?**

The two problems that motivate us here, with regard to the MCP, are to be able to prove that the MCP is solvable for three missionaries and three cannibals (we denote it $MCP(3,3)$), and is not solvable for the respective groups being of cardinality four and four (denoted $MCP(4,4)$). The first one requires us to reason about effects of actions, while the second one requires us to also reason about the possible actions at our disposal.

A compact formalism for the MCP is detailed in [Ami97] and a more elaborate formalism with proofs is presented in [MA96b]. Here we reproduce some of the details that will serve us in understanding the difficulties encountered with Elaboration Tolerance.

Our language will include (besides the constants of the above framework) the following object constants: M, C, $Boat$, $Banks$, which are the sets of Missionaries and Cannibals, the Boat, and the set of Banks, respectively.

For simplification, we restrict ourselves to actions of the form

$$A(group, bank) = using\_tool(Boat, go(group, bank))$$

Notice that the Boat is fixed, and we therefore implicitly assume that the boat is needed, that there is exactly one, and that there are no other actions possible. We further restrict the set of actions to be

$$Actions(s) = \left\{ A(group, bank) \;\middle|\; \begin{array}{l} bank \in Banks \wedge \\ group \subset inhabitants(opp(bank), s) \end{array} \right\} \tag{1}$$

Let A be a macro for $A(group, bank)$. We list the preconditions and effects of A.

Axioms related to *going*

$$\begin{array}{l} \neg Abq(A, s) \rightarrow Can(A, s) \\ Can(A, s) \rightarrow A \in Actions(s) \\ Can(A, s) \rightarrow 0 < card(group) \\ Can(A, s) \rightarrow (\forall p \in group)(location(p, s) = opp(bank)) \\ Can(A, s) \rightarrow (\forall p \in group)(location(p, Result(A, s)) = bank) \end{array} \tag{2}$$

Axioms related to the boat

$$\begin{array}{l} Can(A, s) \rightarrow location(Boat, Result(A, s)) = bank \\ Can(A, s) \rightarrow card(group) \leq capacity(Boat) \\ Can(A, s) \rightarrow water\text{-}crosser(Boat) \\ Can(A, s) \rightarrow available(Boat, group, s) \end{array} \tag{3}$$

To demonstrate our contribution, the above details will suffice, but for clarity we will note that here we listed only some of factors in the story (for example, we ignored the condition on the cannibals not outnumbering the missionaries).

With this framework, we are able to show that the $MCP(3,3)$ is solvable and that $MCP(4,4)$ is not (see [Ami97] or [MA96b] for details). The solvability of the first is a simple exercise in using the qualification and frame solutions (whichever one you take). For each situation we encounter we calculate the exact locations of each of the missionaries and cannibals, satisfy the preconditions for the next action, apply that action, and get a new situation for which the process continues.

Proving that $MCP(4,4)$ is not solvable requires us to reason about the possible actions. Since we explicitly limited our actions to those of the form

$A(group, bank)$, this is also a not-so-complicated result (see [MA96b]). We reason on the accessible situations (by means of applying actions to get to new situations) and their properties. Since we know exactly what are actions may be, we can get a quite limited family of situations, grouped according to their FrameFluent values. None of these groups has the required end condition, and thus the proof follows.

3 A possible consideration of elaboration of actions

In [Cos97] it is shown that some elaborations of actions can be viewed as adding or weakening preconditions and effects. We call such elaborations *Direct Elaborations*. It is argued in [Cos97] that there seem to be a tradeoff between the ability to add and drop effects, and the same goes with the ability to add and drop preconditions.

We divide *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., [LR94],[KL95]) support adding effects and preconditions in an easy manner. For our example this is done as follows (we use the same macro we used above for A):

Oars are needed for the operation of the boat

$$Can(A, s) \rightarrow has\_oars(Boat)$$

Each cross of the river makes the cannibals more hungry

$$Can(A, s) \rightarrow \forall c \in C \; hunger(c, Result(A, s)) > hunger(c, s)$$

After such additions to the theory, the solutions to the Qualification and Persistence problems remains the same, and thus the proof of $MCP(3,3)$ and $\neg MCP(4,4)$[1] is carried out using the same techniques.

The problem that we tackle is that, as shown in [Cos97], theories of action allow for only one of the two in the *add-drop* tradeoff. Retracting or changing previously asserted effects or preconditions, without actually pulling the right part of the formula out, is impossible.

The rest of the paper is dedicated to showing that our proposed formalism does allow for all the above kinds of elaborations of actions, and others.

4 Elaboration of actions

Let $A_0 = go(group, bank)$ be the action of crossing the river (no boat is mentioned or assumed). We are motivated by the observation that the action $using\_tool(Boat, A_0)$ is an elaboration of A_0, and that we can have both actions (A_0 and its elaboration) as two separate actions in the same formalism.

Having that intuition in mind we can see how the idea proceeds: Before we mentioned the elaboration of A_0, we had a characterization of preconditions and effects of A_0, and after the elaboration there is no reason to change our mind with regard to A_0. On the other hand the action $using\_tool(Boat, A_0)$ is a "fresh" action, on which we initially know nothing. Our task now will be to show how we can reason about this new action's properties[2].

4.1 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.

[1]i.e., proving that $MCP(4,4)$ is not solvable.

[2]Notice that this elaboration does not seem to fit easily into one of the four categories described above (e.g., adding 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.

Definition 4.1 *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:

$$\begin{aligned}&Elaborations(go(Group, Lb)) = \{using\_tool(Boat)\}\\&Elaborations(elaborate(go(Group, Lb),\\&\qquad using\_tool(Boat))) = \{using\_tool(One\text{-}oar)\}\end{aligned}$$

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

$$\textbf{Circ}\ \lambda ae\ elaborate(a, e) \tag{4}$$

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

4.2 Elaboration Axioms

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

$$property(a') \equiv ((property(a) \land \varphi) \lor \psi) \qquad (5)$$

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

EXAMPLE $Can(a', s) \equiv (Can(a, s) \land \varphi) \lor \psi$ with $\psi = False$ and $a' = elaborate(a, using\_tool(boat))$ $\varphi = precond(using\_tool(boat))$.

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. $using\_tool(tool)$ - 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.

$$Can(elaborate(action, using\_tool(tool)), s) \implies Can(action, s)$$

$$\begin{array}{l} Can(elaborate(action, using\_tool(tool)), s) \implies \\ \qquad accessible(tool, Actor(action), s) \\ location(tool, s) = location(actor, s) \implies accessible(tool, actor, s) \\ Can(elaborate(a, using\_tool(tool)), s) \implies usable(tool, s) \\ \neg abq_u(tool, s) \implies usable(tool, s) \end{array} \tag{6}$$

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

4.3 $MCP(3,3)$ is still solvable

Let us rewrite (2) and (3) with our new method.

Let A_0, A be macros for $go(group, bank), using(Boat, go(group, bank))$ respectively. We list the preconditions and effects of A_0, A.

Axioms related to *going*

$$\begin{array}{l} \neg Abq(A, s) \rightarrow Can(A, s) \\ Can(A_0, s) \rightarrow A_0 \in Actions(s) \\ Can(A_0, s) \rightarrow 0 < card(group) \\ Can(A_0, s) \rightarrow (\forall p \in group)(location(p, s) = opp(bank)) \\ Can(A_0, s) \rightarrow (\forall p \in group)(location(p, Result(A_0, s)) = bank) \end{array} \tag{7}$$

Axioms related to the boat

$$\begin{array}{l} Can(A, s) \rightarrow location(Boat, Result(A, s)) = bank \\ Can(A, s) \rightarrow card(group) \leq capacity(Boat) \\ Can(A, s) \rightarrow water\text{-}crosser(Boat) \\ Can(A, s) \rightarrow available(Boat, group, s) \end{array} \tag{8}$$

Elaboration related axioms

$$using\_tool(Boat) \in Elaborations(A_0) \tag{9}$$

Using these axioms and (6) we can prove $MCP(3,3)$. Notice that (6) is used to have inheritance of preconditions from A_0 to A). Also notice that the axioms (7) and (8) now refer to different actions: some for the action without the boat, and some to the action that includes the boat.

5 $MCP(4,4)$ is still unsolvable

This section is dedicated to the problem of restricting the set of actions possible in our problem. This restriction is essential if we wish to prove that $MCP(4,4)$ is not solvable. Coming back to our example with $A_0 = go(group, bank)$ and $using\_tool(Boat, A_0)$, we have two actions that we need to consider: the original A_0, and the elaboration $using\_tool(Boat, A_0)$. The motivation that will lead us through this section is that we would like to ignore A_0 when reasoning about the problem, since its elaboration should take its place.

5.1 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 5.1 *We define the relation* $<_e$.

$$\begin{array}{l} \forall e \in Elaborations(a)\ (elaborate(a,e) <_e a) \\ \exists a_3\ (a_1 <_e a_3 \land a_3 <_e a_2) \implies a_1 <_e a_2 \end{array} \tag{10}$$

Also, $a_1 \le_e a_2 \iff a_1 = a_2 \lor a_1 <_e a_2$. ■

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

$$Circ[\Sigma; <_e] \tag{11}$$

where Σ is the conjunction of (10), 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 $a' = elaborate(a, using\_tool(Boat))$.

$$a' <_e a$$
$$elaborate(a', using\_tool(one\text{-}oar)) <_e a$$

Figure 1 shows an example situation more clearly.

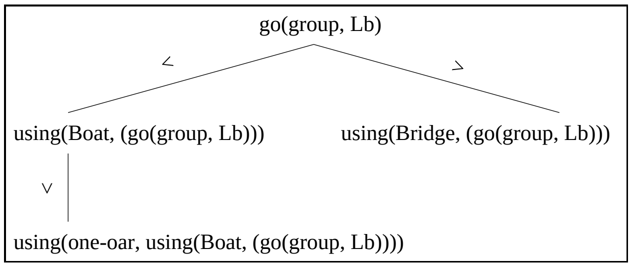
go(group, Lb)
<
>
using(Boat, (go(group, Lb)))
using(Bridge, (go(group, Lb)))
∨
using(one-oar, using(Boat, (go(group, Lb))))

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 ($<_e$-wise) action exists, picking that one might do. Also, if we have several minimal actions, they might all be available (e.g., $elaborate(a, using\_tool(one\text{-}oar))$ and $elaborate(a, using\_tool(two\text{-}oars))$). As mentioned above, the situation is not that simple (although for now we will assume so), and we discuss that in more length later.

5.2 The most elaborate action

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

Definition 5.2 *Let $a \in Actions$. a is* Basic, *if it has no generalization (according to $<_e$).*

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

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

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

■

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

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).

$$chosen(a, s) \iff Minimal(a) \tag{12}$$

6 Embedding *chosen* in the selected action framework

6.1 Adding Elaboration Axioms

In a developed system (like the MCP in [MA96a]), 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

$$\begin{array}{l}
can(elaborate(a, e), s) \implies ((can(a, s) \land \varphi^{Pre}_{a,e}(a, s)) \lor \psi^{Pre}_{a,e}(a, s)) \\
\\
Applicable(a, s) \iff (chosen(a, s) \land can(a, s)) \\
Applicable(a, s) \implies Postconds(a, s) \\
\\
Postconds(elaborate(a, e), s) \implies \\
\qquad (Postconds(a, s) \land \varphi^{Post}_{a,e}(a, s)) \lor \psi^{Post}_{a,e}(a, s))
\end{array} \tag{13}$$

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 (5) (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.[3] Now, we replace the effect axioms of the form $Can(a, s) \implies F(Result(a, s))$ with

$$Applicable(a, s) \implies F(Result(a, s))$$

6.2 Examples

In the beginning of this paper 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 $A(group, bank) = using\_tool(Boat, go(group, bank))$.

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

$$Applicable(A(group, bank), s) \implies Happy(group, Result(a, s))$$

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

$$\begin{array}{l} using\_tool(Oars) \in Elaborations(A(group, bank)) \\ Can(using\_tool(Oars, A(group, bank)), s) \implies \\ \qquad Can(A(group, bank), s) \wedge in(Oars, Boat) \end{array}$$

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

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

[3]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.

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).

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

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

7 Discussion

We presented the problem of elaborating actions and proposed the method of *Action Elaboration* as achieving more *Elaboration Tolerance.* We began with a presentation of a simple formalization of the MCP. We then presented the notion of a reified elaboration, and showed that after elaborating in the proposed method, we can still prove $MCP(3,3)$ and $\neg MCP(4,4)$.

With regard 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 (5). 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 (5) is somewhat oriented towards precondition elaboration than effects. It is our view that this latter point is much dependent on the exercised action formalism, but it is a problem that must be addressed if we are to get Elaboration Tolerance.

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?

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 $with\text{-}stages(stages)$.

The techniques applied in this work resemble both theories of Context (cf [McC93b]) and of Inheritance. In this preliminary work we do not explore these relationships, but the use of our function *elaborate* is very similar to the elaboration of contexts into other contexts, although here we deal with it in a quite restricted scope. Inheritance is also clearly embodied in our Elaboration Axioms, and we expect that Lin's note with regard to *default elaboration axiom* will prove to be closely related with this issue.

8 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.

9 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. We also wish to thank the anonymous referees for their many important notes and insights. This research was supported by an ARPA (ONR) grant N00014-94-1-0775.

References

[Ami97] Eyal Amir. Formalizing Action using Set Theory and Pointwise Circumscription. In NRAC-97', 1997.

[Cos97] Tom Costello. Beyond minimizing change. In *Proceedings of AAAI-97*, 1997.

[KL95] G. N. Kartha and V. Lifschitz. A simple formalization of actions using circumscription. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995.

[Lif93] Vladimir Lifschitz. Circumscription. In J.A.Robinson D.M. Gabbay, C.J.Hogger, editor, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning.* Oxford University Press, 1993.

[LR94] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 1994.

[MA96a] John McCarthy and Eyal Amir. Missionaries and Cannibals: Making it Elaboration Tolerant. http://www-formal.stanford.edu/eyal/nmr/m4.ps, 1996.

[MA96b] John McCarthy and Eyal Amir. Working through the Missionaries and Cannibals. http://www-formal.stanford.edu/eyal/nmr/m4-nmr3.ps, 1996.

[McC86] John McCarthy. Applications of Circumscription to Formalizing Common Sense Knowledge. *Artificial Intelligence*, 28:89–116, 1986. Reprinted in [McC90].

[McC90] John McCarthy. *Formalization of common sense, papers by John McCarthy edited by V. Lifschitz.* Ablex, 1990.

[McC93a] John McCarthy. History of circumscription. *Artificial Intelligence*, 59(1):23–26, 1993.

[McC93b] John McCarthy. Notes on Formalizing Context. In *IJCAI-93*, 1993. Available on http://www-formal.stanford.edu/jmc/.

[Rei91] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.

[SS94] Erik Sandewall and Yoav Shoham. Nonmonotonic Temporal Reasoning. In D.M. Gabbay, C.J.Hogger, and J.A.Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 4: Epistemic and Temporal Reasoning.* Oxford University Press, 1994.