# Equational Logic and Theories of Action

Steffen Hölldobler
KI, Informatik, TU Dresden
D–01062 Dresden (Germany)
sh@inf.tu-dresden.de

**Abstract**

This is a survey on reasoning about situations, actions, and causality within equational logic. Its main feature is the representation of situations as multisets of resources. Such resources are consumed whenever the conditions of an action are satisfied and are produced whenever the action is applied. It is demonstrated that the well-known frame and ramification problems can be elegantly solved within such a framework. Moreover, parallel actions as well as hierarchical planning problems can be naturally represented in this approach.

## 1 Introduction

The design of rational agents which perceive and act upon their environment is one of the main goals of Intellectics, ie. Artificial Intelligence and Cognition [3]. Inevitably, such rational agents need to represent and reason about situations, actions, and causality, and it comes as no surprise that these topics have a long history in Intellectics. Already in 1963 John McCarthy proposed a predicate logic formalization, viz. the situation calculus [16, 17], which has been extensively studied and extended ever since (see eg. [14, 18]). The core idea underlying this line of research is that a situation is a snapshot of the world and that actions mapping situations onto situations are the only means for changing situations. As it is impossible to completely describe the world at a particular time or to completely specify an action each situation and each action can only be partially known. This gives rise to several severe problems like the frame, ramification, qualification, and the prediction problem.

All these problems have a cognitive as well as a technical aspect. We are cognitively interested in how humans solve these problems as we are faced with them as well, and we technically interested in how we can adequately compute solutions to these problems. Computation requires representation and reasoning. Within logical formalisms actions are typically represented as axioms such that the question whether goals can be reached by applying certain actions is a logical consequence of these axioms and the facts about a situation (see [16]). Within the situation calculus the facts about a situation are represented as predicates. From a logical point of view, predicates are properties and properties do not change within a logical reasoning process. But as we explicitely want to model change, McCarthy proposed to add to each predicate symbol an additional argument representing the situation, in which the correspond-

ing predicate is supposed to hold. Although this is adequate from a representational point of view, it imposed a severe constraint on the reasoning formalism. Whenever an action is applied all facts about a situation have to be recomputed.

In recent years several new methods have been proposed to reason about situations, actions, and causality [2, 7, 12]. They are all based on the same idea, viz. that facts about a situation are represented as resources. Resources can be consumed and produced. They are consumed whenever the conditions of an action are to be satisfied and they are produced as effects of an action. In this paper we will outline one of these approaches, viz. an equational logic formalization first proposed in [12].

In the following Section 2 we will give the world model and define situations, actions, and causality. In particular, we will discuss the problems arising from the fact that situations and actions are only partially known. The equational logic representation and the reasoning process based thereon are presented in Section 3. Further extensions of this approach are sketched in Section 4. A brief discussion in Section 5 completes the overview given in this paper.

## 2 Worlds, Situations, Actions, and Causality

To model a changing world we need a logic for representing and reasoning about the world and a logic for representing and reasoning about the change. As far as the former is concerned we will simply use propositional logic. In the literature facts about a world are usually called *fluents* and a *fluent formula* is a propositional logic formula over the set of fluents $\mathcal{F}$.[1] Besides the usual notion of consistency in propositional logic it is sometimes convenient to model conflicts in the world as inconsistency as well. For example, the fluents $a$ and $d$ may be used to represent the facts that certain individuals are alive and dead, respectively. As an individual cannot be alive and dead at the same time, we specify with the help of the fluent formala $\overline{a} \rightarrowtail d$ that such a world is inconsistent as $a \wedge d \rightarrowtail a \wedge \overline{a} \rightarrowtail \bot$. The *domain constraints* $\mathcal{D}$ are simply a set of such fluent formulae.

A *world state*, or simply *state*, is a maximal and consistent set of fluent literals with respect to $\mathcal{D}$. Let $\mathcal{C}$ denote the set of states. A *dynamical system* consists of a set of fluents $\mathcal{F}$, a set of action names $\mathcal{A}$, a set of domain constraints $\mathcal{D}$, and a partial state transition function $\Phi : \mathcal{A} \times \mathcal{C} \rightarrowtail \mathcal{C}$. It is said to be *deterministic* iff $\Phi$ is deterministic. Figure 1 shows an example of a simple deterministic dynamical system.

Each element $(A, Z, Z') \in \Phi$ can be conveniently represented by a so-called *action description* $\langle C, A, E\rangle$ such that $C \subseteq Z$ and $Z' = (Z \setminus C) \cup E$. $C$ and $E$ are called *conditions* and *effects*, respectively. Without loss of generality we may require that there are no action descriptions $\langle C_1, A, E_1\rangle$ and $\langle C_2, A, E_2\rangle$ such that $C_1 = C_2$.

[1] Without loss of generality we may assume that the negation sign occurs only in front of atoms. Thus, we may build fluent formulas from fluent literals and the connectives $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$.

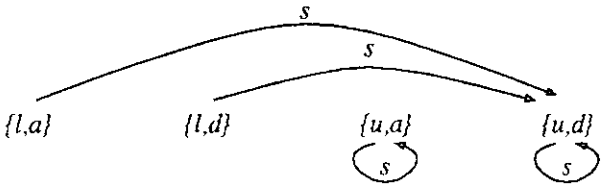


Figure 1: A simple deterministic dynamical system with fluents $\mathcal{F} = \{l, u, a, d\}$ representing the facts that a gun is *l*oaded or *u*nloaded, and that an individual is *a*live or *d*ead, respectively. The only action is a *s*hoot action represented by $\mathcal{A} = \{s\}$. The domain constraints enforce that a gun cannot be loaded and unloaded at the same time, and that an individual cannot be dead and alive at the same time, ie. $\mathcal{D} = \{\overline{l} \rightarrowtail u,\ \overline{a} \rightarrowtail d\}$. The figure shows the four maximal and consistent states as well as the transition function $\Phi$ of this shooting scenario.

Returning to the example shown in Figure 1 we find that

$$\{\ \langle \emptyset, s, \emptyset\rangle,\ \langle \{l\}, s, \{u\}\rangle,\ \langle \{l, a\}, s, \{u, d\}\rangle\ \} \tag{1}$$

is a complete action description for the shooting scenario.

An action description $\langle C, A, E\rangle$ is applicable to a state $Z$ iff $C \subseteq Z$ and if applied yields the state $Z' = (Z \setminus C) \cup E$. As the shooting scenario already exemplifies there may be more than one description of a certain action applicable to a given state. Such potential conflicts can be avoided by defining a partial order on action descriptions and applying only the most specific action according to this ordering [13]. Formally, an action description $\langle C_1, A, E_2\rangle$ is *more specific* than a description $\langle C_2, A, E_2\rangle$ iff $C_2 \subset C_1$. The *successor* of a state $Z$ wrt an action $A$ is obtained by applying the most specific description of $A$ to $Z$. Thus, in the shooting scenario the application of the shoot action to $\{l, a\}$ yields $\{u, d\}$ as intended.

If the world were a deterministic dynamical system and were accessible to a planning agent, then all the agent would have to do in order to reason about causal relationships is to compute complete action descriptions and apply them accordingly. There is a technical problem, though, which has to be addressed, viz. the problem to efficiently implement the computation of a successor situation, and we will come back to this so-called *technical frame problem* in Section 3.3. A more striking problem arises from the fact that the word is generally inaccessible and non-deterministic. In other words, an agent has only partial knowledge of the current state as well as of the available actions, and the application of an action may lead to effects which cannot be determined in advance. Ignoring the latter for a moment we will assume that the world is deterministic, but that the agent has only partial knowledge about the world and the actions.

As the current state $Z$ is generally not completely known, the best our agent can hope for is to know a consistent subset $S \subseteq Z$ of it. Let us call such a subset a

*situation*. Thus, a situation $S$ represents all states $Z$, where $S \subseteq Z$. As a situation is a set of fluents, actions can be applied to situations as well. Let $(C, A, E)$ be the most specific description of the action $A$ applicable to $S$ and $S' = (S \backslash C) \cup E$ be the successor of $S$. It can be easily demonstrated that $S'$ may represent states, which cannot be reached by applying $A$ to the states represented by $S$. For example, the situation $S = \{l\}$ represents the states $\{l, a\}$ and $\{l, d\}$ in the previously mentioned shooting scenario. Applying $\langle\{l\}, s, \{u\}\rangle$, which is the most specific action description in this case, to $S$ yields $S' = \{u\}$. $S'$ represents the states $\{u, a\}$ and $\{u, d\}$, but only the latter one can be reached in the world be applying the shoot action to $\{l, a\}$ and $\{l, d\}$. In this simple scenario the agent could reason to find that $d$ must hold in $S'$ provided that he knows that his action descriptions are complete.[2] However, this will be impossible if the example is slightly changed,[3] or if the agent cannot assume that his action descriptions are complete.

Conversely, if $S = \{a\}$ then the most specific desciption of $s$ applicable to $S$ is $\langle\emptyset, s, \emptyset\rangle$, which would lead to the successor situation $S' = S$. A sceptical agent would not draw this conclusion as the gun might be loaded in the state $\{a, l\}$ represented by $S$ and, consequently, it cannot be safely concluded that the individual is alive after shooting. A bold agent will draw the conclusion, but then he should be prepared to revise his conclusion if he observes later that the individual is in fact dead. Such an observation would also allow for deducing additional fluents about the initial state. If after shooting the individual is dead then the gun must have been loaded initially.

So far we have considered only situations as partial representation of states. As already mentioned actions may be only partially specified as well. It is generally unconceivable to assume that the conditions and effects of actions are completely known. As with situations the conditions and effects of an action description may be only subsets of the real but unknown conditions and effects. It comes as no surprise that reasoning with partial descriptions of actions leads to problems and requires to make certain assumptions.

The *frame problem* is the problem whether the fluents which hold in a situation $S$ and are unaffected by an action hold also in the successor situation. According to the *frame* or *persistence assumption* they do hold.[4] For example, if the shoot action is described by

$$\{\ \langle\emptyset, s, \emptyset\rangle,\ \langle\{l\}, s, \{u\}\rangle\ \}$$

only, then its application to the situation $\{l, a\}$ yields $\{u, a\}$ under the persistence assumption. This is not correct wrt to the world given in Figure 1, but recall that the world is inaccessable for the agent.

The *ramification problem* is the problem to decide which fluents are affected by an action. Suppose we extend our shooting scenario by an additional fluent $walking$, which represents the fact that the individual is walking. Clearly, for somebody to be able to walk he must be alive, which can be expressed by the additional domain constraint $w \rightarrow a$. Now, applying the shoot action as given in (1) to the situation $S = \{l, a, w\}$ yields $S' = \{u, d, w\}$ under the persistence assumptions. However, $S'$ is inconsistent wrt the domain constraints as $w \rightarrow a \leftrightarrow \overline{d}$ holds and, consequently, $S'$ is not even a situation. As shown in [19] $S'$ can be turned into a situation by applying the causal relationship $d \Rightarrow \overline{w}$ and thereby replacing $w$ by $\overline{w}$. The resulting set of fluent literals, ie. $\{u, d, \overline{w}\}$, is consistent wrt the domain constraints and, thus, is a situation. Such causal relationships can be derived from the domain constraints, but sometimes additional domain knowledge specifying an influence relation may be required [19].

The *qualification problem* is the problem to decide which fluents are needed to completely specify the conditions of an action. Extending our shooting example once again we may add an action entice with condition $\overline{w}$ and effect $w$. Applied to the situation $\{u, d, \overline{w}\}$ this action yields $S' = \{u, d, w\}$, which is inconsistent as discussed in the previous paragraph. Unfortunately, $S'$ can now not be turned into a consistent situation by applying causal relationships as we neither want to change the direct effects of an action nor shall a dead individual become alive by enticing it. The fact that domain constraints are violated in this case should lead to an additional condition, viz. $a$, stating that only alive individuals can be enticed to walk.

Finally, the *predication problem* is the problem to decide how long a certain fluent holds. Suppose that the agent knows that the individual is alive. How long does this hold? Depending on the nature of an individual it may be just one day or several years. But it is important to realize that the actions performed by an agent are not the only means to change the world.

# 3 Equational Logic Programming and Planning

In this section we like to demonstrate how situations, actions, and causality can be represented in an equational logic. Before developing the equational logic we should answer the question of why to use an equational logic at all. After all the situation calculus has been developed precisely for this reason and has been extensively used within the artificial intelligence community [16, 17, 18]. Within the situation calculus situations are represented by terms and fluents by predicates ranging over situations. For example, in the shooting scenario depicted in Figure 1, the situation $\{l, a\}$ would be represented by the formula $l(s_0) \wedge a(s_0)$, where $s_0$ is a constant. Applying the shoot action to this situation yields $\{u, d\}$, which would be represented by $u(res(s, s_0)) \wedge d(res(s, s_0))$, where $res$ is a function symbol which appied to an action and a situation denotes the successor situation.

From a representational point of view there is nothing wrong with the formalization of situations, actions, and causality within the situation calculus. From a reasoning point of view, however, there is an efficiency problem related to the solution of the

[2] As the states are maximal and consistent, the agent can conclude that either $a$ or $d$, but not both, must hold in the states represented by $S$ and $S'$. A simple case analysis shows, that applying $s$ will always lead to $d$ in the successor state.

[3] Just drop the domain constraint $\overline{a} \leftrightarrow d$, for example.

[4] It is an interesting question whether we make such an assumption in our daily encounters. Apparently we do so in many circumstances.

technical frame problem. As each fluent is labelled by the situation in which the fluent holds, *all* fluents have to be updated whenever an action is applied. In particular, this holds also for all fluents which are not affected by an action and which, according to the persistence assumption, remain unchanged. The nature of this problem is linked to the fact that fluents are represented as properties within the situation calculus. Under a given interpretation properties do or do not hold, but they may not change their truth value during the reasoning process.

Since several years alternative approaches are developed whose main characterization is the feature that fluents are represented as resources [2, 7, 15, 12]. Resources can be consumed and produced. They are consumed whenever the conditions of an action are to be satisfied and they are produced by the effects whenever an action is applied. In the sequel we will present the equational logic approach based on [12] as it has turned out that this approach admits a standard semantics and is equivalent to the linear connection method developed in [2] and the linear logic approach developed in [15], both of which do not have a standard semantics [8].

## 3.1 Representing Fluents and Situations

As a first step towards an equational logic approach to planning we have to choose an appropriate representation for fluents and situations. As already mentioned we like to represents fluents by resources and, consequently, situations are represented by multisets. Formally, each fluent literal is reified and represented by a constant. The empty multiset is represented by the constant $\emptyset$ and non-empty multisets are represented using the binary function symbol $\circ$, which is required to be associative (A), commutative (C), and to admit the constant $\emptyset$ as a unit element (1). A situation $S = \{f_1, \ldots, f_n\}$ is now represented by the term $\tau_S = f_1 \circ \ldots \circ f_n \circ \emptyset$, where $f_1, \ldots, f_n$, $n \geq 0$, are fluent literals.

The underlying equational theory (AC1) is built into the unification computation and, thus, when reasoning about situations, actions, and causality we will apply SLDE-resolution [10] as long as the logic programs are definite and SLDENF-resolution [20] as soon as negation is involved.

## 3.2 Representing Actions and Causality

As a second step we have to choose representations for actions and causality. Each action description $\langle C, A, E\rangle$ is represented by a fact

$$action(\tau_C, A, \tau_E). \tag{2}$$

For example, the action description of the shooting scenario given in (1) is represented by

$$\begin{array}{l} action(\emptyset, s, \emptyset) \\ action(l, s, u) \\ action(l \circ a, s, u \circ d) \end{array}$$

The question whether an action description $\langle C, A, E\rangle$ is applicable to situation $S$, ie. whether $C \subseteq S$, can now be mapped onto the AC1-unification problem

$$\tau_S =_{AC1} \tau_C \circ V, \tag{3}$$

where $V$ is a variable. If this unification problem is solvable with substitution $\sigma$ then $\sigma V =_{AC1} \tau_{S \setminus C}$. In other words, $\sigma V$ represents $S \setminus C$. Consequently, the application of an action description $\langle C, A, E\rangle$ to a situation $S$ yielding the successor situation $S' = (S \setminus C) \cup E$ can be computed as $\sigma V \circ E$. Alltogether, with the ternary predicate $causes(S, [A_1, \ldots, A_n], G)$ we can express that situation S is transformed into situation G by applying the actions $A_1, \ldots, A_n$, $n \geq 0$, as follows.

$$\begin{array}{lll} causes(S, [], S). & & \\ causes(S, [A|P], G) & \leftarrow & action(C, A, E), \\ & & C \circ V =_{AC1} S, \\ & & causes(V \circ E, P, G). \end{array} \tag{4}$$

Similarly, the check whether the action description $\langle C, A, E\rangle$ is not the most specific description of $A$ applicable to situation $S$ can be performed by the following equational logic program.

$$\begin{array}{lll} non\_specific(A, C, S) & \leftarrow & action(C', A, E'), \\ & & C' \circ V =_{AC1} S, \\ & & C \circ W =_{AC1} C', \\ & & \neg W =_{AC1} \emptyset. \end{array} \tag{5}$$

This check can be built into (4) as follows.

$$\begin{array}{lll} causes(S, [], S). & & \\ causes(S, [A|P], G) & \leftarrow & action(C, A, E), \\ & & C \circ V =_{AC1} S, \\ & & \neg non\_specific(A, C, S), \\ & & causes(V \circ E, P, G). \end{array} \tag{6}$$

Finally, as equations occur in the body of program clauses we have to add the axiom of reflexivity.

$$X =_{AC1} X. \tag{7}$$

It has been shown in [13] that the completion of the equational theory AC1 and the logic program (2)∪(5)∪(6)∪(7) is a model for $causes(\tau_S, [a_1, \ldots, a_n], \tau_G)$ iff the initial situation $S$ can be transformed into the goal situation $G$ by applying the actions $a_1, \ldots, a_n$, $n \geq 0$. In particular, in each step the most specific action description is applied. Moreover, it has been shown that if in the goal $\leftarrow causes(s, [a_1, \ldots, a_n], g)$ the term $s$ is ground, then no SLDENF-derivation of this goal wrt the equational theory AC1 and (2) ∪ (5) ∪ (6) ∪ (7) flounders or is infinite, and that SLDENF-resolution is a correct and complete calculus for reasoning with most specific action descriptions.

## 3.3 Solving the Frame Problem

As already mentioned conventional approaches to modelling situations, actions, and causality based on the situation calculus need additional axioms — often called frame axioms [17], law of inertia [14], or successor state axioms [18] — to compute what actions do *not* change. Such additional axioms are unnecessary in the equational approach. The technical frame problem is elegantly solved by mapping it onto the unification problem (3). All fluents which remain unchanged are bound to the variable $V$. For example, let $S$ be the situation $\{l, a, b_1, b_2\}$, where $b_1$ and $b_2$ denote two blocks, then the unification problem

$$l \circ a \circ b_1 \circ b_2 =_{AC1} V \circ l \circ a \tag{8}$$

has a unique most general unifier $\{V \mapsto b_1 \circ b_2\}$ modulo AC1. Consequently, the action description $\langle\{l, a\}, s, \{u, d\}\rangle$ can be applied to $S$ yielding $\{u, d, b_1, b_2\}$.

This solution of the frame problem is ultimately linked to the fact that fluents are represented as resources, ie. that $\circ$ is an AC1–symbol. One could be tempted to model situations with the help of an ACI1–symbol, say $\cdot$, ie. a symbol which is not only AC1 but also idempotent (I). After all, situations are sets and a distinguished feature of sets is that multiple occurrences of an element are not taken into consideration. But then, the unification problem

$$l \cdot a \cdot b_1 \cdot b_2 =_{ACI} V \cdot l \cdot a$$

corresponding to (8) has additional most general solutions like $\{V \mapsto l \cdot b_1 \cdot b_2\}$. In other words, $V$ does no longer represent only those fluents which remain unchanged.

As discussed in [11] using an AC1 instead of an ACI1–symbol for representing situations solves the frame problem just as the elimination of the weakening and contraction rules from the Gentzen calculus or the restriction to linear connections in the connection method solves it as well. The reason is always the same: fluents are represented as resources which are consumed and produced by actions.

## 3.4 Solving the Ramification Problem

Solving the technical frame problem is only the first step towards computing the successor situation. We have to solve the ramification problem as well and we will show how this can be done within the equational logic approach following [19].

The ramification problem arises as soon as the domain constraints enforce dependencies which are not reflected in the action descriptions. Whereas in a simple blocks world scenario it is possible to formally prove that the application of an action to a consistent situation yields again a consistent situation (see [12]) this is often impractical in more involved scenarios.[5] As already shown by the walking–example in Section 2 we will make use of so–called causal relationships, which can be automatically derived from the domain constraints using an additional influence relation. The influence relation is domain dependent and specifies which fluents are influenced by others. As an example Figure 2 depicts an electric circuit with two switches and a light bulb taken from [14].

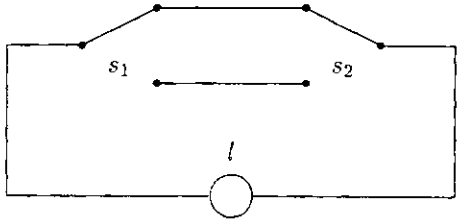


Figure 2: A simple electric circuit with $\mathcal{F} = \{s_1, s_2, l\}$ representing that *s*witch 1, *s*witch 2 are in the up–position, and that the *l*ight bulb is on, respectively. There are two actions: both switches can be toggled which is represented by $\mathcal{A} = \{t_1, t_2\}$. The light is on iff both switches are in the same position, which can be expressed by the domain constraint $\mathcal{D} = \{l \leftrightarrow (s_1 \leftrightarrow s_2)\}$. Simple physics tells us that changing the position of a switch causes the light to go on or off but not vice versa. Hence, the influence relation is defined by $\{(s_1, l), (s_2, l)\}$.

Formally, a *clausal relationship* is an expression of the form $F : e \Rightarrow f$, where $F$ is a fluent formula and $e$, $f$ are fluent literals. Let $S$, $E$ be sets of fluent literals, then such a relationship is *applicable* to $(S, E)$ iff $S \models F$, $\overline{f} \in S$, and $e \in E$. Its application yields $(S', E')$, where $S' = (S \setminus \{\overline{f}\}) \cup \{f\}$ and $E' = E \cup \{f\}$.

We can now extend the definition of a successor situation. Let $S$ be a situation and $\langle C, A, E\rangle$ be a most specific action description applicable to $S$. $S'$ is a *successor situation* of $S$ wrt $A$ iff there exists a set of fluent literals $E'$ such that $(S', E')$ can be obtained from $((S \setminus C) \cup E, E)$ by applying causal relationships. In other words, after applying the action $A$ to $S$ causal relationships are applied until a consistent set of fluent literals is reached.

Returning to the electric circuit domain let $\langle\overline{s_1}, t_1, s_1\rangle$ be an action description. $t_1$ is applicable in $S = \{\overline{s_1}, s_2, \overline{l}\}$ yielding the inconsistent set $S^* = \{s_1, s_2, \overline{l}\}$. Now, applying the causal relationship $s_2 : s_1 \Rightarrow l$ to $(S^*, \{s_1\})$ leads to the successor situation $S' = \{s_1, s_2, l\}$.

The equational logic approach developed so far can easily be extended to compute with causal relationships. First, we have to deal with the fluent formula $F$ occurring in a causal relationship and to check whether $F$ holds in a given situation $S$. This

[5] From a theoretical point of view one could compile the domain constraints into refined action descriptions making use of the specifity–criterion, but this may lead to an intractable number of action descriptions.

is accomplished by the binary predicate *holds* as follows.[6]

$$
\begin{array}{lcl}
holds(F, F \circ V). & & \\
holds(F \wedge G) & \leftarrow & holds(F,S),\ holds(G,S). \\
holds(F \vee G) & \leftarrow & holds(F,S). \\
holds(F \vee G) & \leftarrow & holds(G,S). \\
holds(F \rightarrow G) & \leftarrow & holds(G,S). \\
holds(F \rightarrow G) & \leftarrow & \neg holds(F,S). \\
holds(F \leftrightarrow G) & \leftarrow & holds(F \rightarrow G,S),\ holds(G \rightarrow F,S).
\end{array} \tag{9}
$$

Based on this program we can now compute whether a given situation is consistent wrt to the domain constraints $\mathcal{D} = \{D_1, \ldots, D_n\}$.

$$
consistent(S) \leftarrow holds(\tau_{D_1}, S), \ldots, holds(\tau_{D_n}). \tag{10}
$$

The application of a causal relationship $F : e \Rightarrow f$ to a pair of sets $(S, E)$ is implemented by the predicate *ramify*.

$$
ramify(S \circ \overline{f}, E \circ e, S \circ f, E \circ e \circ f) \leftarrow holds(\tau_C, S \circ \overline{f}). \tag{11}
$$

Such a clause is generated for each causal relationship. Recursive applications of causal relationships are implemented by the predicate *consistify*.

$$
\begin{array}{lcl}
consistify(S,E,S) & \leftarrow & consistent(S). \\
consistify(S,E,T) & \leftarrow & ramify(S,E,S',E'),\ consistify(S',E',T).
\end{array} \tag{12}
$$

We can now build this relation into the definition of the *causes* -predicate (6).

$$
\begin{array}{lcl}
causes(S,[],S). & & \\
causes(S,[A|P],G) & \leftarrow & action(C,A,E), \\
 & & C \circ V =_{ACI} S, \\
 & & \neg non\_specific(A,C,S), \\
 & & consistify(V \circ E, E, S'), \\
 & & causes(S',P,G).
\end{array} \tag{13}
$$

Alltogether we obtain an equational logic program which solves the frame as well as the ramification problem.

# 4 Extensions

The equational logic approach presented in the previous section has been extended in various ways. So far we have only dealt with deterministic systems and situations were always sets of fluent literals. These sets are represented with the help of a binary function symbol $\circ$, which corresponds to a non-idempotent conjunction.

[6] One should recall that the negation sign may occur only in front of atoms.

Technically, there is no need to restrict the equational logic to non-idempotent conjunctions. We may add additional function symbols corresponding to idempotent conjunction and disjunction as well as to non-idempotent disjunction and combine these using appropriate distribution laws. In such a framework it is then possible to represent non-deterministic actions like roling a dice, which leads to six successor situations. Based on this representation we may then model sceptical or credulous agents depending on whether they believe a fluent if it holds in each or only one successor situation, respectively [9, 5].

In [4] it was shown how the approach can be extended to parallel actions. In particular, if concurrently executed actions yield conflicting effects then this is interpreted as an implicit uncertainty in the action descriptions. Such actions are treated non-deterministically wrt to the conflicting effects.

So far only fluents were reified and modelled as resources, whereas actions were represented by atoms (see (2)). According to the laws of predicate logic such atoms cannot be changed during a reasoning process and, moreover, they can be used over and over again. Unfortunately, this may not correspond to a realistic world model. Consider for example the action

$$
action(q \circ q \circ q, g, l) \tag{14}
$$

modelling a vending machine, where one can *g*et a *l*emonade for three *q*uaters.[7] In contrast to predicate logic such an action may not be applicable infinitely many times in the real world because the vending machine may run out of lemonade and needs to be restocked. In other words, action (14) itself should be treated as a resource which is consumed whenever a customer gets himself a lemonade and is produced whenever the vending machine is restocked. Such a model has been developed in [6]. It additionally allows for refining actions by adding conditions and/or effects, and for combining simple actions to yield complex ones. In other words, it models hierarchical planning scenarios. Moreover, reasoning about situations, actions, and causality in such scenarios can be mapped onto an appropriately extended chemical abstract machine [1], which thus may serve as an asynchronous and abstract machine model.

# 5 Discussion

The equational logic approach has a computational advantage over solutions based on the situation calculus as facts about a situation are represented as resources [21]. It has a semantical advantage over solutions based on the linear connection method [2] or linear logic [7] as it admits a standard semantics. Nevertheless there are a variety of open questions.

[7] As situations are represented as multisets within the equational logic approach we may take advantage of the fact that multiple occurrences of an element within multisets are distinguishable. Thus, a situation where we have three quaters can be represented by $q \circ q \circ q$.

We have not yet completed the design of a rational agent interacting with its environment and based on the equational logic approach. We have to develop adequate solutions for the qualification and the prediction problem. The equational logic approach is purely theoretical so far. There exists a (naive) PROLOG-implementation, but we really have to build the equational theory in question into an appropriate abstract machine. We believe that the chemical abstract machine [1] is a good candidate model for our purposes, but it has to be extended. Last, not least, whereas our formalism is based on situations chanced by actions, nowadays practical planners are based on the idea of refining partial plans. We believe that the equational logic approach can be extended to represent and reason about partial plans if we manage to integrate partially ordered sets.

## References


[1] G. Berry and G. Boudol. The chemical abstract machine. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, pages 81–94, 1990.

[2] W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

[3] W. Bibel. Intellectics. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 705–706. John Wiley, New York, 1992.

[4] S.-E. Bornscheuer and M. Thielscher. Representing concurrent actions and solving conflicts. In B. Nebel and L. Dreschler-Fischer, editors, *KI–94: Advances in Artificial Intelligence*, volume 861 of *Lecture Notes in Artificial Intelligence*, pages 16–27. Springer, 1994.

[5] S. Brüning, G. Große, S. Hölldobler, J. Schneeberger, U. Sigmund, and M. Thielscher. Disjunction in plan generation by equational logic programming. In A. Horz, editor, *Beiträge zum 7. Workshop Planen und Konfigurieren*, pages 18–26. Arbeitspapiere der GMD 723, 1993.

[6] K. Eder, S. Hölldobler, and M. Thielscher. An abstract machine for reasoning about situations, actions, and causality. In R. Dyckhoff, H. Herre, and P. Schroeder-Heister, editors, *Proceedings of the International Workshop on Extensions of Logic Programming*, volume 1050 of *Lecture Notes in Artificial Intelligence*, pages 137–151. Springer, 1996.

[7] J. Y. Girard. Linear logic. *Journal of Theoretical Computer Science*, 50(1):1 – 102, 1987.

[8] G. Große, S. Hölldobler, and J. Schneeberger. Linear deductive planning. *Journal of Logic and Computation*, 1996. (to appear).

[9] G. Große, S. Hölldobler, J. Schneeberger, U. Sigmund, and M. Thielscher. Equational logic programming, actions, and change. In K. Apt, editor, *Proceedings of the International Joint Conference and Symposium on Logic Programming*, pages 177–191. MIT Press, 1992.

[10] S. Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *Lecture Notes in Artificial Intelligence*. Springer, 1989.

[11] S. Hölldobler. On deductive planning and the frame problem. In A. Voronkov, editor, *Proceedings of the Conference on Logic Programming and Automated Reasoning*, pages 13–29. Springer, LNCS, 1992.

[12] S. Hölldobler and J. Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990. A short version appeared in the Proceedings of the German Workshop on Artificial Intelligence, Informatik Fachberichte *216*, pages 63-73, 1989.

[13] S. Hölldobler and M. Thielscher. Computing change and specificity with equational logic programs. *Annals of Mathematics and Artificial Intelligence*, 14:99–133, 1995.

[14] V. Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[15] M. Masseron, C. Tollu, and J. Vauzielles. Generating plans in linear logic. In *Foundations of Software Technology and Theoretical Computer Science*, pages 63–75. Springer, LNCS *472*, 1990.

[16] J. McCarthy. Situations and actions and causal laws. Stanford Artificial Intelligence Project: Memo 2, 1963.

[17] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463 – 502. Edinburgh University Press, 1969.

[18] 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 — Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[19] M. Thielscher. Computing ramification by postprocessing. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1994–2000, 1993.

[20] M. Thielscher. On the completeness of SLDENF-resolution. *Journal of Automated Reasoning*, 1996. (To appear).

[21] S. Hölldobler und M. Thielscher. Properties vs. resources: Solving simple frame problems. Technical Report AIDA–96–03, Intellektik, Informatik, TH Darmstadt, 1996.