Reasoning about Concurrent Interaction*

Lars Karlsson and Joakim Gustafsson
Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
Tel: +46-13-28 24 28, Fax: +46-13-28 26 66
Email: {larka,joagu}@ida.liu.se

October 19, 1999

Abstract

In this paper we present TAL-C, a logic of action and change for worlds with concurrency. TAL-C has a first-order semantics and proof theory. It builds on an existing logic TAL, which includes the use of dependency laws for dealing with ramification. It is demonstrated how TAL-C can represent a number of phenomena related to action concurrency: action duration, how the effects of one action interferes with or enables another action, synergistic effects of concurrent actions, conflicting and cumulative effect interactions, and resource conflicts. A central idea is that actions are not described as having effects that directly alter the world state. Instead, actions produce influences, and the way these influences alter the world state are described in specialized influence laws. Finally, we address how TAL-C scenarios can be written to support modularity.

1 Introduction

To an agent operating in a complex multi-agent environment, it is important to be able to reason about how the environment changes due to the actions that the agents perform. Most of the work in reasoning about action and change has been done under the (sometimes implicit) assumption that there is a single agent that performs sequences of actions. This is a comparatively easier problem than reasoning about action and change under a multi-agent assumption, or under a non-sequentiality assumption for a single agent, which by necessity introduces the complication that one or several agents can perform actions concurrently. Concurrency, in turn, can involve a wide range of interactions between actions, which makes it unlikely that there is one single, uniform technique of general applicability.

*To appear in *Journal of Logic and Computation*, Vol. 9, 1999.

1.1 TAL-C

This paper presents an approach to reasoning about action and change which supports the description of concurrent actions with nontrivial interactions. The approach is based on TAL (*Temporal Action Logic*, formerly called PMON) which in its original form [41, 5] covers worlds with a natural numbers time domain and sequentially executed actions whose effects can be context-dependent (different initial states can lead to different effects) and nondeterministic (the same initial state can lead to several different effects). From the perspective of concurrency, an important property of TAL is that actions have durations (that is occur over an interval of time). TAL has recently been extended to support the description of ramifications, or indirect effects [16], and to deal with qualified action descriptions [7]. In this paper we present TAL-C, a development of the TAL/PMON formalism, which combines ramification and concurrency. Parts of the results in this paper have also been exploited for dealing with delayed effects [21]. In TAL-C, the description of concurrent interactions are done on the level of features (state variables). The central idea is that actions are not modeled to directly change the world state, but to produce influences on features. For each feature one can then use influence laws to specify how it is affected by its various associated influences. Besides providing a flexible and versatile means for describing concurrent interactions, the use of influences and influence laws permits describing the properties of actions, dependencies and features in isolation, thereby supporting modularity. The major merit of TAL-C is that it combines (a) a standard first-order semantics and proof theory; (b) a notion of explicit time, which makes it possible to reason about the durations and timing of actions and effects; and (c) modeling of a number of important phenomena related to concurrency, in addition to ramification and nondeterminism.

1.2 The two language levels of TAL-C

TAL-C, like its predecessors, is a formalism that consists of two languages. First, there is the surface language $\mathcal{L}(SD)$ which provides a number of macros that make it possible to describe TAL-C scenarios in a concise way. Scenarios in $\mathcal{L}(SD)$ are translated to the standard first-order language $\mathcal{L}(FL)$ by expanding these macros, and standard first-order deduction can then be used for reasoning about the scenarios. $\mathcal{L}(SD)$ is used throughout most of this paper; the translation to $\mathcal{L}(FL)$ is presented in appendix A. Therefore, some definitions are given preliminary presentations in $\mathcal{L}(SD)$ but are actually encoded as macro expansions in the translation to $\mathcal{L}(FL)$.

1.3 Organization of the paper

The rest of the paper is organized as follows. In section 2, TAL is introduced with an example and a definition of the syntax of the surface language $\mathcal{L}(SD)$. In section 3, a number of interesting cases of concurrency are identified and discussed. Section 4 outlines an approach to concurrency and introduces the concept of influences, and section 5 introduces the extensions to TAL that make TAL-C. In section 6, the means for solving the problems identified in section 3 are developed. Section 7 addresses modularity in the context of TAL-C. Section 8 provides an overview of previous work on concurrency, and section 9 contains some conclusions. Finally, appendix A presents a translation from the surface language $\mathcal{L}(SD)$ to the first-order language $\mathcal{L}(FL)$.

2 Preliminaries

In TAL, the world consists of objects, features that have time-variant values, and actions that can be executed by agents and that affect the values of features over time. A TAL scenario is a structured collection of statements referring to the dynamics of the world in terms of action laws and dependency laws, action occurrences and their timing, and observations of feature values at different time-points. A scenario encodes a specific course of actions, and therefore TAL is a narrative-based formalism (other examples are Event Calculus [23, 19] and Allen's logic [2]; see also Karlsson [20] on representing TAL narratives as first-order objects). The surface language $\mathcal{L}(SD)$ for sequential scenarios is presented in this section, and extensions for concurrency are introduced in section 5.

2.1 Scenarios in TAL

The following is a scenario in $\mathcal{L}(SD)$. It describes a world with two types of actions (LightFire and PourWater), and a number of agents (bill and bob) and other objects (wood1). For notational convenience, all variables appearing free are implicitly universally quantified.

$$\begin{array}{ll} \text{acs1} & [s,t]\mathsf{LightFire}(a,x) \Rightarrow \\ & \quad ([s]\mathsf{dry}(x) \wedge \mathsf{wood}(x) \Rightarrow R((s,t]\mathsf{fire}(x))) \\ \text{acs2} & [s,t]\mathsf{PourWater}(a,x) \Rightarrow (R([s,t]\neg\mathsf{dry}(x))\wedge \\ & \quad ([s]\mathsf{fire}(x) \Rightarrow R((s,t]\neg\mathsf{fire}(x)))\,) \\ \text{obs1} & [0]\mathsf{dry}(\mathsf{wood1}) \wedge \neg\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1}) \\ \text{occ1} & [2,5]\mathsf{LightFire}(\mathsf{bill},\mathsf{wood1}) \\ \text{occ2} & [6,7]\mathsf{PourWater}(\mathsf{bob},\mathsf{wood1}) \end{array} \tag{1}$$

Notice that all lines are labeled. The labeling reflects the structure of the scenario; different types of statements serve different purposes.

Acs1 and *acs2* are action laws, which describe the effects of specific action types under different conditions. The first action law states that if an agent a lights a fire using some wood x, and if the wood is dry, then the result will be that the wood is on fire. The expression $[s,t]\mathsf{LightFire}(a,x)$ denotes the action in question where $[s,t]$ is its time interval, and $[s]\mathsf{dry}(x) \land \mathsf{wood}(x)$ denotes that the features $\mathsf{dry}(x)$ and $\mathsf{wood}(x)$ hold at time-point s. The statement $R((s,t]\mathsf{fire}(x))$ denotes that the feature $\mathsf{fire}(x)$ is reassigned to become true somewhere in the interval $(s,t]$, and in particular that it is true at the last time-point t of the interval.[1] The terms s and t are time-point variables, and are assumed to be universally quantified (as are a and x). The second action law states that if somebody pours water on an object, then the object will no longer be dry, and will cease being on fire.

Obs1 is an observation statement. It states that the wood (denoted $\mathsf{wood1}$) is dry and not burning at the initial time-point 0. Observations are assumed to be correct, and can refer to arbitrary time-points or intervals; in the latter case, the notation $[\tau,\tau']\phi$ is used (e.g. $[0,6]\mathsf{dry}(\mathsf{wood1})$). *Occ1* and *occ2* are occurrence statements. They describe what actions actually occur in a scenario. A fire is lit by the agent bill during the temporal interval $[2,5]$, and then the agent bob pours water on the wood during the temporal interval $[6,7]$. No actions besides those explicitly appearing in the occurrence statements are assumed to occur in the scenario. By using non-numerical temporal constants, such as $[\mathsf{s}_1,\mathsf{t}_1]\mathsf{LightFire}(\mathsf{bill},\mathsf{wood1})$, the exact timing of an action can be left unspecified.

It is possible for features to have domains other than boolean truth values. In that case, the notation $\phi\hat{=}\omega$ is used to state that the feature ϕ has the value ω, like in the statement $[5]\mathsf{traffic\text{-}light}\hat{=}\mathsf{green}$. The same notation is applicable (but optional) to booleans, e.g. $[s]\mathsf{dry}(x)\hat{=}\mathsf{T}$.

It is commonly recognized that in domains where there are more complex dependencies between features, specifying all possible direct and indirect effects of an action is not feasible. The problem of specifying all effects of actions in a compact manner is called the ramification problem [15]. In TAL, dependency laws are used to deal with this problem (for other approaches to ramification, see e.g. [31, 44, 32]). With dependency laws, one can specify general relations between features once, rather than having to repeat them in each relevant action law.[2] The following are two examples that could complement the description of the fire lighting action in (1). *Dep1* states that if an object starts burning, then it also starts smoking, and *dep2* states that

[1]Previously, the notation $[s,t]\mathsf{fire}(x) := T$ has been used for reassignment [41]. However, in order to be coherent with the notation for the additional operations on features that are introduced in this paper, $R((s,t]\mathsf{fire}(x))$ has been preferred.

[2]Note that there are restrictions as to how dependency laws can be combined in TAL, in order to avoid cycles of instantaneous dependencies that can result in "spontaneous triggering" of dependency laws. Theories that do not contain any dependency cycles are called stratified (see [16]).

if an object starts smoking and the damper is closed (or the object is smoking and the damper becomes closed) then the agents' eyes become sore. The C_T operator denotes that the expression inside was false at the previous time point and has just become true: $C_T([t]\alpha) =_{def} (\forall t'[t = t'+1 \Rightarrow [t']\neg\alpha] \wedge [t]\alpha)$. Again, free variables are implicitly universally quantified.

$$\begin{array}{ll} \text{dep1} & C_T([t]\mathsf{fire}(x)) \Rightarrow R([t]\mathsf{smoking}(x)) \\ \text{dep2} & C_T([t]\mathsf{smoking}(x) \wedge \neg\mathsf{damper\text{-}open}) \Rightarrow R([t]\mathsf{eyes\text{-}sore}(a)) \end{array} \tag{2}$$

Reassignment (R) plays an important role in the solution to the frame problem [34] in TAL. Reassignment expresses change, and unless a feature is involved in reassignment, it is assumed not to change. The reassignment operator is defined as follows,[3]

$$\begin{array}{l} R((\tau,\tau']\alpha) =_{def} (X((\tau,\tau']\alpha) \wedge [\tau']\alpha) \\ R([\tau]\alpha) =_{def} (X([\tau]\alpha) \wedge [\tau]\alpha). \end{array} \tag{3}$$

X is an operator that represents "occlusion" of the features in α, whereas the right-most parts of the definitions denote that α holds at the end of the interval. Occlusion represents an exception from the general principle of persistence, so features that are occluded are allowed to change from one time-point to the next. By minimizing occlusion, the time-points where a specific feature can change value are restricted to those time-points where the feature in question explicitly appears in a reassignment. The fact that features normally do not change is encoded in the no-change axiom below. It states that unless the feature f is occluded at time-point $t+1$, it must have the same value at $t+1$ as at time-point t.[4]

$$\forall t, f, v[\neg X([t+1]f) \Rightarrow ([t]f \hat{=} v \equiv [t+1]f \hat{=} v)] \tag{4}$$

To illustrate how one can reason in TAL, consider the scenario in (1). From 0 to 2, there is no reassignment and therefore no occlusion, so one can with the aid of (4) infer

$$[0,2]\mathsf{dry}(\mathsf{wood1}) \wedge \neg\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1}).$$

From lines *acs1* and *occ1*, it follows that

$$[2]\mathsf{dry}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1}) \Rightarrow R((2,5]\mathsf{fire}(\mathsf{wood1}))$$

holds, which yields $[5]\mathsf{fire}(\mathsf{wood1})$. As the two other features are not occluded, due to (4) one has that

$$[3,5]\mathsf{dry}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1})$$

[3]The following definitions are actually encoded in the translation process from $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$ in appendix A. The $\mathcal{L}(SD)$ versions in this section are preliminary.

[4]This version of the axiom is preliminary. See appendix A.

holds, and then that

$$[6]\mathsf{dry}(\mathsf{wood1}) \wedge \mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1})$$

holds. From *acs2* and *occ2*, it follows that

$$R((6,7]\neg\mathsf{dry}(\mathsf{wood1})) \wedge R((6,7]\neg\mathsf{fire}(\mathsf{wood1}))$$

holds, which yields

$$[7]\neg\mathsf{dry}(\mathsf{wood1}) \wedge \neg\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1}).$$

Finally, as nothing happens after 7,

$$[t]\neg\mathsf{dry}(\mathsf{wood1}) \wedge \neg\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1})$$

holds for all $t \geq 7$.

2.2 The language $\mathcal{L}(SD)$

This section defines the surface language $\mathcal{L}(SD)$ for sequential scenarios. Extensions for concurrency are introduced in section 5, and the translation to the first-order language $\mathcal{L}(FL)$ is presented in appendix A. We use the overline as abbreviation of a sequence, when the contents of the sequence is obvious. For example, $f(\overline{x}, \overline{y})$ means $f(x_1, ..., x_n, y_1, ..., y_m)$.

Definition 1 (vocabulary) A TAL vocabulary $\nu = \langle C, F, A, T, V, R, S, \sigma \rangle$ is a tuple where C is a set of constant symbols, F is a set of feature symbols, A is a set of action symbols, T is a set of temporal function symbols, V is a set of value function symbols, R is a set of relation symbols, S is a set of basic sorts and σ is a function that maps each member of these symbol sets to a sort declaration of the form $\mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ or $\mathcal{S}_1 \times \ldots \times \mathcal{S}_n \rightarrow \mathcal{S}_{n+1}$ where $\mathcal{S}_i \in S$.

Definition 2 (basic sorts) There are a number of sorts for values $\mathcal{V}_i$, including the boolean sort $\mathcal{B}$ with the constants $\{\mathsf{T}, \mathsf{F}\}$. There are a number of sorts for features $\mathcal{F}_i$, each one associated with a value domain $dom(\mathcal{F}_i) = \mathcal{V}_j$ for some j, a sort for actions $\mathcal{A}$, and a temporal sort $\mathcal{T}$.

$\mathcal{T}$ is assumed to be an interpreted sort, but can be axiomatized in first-order logic as a subset of Presburger arithmetic [22] (natural numbers with addition).

Definition 3 (terms) A *value term* ω is a variable v or a constant v of sort $\mathcal{V}_i$ for some i, or an expression $\mathsf{g}(\omega_1, \ldots, \omega_n)$ where $\mathsf{g} : \mathcal{V}_{k_1} \times \ldots \times \mathcal{V}_{k_n} \rightarrow \mathcal{V}_i$ is a value function symbol and each ω_j is of sort $\mathcal{V}_{k_j}$. A *temporal term* τ is a variable t or a constant $0, 1, 2, 3, \ldots$ or $\mathsf{s}_1, \mathsf{t}_1, \ldots$, or $\tau_1 + \tau_2$, all of sort $\mathcal{T}$. A fluent term ϕ is a feature variable f or an expression $\mathsf{f}(\omega_1, \ldots, \omega_n)$ where $\mathsf{f} : \mathcal{V}_{k_1} \times \ldots \times \mathcal{V}_{k_n} \rightarrow \mathcal{F}_i$ is a feature symbol and each ω_j is of sort $\mathcal{V}_{k_j}$.

Definition 4 (temporal and value formulae) If τ, τ' are temporal terms, then $\tau = \tau'$, $\tau < \tau'$ and $\tau \leq \tau'$ are *temporal formulae.* A *value formula* is of the form $\omega = \omega'$ where ω and ω' are value terms, or $\mathsf{r}(\omega_1, \ldots, \omega_n)$ where $\mathsf{r} : \mathcal{V}_{k_1} \times \ldots \times \mathcal{V}_{k_n}$ is a relation symbol and each ω_j is of sort $\mathcal{V}_{k_j}$.

Definition 5 (fluent formula) An *elementary fluent formula* has the form $\phi \hat{=} \omega$ where ϕ is a fluent term of sort $\mathcal{F}_i$ and ω is a value term of sort $dom(\mathcal{F}_i)$. A *fluent formula* is an elementary fluent formula or a combination of fluent formulae formed with the standard logical connectives and quantifiers.

The elementary fluent formula $\phi \hat{=} \mathsf{T}$ can be abbreviated ϕ.

Definition 6 (timed formulae) Let τ, τ' be temporal terms and α a fluent formula. Then $[\tau, \tau']\alpha$, $(\tau, \tau']\alpha$ and $[\tau]\alpha$ are *fixed fluent formulae*, $C_T([\tau]\alpha)$ is a *becomes formula*, $R((\tau, \tau']\alpha)$, $R([\tau, \tau']\alpha)$ and $R([\tau]\alpha)$ are *reassignment formulae*, and $X((\tau, \tau']\alpha)$, $X([\tau, \tau']\alpha)$ and $X([\tau]\alpha)$ are *occlude formulae.*

Definition 7 (static formula) A logical combination (including quantifiers) of temporal and value formulae, fixed fluent formulae and/or becomes formulae is called a *static formula.*

Definition 8 (change formula) A *change formula* is a formula that has (or is rewritable to) the form $\mathcal{Q}\overline{v}(\alpha_1 \vee \ldots \vee \alpha_n)$ where $\mathcal{Q}\overline{v}$ is a sequence of quantifiers with variables, and each α_i is a conjunction of static, occlusion and reassignment formulae. The change formula is called *balanced* iff the following two conditions hold. (a) Whenever a feature $f(\overline{\omega})$ appears inside a reassignment or occlusion formula in one of the α_i disjuncts, then it must also appear in all other α_i's inside a reassignment or occlusion formula with exactly the same temporal argument. (b) Any existentially quantified variable v in the formula, whenever appearing inside a reassignment or occlusion formula, only does so in the position $\phi \hat{=} v$.

Definition 9 (application formula) An *application formula* is any of the following: (a) a balanced change formula; (b) $\Lambda \Rightarrow \Delta$, where Λ is a static formula and Δ is a balanced change formula; or (c) a combination of elements of types (a) and (b) formed with $\wedge$ and $\forall$.

Definition 10 (occurrence formula) An *occurrence formula* has the form $[\tau, \tau']\Phi(\overline{\omega})$, where τ and τ' are elementary time-point expressions, Φ is an action name of sort $\mathcal{V}_1 \times \ldots \times \mathcal{V}_n \rightarrow \mathcal{A}$ and the value terms in $\overline{\omega}$ are of matching sorts.

Definition 11 (scenario components) An action law (labeled *acs*) has the form $\forall t, t', \overline{x}, \overline{y}[\,[t, t']\Phi(\overline{x}) \Rightarrow \Psi(\overline{x}, \overline{y})]$ where $[t, t']\Phi(\overline{x})$ is an occurrence formula and $\Psi(\overline{x}, \overline{y})$ is an application formula. A dependency law (labeled *dep*) has the form $\forall t, \overline{x}[\Psi(\overline{x})]$ where $\Psi(\overline{x})$ is an application formula. An observation (labeled *obs*) is a static formula. An occurrence (labeled *occ*) is an occurrence formula $[\tau, \tau']\Phi(\overline{\omega})$ where τ, τ', $\overline{\omega}$ all are variable-free terms.

3 Variations on the concurrency theme

In this section, we release the sequentiality assumption from the previous section and identify a number of issues that a language for scenarios with concurrency should be able to handle. We observe that TAL is sufficient for handling some of these issues, namely action duration and concurrent execution of independent actions. When it comes to interacting actions, we observe that TAL is not sufficient in a number of cases. This observation forms the basis for the discussion on how to extend TAL to handle concurrency in the subsequent sections. Notice that although we address mainly actions, the discussion applies also to dependencies.

In many domains, it is a fact that actions take time. As long as actions occur sequentially, the only way actions can interact is when the effects of one action affect the context in which a later action is executed. Therefore, it can make sense to abstract away action durations in the case of sequentiality, as is done in for instance basic situation calculus [38]. However, in the case of concurrency, the durations of actions are important for a number of reasons. First, the way the durations of two or more actions overlap can determine how they interact. Second, an action can overlap with two or more other actions without these latter actions overlapping. Third, what happens in the duration of an action can be important for how it interacts with other concurrent actions. TAL has explicit time and actions in TAL have duration.

Concurrent actions can be independent, and involve disjoint sets of features. In this case, the combined effect of the concurrent actions is simply the union of the individual effects, as in the example below.

$$\begin{array}{ll}
\text{acs1} & [s,t]\mathsf{LightFire}(a,x) \Rightarrow \\
 & \quad ([s]\mathsf{dry}(x) \wedge \mathsf{wood}(x) \Rightarrow R((s,t]\mathsf{fire}(x))) \\
\text{acs2} & [s,t]\mathsf{PourWater}(a,x) \Rightarrow \\
 & \quad R((s,t]\neg\mathsf{dry}(x)) \wedge R((s,t]\neg\mathsf{fire}(x)) \\
\text{obs1} & [0]\mathsf{dry}(\mathsf{wood1}) \wedge \neg\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1}) \\
\text{obs2} & [0]\mathsf{dry}(\mathsf{wood2}) \wedge \neg\mathsf{fire}(\mathsf{wood2}) \wedge \mathsf{wood}(\mathsf{wood2}) \\
\text{occ1} & [2,7]\mathsf{LightFire}(\mathsf{bill},\mathsf{wood1}) \\
\text{occ2} & [2,7]\mathsf{LightFire}(\mathsf{bob},\mathsf{wood2})
\end{array} \tag{5}$$

Here TAL yields the conclusion $[7](\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{fire}(\mathsf{wood2}))$ as intended. Concurrency of independent actions does not pose a problem for TAL [45], nor should it for most other formalisms that do not rely on some kind of explicit frame axioms. The difficult problems arise when concurrent actions are not independent.

We address three different problems related to concurrent execution of interdependent actions. The first problem is due to the fact that the conditions under which an action is executed are not always stable, but may be altered by the effects of other concurrent actions. Consider a slight modification of (5), where bob pours water on the fire wood while bill is lighting the fire. The intuitive conclusion is that the wood should not be on fire at 7. We formalize this scenario in TAL by modifying some lines in the scenario (5) above.[5]

$$\begin{array}{ll} \text{occ1} & [2,6]\mathsf{LightFire}(\mathsf{bill},\mathsf{wood1}) \\ \text{occ2} & [3,5]\mathsf{PourWater}(\mathsf{bob},\mathsf{wood1}) \end{array} \tag{6}$$

The modified scenario allows us to infer that the wood is actually on fire: $[7]\mathsf{fire}(\mathsf{wood1})$. The reason is that the effect of the $\mathsf{LightFire}(\mathsf{bill},\mathsf{wood1})$ action is determined only by the state at time-point 2 whereas the wood does not become wet until time-point 5. Thus, just referring to the starting state in preconditions in action laws is apparently not sufficient. One needs to take into account that the conditions under which the action is executed may be altered by the direct and indirect effects of other actions while the action is going on.

The effects of one action on the conditions of another action need not always be harmful. Often, the execution of one action can enable the successful execution of another simultaneous action. For instance, turning the latch of a door might enable opening the door. Sometimes, the enabling is mutual, and two (or more) actions have synergistic effects.

A slight modification of the scenario above illustrates the second problem, which is due to the way effects of actions are represented with reassignment. Assume that the two last lines in (5) are replaced with the following.

$$\begin{array}{ll} \text{occ1} & [3,7]\mathsf{LightFire}(\mathsf{bill},\mathsf{wood1}) \\ \text{occ2} & [3,7]\mathsf{PourWater}(\mathsf{bob},\mathsf{wood1}) \end{array} \tag{7}$$

That is to say, the lighting and the pouring actions have the same duration. Now, from *acs1* and *occ1* one can infer the effect $[7]\mathsf{fire}(\mathsf{wood1})$ and from *acs2* and *occ2* one can infer the effect $[7]\neg\mathsf{fire}(\mathsf{wood1})$. Notice that these two effects are both asserted to be direct and indefeasible. Thus, the scenario becomes inconsistent. The conclusion one would like to obtain is again that the wood is not on fire.

A variation of effect interaction is when several actions affect the same feature in a cumulative way, that is when the total effect of the actions is an aggregate of the individual effects. Reassignment represents changes to absolute values (although these values can be calculated relative to other values), and obviously, aggregation of absolute values is not very meaningful. For instance, consider the following scenario with a box of coins.

$$\begin{array}{ll} \text{acs1} & [s,t]\mathsf{TakeCoin}(a,b) \Rightarrow \\ & \quad [s]\mathsf{coins}(b)\hat{=}(n+1) \Rightarrow R([s,t]\mathsf{coins}(b)\hat{=}n) \\ \text{obs1} & [0]\mathsf{coins}(\mathsf{box1})\hat{=}2 \\ \text{occ1} & [2,3]\mathsf{TakeCoin}(\mathsf{bill},\mathsf{box1}) \\ \text{occ2} & [2,3]\mathsf{TakeCoin}(\mathsf{bob},\mathsf{box1}) \end{array} \tag{8}$$

[5]We only present the modified or added lines in the subsequent scenarios.

Both *occ1* and *occ2* produce the effect $R((2,3]\mathsf{coins}(\mathsf{box1})\hat{=}1)$. Notice that this effect states that $\mathsf{coins}(\mathsf{box1})$ has the absolute value 1, and not that $\mathsf{coins}(\mathsf{box1})$ changes by 1. Therefore, TAL yields $[3]\mathsf{coins}(\mathsf{box1})\hat{=}1$, while the intuitive conclusion is that there are 0 coins in the box at time-point 3.

The third problem is that the conditions for two concurrent actions might interfere. In particular, actions might compete for such things as space, objects and energy. For instance, if lighting a fire requires the use of two hands, then a two-handed agent is not able to light two fires concurrently. One way of addressing this type of conflicts is in terms of limited resources.

4 From action laws to laws of interaction

Based on the observations in the previous section, we argue that the way action laws are formulated in TAL (and many other formalisms) is not appropriate in the case of concurrency. In this section, two potential solutions are discussed. The first solution is to deal with interactions on the level of actions by allowing more expressive action laws. The second solution is to deal with interactions on the level of effects and features, by expressing effects in a less direct and absolute manner than direct reassignment of a feature. One can also imagine numerous combined approaches, for instance where conflicts can be detected on the level of features and then resolved on the level of actions, but that will not be discussed here.

4.1 Interactions on the level of actions

As the description of the effects of actions is usually centered around actions in most formalisms, it might seem like an obvious approach to also deal with concurrent interactions on the level of actions. This approach can be realized with action laws that refer to combinations of action occurrences, as in the work of Baral and Gelfond [3], Li and Pereira [27], Bornscheuer and Thielscher [4], and Reiter [39]. The following action laws, encoded in a hypothetical extended version of TAL, illustrate how one can overcome the problems in (6) and (7). Observe how *acs1* cancels the effect of $\mathsf{LightFire}$ in the presence of $\mathsf{PourWater}$.

$$\begin{array}{ll} \text{acs1} & [s,t]\mathsf{LightFire}(a,x)\wedge \\ & \neg\exists a',s',t'[(s \le s' < t \vee s < t' \le t) \wedge [s',t']\mathsf{PourWater}(a',x)] \Rightarrow \\ & \quad ([s]\mathsf{dry}(x) \wedge \mathsf{wood}(x) \Rightarrow R((s,t]\neg\mathsf{fire}(x))) \\ \text{acs2} & [s,t]\mathsf{PourWater}(a,x) \Rightarrow \\ & \quad R((s,t]\neg\mathsf{dry}(x)) \wedge R((s,t]\neg\mathsf{fire}(x)) \end{array} \tag{9}$$

Unfortunately, this solution has a number of weaknesses. It is no longer possible to describe actions in isolation, which weakens the case for modularity in action descriptions. All potentially interacting action combinations have to be identified and explicitly put into action laws. If actions have duration, the number of such combinations is not just determined by the number of actions, but also by the number of ways two or more actions can overlap in time.

Other factors also contribute to additional complexity. If an action has several effects, then one might not want an interference regarding just one feature to neutralize all the effects of the action. Furthermore, if the action interfered with starts before the interfering action, then the parts of the effects of the former action that are defined to occur before the starting time of the latter action should not be prevented, as this would imply causality working backwards in time.

An additional problem is that interactions are not confined to occur exclusively between the direct effects of actions, but might also involve indirect effects. Taking into consideration all potential interactions between combinations of actions and dependency laws would simply not be feasible for most nontrivial domains. Further, this would multiply all the previously mentioned complications.

4.2 Interaction on the level of features

As an alternative to an action-centered approach, we propose an approach based on the assumption that interactions resulting from concurrency are best modeled on the level of features, and not on the level of actions. The central ideas are as follows.

1. Actions provide an interface between the agent and the environment.

2. An action law does not explicitly encode the immediate effects that the action has on the state of the world. Instead, action laws encode what *influences* the action brings upon the environment. For instance, instead of stating $R((s,t]\mathsf{fire}(x))$ as an effect of the action $[s,t]\mathsf{LightFire}(a,x)$, one states $I((s,t]\mathsf{fire}^*(x,\mathsf{T}))$ where $\mathsf{fire}^*(x,\mathsf{T})$ represents an influence to make the feature $\mathsf{fire}(x)$ true (the I operator is similar to R, but denotes that the expression inside is true throughout the interval). An influence represents an inclination for a feature to take on a certain value or to change in a certain direction. Thus, it is more correct to consider action occurrence statements as representing action attempts that might fail to have their expected effects due to external interference rather than representing successfully executed actions.

3. Similarly, dependencies are modified to result in influences rather than actual change.

4. The actual effects that these influences (and indirectly the actions than caused them) have on the environment are then specified in a special type of dependency laws called influence laws. For instance,

$[t]\mathsf{fire}^*(x, \mathsf{T}) \Rightarrow R([t]\mathsf{fire}(x))$. Generally, each feature is associated with a number of different influences. The behavior of a feature can be specified in a number of influence laws that describe how this feature is affected by different individual influences and combinations of influences. Thus, descriptions of features and how they change due to influences play a central role in TAL-C.

This approach implies that the emphasis of the world description has shifted from actions to features, and from action laws to influence laws. Further, it can be realized with a minimum of modifications of the TAL language, and in particular the first-order nature of TAL can be retained (see appendix A). Influence laws have the same form as dependency laws, which are already an integral part of the language, and influences can actually be represented as features. The difference between influences and other features is purely conceptual; no new syntactic or semantic constructions particular to influences are required (although some new constructs applicable to features in general will be introduced). The term "actual features" will be reserved for features that are not influences.

Notice that the use of influences as intermediaries of change serves two purposes. First, it makes it possible to avoid logical contradiction when two or more actions and dependencies affect the same feature. For instance, the combination $[5]\mathsf{fire}^*(\mathsf{wood1}, \mathsf{T})$ and $[5]\mathsf{fire}^*(\mathsf{wood1}, \mathsf{F})$ is logically consistent, but the combination $[5]\mathsf{fire}(\mathsf{wood1})$ and $[5]\neg\mathsf{fire}(\mathsf{wood1})$ is not. But there is more to concurrent interactions than preserving consistency. The actual outcomes of interactions need to be represented somehow, and a rich phenomenon like concurrent interactions requires a flexible representation that goes beyond the most stereotypical cases. This is supported in TAL-C by the fact that influences are first-order objects, which can be referred to in influence laws.

The use of influences in TAL-C has some resemblance to the way physical systems are often modeled. For instance, in mechanics the position/speed of a physical body is influenced by forces, and in hydraulics the levels of/flows between tanks are influenced by pressures. Notice that in physics, influences often behave cumulatively. For instance, several forces can influence an object in a mechanical system at the same time, and these forces can be aggregated using vector addition. The term "influences" is explicitly used in qualitative reasoning about physical systems by (among others) Forbus [10]. For instance, in Forbus's qualitative process theory, the expressions I+(amount-of(dest),A[flow-rate]) and I-(amount-of(source),A[flow-rate]) denote that the flow rate between two interconnected tanks influences the amount of liquid in the tanks to increase respectively decrease. If a tank t has several pipes connected, then amount-of(t) will be subject to several influences. The flow rate is in turn proportional to the difference of pressure in the two tanks (flow-rate $\propto_{Q+}$ A[pressure(src)]$-$A[pressure(dest)]). Yet, the use of influences in TAL-C is not identical to the use in physical modeling. Although influences surely can be used in TAL-C in ways compatible with quantitative and qualitative physical modeling, they need not always faithfully represent actual physical entities or behave cumulatively.

5 Extending TAL to TAL-C

This section presents the differences between TAL and TAL-C. These include a new class of features, a new effect operator I and two new classes of scenario statements.

5.1 Persistent and durational features

In reasoning about action and change, features are typically considered to be persistent. That means that they only change under special conditions, such as during the execution of an action. In order to facilitate the use of influences, TAL-C is in addition equipped with a second type of features called durational features. These normally have a default value, and they can only have a non-default value under certain circumstances, such as during the execution of an action.[6] The predicates $Per(f)$ and $Dur(f,v)$ represent that a feature is persistent respectively durational with default value v. These predicates can be augmented with a temporal argument to support features with variable behavior, such as variable default value. The default behavior of persistent features is defined as follows.[7]

$$\forall t,f,v[Per(f) \Rightarrow (\neg X([t+1]f) \Rightarrow ([t]f \hat{=} v \equiv [t+1]f \hat{=} v))] \tag{10}$$

The default behavior of durational features is defined as follows,

$$\forall t,f,v\,[Dur(f,v) \Rightarrow (\neg X([t]f) \Rightarrow [t]f \hat{=} v)]. \tag{11}$$

Notice that the distinction between persistent and durational features is in principle orthogonal to the distinction between actual features and influences, although in practice actual features are mostly persistent and influences are mostly durational. Naturally, one need not be confined to the two types of features presented here, but they suffice for the purposes of this paper. We should also mention that the distinction between persistent and durational features have proven useful for more than concurrency. In particular, it has proven fruitful for addressing the qualification problem [7].

[6]The representation of features that are only momentarily true has previously been addressed by for instance Lifschitz and Rabinov [30] and Thielscher [43].

[7]Recall that the occlusion operator X denotes that a feature is not subject to its default assumptions at a given time-point. Furthermore, the following definitions are preliminary; the final versions are presented in appendix A.

5.2 Syntactical additions

In addition to the reassignment operator R, we provide a new operator I which is typically (but not necessarily always) used for durational features. It is used to state that something holds over an interval.

$$I((\tau,\tau']\alpha) =_{def} X((\tau,\tau']\alpha) \wedge \forall t(\tau < t \leq \tau' \Rightarrow [t]\alpha) \tag{12}$$

This specifies that the features in α are exempt from their default behaviors and that the formula α is true at all time-points from $\tau + 1$ to τ'.

The definitions of a change formula and a balanced change formula are extended to include durational formulae of the forms $I((\tau,\tau']\alpha)$, $I([\tau,\tau']\alpha)$ and $I([\tau']\alpha)$, and with the same restrictions as for R and X formulae. Two new kinds of scenario statements are introduced: domain formulae (*dom*) that can contain *Per* and *Dur* elements and value formulae, and influence laws (*inf*), which have the same syntax as dependency laws.

5.3 An example

In the following scenario, there is a durational feature fire^* representing influences on the actual feature fire. Henceforth, we will follow the convention of representing the influences on an actual feature $\mathsf{f}(\overline{\omega})$ with $\mathsf{f}^*(\overline{\omega}, v)$, where v is a value in the domain of f.

$$\begin{array}{ll}
\text{dom1} & Per(\mathsf{fire}(x)) \wedge Dur(\mathsf{fire}^*(x,v), \mathsf{F}) \\
\text{acs1} & [s,t]\mathsf{LightFire}(a,x) \Rightarrow I((s,t]\mathsf{fire}^*(x,\mathsf{T})) \\
\text{inf1} & [s,s+3]\mathsf{fire}^*(x,\mathsf{T}) \wedge \neg\mathsf{fire}^*(x,\mathsf{F}) \Rightarrow R([t+3]\mathsf{fire}(x)) \\
\text{inf2} & [s]\mathsf{fire}^*(x,\mathsf{F}) \Rightarrow R([s]\neg\mathsf{fire}(x)) \\
\text{occ1} & [2,6]\mathsf{LightFire(Bob,wood1)}
\end{array} \tag{13}$$

Notice how *acs1* does not immediately cause fire to be true. Instead, it produces an influence to make fire true, using the I operator. How influences on fire then affect fire is described in *inf1* and *inf2*. It takes 4 consecutive time-points to make fire true, while it takes just 1 time-point to make it false. The influence to make fire false always has precedence over the influence to make fire true. As a matter of fact, *inf1* and *inf2* are general enough to handle any conflict that can occur between a group of actions/dependencies that try to make fire both true and false at the same time. Thus, one can in principle add arbitrary action laws and dependency laws influencing fire without any worries that they lead to inconsistency. Of course, it might still be desirable to refine or modify the way conflicts between influences are treated as a domain is elaborated and more actions and dependencies are introduced. In the examples to follow, we will continue to use influences in a manner that permits easy extension of scenarios, although this practice leads to somewhat larger scenarios. This issue is elaborated further in section 7.

6 Variations on the concurrency theme revisited

In section 3, a number of concurrent interactions that our original TAL formalism could not handle were identified. In this section, we show how these interactions can be represented in TAL-C. We should emphasize that although we present specific examples, the techniques employed in this section are applicable to frequently reoccurring classes of interactions.

6.1 Interactions from effects to conditions

Scenario (6) was an example of the effects of one action interfering with the execution of another concurrent action. While Bill was lighting a fire, Bob poured water on the wood. This type of interference can be handled by including a suitable condition in the influence law that makes the fire feature true.

The following two laws state that the fact that the wood is not dry produces an influence $\mathsf{fire}^*(x, \mathsf{F})$ to extinguish the fire (if there is one), and that the influence $\mathsf{fire}^*(x, \mathsf{T})$ for starting the fire has to be applied without interference for an extended period of time to affect the feature $\mathsf{fire}(x)$. The non-interference condition in this case is that $\mathsf{fire}^*(x, \mathsf{F})$ stays false.

$$\begin{array}{ll}
\text{dep1} & [s]\neg\mathsf{dry}(x) \Rightarrow I([s]\mathsf{fire}^*(x,\mathsf{F})) \\
\text{inf1} & [s, s+3]\mathsf{fire}^*(x,\mathsf{T}) \land \neg\mathsf{fire}^*(x,\mathsf{F}) \land \mathsf{wood}(x) \Rightarrow \\
 & \quad R([s+3]\mathsf{fire}(x))
\end{array} \tag{14}$$

Below is the complete modified version of scenario (6). The action laws *acs1* and *acs2* and dependency law *dep1* produce influences, and the effects that these influences have, alone and in combination, are specified in *inf1*, *inf2*, *inf3* and *inf4*.

$$\begin{array}{ll}
\text{dom1} & \mathit{Per}(\mathsf{fire}(x)) \land \mathit{Dur}(\mathsf{fire}^*(x,v),\mathsf{F}) \\
\text{dom2} & \mathit{Per}(\mathsf{dry}(x)) \land \mathit{Dur}(\mathsf{dry}^*(x,v),\mathsf{F}) \\
\text{dom3} & \mathit{Per}(\mathsf{wood}(x)) \\
\text{acs1} & [s,t]\mathsf{LightFire}(a,x) \Rightarrow I((s,t]\mathsf{fire}^*(x,\mathsf{T})) \\
\text{acs2} & [s,t]\mathsf{PourWater}(a,x) \Rightarrow I((s,t]\mathsf{dry}^*(x,\mathsf{F})) \\
\text{dep1} & [s]\neg\mathsf{dry}(x) \Rightarrow I([s]\mathsf{fire}^*(x,\mathsf{F})) \\
\text{inf1} & [s, s+3]\mathsf{fire}^*(x,\mathsf{T}) \land \neg\mathsf{fire}^*(x,\mathsf{F}) \land \mathsf{wood}(x) \Rightarrow \\
 & \quad R([s+3]\mathsf{fire}(x)) \\
\text{inf2} & [s]\mathsf{fire}^*(x,\mathsf{F}) \Rightarrow R([s]\neg\mathsf{fire}(x)) \\
\text{inf3} & [s, s+3]\mathsf{dry}^*(x,\mathsf{T}) \land \neg\mathsf{dry}^*(x,\mathsf{F}) \Rightarrow R([s+3]\mathsf{dry}(x)) \\
\text{inf4} & [s]\mathsf{dry}^*(x,\mathsf{F}) \Rightarrow R([s]\neg\mathsf{dry}(x)) \\
\text{obs1} & [0]\neg\mathsf{fire}(\mathsf{wood1}) \land \mathsf{dry}(\mathsf{wood1}) \land \mathsf{wood}(\mathsf{wood1}) \\
\text{occ1} & [2,6]\mathsf{LightFire}(\mathsf{bill},\ \mathsf{wood1}) \\
\text{occ2} & [3,5]\mathsf{PourWater}(\mathsf{bob},\ \mathsf{wood1})
\end{array} \tag{15}$$

The fact that the wood is not on fire at 7 can be inferred as follows (we provide a first-order proof in appendix A). Due to *occ2* and *acs2*, the condition

$(3,5]\mathsf{dry}^*(\mathsf{wood1},\mathsf{F})$ holds. This condition and *inf4* yield $[4,5]\neg\mathsf{dry}(\mathsf{wood1})$, and as dry is persistent, $[6]\neg\mathsf{dry}(\mathsf{wood1})$ and $[7]\neg\mathsf{dry}(\mathsf{wood1})$. *Dep1* then yields $[7]\mathsf{fire}^*(\mathsf{wood1},\mathsf{F})$. Finally, *inf2* gives $[7]\neg\mathsf{fire}(\mathsf{wood1})$. Notice that although $[3,6]\mathsf{fire}^*(\mathsf{wood1},\mathsf{T})$ holds, the condition $[s,s+3](\mathsf{fire}^*(\mathsf{wood1},\mathsf{T}) \wedge \neg\mathsf{fire}^*(\mathsf{wood1},\mathsf{F}) \wedge \mathsf{wood}(\mathsf{wood1}))$ does not hold for any $s \leq 3$, and this the only condition (in *inf1*) under which $\mathsf{fire}(\mathsf{wood1})$ can become true.

The case when an effect of one action enables the effect of another action can also be handled with conditional influence laws. For instance, the following influence law states that opening a door requires initially keeping the latch open (the example is originally due to Allen [2]).

$$\text{infl} \quad [t]\mathsf{latch\text{-}open} \wedge [t,t+5]\mathsf{open}^*(\mathsf{T}) \Rightarrow R([t+5]\mathsf{open}) \tag{16}$$

A variation of enablement is when the concurrent execution of two or more actions may mutually enable a common effect that none of them could have in isolation. This phenomenon is referred to as synergistic effects. It can also be the case that the concurrent execution of several actions may prevent effects that each of the actions would have in isolation. In TAL-C, this can be achieved with the use of dependency laws. One example which contains both synergistic enablement and prevention is the scenario with a soup bowl standing on a table (the version presented here is an elaboration of the original scenario, which is due to Gelfond, Lifschitz and Rabinov [12]). The table has four sides: l for left, r for right, f for front and b for back. The variables a, x and r represent the agent, the table and a side of the table, respectively. The table can be lifted at any side (*acs1*). The actual feature $\mathsf{lift\text{-}s}(x,r)$ represents that the table x is lifted on side r. If the table is lifted at two opposite sides, then it is lifted from the ground (*dep1*), but if is not lifted at opposite sides, then it is tilted (*dep2*). If there is a soup bowl on the table and the table is tilted, then the soup is spilled (*dep3*). The relation

opp, defined in *dom6*, specifies when two sides are opposite.

$$\begin{array}{ll}
\text{dom1} & Dur(\mathsf{lift\text{-}s}(x,r),\mathsf{F}) \wedge Dur(\mathsf{lift\text{-}s}^*(x,r,v),\mathsf{F}) \\
\text{dom2} & Dur(\mathsf{tilted}(x),\mathsf{F}) \wedge Dur(\mathsf{tilted}^*(x,v),\mathsf{F}) \\
\text{dom3} & Dur(\mathsf{lifted}(x),\mathsf{F}) \wedge Dur(\mathsf{lifted}^*(x,v),\mathsf{F}) \\
\text{dom4} & Per(\mathsf{spilled}(x)) \wedge Dur(\mathsf{spilled}^*(x,v)) \\
\text{dom5} & Per(\mathsf{soup}(x)) \wedge Per(\mathsf{table}(x)) \wedge Per(\mathsf{on}(x,y)) \\
\text{dom6} & \mathsf{opp}(r,r') \equiv \\
 & [\langle r,r'\rangle = \langle \mathsf{l},\mathsf{r}\rangle \vee \langle r,r'\rangle = \langle \mathsf{r},\mathsf{l}\rangle \vee \langle r,r'\rangle = \langle \mathsf{f},\mathsf{b}\rangle \vee \langle r,r'\rangle = \langle \mathsf{b},\mathsf{f}\rangle] \\
\text{acs1} & [s,t]\mathsf{Lift}(a,x,r) \Rightarrow I((s,t]\mathsf{lift\text{-}s}^*(x,r,\mathsf{T})) \\
\text{dep1} & [t](\mathsf{table}(x) \wedge \exists r_1,r_2[\mathsf{lift\text{-}s}(x,r_1) \wedge \mathsf{lift\text{-}s}(x,r_2) \wedge \\
 & \quad \mathsf{opp}(r_1,r_2)]) \Rightarrow I([t]\mathsf{lifted}^*(x,\mathsf{T})) \\
\text{dep2} & [t](\mathsf{table}(x) \wedge \mathsf{lift\text{-}s}(x,r_1) \wedge \neg\exists r_2,r_3[\mathsf{lift\text{-}s}(x,r_2) \wedge \\
 & \quad \mathsf{lift\text{-}s}(x,r_3) \wedge \mathsf{opp}(r_2,r_3)]\,) \Rightarrow I([t]\mathsf{tilted}^*(x,\mathsf{T})) \\
\text{dep3} & [t]\mathsf{tilted}(y) \wedge \mathsf{on}(x,y) \wedge \mathsf{soup}(x) \Rightarrow R([t+1]\mathsf{spilled}^*(x,\mathsf{T})) \\
\text{inf1} & [t]\mathsf{lift\text{-}s}^*(x,r,\mathsf{T}) \Rightarrow I([t]\mathsf{lift\text{-}s}(x,r)) \\
\text{inf2} & [t]\mathsf{tilted}^*(x,\mathsf{T}) \Rightarrow I([t]\mathsf{tilted}(x)) \\
\text{inf3} & [t]\mathsf{spilled}^*(x,\mathsf{T}) \Rightarrow I([t]\mathsf{spilled}(x)) \\
\text{inf4} & [t]\mathsf{lifted}^*(x,\mathsf{T}) \Rightarrow I([t]\mathsf{lifted}(x)) \\
\text{obs1} & [0]\mathsf{table(t1)} \wedge \mathsf{soup(s1)} \wedge \mathsf{on(s1,t1)} \\
\text{occ1} & [3,6]\mathsf{Lift(bill,t1,l)} \\
\text{occ2} & [3,6]\mathsf{Lift(bob,t1,r)}
\end{array} \qquad (17)$$

In this scenario, one can infer from *occ1*, *occ2*, *inf1* and *inf2* that $\mathsf{lift\text{-}s}(\mathsf{t1},\mathsf{l}) \wedge \mathsf{lift\text{-}s}(\mathsf{t1},\mathsf{r})$ holds from 4 to 6. Thus, the condition $\exists r_1,r_2[\mathsf{lift\text{-}s}(x,r_1) \wedge \mathsf{lift\text{-}s}(x,r_2) \wedge \mathsf{opp}(r_1,r_2)]$ is satisfied in this time interval, enabling the effect $\mathsf{lifted(t1)}$ from *dep1* and *inf4*, while preventing the effect $\mathsf{tilted(t1)}$ according to *dep2* and *inf2*.

The table lifting scenario encodes several other potential interactions between lifting actions. If the two lifting actions are not synchronized, the table is tilted and the soup is spilled. For instance, if *occ2* is altered to $[4,7]\mathsf{Lift(bob,t1,r)}$, then one can infer $[4](\mathsf{table(t1)} \wedge \mathsf{lift\text{-}s(t1,\ l)} \wedge \neg\exists r_2,r_3[\mathsf{lift\text{-}s}(\mathsf{t1},r_2) \wedge \mathsf{lift\text{-}s}(\mathsf{t1},r_3) \wedge \mathsf{opp}(r_2,r_3)]\,)$. According to *dep2* and *inf2*, this produces the effect $[4]\mathsf{tilted(t1)}$, and the soup is spilled. Also notice that if the table is lifted from three sides, (for instance, add *occ3* $[4,7]\mathsf{Lift(ben,t1,f)}$ to the scenario) then the condition $\exists r_1,r_2[\mathsf{lift\text{-}s}(x,r_1) \wedge \mathsf{lift\text{-}s}(x,r_2) \wedge \mathsf{opp}(r_1,r_2)]$ is still satisfied, which implies that the table is lifted and not tilted. However, if the only occurrences are *occ1* and *occ3*, then that condition does not hold and the table is tilted and the soup is spilled.

Notice that it is possible to write influence laws that determine directly from the $\mathsf{lift\text{-}s}^*$ influence whether the table is lifted or the soup is spilled. Anyhow, we have preferred to explicitly represent the causal chain from lifting to spilling and tilting, as it makes it easier to extend the scenario to include actions that for instance counter-act the lifting by pressing down a side of the table or that stabilize the bowl.

6.2 Interactions between effects

The previous subsection addressed interactions between effects and conditions of actions and dependencies. As observed in section 3, another problem is when two or more actions or dependencies are affecting the same feature. Such combinations of effects can be conflicting or cumulative.

6.2.1 Conflicting effects

Returning to the fire lighting scenario (15), it can be observed that the use of influence laws in that scenario also solves the problem of conflicting effects that was observed in (7) in section 3. There were two influence laws in (15) that determined the result of conflicting influences on the fire feature:

$$\begin{array}{ll} \text{inf1} & [s, s+3]\mathsf{fire}^*(x, \mathsf{T}) \wedge \neg\mathsf{fire}^*(x, \mathsf{F}) \wedge \mathsf{wood}(x) \Rightarrow \\ & \quad R([s+3]\mathsf{fire}(x)) \\ \text{inf2} & [s]\mathsf{fire}^*(x, \mathsf{F}) \Rightarrow R([s]\neg\mathsf{fire}(x)) \end{array} \tag{18}$$

Now assume that the occurrences in (15) are modified as follows.

$$\begin{array}{ll} \text{occ1} & [2, 6]\mathsf{LightFire(bill,\ wood1)} \\ \text{occ2} & [4, 6]\mathsf{PourWater(bob,\ wood1)} \end{array} \tag{19}$$

In this case, one can infer $(2, 6]\mathsf{fire}^*(\mathsf{wood1}, \mathsf{T})$ which normally has the effect $[6]\mathsf{fire}(\mathsf{wood1})$ (*inf1*). One can also infer $[6]\neg\mathsf{dry}(\mathsf{wood1})$ and $[6]\mathsf{fire}^*(\mathsf{wood1}, \mathsf{F})$, which normally results in $[6]\neg\mathsf{fire}(\mathsf{wood1})$ (*inf2*). The conflict between these two influences is resolved in *inf1* and *inf2*, to the advantage of the latter.

The fire lighting scenario illustrates a conflict involving just two opposite influences. However, there might also be conflicts involving arbitrarily large number of influences. For instance, consider the following scenario where several agents try to pick up the same object. The feature $\mathsf{pos}(x)$ represents the position of an object x, and its value domain of positions includes both locations (e.g. floor) and agents (e.g. bill, bob). An object can only have one position, so when more than one agent is trying to take the object, then there is a conflict. This conflict is resolved in *inf1* in the scenario.

$$\begin{array}{ll} \text{dom1} & \mathit{Dur}(\mathsf{pos}^*(x, a), \mathsf{F}) \wedge \mathit{Per}(\mathsf{pos}(x)) \\ \text{acs1} & [s, t]\mathsf{Pickup}(a, x) \Rightarrow I((s, t]\mathsf{pos}^*(x, a)) \\ \text{inf1} & [t]\mathsf{pos}(x)\hat{=}\mathsf{floor} \wedge \exists p[\,[t+1]\mathsf{pos}^*(x, p)\,] \Rightarrow \\ & \quad \exists p[\,[t+1]\mathsf{pos}^*(x, p) \wedge R([t+1]\mathsf{pos}(x)\hat{=}p)\,] \\ \text{obs1} & [0]\mathsf{pos(wallet)}\hat{=}\mathsf{floor} \\ \text{occ1} & [2, 3]\mathsf{Pickup(bill,wallet)} \\ \text{occ2} & [2, 3]\mathsf{Pickup(bob,wallet)} \end{array} \tag{20}$$

Inf1 states that "if the object x is on the floor and at least one agent is trying to take x then one of the agents who are trying to take x will actually have x". The result in this case is nondeterministic, and this is perhaps the best way to treat conflicts when one lacks detailed information of what the actual result would be. Notice that the consequent of *infl* only changes the value of $\mathsf{pos}(x)$, and not the value of the influence $\mathsf{pos}^*(x,p)$. The $\mathsf{pos}^*(x,p)$ component of the consequent is in effect a filter on what values p that $\mathsf{pos}(x)$ can be reassigned to.

It is equally possible to state that no effect occurs in the case of conflict:

$$\begin{aligned}\text{infl}\quad &([t]\mathsf{pos}(x)\hat{=}\mathsf{floor} \wedge [t+1]\mathsf{pos}^*(x,p)\wedge\\ &\neg\exists p'[\,[t+1]\mathsf{pos}^*(x,p') \wedge p \neq p'\,]) \Rightarrow R([t+1]\mathsf{pos}(x)\hat{=}p)\end{aligned} \tag{21}$$

Finally, some values might be preferred to other values. For instance, we can enhance the picking-up scenario by giving preference to stronger agents, as follows. The relation $\mathsf{stronger}$ encodes a partial ordering on agents based on their relative strength.

$$\begin{aligned}\text{infl}\quad &[t]\mathsf{pos}(x)\hat{=}\mathsf{floor} \wedge \exists p[\,[t+1]\mathsf{pos}^*(x,p)\,] \Rightarrow\\ &\exists p[\,[t+1]\mathsf{pos}^*(x,p) \wedge \neg\exists p'[\,[t+1]\mathsf{pos}^*(x,p') \wedge \mathsf{stronger}(p',p)\,]\wedge\\ &R([t+1]\mathsf{pos}(x)\hat{=}p)\,]\end{aligned} \tag{22}$$

6.2.2 Cumulative effects

Another common phenomenon besides conflicting influences is when influences signify some relative change, and therefore multiple influences can be combined in a cumulative way. We have already mentioned that in mechanics, multiple forces on an object can be combined using vector addition, and the vectorial sum determines changes in the objects speed and position.

Here, we present another example, which involves a box from which agents can take coins. In order to specify cumulative effects, we need to introduce a minimal portion of set theory, including set membership (*in*), the empty set (*empty*) and subtraction of one element from a set (*remove*). A set theory that is sufficient for our purpose is obtained using the following two axioms. The sets contain only features of a specific feature sort, so the axioms have to be restated for different sorts. The variable σ represents sets.

$$\forall \sigma, f, f'[\mathit{in}(f, \mathit{remove}(f',\sigma)) \equiv (\mathit{in}(f,\sigma) \wedge f \neq f')] \tag{23}$$

$$\forall \sigma[\mathit{empty}(\sigma) \equiv \forall f[\neg \mathit{in}(f,\sigma)]\,] \tag{24}$$

Furthermore, we provide the following definition of the sum of feature values over a set of features.

$$\begin{aligned}&\forall t,\sigma,f,m,n[\mathit{in}(f,\sigma) \wedge \mathit{sum}(t,\mathit{remove}(\sigma,f)) = m \wedge [t]f\hat{=}n \Rightarrow\\ &\qquad\qquad \mathit{sum}(t,\sigma) = (m+n)]\\ &\forall t,\sigma[\mathit{empty}(\sigma) \Rightarrow \mathit{sum}(t,\sigma) = 0]\end{aligned} \tag{25}$$

Now we can introduce an influence $\mathsf{coins}^-(a,c)$ with a value domain of natural numbers and default value 0 to represent that an agent a is taking a coin from a container c. In addition, we define the special function $\mathsf{Coins}^{-}(t, c)$ which for a given c represents the set of all $\mathsf{coins}^{-}(a, c)$ with a nonzero value at time-point t.

$$in(\mathsf{coins}^{-}(a, c'), \mathsf{Coins}^{-}(t, c)) \equiv (c = c' \wedge [t]\neg\mathsf{coins}^{-}(a, c) \hat{=} 0) \tag{26}$$

This definition establishes the existence of a set which contains all the non-zero features of the relevant type. The definition of *remove* above then establishes the existence of all subsets of this set, which is sufficient for determining the sum of all features in the set.

The scenario (8) is modified as follows.

dom1	$Dur(\mathsf{coins}^{-}(a, c), 0) \wedge Per(\mathsf{coins}(c))$	(27)
acs1	$[s, t]\mathsf{TakeCoin}(a, c) \Rightarrow I([t]\mathsf{coins}^{-}(a, c) \hat{=} 1)$	
infl	$([t]\mathsf{coins}(c) \hat{=} (m + n) \wedge sum(t{+}1, \mathsf{Coins}^{-}(t + 1, c)) = n) \Rightarrow$ $R([t + 1]\mathsf{coins}(c) \hat{=} m)$	
obs1	$[0]\mathsf{coins}(\mathsf{box1}) \hat{=} 2$	
occ1	$[2, 3]\mathsf{TakeCoin}(\mathsf{bill}, \mathsf{box1})$	
occ2	$[2, 3]\mathsf{TakeCoin}(\mathsf{bob}, \mathsf{box1})$	

The cumulative behavior of the coins feature is encoded in *infl*, which adds together the values of all non-zero $\mathsf{coins}^{-}(a, c)$ for a specific c and adds this sum to $\mathsf{coins}(c)$. From this scenario, one can infer that $\mathsf{Coins}^{-}(3, \mathsf{box1}) = \{\mathsf{coins}^{-}(\mathsf{bill}, \mathsf{box1}), \mathsf{coins}^{-}(\mathsf{bob}, \mathsf{box1})\}$. As $[3]\mathsf{coins}^{-}(\mathsf{bill}, \mathsf{box1}) \hat{=} 1$ and $[3]\mathsf{coins}^{-}(\mathsf{bob}, \mathsf{box1}) \hat{=} 1$ we get $sum(\mathsf{Coins}^{-}(3, \mathsf{box1})) = 2$ and $[3]\mathsf{coins}(\mathsf{box1}) \hat{=} 0$. Obviously, the scenario can be enhanced. For example, more than one coin can be taken by the same agent, coins can be added to the box, and so on.

6.3 Interacting conditions

Finally, there is the problem that the conditions of two or more actions interact. A special case of this is when an agent has a resource that can only be used for one action at a time. For instance, people generally have only two hands, and thus cannot simultaneously perform two actions that each requires the use of both hands, like lighting a fire. A strategy for representing resources is to introduce a feature representing what action the resource is actually used for. In the following scenario, the feature $\mathsf{uses\text{-}hands}(x)$ fulfills this function. There is a value sort of action tokens that duplicates the action sort (e.g. the value $\mathsf{light\text{-}fire}(a, x)$ corresponds to the action $\mathsf{LightFire}(a, x)$). The feature $\mathsf{uses\text{-}hands}(x)$ has action tokens as domain, and the letter e is used for action token variables. The action token noop stands for "no operation".

The use of a resource for an action is divided into two steps. First, the action claims the resource. *Acs1* in the scenario below states that the action of lighting a fire needs the "hands" resource. This need is represented with the influence $\mathsf{uses\text{-}hands}^{*}(a, e)$. Second, the resource is actually used and the action produces some effect. *Dep1* below states that if the hands are used for lighting a fire, then this will produce a fire influence.

$$\begin{array}{ll}
\text{dom1} & Per(\mathsf{fire}(x)) \wedge Dur(\mathsf{fire}^*(x,v),\mathsf{F}) \\
\text{dom2} & Per(\mathsf{wood}(x)) \\
\text{dom3} & Dur(\mathsf{uses\text{-}hands}(a),\mathsf{noop}) \wedge Dur(\mathsf{uses\text{-}hands}^*(a,e),\mathsf{F}) \\
\text{acs1} & [s,t]\mathsf{LightFire}(a,x) \Rightarrow \\
 & \quad I((s,t]\mathsf{uses\text{-}hands}^*(a,\ \mathsf{light\text{-}fire}(a,x))) \\
\text{dep1} & [t]\mathsf{uses\text{-}hands}(a)\hat{=}\mathsf{light\text{-}fire}(a,x) \Rightarrow I([t]\mathsf{fire}^*(x,\mathsf{T})) \\
\text{inf1} & \exists e[\,[t]\mathsf{uses\text{-}hands}^*(a,e)\,] \Rightarrow \\
 & \quad \exists e[\,[t]\mathsf{uses\text{-}hands}^*(a,e) \wedge I([t]\mathsf{uses\text{-}hands}(a)\hat{=}e)\,] \\
\text{inf2} & [s,s+3]\mathsf{fire}^*(x,\mathsf{T}) \wedge \neg\mathsf{fire}^*(x,\mathsf{F}) \wedge \mathsf{wood}(x) \Rightarrow \\
 & \quad R([s+3]\mathsf{fire}(x)) \\
\text{inf3} & [s]\mathsf{fire}^*(x,\mathsf{F}) \Rightarrow R([s]\neg\mathsf{fire}(x)) \\
\text{obs1} & [0]\mathsf{dry}(\mathsf{wood1}) \wedge \neg\mathsf{fire}(\mathsf{wood1}) \wedge \mathsf{wood}(\mathsf{wood1}) \\
\text{occ1} & [2,6]\mathsf{LightFire}(\mathsf{bill},\mathsf{wood1}) \\
\text{occ2} & [2,6]\mathsf{LightFire}(\mathsf{bill},\mathsf{wood2})
\end{array} \tag{28}$$

The distribution of resources is encoded in *inf1*, which states that if at least one action e needs the "hands" resource then some action that needs that resource will have it (compare to *inf1* in (20)). If two or more actions need a resource, then only one of them will have it, and the resource can randomly alter between competing actions. In this example, the value of $\mathsf{uses\text{-}hands}(a)$ alternates randomly between $\mathsf{light\text{-}fire}(\mathsf{bill},\mathsf{wood1})$ and $\mathsf{light\text{-}fire}(\mathsf{bill},\mathsf{wood2})$ from 3 to 6, with the result that both actions fail or only one of them succeeds.

More sophisticated forms of resources than the binary resource above are also possible. For instance, one can utilize the techniques presented in connection with cumulative effects to deal with quantitative and sharable resources, and resources can be renewable or consumable.

6.4 Special vs. general influences

In all examples so far, each feature type[8] has its own set of influences and influence laws. While offering a high level of freedom in handling concurrent interactions, this approach also requires declaring influences for each feature type and writing down a large amount of influence laws. Of course, if one desires a flexible and non-stereotypical treatment of interactions, this is hard to avoid. But TAL-C is also capable of a more uniform and compact treatment of interactions. Instead of declaring separate influences of each actual feature type (e.g. dry^* for dry), one can group together features with the same value domain and the same behavior and let them be of the same feature sort $\mathcal{F}_i$. Next, one introduces a function $* : \mathcal{F}_i \times dom(\mathcal{F}_i) \to \mathcal{F}_j$

[8]We consider all features with the same feature symbol, e.g. fire, to define a type. Several feature types with the same value domain might be of the same sort.

that for a given feature and value represents an influence on the feature to change according to the value (e.g. $*(\mathsf{dry(wood1)},\mathsf{F})$). Thereby, it is possible to specify influence laws that apply to all feature types that are of the sort $\mathcal{F}_i$. The following is an example of a general influence law where $dom(\mathcal{F}_i)$ is the boolean value sort. The variable f (implicitly universally quantified) is of sort $\mathcal{F}_i$. The influence law handles conflicts by making the outcome nondeterministic.

$$\begin{array}{ll} \text{dom} & Dur(*(f,v),\mathsf{F}) \\ \text{inf} & ([t]*(f,\mathsf{T}) \wedge \neg *(f,\mathsf{F}) \Rightarrow R([t]f)) \wedge \\ & ([t]\neg *(f,\mathsf{T}) \wedge *(f,\mathsf{F}) \Rightarrow R([t]\neg f)) \wedge \\ & ([t]*(f,\mathsf{T}) \wedge *(f,\mathsf{F}) \Rightarrow X([t]f)) \end{array} \tag{29}$$

This approach yields a number of influence laws that is proportional to the number of feature sorts, instead of the number of feature types. It can be particularly useful for scenarios with a large number of feature types that exhibit relatively uniform behaviors. In addition, the fact that influence laws are more general and applies to sorts rather than to specific feature types implies a higher degree of reusability.

7 Working with TAL-C scenarios

When encoding a scenario in TAL-C, a bottom-up approach involving the four following levels can be used. (1) Identify relevant features of the world and their value domains. (2) For each feature, determine its normal (non-influenced) behavior, its potential influences and how these affect the feature alone and in combination. (3) Identify actions and dependencies in the world and how these influence features. (4) Determine what holds and occurs in the world, and what individuals there are.

Often, it is not possible to work strictly sequentially from level 1 to level 4. The elaboration of a complex scenario is an iterative and incremental process, where the four levels above are intertwined and decisions made earlier can be reconsidered. Therefore, to estimate how demanding this elaboration process would be in TAL-C, it is relevant to analyze what implications additions or modifications at different levels would have on an existing scenario as a whole. Although there are some initial studies on the subject [33], there exist no systematic methods for performing this kind of estimate. Therefore, the following observations are based mainly on practical experience and common sense.

To provide a background for the discussion to follow, we need to make some additional assumptions about the form of a scenario. We should emphasize that these assumptions represent good conventions that have been followed in this paper, but they are not formally part of the TAL-C definition.

An action law for an action A has the following form, where $\Lambda^A(\overline{x})$ contains only actual features and $\Delta^A(\overline{x})$ contains only influences:

$$\text{acs} \quad [t, t']\mathsf{A}(\overline{x}) \Rightarrow (\Lambda^A(\overline{x}) \Rightarrow \Delta^A(\overline{x})) \tag{30}$$

A dependency law has the following form, with the corresponding restrictions on $\Lambda_k(\overline{x})$ and $\Delta_k(\overline{x})$.

$$\text{dep} \quad \Lambda_k(\overline{x}) \Rightarrow \Delta_k(\overline{x}) \tag{31}$$

A domain statement for a specific feature has one of the two following forms:

$$\begin{array}{ll} \text{dom} & Per(\mathsf{f}(\overline{x})) \\ \text{dom} & Dur(\mathsf{f}(\overline{x}), \boldsymbol{v}) \end{array} \tag{32}$$

Finally, each feature type f has a number of influence laws of the form

$$\text{inf}_i \quad \Delta_i^f(\overline{x}) \Rightarrow \Lambda_i^f(\overline{x}) \tag{33}$$

where the consequent $\Lambda_i^f(\overline{x})$ contains references to no other actual feature but $\mathsf{f}(\overline{x})$ and this feature occurs only inside reassignment, interval and occlude formulae. $\Lambda_i^f(\overline{x})$ may also contain influences that belong to $\mathsf{f}(\overline{x})$, but then only inside static subformulae. The antecedent $\Delta_i^f(\overline{x})$ contains only influences that belong to $\mathsf{f}(\overline{x})$ (e.g. $\mathsf{f}^*(\overline{x}, v)$) and actual features. Each group of influence laws represents a module that describes the behavior of a specific feature f together with the *dom* statement for that feature, and any conflicts or other interactions are handled locally within that module. Finally, we assume that each influence only belongs to one actual feature.

Given the assumptions above, we can draw the following conclusions about the impact an addition/modification will have on a scenario.

Adding a new feature (level 1 according to the enumeration above), does in itself not affect anything else. It is obviously followed by adding new influences and influence laws (level 2) and sometimes also adding/modifying actions and dependencies (level 3). These operations are discussed below.

Adding or altering the default behavior of a feature (level 2) is local to the domain statement specifying the default behavior in question. Adding a new type of influence for a feature implies altering the influence laws for the feature in question, but does not affect the influence laws for other features. If care is taken in the choice of influences, additions of influences should seldom occur, and the types of influences for a given feature should remain more or less constant. As a parallel, the physical property of an object's speed can be influenced by a large amount of imaginable actions and conditions. Yet, one single type of influence ("force") suffices to determine changes in speed. Further, altering the interactions between influences for a particular feature is local to the influence laws (i.e. the module) of that feature, but does not affect the default behavior or any action or dependency laws.

Modifying or adding an action law or dependency law (level 3) does not affect any other existing action or dependency laws as any interactions are delegated to the influence laws, nor does it affect existing influence laws. The exception is of course when the new action or dependency law requires the introduction of a new type of influence for a particular feature, in which case the influence laws of that feature have to be extended. Finally, adding action occurrences and observations (level 4) does not affect anything at the preceding levels.

In summary, additions or modifications are in general local operations which preserve modularity in TAL-C. Different features can be described in isolation, and given a set of features and associated influences, different actions and dependencies can be described in isolation. This property is mainly due to two features of the logic, namely that interactions between actions and dependencies are channeled through influences, and further that the respective sets of influences of different features are disjoint, and therefore the behaviors of features can be specified in normal behavior and influence laws that are independent of those of other features. It is encouraging to achieve this level of modularity, considering the fact that we are addressing complicated causal dependencies and concurrent interactions.

8 Other work on concurrency

Hendrix's work [17] is an early attempt to represent continuous and simultaneous processes, using a STRIPS-like [9] language. Unlike STRIPS, Hendrix's formalism involves notions of explicit time, duration, and simultaneous and extraneous activity. A process has preconditions and continuation conditions that determine when the process can be initiated and for how long it goes on. Hendrix distinguishes between instantaneous effects at the initiation and termination moments of the process and gradual effects that occur while the process is going on. However, there are no means for determining what happens when more than one process affects the same feature ("parameter" in Hendrix's terminology). More sophisticated representations of physical process were later developed in qualitative reasoning, were Forbus [10] has already been mentioned.

In the work of Georgeff [13] there are world states that are linked together in histories. In a world state, features can hold and one or more events (actions) can occur. Specific features can explicitly be declared to be independent of specific events, and a persistence axiom, similar to the nochange axiom in TAL, states that if a feature p is independent of all events in a state, then it will not have changed in the next state. Georgeff then introduces the concept of *correctness conditions*. If the correctness condition p of an event e is independent of another event e', then e' will not interfere with (prevent) e. Thus, Georgeff's formalism can define when two events (actions) can and cannot occur simultaneously, and what the result is when two independent events are executed simultaneously. A limitation is the lack of an explicit notion of duration, so events cannot overlap partially. Georgeff also considers processes, which essentially are related groups of events with limited interaction with events outside the process.

Structural relations between events is the central theme in work by Lansky [26]. In GEM, there is an explicit representation of event location; events can belong to elements, which are loci of forced sequential activity, and which in turn can belong to groups. In essence, groups represent boundaries of causal access. Events inside a group can only interact with external events via specific ports (causal holes). Thereby, the possible concurrent interactions between events can be restricted.

Pelavin's work on a logic for planning with simultaneous actions with duration [35] is based on Allen's interval temporal logic [2], in which properties and actions are associated with intervals of time. Pelavin's formalism has quite a complex non-standard semantics, where the central entities are world histories. A closeness function defines how the addition of actions to a world history results in new world histories. On the syntactic level, there are modal operators on world histories: $IFTRIED(pi, P)$ denoting that the condition P would hold if the actions in pi are executed, and $INEV(i, P)$ stating that the condition P inevitably holds at time i (is independent of anything happening after i). These operators can be used for quite sophisticated descriptions of actions, including interference where one action prevents another, and cumulative effects. However, what does not change due to an action has to be explicitly encoded, and there is no concept of dependency laws.

Thielscher [43] presents a theory of dynamic systems, where state transitions can occur naturally in addition to being caused by actions. Fluents are divided into two sets. There are *persistent fluents*, which are subject to inertia and only change when directly influenced, and there are *momentary fluents* that become false if nothing affects them. A subset of the momentary fluents are the *action fluents*. Causal laws are specified in a STRIPS-style manner, with a precondition, a set of persistent fluents to become true, a set of persistent fluents to become false and a set of momentary fluents to become true in the following state. Thielscher addresses some aspects of concurrency, but a versatile way of handling time is lacking. Durations and delays cannot be easily modeled, due to the STRIPS-style operational nature of the language. The paper also discusses one type of concurrent conflicts the formalism can handle, but no general way to handle other types of concurrent conflicts are mentioned.

Ferber and Müller [8] presents a theory for dynamic multi-agent environments with a distinction between influences and state. The world develops in two-step cycles: there is a set of operators (corresponding to actions and events) that for given influences and conditions on the state yield new influences; and a set of laws that for given conditions on the state and influences transform the state. The state component develops according to a persistence assumption, whereas influences are transient (like persistent respectively durational features in TAL-C). The theory is then augmented with agent behaviors, which are functions from influence sets (percepts) to influence sets (responses). A STRIPS-style operational formalism is used in the paper, but the authors explain that the general principles should apply to other types of formalisms as well.

Among the work done in situation calculus, Pinto's [36] modeling of concurrency is particularly interesting. Pinto addresses the use of resources and exploits state constraints (of a weaker kind than the dependency laws in this paper) to deal with effect interaction, although in the context of instantaneous actions. In a recent paper [37], Pinto proposes an approach where he uses natural events (in the style of Reiter [39]) to model causal dependecies, and then he uses causal dependencies to model concurrent interactions. In this approach, natural events can play a role similar to the one of influences in TAL-C, i.e. as intermediaries of change. What is particularily interesting is that Pinto motivates his approach with arguments for modularity; using causal dependencies to model concurrent interactions on the level of fluents allow us to describe actions in isolation.

Finally, we should also mention Baral and Gelfond's propositional language $\mathcal{A}_C$ [3] and its relatives by Li and Pereira [27] and Bornscheuer and Thielscher [4]. $\mathcal{A}_C$, an extension of Gelfond's and Lifschitz's language $\mathcal{A}$ [11], relies on action rules (e-propositions) of the form $\{A_1, \ldots, A_n\}$ **causes** e **if** $p1_, \ldots, p_n$ to describe concurrent interactions. Rules for the same fluent with more specific action parts override less specific ones. This makes $\mathcal{A}_C$ suitable for representing synergistic and conflicting effects. On the other hand, actual cancellation of effects requires the use of explicit frame axioms, like in the soup bowl example [3] where the rule $\{lift\_left, lift\_right\}$ **causes** $\neg$*spilled* **if** $\neg$*spilled* overrides the rules $\{lift\_left\}$ **causes** *spilled* and $\{lift\_right\}$ **causes** *spilled*. None of [3, 27, 4] address ramification or action duration, and only Bornscheuer's and Thielscher's version addresses nondeterminism. The three languages differ mainly in the treatment of concurrent conflicts that are not resolved by any e-proposition. According to Baral and Gelfond, the entire resulting state is undefined, according to Li and Pereira, the result of the conflicting actions is undefined, and according to Bornscheuer and Thielscher, the resulting value of the conflicting feature is nondeterministic.

The fact that more specific rules override less specific ones in $\mathcal{A}_C$ makes it possible to add new rules without having to modify existing ones, thereby contributing to elaboration tolerance. Compared to TAL-C, is possible to specify scenarios where certain modifications are easier to do in $\mathcal{A}_C$ (and vice versa). However, actions cannot be described in isolation, which has to be considered a major drawback from a modularity perspective. Also recall the discussion in section 4.1.

9 Conclusions

In this paper, we have presented TAL-C, which is a logic for describing action scenarios that involve action concurrency and causal dependencies between features. What distinguishes TAL-C from previous work is the combination of the following factors: (a) TAL-C has a standard first-order semantics and proof theory. (b) TAL-C has a notion of explicit time, which makes it possible to reason about the durations of actions and other interesting temporal properties and relations. (c) TAL-C is able to model a number of important phenomena related to concurrency. We should also mention that several of the examples in this paper have been tested using an implementation of TAL and TAL-C, called VTAL (available on WWW at `http://anton.ida.liu.se/vital/vital.html` as a Java applet).

Technically, TAL-C is closely related to TAL with ramification [16], although the surface language $\mathcal{L}(SD)$ has been modified and extended. In the base language $\mathcal{L}(FL)$, the same predicates are still used, with the addition of Per and Dur. Most important, the same simple circumscription policy is still applicable, which implies that we can reason about concurrent interactions in first-order logic.

The main difference between TAL and TAL-C is conceptual in nature and based on how action laws are defined and used. We have demonstrated how traditional action laws suffer from a number of problems, in particular due to the fact that preconditions in action laws refer to the state before the action is executed, and that the effects are absolute and indefeasible. The solution involving action laws with multiple actions was rejected due to lack of precision and scaling. Instead, we proposed an approach were actions produce influences instead of actual effects. The way these influences change the world, both alone and in interaction with other influences can then be specified in influence laws for individual features. TAL-C has been demonstrated on a number of nontrivial concurrency-related problems. The use of influence laws in TAL-C provides a flexible tool for describing what happens when a feature is subject to influence from several actions or dependencies simultaneously. Additions and modifications to a TAL-C scenario are local operations which preserve modularity.

An important topic for future research is to adopt TAL-C for continuous time and value domains, in order to describe worlds with piece-wise continuous dynamics (for related work of this problem, see e.g. Sandewall [40] and Shanahan [42]). Such an adaptation will probably involve a richer representation of the default behaviors of features than the distinction between persistent and durational features used in this paper. Future research also include systematically exploring common patterns of concurrency and establishing semantics for different classes of worlds with concurrent actions, and studying how locality [26] can be exploited in a formalism such as TAL-C.

Acknowledgements

This work has been supported by the Wallenberg foundation and by the Swedish Research Council for Engineering Sciences (TFR). Many thanks to Silvia Coradeschi, Jonas Kvarnström, Patrick Doherty and Patrik Haslum for useful comments on previous versions.

A Translation from $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

This section presents the translation from the surface language $\mathcal{L}(SD)$, which is intended for describing scenarios, to $\mathcal{L}(FL)$ which is used for inferences. The translation is based on [6]. $\mathcal{L}(FL)$ is a first-order language with equality consisting of the predicates $Holds_i : \mathcal{T} \times \mathcal{F}_i \times Dom(\mathcal{F}_i)$, $Occlude_i : \mathcal{T} \times \mathcal{F}_i$ (normally, the index i is omitted) and $Occurs : \mathcal{T} \times \mathcal{T} \times \mathcal{A}$; further $Per_i : \mathcal{F}_i$, $Dur_i : \mathcal{F}_i \times Dom(\mathcal{F}_i)$ and all predicates relating to the value domains and temporal domain from $\mathcal{L}(SD)$. There is an isomorfism of sorts and symbols between $\mathcal{L}(SD)$ and $\mathcal{L}(FL)$.

A.1 Translation function

Definition 12 *Tran* is called the *translation function*, and is defined as follows (the obvious parts have been left out). All variables occurring only on the right-hand side are assumed to be previously unused variables.

$$
\begin{aligned}
Tran([\tau]f(\overline{\omega})\hat{=}v) &= Holds(\tau, f(\overline{\omega}), v) && (34)\\
Tran([\tau]\neg\alpha) &= \neg Tran([\tau]\alpha) && (35)\\
Tran([\tau]\alpha\mathcal{C}\beta) &= Tran([\tau]\alpha)\ \mathcal{C}\ Tran([\tau]\beta) && (36)\\
&\quad \text{where } \mathcal{C} \in \{\wedge, \vee, \Rightarrow, \equiv\}. &&\\
Tran([\tau]\mathcal{Q}v[\alpha]) &= \mathcal{Q}v[Tran([\tau]\alpha)] \text{ where } \mathcal{Q} \in \{\forall, \exists\}. && (37)\\
Tran([\tau, \tau']\alpha) &= \forall t'[\tau \leq t' \leq \tau' \Rightarrow Tran([t']\alpha)] && (38)\\
Tran((\tau, \tau']\alpha) &= \forall t'[\tau < t' \leq \tau' \Rightarrow Tran([t']\alpha)] && (39)\\
Tran(X([\tau]f(\overline{\omega}))) &= Occlude(\tau, f(\overline{\omega})) && (40)\\
Tran(X([\tau]f(\overline{\omega})\hat{=}v)) &= Occlude(\tau, f(\overline{\omega})) && (41)\\
Tran(X([\tau]\neg\alpha)) &= Tran(X([\tau]\alpha)) && (42)\\
Tran(X([\tau]\alpha\mathcal{C}\beta)) &= Tran(X([\tau]\alpha)) \wedge Tran(X([\tau]\beta)) && (43)\\
&\quad \text{where } \mathcal{C} \in \{\wedge, \vee, \Rightarrow, \equiv\}. &&\\
Tran(X([\tau]\mathcal{Q}v[\alpha])) &= \forall v[Tran(X([\tau]\alpha))] && (44)\\
&\quad \text{where } \mathcal{Q} \in \{\forall, \exists\}. &&
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Tran}(X((\tau,\tau']\alpha)) &= \forall t'[\ \tau < t' \leq \tau' \Rightarrow \\
&\qquad \mathit{Tran}(X([t']\alpha))] && (45)\\
\mathit{Tran}(X([\tau,\tau']\alpha)) &= \forall t'[\ \tau \leq t' \leq \tau' \Rightarrow \\
&\qquad \mathit{Tran}(X([t']\alpha))] && (46)\\
\mathit{Tran}(R((\tau,\tau']\alpha)) &= \mathit{Tran}(X((\tau,\tau']\alpha)) \wedge \mathit{Tran}([\tau]\alpha) && (47)\\
\mathit{Tran}(R([\tau,\tau']\alpha)) &= \mathit{Tran}(X([\tau,\tau']\alpha)) \wedge \mathit{Tran}([\tau]\alpha) && (48)\\
\mathit{Tran}(R([\tau]\alpha)) &= \mathit{Tran}(X([\tau],\alpha)) \wedge \mathit{Tran}([\tau]\alpha) && (49)\\
\mathit{Tran}(I((\tau,\tau']\alpha)) &= \mathit{Tran}(X((\tau,\tau']\alpha)) \wedge \\
&\quad \mathit{Tran}((\tau,\tau']\alpha) && (50)\\
\mathit{Tran}(I([\tau,\tau']\alpha)) &= \mathit{Tran}(X([\tau,\tau']\alpha)) \wedge \\
&\quad \mathit{Tran}([\tau,\tau']\alpha) && (51)\\
\mathit{Tran}(I([\tau]\alpha)) &= \mathit{Tran}(X([\tau],\alpha)) \wedge \mathit{Tran}([\tau]\alpha) && (52)\\
\mathit{Tran}(C_T([\tau]\alpha)) &= \forall t'[\tau = t' + 1 \Rightarrow \mathit{Tran}([t']\neg\alpha)] \wedge \\
&\quad \mathit{Tran}([\tau]\alpha) && (53)\\
\mathit{Tran}([\tau,\tau']\Phi(\overline{\omega})) &= \mathit{Occurs}(\tau,\tau',\Phi(\overline{\omega})) && (54)
\end{aligned}
$$

Notice the translation of the X operator, in particular lines (42–44), which always occludes all features inside a reassignment or interval formula.

A.2 Circumscription

The second-order circumscription of a number of predicates $\overline{P} = P_1, \ldots, P_n$ in the theory $\Gamma(\overline{P})$ is denoted $\mathit{Circ}_{SO}(\Gamma(\overline{P}); \overline{P})$ (see Lifschitz [28]). Intuitively, $\mathit{Circ}_{SO}(\Gamma(\overline{P}); \overline{P})$ represents a (second-order) theory containing $\Gamma(\overline{P})$ and where the extensions of the predicates $\overline{P}$ are minimal.

Definition 13 Transformation of scenarios from $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$:

1. Let *dom*, *acs*, *dep*, *inf*, *obs* and *occ* be the sets of statements with labels *dom*, *acs*, *dep*, *obs* and *occ* respectively, completed with universal quantification for variables occurring free.

2. Let $\Gamma_{dom} = \mathit{Tran}(dom)$, $\Gamma_{acs} = \mathit{Tran}(acs)$, $\Gamma_{dep} = \mathit{Tran}(dep)$, $\Gamma_{inf} = \mathit{Tran}(inf)$, $\Gamma_{obs} = \mathit{Tran}(obs)$ and $\Gamma_{occ} = \mathit{Tran}(occ)$.

3. Let Γ be $\mathit{Circ}_{SO}((\Gamma_{acs} \cup \Gamma_{dep} \cup \Gamma_{inf})(\overline{\mathit{Occlude}}); \overline{\mathit{Occlude}}) \cup \Gamma_{dom}$ $\cup \mathit{Circ}_{SO}(\Gamma_{occ}(\mathit{Occurs}); \mathit{Occurs}) \cup \Gamma_{obs} \cup \Gamma_{fl} \cup \Gamma_{fnd}$. Γ is the theory that is used for proofs in TAL-C.

The set Γ_{fl} contains the $\mathcal{L}(FL)$ equivalents of (10) and (11), plus two more axioms relating to Per and Dur.

$$\begin{aligned}
&\Gamma_{fl} = \bigcup_i \{ \qquad (55)\\
&\forall f_i, t, v_i\, [\, Dur_i(f_i, v_i) \Rightarrow \\
&\qquad\qquad (\neg Occlude_i(t, f_i) \Rightarrow (Holds_i(t, f_i, v_i)))],\\
&\forall f_i, t, v_i [\, Per_i(f_i) \Rightarrow (\neg Occlude_i(t+1, f_i) \Rightarrow \\
&\qquad\qquad (Holds_i(t, f_i, v_i) \equiv Holds_i(t+1, f_i, v_i)))],\\
&\forall f_i, t, v_i, v_i'\; Dur_i(f, v_i) \wedge Dur_i(f, v_i') \Rightarrow v_i = v_i',\\
&\forall f_i [Per_i(f_i) \oplus \exists v_i\, Dur_i(f_i, v_i)]\ \}
\end{aligned}$$

Finally, the set Γ_{fnd} consists of foundational axioms for unique names for actions, features and values, and constraints that a feature has exactly one value at each time-point.

An important property of the circumscribed theory Γ is that although it is a second-order theory due to the second-order nature of circumscription, it can be reduced to an equivalent first-order theory, and in a very convenient form. The following is a principal account for this reduction based on [6], where the proofs in [6] are directly applicable. Due to the definition of action laws and dependency laws, occlusion can only occur on the right-hand side Δ of an implication $\Gamma \Rightarrow \Delta$. Furthermore, due the restrictions on balanced change formulae and the definition of the $Tran$ function, occlusion only occurs positive in Δ, and if it occurs in a disjunction inside Δ, then it occurs identically on both sides. Therefore, the $Occlude_i$ parts, using the law of distributivity, can be separated from the rest of Γ, resulting in $\Lambda \Rightarrow (\bigwedge_i \Delta_i^{occ}) \wedge \Delta^h$, and from there to $\Lambda \Rightarrow \Delta^h \wedge (\bigwedge_i \Lambda \Rightarrow \Delta_i^{occ})$. Thus, it can be shown that each expanded action law or dependency law is equivalent to a formula $\Lambda \Rightarrow \Delta^h \wedge (\bigwedge_i \forall t, f_i[\Lambda' \Rightarrow Occlude_i(t, f_i)])$. Now, there are two useful theorems by Lifschitz [29], the first stating that if B does not contain P, then $Circ_{SO}(\Gamma(P) \wedge B; P) \equiv Circ_{SO}(\Gamma(P); P) \wedge B$, and the second stating that if $F(\overline{x})$ does not contain any P then $Circ_{SO}(\forall \overline{x} F(\overline{x}) \Rightarrow P(\overline{x}); P) \equiv (\forall \overline{x} F(\overline{x}) \equiv P(\overline{x}))$. From these theorems and the equivalent form above follows that $Circ_{SO}((\Gamma_{acs} \cup \Gamma_{dep})(\overline{Occlude}); \overline{Occlude}) = (\bigwedge_k \Lambda_k \Rightarrow \Delta_k^h) \wedge (\bigwedge_i \forall t, f_i[(\bigvee_k \Lambda_{ik}') \equiv Occlude_i(t, f_i)]$, which is first-order.

A.3 Example

The following is the $\mathcal{L}(FL)$ translation of scenario (15). In this case, there is only one feature sort, which has a boolean value domain.

$$\begin{aligned}
&\Gamma_{dom} = \{ \qquad (56)\\
&\text{dom1 } \forall x, v[Per(\mathsf{fire}(x)) \wedge Dur(\mathsf{fire}^*(x, v), \mathsf{F})]\\
&\text{dom2 } \forall x, v[Per(\mathsf{dry}(x)) \wedge Dur(\mathsf{dry}^*(x, v), \mathsf{F})]\ \}
\end{aligned}$$

$$
\begin{array}{l}
\Gamma_{acs+dep+inf} = \{ \\
\text{acs1 } \forall s,t,a,x[\mathit{Occurs}(s,t,\mathsf{LightFire}(a,x)) \Rightarrow \\
\qquad \forall t'[s < t' \leq t \Rightarrow \mathit{Holds}(t', \mathsf{fire}^*(x,\mathsf{T}),\mathsf{T})] \wedge \\
\qquad \forall t'[s < t' \leq t \Rightarrow \ \mathit{Occlude}(t', \mathsf{fire}^*(a,x,\mathsf{T}))], \\
\text{acs2 } \forall s,t,a,x[\mathit{Occurs}(s,t,\mathsf{PourWater}(a,x)) \Rightarrow \\
\qquad \forall t'[s < t' \leq t \Rightarrow \mathit{Holds}(t', \mathsf{dry}^*(x,\mathsf{F}),\mathsf{T})] \wedge \\
\qquad \forall t'[s < t' \leq t \Rightarrow \ \mathit{Occlude}(t', \mathsf{dry}^*(x,\mathsf{F}))], \\
\text{dep1 } \forall s,x[\neg \mathit{Holds}(s,\mathsf{dry}(x),\mathsf{T}) \Rightarrow \mathit{Holds}(s,\mathsf{fire}^*(x,\mathsf{F}),\mathsf{T})] \\
\text{inf1 } \forall s,x(\forall t'[s \leq t' \leq s+3 \Rightarrow (\mathit{Holds}(t', \mathsf{fire}^*(x,\mathsf{T})),\mathsf{T}) \wedge \\
\qquad \neg \mathit{Holds}(t', \mathsf{fire}^*(x,\mathsf{F})),\mathsf{T}) \wedge \mathit{Holds}(t', \mathsf{wood}(x),\mathsf{T}))] \Rightarrow \\
\qquad\quad \mathit{Occlude}(s+3,\mathsf{fire}(x)) \wedge \mathit{Holds}(s+3,\mathsf{fire}(x),\mathsf{T})), \\
\text{inf2 } \forall s,x[\mathit{Holds}(s,\mathsf{fire}^*(x,\mathsf{F}),\mathsf{T}) \Rightarrow \\
\qquad (\mathit{Occlude}(s,\mathsf{fire}(x)) \wedge \neg \mathit{Holds}(s,\mathsf{fire}(x),\mathsf{T}))], \\
\text{inf3 } \forall s,x(\forall t'[s \leq t' \leq s+3 \Rightarrow \\
\quad (\mathit{Holds}(t', \mathsf{dry}^*(x,\mathsf{T})),\mathsf{T}) \wedge \neg \mathit{Holds}(t', \mathsf{dry}^*(x,\mathsf{F})),\mathsf{T})] \Rightarrow \\
\ (\mathit{Occlude}(s+3,\mathsf{dry}(x)) \wedge \mathit{Holds}(s+3,\mathsf{dry}(x),\mathsf{T})), \\
\text{inf4 } \forall s,x[\mathit{Holds}(s,\mathsf{dry}^*(x,\mathsf{F}),\mathsf{T}) \Rightarrow \\
\qquad (\mathit{Occlude}(s,\mathsf{dry}(x)) \wedge \neg \mathit{Holds}(s,\mathsf{dry}(x),\mathsf{T}))]\}
\end{array}
\tag{57}
$$

$$
\begin{array}{l}
\Gamma_{occ} = \{ \ \text{occ1 } \mathit{Occurs}(2,6,\mathsf{LightFire}(\mathsf{bill},\mathsf{wood1})), \\
\qquad\quad \text{occ2 } \mathit{Occurs}(3,5,\mathsf{PourWater}(\mathsf{bob},\mathsf{wood1}))\}
\end{array}
\tag{58}
$$

$$
\begin{array}{l}
\Gamma_{obs} = \{\text{obs1 } \mathit{Holds}(0,\mathsf{dry}(\mathsf{wood1}),\mathsf{T}) \wedge \\
\qquad\qquad \neg \mathit{Holds}(0,\mathsf{fire}(\mathsf{wood1}),\mathsf{T}) \wedge \\
\qquad\qquad \mathit{Holds}(0,\mathsf{wood}(\mathsf{wood1}),\mathsf{T})\}
\end{array}
\tag{59}
$$

Circumscribing *Occurs* in Γ_{occ} and *Occlude* in $\Gamma_{acs} \cup \Gamma_{dep} \cup \Gamma_{inf}$ yield the following exact descriptions of the two predicates, which together with the original theory and the additional components constitute Γ. Notice that the *Occlude* part specifies exactly the exceptions to the default rules for persistent and durational features, as expressed in the two first axioms in Γ_{fl}.

$$
\begin{array}{l}
\forall s,t,e\ [\mathit{Occurs}(s,t,e) \equiv \\
\qquad ((s = 2 \wedge t = 6 \wedge e = \mathsf{LightFire}(\mathsf{bill},\mathsf{wood1})) \vee \\
\qquad (s = 3 \wedge t = 5 \wedge e = \mathsf{PourWater}(\mathsf{bob},\mathsf{wood1})))]
\end{array}
\tag{60}
$$

$$
\begin{aligned}
&\forall t', f\ (Occlude(t', f) \equiv \exists s, t, a, x[\\
&(s < t' \leq t \wedge f = \mathsf{fire}^*(x, \mathsf{T}) \wedge \\
&\quad Occurs(s, t, \mathsf{LightFire}(a, x))) \vee \\
&\ (s < t' \leq t \wedge f = \mathsf{dry}^*(x, \mathsf{F}) \wedge \\
&\quad Occurs(s, t, \mathsf{PourWater}(a, x))) \vee \\
&(f = \mathsf{fire}(x) \wedge t' = s + 3 \wedge \forall t'[s \leq t' \leq s + 3 \Rightarrow \\
&\qquad (Holds(t', \mathsf{fire}^*(x, \mathsf{T})), \mathsf{T}) \wedge \\
&\qquad \neg Holds(t', \mathsf{fire}^*(x, \mathsf{F}), \mathsf{T}) \wedge Holds(t', \mathsf{wood}(x), \mathsf{T})]) \vee \\
&(f = \mathsf{fire}(x) \wedge t' = s \wedge Holds(s, \mathsf{fire}^*(x, \mathsf{F}), \mathsf{T})) \vee \\
&(f = \mathsf{dry}(x) \wedge t' = s + 3 \wedge (\forall t'[s \leq t' \leq s + 3 \Rightarrow \\
&\quad (Holds(t', \mathsf{dry}^*(x, \mathsf{T}), \mathsf{T}) \wedge \neg Holds(t', \mathsf{dry}^*(x, \mathsf{F}), \mathsf{T}))])) \vee \\
&(f = \mathsf{dry}(x) \wedge t' = s \wedge Holds(s, \mathsf{dry}^*(x, \mathsf{F}), \mathsf{T}))])
\end{aligned} \tag{61}
$$

The proof for $\neg Holds(7, \mathsf{fire}(\mathsf{wood1}), \mathsf{T})$ is as follows.

1. $Occurs(3, 5, \mathsf{PourWater}(\mathsf{bob}, \mathsf{wood1}))$ is true according to *occ2*.
2. From 1 and *acs2*, one can infer $Holds(5, \mathsf{dry}^*(\mathsf{wood1}, \mathsf{F}), \mathsf{T})$.
3. From 2 and *inf4*, it follows that $\neg Holds(5, \mathsf{dry}(\mathsf{wood1}), \mathsf{T})$ is true.
4. From (60) and (61), it follows that $\neg Occlude(t, \mathsf{dry}^*(\mathsf{wood1}, \mathsf{F}))$, $\neg Occlude(t, \mathsf{dry}^*(\mathsf{wood1}, \mathsf{T}))$ and consequently $\neg Occlude(t, \mathsf{dry}(\mathsf{wood1}))$ are true for $t = 6$ and $t = 7$.
5. From 3, 4, $Per(\mathsf{dry}(\mathsf{wood1}))$ (*dom2*) and (55), it follows that $\neg Holds(t, \mathsf{dry}(\mathsf{wood1}), \mathsf{T})$ is true for $t = 6$ and $t = 7$.
6. From 5 and *dep1*, it follows that $Holds(7, \mathsf{fire}^*(\mathsf{wood1}, \mathsf{F}), \mathsf{T})$ is true.
7. From 6 and *inf2*, it follows that $\neg Holds(7, \mathsf{fire}(\mathsf{wood1}), \mathsf{T})$ is true.

References

[1] Allen, Kautz, Pelavin, and Tenenberg, editors. *Reasoning About Plans.* Morgan Kaufmann, 1991.

[2] James F. Allen. Temporal reasoning and planning. In Allen et al. [1].

[3] C. Baral and M. Gelfond. Representing concurrent actions in extended logic programming. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, 1993.

[4] Sven-Erik Bornscheuer and Michael Thielscher. Explicit and implicit indeterminism. Reasoning about uncertain and contradictory specifications of dynamic systems. *Journal of Logic Programming, Special issue: reasoning about action and change*, 31(1–3), 1997.

[5] Patrick Doherty. Reasoning about action and change using occlusion. In *Proceedings of the Eleventh European Conference on Artificial Intelligence*. John Wiley & Sons, 1994.

[6] Patrick Doherty. PMON+: A fluent logic for action and change. Technical Report 96-33, Department of Computer and Information Science, Linköping University, 1996. Available at Linköping University Electronic Press, `http://www.ep.liu.se/`.

[7] Patrick Doherty and Jonas Kvarnström. Tackling the qualification problem using fluent dependency constraints: Preliminary report. In *TIME'98*, Florida, 1998.

[8] Jacques Ferber and Jean-Pierre Müller. Influences and reaction: a model of situated multi-agent systems. In Mario Tokoro, editor, *Proceedings of the Second International Conference on Multi-Agent Systems*, Kyoto, December 1996. AAAI Press.

[9] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[10] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24, 1984.

[11] M Gelfond and V Lifschitz. Representing action and change using logic programs. *Journal of Logic Programming*, 17:301–321, 1993.

[12] M. Gelfond, V. Lifschitz, and A. Rabinov. What are the limitations of the situation calculus? In *Working notes, AAAI Spring Symposium, Series. Symposium: Logical Formalization of Commonsense Reasoning*, 1991.

[13] Michael P. Georgeff. Actions, processes, and causality. In Georgeff and Lansky [14].

[14] M.P. Georgeff and A.L. Lansky, editors. *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*. Morgan Kaufmann, 1987.

[15] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: a possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.

[16] Joakim Gustafsson and Patrick Doherty. Embracing occlusion in specifying the indirect effects of actions. In KR96 [24].

[17] Gary Hendrix. Modelling simultaneous actions and continuous processes. *Artificial Intelligence*, 4:145–180, 1973.

[18] *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, 1995.

[19] Antonias Kakas and Rob Miller. A simple declarative language for describing narratives with actions. *Journal of Logic Programming, Special issue: reasoning about action and change*, 31(1–3), 1997.

[20] Lars Karlsson. Anything can happen: on narratives and hypothetical reasoning. In KR98 [25].

[21] Lars Karlsson, Joakim Gustafsson, and Patrick Doherty. Delayed effects of actions. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence.* John Wiley & Sons, 1998.

[22] M. Koubarakis. Complexity results for first-order theories of temporal constraints. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference.* Morgan Kaufmann, 1994.

[23] R. A. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[24] *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference.* Morgan Kaufmann, 1996.

[25] *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference.* Morgan Kaufmann, 1998.

[26] Amy L. Lansky. A representation of parallel activity based on events, structure, and causality. In Georgeff and Lansky [14].

[27] Renwei Li and Luís Moniz Pereira. Temporal reasoning and abductive logic programming. In *Proceedings of the Twelfth European Conference on Artificial Intelligence.* John Wiley & Sons, 1996.

[28] Vladimir Lifschitz. Computing circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 121–127. Morgan Kaufmann, 1985.

[29] Vladimir Lifschitz. Circumscription. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3 of *Handbook of Artificial Intelligence and Logic Programming*, pages 179–193. Oxford University Press, 1993.

[30] Vladimir Lifschitz and Arkady Rabinov. Things that change by themselves. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 864–867. Morgan Kaufmann, 1989.

[31] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In IJCAI95 [18].

[32] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, California, 1997.

[33] John McCarthy. Elabortation tolerance. In *Common Sense 98*, Queen Mary and Westfield College, London, 1998.

[34] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[35] Richard N. Pelavin. Planning with simultaneous actions and external events. In Allen et al. [1].

[36] Javier A. Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, University of Toronto, 1994.

[37] Javier A. Pinto. Concurrent actions and interactions. In KR98 [25].

[38] Ray 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 359–360. Academic Press, 1991.

[39] Raymond Reiter. Natural actions, concurrency and continuous time in the situation calculus. In KR96 [24].

[40] Erik Sandewall. Combining logic and differential equations for describing real-world systems. In *Principles of Knowledge Representation and Reasoning: Proceedings of the First International Conference*. Morgan Kaufmann, 1989.

[41] Erik Sandewall. *Features and Fluents.* Oxford Press, 1994.

[42] Murray Shanahan. Representing continuous change in the event calculus. In *Proceedings of the Ninth European Conference on Artificial Intelligence*. John Wiley & Sons, 1990.

[43] Michael Thielscher. The logic of dynamic systems. In IJCAI95 [18].

[44] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[45] Choong-Ho Yi. Towards assessment of logics for concurrent actions. Presented at the AAAI 1995 Spring Symposium: Extended Theories of Action, Stanford University, 1995.