Cumulative Effects of Concurrent Actions on Numeric-Valued Fluents

Esra Erdem
Institute of Information Systems,
Vienna University of Technology, Vienna, Austria
esra@kr.tuwien.ac.at

Alfredo Gabaldon
National ICT Australia
Sydney, Australia
alfredo.gabaldon@nicta.com.au

Abstract

We propose a situation calculus formalization of action domains that include numeric-valued fluents (so-called additive or measure fluents) and concurrency. Our approach allows formalizing concurrent actions whose effects increment/decrement the value of additive fluents. For describing indirect effects, we employ mathematical equations in a manner that is inspired by recent work on causality and structural equations.

1 Introduction

In this paper we study the problem of formalizing action domains that include numeric-valued fluents and concurrency, in the situation calculus [McCarthy, 1963]. These fluents, known as *additive fluents* [Lee & Lifschitz, 2003] or *measure fluents* [Russel & Norvig, 1995], are used for representing measurable quantities such as weight or speed. An obvious practical application of reasoning about additive fluents is planning with resources, which usually are measurable quantities whose value is incremented/decremented by the execution of actions.

The ability to build plans in concurrent domains with numeric-valued fluents is crucial in real world applications. However, there has not been much work on formal accounts of this problem. Although there are several planning systems designed to work in concurrent domains with resources,[1] most of them simplify the problem by requiring that concurrent actions be *serializable*. That is, actions are allowed to execute concurrently as long as their effect is equivalent to the effect of executing the same actions consecutively. This assumption eliminates practically all the semantic issues of the problem. On the other hand, this requirement precludes planners from solving many interesting problems. Consider for instance a simple problem where there are two resources R_1, R_2 and actions A, B such that A consumes one unit of R_1 and produces one unit of R_2, and B consumes one unit of R_2 and produces one of R_1. Suppose also that there is the constraint $R_i > 0$ at all times, and that they are initially $R_1 = R_2 = 1$. The simple plan consisting of the concurrent execution of A and B is not serializable, hence out of the scope of most planning systems.

[1][Koehler, 1998; Rintanen & Jungholt, 1999; Kvarnström, Doherty, & Haslum, 2000; Bacchus & Ady, 2001; Do & Kambhampati, 2003] are recent examples.

In addition to a more general account of concurrency with additive fluents, we are also interested in allowing certain forms of indirect effects of actions (ramifications) on additive fluents. For instance, when a robot adds some water into a small container, and the water overflows into a larger container, the increment in the amount of water in the large container can be viewed as an indirect effect of the robot's action. Given that the total amount of water in both containers is preserved, one may want to capture indirect effects by means of a mathematical equation. However, one is immediately confronted with a problem similar to the problem that led to the introduction of explicit notions of causality in action theories (see [Lin, 1995; McCain & Turner, 1995; Thielscher, 1997] among others): mathematical equations are symmetric and thus cannot express the causal relationship among the fluents in the equation. In this paper we present a formalization of indirect effects of concurrent actions on additive fluents. Our approach is in some respects based on the work of [Iwasaki & Simon, 1986; Halpern & Pearl, 2001] on causal reasoning with *structural equations*.

Our formalization for reasoning about the effect of concurrent actions on additive fluents builds on the work of [Reiter, 2001] and [Lee & Lifschitz, 2003]. We generalize Reiter's *basic action theories* in the concurrent situation calculus [Reiter, 2001] with an account of additive fluents that is inspired on [Lee & Lifschitz, 2003], which, on the other hand, restricts additive fluents to range over finite sets of integers and does not consider the kind of indirect effects of actions on these fluents that we do.

2 Action Theories with Additive Fluents

We axiomatize action domains as basic action theories of the concurrent situation calculus [Reiter, 2001]. In addition to situations, objects, and primitive actions, the concurrent situation calculus includes a sort for concurrent actions, which are treated as sets of primitive actions. For a detailed description of the language and of basic action theories we refer the reader to [Reiter, 2001].

Additive fluents in the situation calculus are functional fluents that take numerical values, usually within a range. We will assume that for each additive fluent f, a *range constraint* $[L_f, U_f]$ is given, meaning that in every situation s, $L_f \leq f(s) \leq U_f$. These range constraints will usually be treated as qualification constraints, i.e., as additional action preconditions. Later when we consider indirect effects, we will see how these constraints also play a role there.

2.1 Direct effect axioms

For describing direct effects of actions on additive fluents, we introduce a function $\mathit{contr}_f(\vec{x}, a, s)$ for each additive fluent f.Intuitively, $\mathit{contr}_f(\vec{x}, a, s)$ is the amount that the action a contributes to the value of f when executed in situation s.

According to [Reiter, 1991], successor state axioms for functional fluents are sometimes derived from *effect axioms* of the form $\gamma(\vec{x}, v, a, s) \supset f(\vec{x}, \mathit{do}(a, s)) = v$. Similarly, we describe the effects of primitive actions on additive fluents by axioms of the form:

$$\kappa_f(\vec{x}, v, a, s) \supset \mathit{contr}_f(\vec{x}, a, s) = v \tag{1}$$

where $\kappa_f(\vec{x}, v, a, s)$ is a first-order formula whose only free variables are $\vec{x}, v, a, s$, does not mention function contr_g for any g, and s is its only term of sort situation. For instance, when a robot r dumps a container B with n liters of water, this action causes its contents to decrease by n:

$$(\exists r)[a = \mathit{dumpB}(r) \wedge n = -B(s)] \supset \mathit{contr}_B(a, s) = n.$$

From such effect axioms, we intend to derive successor state axioms for additive fluents by the same kind of transformation in [Reiter, 1991], which is based on an explanation closure assumption.

2.2 Successor state axioms

The effect axioms (1) describe the effects of atomic actions on additive fluents. We can obtain successor state axioms for these fluents in the concurrent situation calculus as follows.

As a first step, similar to how effect axioms for regular fluents are handled in [Reiter, 2001], we assume that if a primitive action has an effect on an additive fluent, then there is one effect axiom of the form (1) describing this effect, and that otherwise the effect of the action is to contribute zero to the additive fluent. This assumption allows us to derive a definitional axiom of the following form for each function contr_f:

$$\begin{aligned} \mathit{contr}_f(\vec{x}, a, s) = v \equiv \kappa_f(\vec{x}, v, a, s) \vee \\ v = 0 \wedge \neg(\exists v')\kappa_f(\vec{x}, v', a, s). \end{aligned} \tag{2}$$

Frequently the formula $\kappa_f(\vec{x}, v, a, s)$ is a disjunction of the form $a = \alpha_1 \wedge \kappa_{\alpha_1, f}(\vec{x}, v_1, a, s) \vee \ldots \vee a = \alpha_k \wedge \kappa_{\alpha_k, f}(\vec{x}, v_k, a, s)$. When this is the case, we write axiom (2) as follows:

$$\mathit{contr}_f(\vec{x}, a, s) = \begin{cases} v_1 & \text{if } a = \alpha_1 \wedge \kappa_{\alpha_1, f}(\vec{x}, v_1, a, s) \\ \ldots & \\ v_k & \text{if } a = \alpha_k \wedge \kappa_{\alpha_k, f}(\vec{x}, v_k, a, s) \\ 0 & \text{otherwise} \end{cases}$$

Example 1 Consider the missionaries and cannibals problem with two boats. The number of missionaries *Mi* or cannibals *Ca* at *Bank1* or *Bank2* of the river is described by the additive fluent $\mathit{num}(g, l, s)$. The action of crossing the river is described by $\mathit{cross}(b, l, nm, nc)$ ("nm number of missionaries and nc number of cannibals are crossing the river by boat b to reach the location l").

The only action in the domain, $cross$, has a direct effect on the additive fluent *num*:

$$\mathit{contr}_{\mathit{num}}(g, l, a, s) = \begin{cases} n_1 & \text{if } (\exists b, n_2) a = \mathit{cross}(b, l, n_1, n_2) \wedge g = \mathit{Mi} \\ n_2 & \text{if } (\exists b, n_1) a = \mathit{cross}(b, l, n_1, n_2) \wedge g = \mathit{Ca} \\ -n_1 & \text{if } (\exists b, n_2, l') a = \mathit{cross}(b, l', n_1, n_2) \wedge \\ & \qquad g = \mathit{Mi} \wedge l \neq l' \\ -n_2 & \text{if } (\exists b, n_1, l') a = \mathit{cross}(b, l', n_1, n_2) \wedge \\ & \qquad g = \mathit{Ca} \wedge l \neq l' \\ 0 & \text{otherwise} \end{cases}$$

Once the axioms defining contr_f are in place, the successor state axioms for additive fluents are straight forward to write. What remains is to add up the contributions of all the primitive actions in a concurrent action. Such a sum defines the following function:

$$\mathit{cContr}_f(\vec{x}, c, s) = \sum_{a \in c} \mathit{contr}_f(\vec{x}, a, s).$$

The successor state axiom for each additive fluent f is

$$f(\vec{x}, \mathit{do}(c, s)) = f(\vec{x}, s) + \mathit{cContr}_f(\vec{x}, c, s). \tag{3}$$

Example 2 Consider again the missionaries and cannibals problem of Example 1. The location of a boat b is described by the non-additive functional fluent $\mathit{loc}(b, s)$. For this fluent, the successor state axiom is of the usual form:

$$\begin{aligned} \mathit{loc}(b, \mathit{do}(c, s)) = l \equiv (\exists n_1, n_2)(\mathit{cross}(b, l, n_1, n_2) \in c) \vee \\ \neg(\exists n_1, n_2, l')(\mathit{cross}(b, l', n_1, n_2) \in c) \ \wedge \mathit{loc}(b, s) = l. \end{aligned}$$

For the additive fluent *num*, the successor state axiom is of the form (3):

$$\mathit{num}(g, l, \mathit{do}(c, s)) = \mathit{num}(g, l, s) + \mathit{cContr}_{\mathit{num}}(g, l, c, s).$$

2.3 Action preconditions

In a basic action theory as described above, a concurrent action is possible only if each of its primitive actions is possible. However, a set of primitive actions each of which is individually possible may be impossible when executed concurrently. To handle such cases, we describe the conditions under which the primitive actions in c conflict with each other, denoted by $\mathit{conflict}(c)$, and require their negation as additional preconditions of c. For example, in the blocks world, a concurrent action containing the two primitive actions $\mathit{stack}(x, z)$ and $\mathit{stack}(y, z)$ $(x \neq y)$ has a conflict, denoted by:

$$\begin{aligned} &\mathit{conflict}(c) \stackrel{\text{def}}{=} \\ &\qquad (\exists x, y, z)[\mathit{stack}(x, z) \in c \wedge \mathit{stack}(y, z) \in c \wedge x \neq y], \end{aligned}$$

so we include $\neg\mathit{conflict}(c)$ as a precondition for c.

Another requirement for a concurrent action to be possible is that it must result in a situation that satisfies the range constraints on additive fluents. We use $RC(s)$ to denote the conjunction of the range constraints on each additive fluent f:

$$\bigwedge_f L_f \leq f(s) \leq U_f$$

conjoined with additional qualification constraints if given (see Example 3).

For additive fluents, most conflicts are covered by treating the range constraints as a precondition. For example, suppose that there is a fluent f, with the range constraint $[0, 10]$ and the initial value $f(S_0) = 5$, and actions A which doubles the current value of f when executed and B which contributes 5 to f. Due to the range constraint, although each action is possible in S_0, the concurrent action $\{A, B\}$ is not. On the other hand, actions that *set* additive fluents to absolute values are an exception. A concurrent action that includes an action that sets the value of a fluent, e.g., "dump bucket," and an action that contributes to the same fluent, e.g., "pour into bucket," has a conflict that needs to be encoded explicitly by *conflict*(c).

To exclude both sorts of conflicting cases among actions, we include in an action theory a precondition axiom of the form

$$\begin{aligned}\mathit{Poss}(c, s) \equiv (\exists a)(a \in c) \wedge (\forall a \in c)\mathit{Poss}(a, s) \wedge \\ \neg\mathit{conflict}(c, s) \wedge \mathcal{R}^1[\mathit{RC}(\mathit{do}(c, s))].\end{aligned} \tag{4}$$

Here, $\mathcal{R}^1[W]$ is a formula equivalent to the result of applying one step of Reiter's regression procedure [Reiter, 1991]. Intuitively, by applying one regression step we obtain a formula that is relative to s and is true iff W is true in $do(c, s)$. If the regressed formula holds in s, it is guaranteed that, after executing c, the constraints RC will hold. Regressing W is necessary in order to obtain an axiom of the form $Poss(c, s) \equiv \Pi(c, s)$ where $\Pi(c, s)$ is a formula whose truth value depends on situation s and on no other situation.

A single primitive action A can be viewed as a singleton concurrent action $\{A\}$. Thus, reasoning about executable sequences is done in terms of concurrent actions only [Reiter, 2001]:[2]

$$\mathit{executable}(s) \stackrel{\text{def}}{=} (\forall c, s^*)(\mathit{do}(c, s^*) \sqsubseteq s \supset \mathit{Poss}(c, s^*)).$$

Example 3 Continuing with the axiomatization of the missionaries and cannibals problem, given the capacity of each boat by a situation independent function $capacity(b)$, we have the following precondition axiom for $cross$:

$$\begin{aligned}\mathit{Poss}(\mathit{cross}(b, l, n_1, n_2), s) \equiv \mathit{loc}(b, s) \neq l \wedge \\ n_1 + n_2 \neq 0 \ \wedge\ n_1 + n_2 \leq \mathit{capacity}(b).\end{aligned}$$

One possible conflict we must consider is two cross actions to different locations but with the same boat:

$$\begin{aligned}\mathit{conflict}(c, s) \stackrel{\text{def}}{=} (\exists l, l_1)(l \neq l_1) \wedge \\ (\exists b, n_1, n_2)\mathit{cross}(b, l, n_1, n_2) \in c \wedge \\ \mathit{cross}(b, l_1, n_1, n_2) \in c.\end{aligned}$$

In this example, the constraints $RC(s)$ are more interesting than just upper and lower bounds on the additive fluents, since there are additional constraints on the numbers of missionaries relative to cannibals: missionaries must not be outnumbered by cannibals. We include the following constraint:

$$\begin{aligned}RC(s) \stackrel{\text{def}}{=} \neg(\exists l)(\mathit{num}(Ca, l, s) > \mathit{num}(Mi, l, s) \wedge \\ \mathit{num}(Mi, l, s) > 0) \wedge \\ (\forall g, l)(0 \leq \mathit{num}(g, l, s) \leq \mathit{MaxNumber}).\end{aligned}$$

The constant *MaxNumber* is the upper bound on fluent *num* for both cannibals and missionaries.

3 Ramification Constraints on Additive Fluents

A domain that does not contain any actions with ramifications can be described as an action theory in the concurrent situation calculus as discussed in the previous sections. How do we describe in the concurrent situation calculus a domain that contains an action with indirect effects on some fluents? In this section we provide an answer to this question for a particular representation of ramifications.

Example 4 Suppose that we have a small container and a large container for storing water. The small container is suspended over the large container so that, when the small container is full of water, the water poured into the small container overflows into the large container. Suppose also that there are three taps: one directly above the small container, by which some water can be added to the containers from an external source, one on the small container, by which some water can be released from the small container into the large container, and a third tap on the large container to release water to the exterior. We want to formalize this domain in the concurrent situation calculus.

The amount of water in the small and the large containers is represented by the additive fluents: *small*(s) and *large*(s). Another additive fluent, *total*(s), represents the total amount of water in the containers.

We introduce the action *add*(n) to describe the action of adding n liters of water to the containers by opening the tap over them, and the actions *releaseS*(n) and *releaseL(n)*, resp., to describe the action of releasing n liters of water from the small, resp. large, container by opening its tap.

We can describe the direct contributions of *add*(n), *releaseS*(n) and *releaseL*(n) by axioms of form (1). The action *add*(n) contributes directly to *total*:

$$(\exists n)[a = \mathit{add}(n) \wedge v = n] \supset \mathit{contr}_{\mathit{total}}(a, s) = v$$

and to *small*:

$$(\exists n)[a = \mathit{add}(n) \wedge v = n] \supset \mathit{contr}_{\mathit{small}}(a, s) = v.$$

The action *releaseS*(n) contributes directly to *small*:

$$(\exists n)[a = \mathit{releaseS}(n) \wedge v = -n] \supset \mathit{contr}_{\mathit{small}}(a, s) = v$$

and to *large*:

$$(\exists n)[a = \mathit{releaseS}(n) \wedge v = n] \supset \mathit{contr}_{\mathit{large}}(a, s) = v. \tag{5}$$

The action *releaseL* contributes to *large*:

$$(\exists n)[a = \mathit{releaseL}(n) \wedge v = -n] \supset \mathit{contr}_{\mathit{large}}(a, s) = v$$

and similarly to *total*:

$$(\exists n)[a = \mathit{releaseL}(n) \wedge v = -n] \supset \mathit{contr}_{\mathit{total}}(a, s) = v.$$

From these direct contribution axioms, we obtain definitional axioms of the form (2):

$$\mathit{contr}_{\mathit{small}}(a, s) = \begin{cases} n & \text{if } a = \mathit{add}(n) \\ -n & \text{if } a = \mathit{releaseS}(n) \\ 0 & \text{otherwise} \end{cases}$$

[2]Intuitively, an expression $s \sqsubseteq s'$ means that s is a subsequence of s'.

$$contr_{large}(a,s) = \begin{cases} n & \text{if } a = releaseS(n) \\ -n & \text{if } a = releaseL(n) \\ 0 & \text{otherwise} \end{cases}$$

$$contr_{total}(a,s) = \begin{cases} n & \text{if } a = add(n) \\ -n & \text{if } a = releaseL(n) \\ 0 & \text{otherwise} \end{cases}$$

3.1 Range constraints and ramification

In earlier sections, the range restrictions were treated as qualification constraints: if executing an action will falsify them, the action is consider impossible. In this example, however, the upper bound on the value of *small* plays a different role. Actions that seemingly would increase the value of *small* over U_{small} should not be considered impossible, but actually to increase its value *up to* U_{small}.

This fact will be captured explicitly in the definition of the concurrent contribution of actions to *small* as follows:

$$cContr_{small}(\vec{x},c,s) = \begin{cases} U_{small} - small(s) & \text{if } sum_{small} > U_{small} - small(s) \\ sum_{small} & \text{otherwise} \end{cases}$$

where sum_{small} stands for $\sum_{a \in c} contr_{small}(\vec{x},a,s)$.

In general, functions $cContr_f$ are defined as follows:

$$cContr_f(\vec{x},c,s) = \begin{cases} U_f - f(\vec{x},s) & \text{if } sum_f > U_f - f(\vec{x},s) \\ L_f - f(\vec{x},s) & \text{if } sum_f < L_f - f(\vec{x},s) \\ sum_f & \text{otherwise} \end{cases}$$

where sum_f stands for $\sum_{a \in c} contr_f(\vec{x},a,s)$, and the first two lines in the right-hand side being present only if the range restriction U_f, resp. L_f, are a source of ramifications. Note that if the range restrictions play no role in ramifications and the two lines are thus missing, the definition of $cContr_f$ is just as shown earlier.

3.2 Contribution equations

The next question in formalizing the ramifications is how to describe the causal influence among the fluents. In our water container example, the relation among the fluents could be described by the equation:

$$total(s) = small(s) + large(s)$$

which must hold in all situations s. However, this equation does not capture the arrangement of the containers that makes water flow from the small container into the large one. The reason is clear: such algebraic equations are symmetric and are not meant to describe how changes in one fluent causally influence other fluents in the equation.

Causal reasoning with equations has been considered before in AI. [Iwasaki & Simon, 1986] (subsequently IS) considers the problem of making explicit the causal relation among variables in an equation describing a *mechanism*—a component of a device or system. IS assumes each mechanism is described by a single *structural equation* describing how variables influence other variables. [Halpern & Pearl, 2001] (subsequently HP) also uses structural equations, in this case with the purpose of representing causal relations among random variables for modeling counterfactuals.

Our approach to handling indirect effects on additive fluents has been influenced by IS and HP. In order to represent indirect effects on fluents, we will use equations in a similar fashion as structural equations are used in the aforementioned work to describe causal influence among variables. We use structural equations under mainly four assumptions.

1. Similarly to IS and HP, we assume that each equation represents a single mechanism. That is, an equation describes the indirect contribution of actions to one fluent in terms of the contribution to the value of the other fluents in the equation.
2. Both IS and HP require each variable to be classified as either exogenous or endogenous. This is reasonable for the settings they consider where there is no agent intervening with the mechanism represented by the equation. All external intervention is fixed a-priori, which allows classifying variables this way. In our case, external intervention[3] depends on what particular action is executed. Hence a fluent may be exogenous (directly affected) with respect to one primitive action and endogenous with respect to another primitive action, with both actions occurring concurrently. Thus, in our approach we do not assume that fluents can be separated into exogenous and endogenous classes.
3. We do not intend to derive a causal ordering among fluents as IS does for variables. We assume, as done in HP and recent work on causality [Lin, 1995; McCain & Turner, 1995; Thielscher, 1997], that the causal relation among fluents is explicit in the axioms describing the indirect contributions of actions.
4. We assume, as IS and HP do, that the causal influence among fluents is acyclic. Lifting this assumption remains a topic for future work.

With these assumptions, suppose that, in axiomatizing our domain, for describing each causal mechanism, we provide an equation similar to the structural equations in HP: for a fluent f, the equation would have the form $f = \mathcal{E}$ where $\mathcal{E}$ is an expression in terms of the fluents on which f causally depends. In the case of additive fluents, such an expression is in fact a linear combination of functions. Just as the structural equations in HP, an equation such as $f = \mathcal{E}$ is asymmetric in the sense that the equation determines the value of f but not the value of any of the fluents in the right-hand side. We use such equations, however, not to compute the value of fluents, but to compute the contribution to the value of the fluents that results from executing an action. From an equation $f = \mathcal{E}$, we obtain an almost identical equation but instead of written in terms of fluent functions, written in terms of functions $cContr_f$ and an abbreviation $iContr_f(\vec{x},c,s)$ for each fluent f that intuitively denotes the amount that action c indirectly contributes to f in situation s.

If an equation describing indirect effects on a fluent f is not given, then

$$iContr_f(\vec{x},c,s) \stackrel{\text{def}}{=} 0.$$

[3] Here, by external intervention we mean external to the mechanism represented by an equation.

Otherwise, suppose that equation $f = \mathcal{E}(f_1, \ldots, f_n)$ is given, where $\mathcal{E}(f_1, \ldots, f_n)$ is a linear combination of fluents $f_1(\vec{x}_1, s), \ldots, f_n(\vec{x}_n, s)$. Then we define $iContr_f(\vec{x}, c, s)$ as follows:

$$\begin{aligned} &iContr_f(\vec{x}, c, s) \stackrel{\text{def}}{=} \\ &\quad \mathcal{E}(cContr_{f_1}(\vec{x}_1, c, s), \ldots, cContr_{f_n}(\vec{x}_n, c, s)) \\ &\quad - cContr_f(\vec{x}, c, s) \end{aligned}$$

Example 5 Consider Example 4. Suppose that the mechanism of containers is described by the contribution equation

$$large(s) = total(s) - small(s) \tag{6}$$

and the range restrictions on the fluents are as follows:[4]

$$\begin{aligned} &L_{total} = L_{small} = L_{large} = 0, \\ &U_{total} = 6,\ U_{small} = 2,\ U_{large} = 4. \end{aligned}$$

Any concurrent action whose total effect on the fluents results in a situation where these range restrictions are violated is impossible, in accordance with our axiom (4) for $Poss(c, s)$ described earlier, except for restriction U_{small}. If an action's contribution to *small* will result in a larger value than its upper bound U_{small} allows, the action is not rendered impossible, but instead has an indirect effect. The indirect effect of increasing *small* too much is expressed by the following equation which can be obtained from the contribution equation (6):

$$\begin{aligned} &iContr_{large}(c, s) \stackrel{\text{def}}{=} \\ &\quad cContr_{total}(c, s) - cContr_{small}(c, s) \\ &\quad - cContr_{large}(c, s). \end{aligned}$$

Given the initial values

$$total(S_0) = 2,\ small(S_0) = 1,\ large(S_0) = 1,$$

and the concurrent action

$$c = \{add(6), releaseS(1), releaseL(2)\},$$

we obtain:

$$\begin{aligned} &cContr_{total}(c, S_0) = 4, && iContr_{total}(c, S_0) \stackrel{\text{def}}{=} 0, \\ &cContr_{small}(c, S_0) = 1, && iContr_{small}(c, S_0) \stackrel{\text{def}}{=} 0, \\ &cContr_{large}(c, S_0) = -1, && iContr_{large}(c, S_0) \stackrel{\text{def}}{=} 4. \end{aligned}$$

3.3 Successor state axiom with indirect effects

After defining direct and indirect contributions of actions on an additive fluent f, there only remains to define the successor state axiom for such a fluent. We define such an axiom as follows:

$$f(\vec{x}, do(c, s)) = f(\vec{x}, s) + tContr_f(\vec{x}, c, s)$$

where

$$tContr_f(\vec{x}, c, s) \stackrel{\text{def}}{=} cContr_f(\vec{x}, c, s) + iContr_f(\vec{x}, c, s).$$

This axiom replaces (3) in domain axiomatizations.

Example 6 For our container example with the values from Example 5, we obtain $total(c, S_0) = 2 + 4 = 6$, $small(c, S_0) = 1 + 1 = 2$, and $large(c, S_0) = 1 + 3 = 4$.

[4]These must be consistent with the equation (6).

Once the contribution equation has been compiled into the theory, we can prove that the original, symmetric equation is satisfied in every situation provided it is satisfied initially. This can be done by a very simple application of the induction axiom:

$$(\forall P)P(S_0) \wedge (\forall c, s)[P(s) \supset P(do(c, s))] \supset (\forall s)P(s)$$

with

$$P(s) \stackrel{\text{def}}{=} total(s) = small(s) + large(s).$$

Proposition Let $\mathcal{D}$ stand for the water container theory presented through out this section and $eq(s)$ stand for $total(s) = small(s) + large(s)$. Then,

$$\mathcal{D} \models eq(S_0) \supset (\forall s)eq(s).$$

3.4 An example with two mechanisms

Consider a variation of the container example where a container of medium size is inserted between the small container and the large container, so that, when the small container is full, the water poured into the small container overflows into the medium container, and, when the medium container is full, the water overflows into the large container. Here we consider two mechanisms: one consisting of the small container and the medium container, and the other consisting of this inner mechanism and the large container.

The amount of water in the medium container is represented by the additive fluent $medium(s)$. Another additive fluent, $inner(s)$, represents the total amount of water in the small container and the medium container.

We describe the direct contributions of the actions $add(n)$, $releaseS(n)$, and $releaseL(n)$ by the direct effect axioms of Example 4, except (5), and the axioms

$$(\exists n)[a = add(n) \wedge v = n] \supset contr_{inner}(a, s) = v,$$

$$(\exists n)[a = releaseS(n) \wedge v = n] \supset contr_{medium}(a, s) = v.$$

Given the contribution equations

$$\begin{aligned} &medium(s) = inner(s) - small(s), \\ &large(s) = total(s) - inner(s), \end{aligned}$$

one for each mechanism, the indirect effect of adding too much water to the small container is expressed by the following equations:

$$\begin{aligned} &iContr_{medium}(c, s) \stackrel{\text{def}}{=} \\ &\quad cContr_{inner}(c, s) - cContr_{small}(c, s) \\ &\quad - cContr_{medium}(c, s) \end{aligned}$$

$$\begin{aligned} &iContr_{large}(c, s) \stackrel{\text{def}}{=} \\ &\quad cContr_{total}(c, s) - cContr_{inner}(c, s) \\ &\quad - cContr_{large}(c, s). \end{aligned}$$

In addition to the range restrictions on *small* and *large* as in Example 5, consider the following range restrictions

$$\begin{aligned} &L_{medium} = L_{inner} = L_{total} = 0, \\ &U_{medium} = 3,\ U_{inner} = 5,\ U_{total} = 9. \end{aligned}$$

Suppose that initially all containers have 1 unit of water:

$$\begin{aligned}&\mathit{small}(S_0) = \mathit{medium}(S_0) = \mathit{large}(S_0) = 1,\\ &\mathit{inner}(S_0) = 2,\ \mathit{total}(S_0) = 3,\end{aligned}$$

and that the concurrent action

$$c = \{\mathit{add}(8), \mathit{releaseS}(1), \mathit{releaseL}(2)\}$$

is executed at the initial situation. Then we can compute the contributions of this action to each additive fluent:

$$\begin{aligned}
&\mathit{cContr}_{\mathit{small}}(c, S_0) = 1, && \mathit{iContr}_{\mathit{small}}(c, S_0) \overset{\mathrm{def}}{=} 0,\\
&\mathit{cContr}_{\mathit{medium}}(c, S_0) = 1, && \mathit{iContr}_{\mathit{medium}}(c, S_0) \overset{\mathrm{def}}{=} 1,\\
&\mathit{cContr}_{\mathit{large}}(c, S_0) = -2, && \mathit{iContr}_{\mathit{large}}(c, S_0) \overset{\mathrm{def}}{=} 5,\\
&\mathit{cContr}_{\mathit{inner}}(c, S_0) = 3, && \mathit{iContr}_{\mathit{inner}}(c, S_0) \overset{\mathrm{def}}{=} 0,\\
&\mathit{cContr}_{\mathit{total}}(c, S_0) = 6, && \mathit{iContr}_{\mathit{total}}(c, S_0) \overset{\mathrm{def}}{=} 0,
\end{aligned}$$

and the amount of water in each container:

$$\begin{aligned}
&\mathit{small}(c, S_0) = 1 + 1 = 2, && \mathit{inner}(c, S_0) = 2 + 3 = 5,\\
&\mathit{medium}(c, S_0) = 1 + 2 = 3, && \mathit{total}(c, S_0) = 3 + 6 = 8,\\
&\mathit{large}(c, S_0) = 1 + 3 = 4.
\end{aligned}$$

4 Conclusion

In this paper we introduced a formalization of additive fluents in concurrent domains that is based on Reiter's basic action theories in the concurrent situation calculus. This formalization allows reasoning about the effect of actions that increment/decrement integer or even real valued fluents. Moreover, we presented an approach to reasoning about indirect effects through the use of equations that are conceptually similar to the structural equations of [Iwasaki & Simon, 1986; Halpern & Pearl, 2001]. To the best of our knowledge, this is the first attempt at formalizing ramifications of concurrent actions on numeric-valued fluents.

Our approach to ramifications on additive fluents based on equations that express the direction of causal influence explicitly, is in line with recent work on causality in theories of action [Lin, 1995; McCain & Turner, 1995; Thielscher, 1997]. In our equations, causal direction is made explicit by the use of function *iContr* on one side and the use of functions *cContr* on the other side of the equations.

After compiling such ramification constraints in the form of equations into the action theory, in the spirit of [Lin & Reiter, 1994; Lin, 1995; McIlraith, 2000], the constraints become logical consequences of the resulting theory (as shown by Proposition for our example).

Planning with concurrency and resources is currently a subject of intense research and we believe that a formal, logic-based account of the problem is an important contribution. The proposal we have put forward in this paper allows formalizing a much more general class of domains than current planning systems are designed to solve, and thus it is useful for specifying what is a correct solution to a planning problem in such generalized domains.

5 Acknowledgments

We are grateful to Fangzhen Lin for many stimulating discussions on the subject of this paper, including a variation of the container example. We are also grateful to Yves Lespérance for helpful comments on an earlier version of the paper.

References

[Bacchus & Ady, 2001] Bacchus, F., and Ady, M. Planning with resources and concurrency: A forward chaining approach. In *Procs. of IJCAI'01*, pages 417–424, 2001.

[Do & Kambhampati, 2003] Do, M. B., and Kambhampati, S. Sapa: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research* 20:155–194, 2003.

[Halpern & Pearl, 2001] Halpern, J., and Pearl, J. Causes and explanations: a structural-model approach–Part I: Causes. In *Procs. of UAI'01*, pages 194–202, 2001.

[Iwasaki & Simon, 1986] Iwasaki, Y., and Simon, H. Causality in device behavior. *Artificial Intelligence* 29:3–32, 1986.

[Koehler, 1998] Koehler, J. Planning under resource constraints. In *Procs. of ECAI'98*, pages 489–493, 1998.

[Kvarnström, Doherty, & Haslum, 2000] Kvarnström, J.; Doherty, P.; and Haslum, P. Extending TALplanner with concurrency and resources. In *Procs. of ECAI'00*, 2000.

[Lee & Lifschitz, 2003] Lee, J., and Lifschitz, V. Describing additive fluents in action language $\mathcal{C}$+. In *Procs. of IJCAI'03*, pages 1079–1084, 2003.

[Lin & Reiter, 1994] Lin, F., and Reiter, R. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678, 1994.

[Lin, 1995] Lin, F. Embracing causality in specifying the indirect effects of actions. In *Procs. of IJCAI'95*, pages 1985–1993, 1995.

[McCain & Turner, 1995] McCain, N., and Turner, H. A causal theory of ramifications and qualifications. In *Procs. of IJCAI'95*, pages 1978–1984, 1995.

[McCarthy, 1963] McCarthy, J. Situations, actions and causal laws. Technical report, Stanford University. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pages 410–417, 1963.

[McIlraith, 2000] McIlraith, S. An axiomatic solution to the ramification problem (sometimes). *Artificial Intelligence* 116(1–2):87–121, 2000.

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

[Reiter, 2001] Reiter, R. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. Cambridge, MA: MIT Press, 2001.

[Rintanen & Jungholt, 1999] Rintanen, J., and Jungholt, H. Numeric state variables in constraint-based planning. In *ECP*, pages 109–121, 1999.

[Russel & Norvig, 1995] Russel, S., and Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

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