Compositionality and the Frame Problem

Thomas Drakengren
Email: thodr@ida.liu.se

Abstract

The problem of compositionally specifying effects of actions has not yet received an appropriate solution, as shown in this paper. A general semantic-based method is proposed, the *effect law semantics*, which solves this, and has an associated logic for specification of actions. In addition, the semantics completely separates effects of actions from assumptions about inertia (the *frame problem*), and whether actions have their intended effects or not. This makes it suitable for generalisations, such as handling *surprises* and actions that may fail.

This research has been supported by the Swedish Council for the Engineering Sciences (TFR), grant 93-270.

IDA Technical Report 1996
LiTH-IDA-R-96-04
ISSN-0281-4250

Department of Computer
and Information Science,
Linköping University,
S-581 83 Linköping, Sweden

Compositionality and the Frame Problem

Thomas Drakengren
Email: thodr@ida.liu.se

January 24, 1996

1 Introduction

No approaches to reasoning about action have yet addressed the problem of *compositionally* specifying effects of actions. The principle of compositionality for this states that given a formula α (of some suitable logic), the effects described by α is a function of the effects described by subformulae of α. More formally, if $s(\alpha)$ yields what effects are described by α, and we have e.g. that $\alpha \equiv \beta \wedge \gamma$, then $s(\alpha)$ should be determined by $s(\beta)$ and $s(\gamma)$.

Then, why is compositionality important? Suppose that α is a formula describing effects of an action, and $s(\alpha)$ yields those effects. That is, the *meaning* of the formula α is $s(\alpha)$, and any formula α' with $s(\alpha) = s(\alpha')$ should be regarded as equivalent. Now, if the semantics were noncompositional, we could arrive at a formula $\alpha \wedge \beta$ satisfying $s(\alpha \wedge \beta) \neq s(\alpha' \wedge \beta)$, even if $s(\alpha) = s(\alpha')$, meaning that the change in meaning of $\alpha \wedge \beta$ cannot be traced to a change in meaning in some of its subcomponents, which is clearly absurd. (The standard semantics of classical first order logic is compositional, for instance).

In reasoning about action, compositionality is particularly useful if we know the effects of several actions a_i, described by α_i's; the complete effects of all the actions executed together then is described by $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$. Such a possibility of combining effects is crucial when expressing *concurrent actions*.

We should note that a semantics being compositional does not imply that the semantics is adequate: e.g. every semantics that assigns one value v to every formula is compositional, but is clearly useless.

No existing approach actually *claims* to have a compositional semantics for effects, but nevertheless, we shall first show that neither of the approaches by Sandewall [9, 10, 11], Kartha and Lifschitz [4] and Grünwald [1] has connectives which could serve the purpose of composition, and second, that even if this could be fixed, no connective would be capable of expressing the effects of two actions executing in parallel. We choose to investigate these approaches, because they (and those subsumed by them) are the main ones which can handle *nondeterminism*, without which compositionality should follow from that of classical logic.

In order to solve this, we formulate a general modular semantics for actions and effects, the *effect law semantics*, which separates what are effects (e.g. of actions), whether these effects succeed or not, the inertia assumption, and whether all executions of actions are known or not. This opens for future generalisations to releasing assumptions about inertia (handling *surprises*) or succeeding actions (handling *failing actions*). Furthermore, we define a logic for the proposed semantics, which makes possible compositional specification of scenarios.

In Section 2, we check the compositionality properties of existing approaches, in Section 3, the compositional effect law semantics is presented, Section 4 defines a logic for specification of the semantics, and Section 5 concludes the paper.

2 Review of Approaches

This section investigates the compositionality properties of three approaches to reasoning about action. Due to space limitation, we do not describe the approaches in detail. First, we state two examples which will be used in the investigations. Each of the examples presents two subscenarios, which will be combined into a larger scenario.

Example 2.1

1. Part of an example from [11]. A gun is loaded by inserting a bullet, and spinning the chamber afterwards. This is represented by the action *load*, and the feature *loaded*, representing whether the gun is loaded or not.

2. The light is turned on, by the action *turnon*, and *light* represents whether there is light or not.

1 and 2 are combined, by expressing that at some time point t, either *load* or *turnon* is executed, or both of them (inclusive disjunction). □

Example 2.2

1. Two persons find a coin, and decide who gets the coin by tossing it. If heads comes up, person 1 gets the coin, and person 2 gets it otherwise. There is one action, *toss*, for tossing the coin, and two features h_i representing whether person i has a coin or not.

2. A man walks by and puts a coin into person 1's pocket, by the action *putin*.

Combine 1 and 2 into one scenario, yielding that at the same time point t, either person 1 or 2 gets the coin by tossing, and person 1 gets a coin by the man walking by. Intended result: h_1 becomes true after t, and either h_2 becomes true or not (since we do not distinguish between coins). $\Box$

2.1 $\mathcal{AR}_0$ by Kartha and Lifschitz

We check whether we can use the connective $\vee$ to express Example 2.1. Take the two action descriptions

1. *load* **releases** *loaded*
2. *turnon* **causes** *light*.

But we cannot use $\vee$ as an operation on a "caused" expression in the *load* action, since there is none, and an expression on a form like *loadturn* **causes** (**release** *loaded*) $\vee$ *light* is not allowed by the language, and would not have a clear meaning if it were. Further, since we intend to express "inclusive or", we would need to add **releases** statements to prevent the inclusive or to become exclusive or, but it is not clear what should be released here.

Next, we check if the $\wedge$ connective can be used for Example 2.2. The subscenarios are expressed by

1. *toss* **causes** $h_1 \vee h_2$
2. *putin* **causes** h_1.

We combine them using $\wedge$, by *tossput* **causes** $h_1 \wedge (h_1 \vee h_2)$, equivalent to *tossput* **causes** h_1 (classical logic), which is obviously not the intended result. Thus, $\wedge$ would not be doing the correct composition.

2.2 Sandewall's Trajectory Semantics

We check whether we can express Example 2.1, using the connective $\vee$, in Sandewall's logic DFL-2 [10, 11]. The separate descriptions become

1. $[\mathsf{s},\mathsf{t}]load \Rrightarrow [\mathsf{s},\mathsf{t}]\mathbb{X}loaded$
2. $[\mathsf{s},\mathsf{t}]turnon \Rrightarrow [\mathsf{s},\mathsf{t}]\mathbb{X}light \wedge light$.

Composing them with $\vee$, we would get

$$[\mathsf{s},\mathsf{t}]\mathit{loadturn} \Rrightarrow [\mathsf{s},\mathsf{t}]\mathbb{X}\mathit{loaded} \vee (\mathbb{X}\mathit{light} \wedge \mathit{light}).$$

But Sandewall's formulation requires that action laws are equivalent to one on *full trajectory normal form*, implying that occlusion (the $\mathbb{X}$ symbol) is completely determined by the starting state of an action and the value of variables. This is clearly not the case here, and the action law is disallowed.

Next, we see if the $\wedge$ connective can be used for Example 2.2. The subscenarios are expressed by

1. $[\mathsf{s},\mathsf{t}]\mathit{toss} \Rrightarrow \mathbb{X}h_1 \wedge \mathbb{X}h_2 \wedge (h_1 \vee_x h_2)$
2. $[\mathsf{s},\mathsf{t}]\mathit{putin} \Rrightarrow \mathbb{X}h_1 \wedge h_1$,

$\vee_x$ standing for classical exclusive or. We combine them using $\wedge$, by $[\mathsf{s},\mathsf{t}]\mathit{tossput} \Rrightarrow \mathbb{X}h_1 \wedge \mathbb{X}h_2 \wedge h_1 \wedge (h_1 \vee_x h_2)$, equivalent to $[\mathsf{s},\mathsf{t}]\mathit{tossput} \Rrightarrow \mathbb{X}h_1 \wedge \mathbb{X}h_2 \wedge h_1 \wedge \neg h_2$ in classical logic which is obviously not the intended result. Thus, $\wedge$ would not be doing the correct composition, also here.

Also the approach by Grünwald [1], which has been proved to subsume e.g. approaches by Lin [6], Morgenstern and Stein [12] has problems similar to those above.

In the following, we shall first define a compositional semantics for effects, which is used in Section 4 for a specification logic. The composition operations of the semantics are defined to correspond to parallel, alternative and conditional execution of actions, and does not encounter the problems above. Further, it separates effects from what really happens, and from the assumption about *inertia*.

3 Effect Law Semantics

The technical basics underlying the effect law semantics follow.

Definition 3.1 Let $\mathcal{F}$ be a set of *features* f, taking on *feature values*, denoted $\mathcal{V}_f$. Then $\nu = \langle \mathcal{F}, \mathcal{V}_{\mathcal{F}} \rangle$, with $\mathcal{V}_{\mathcal{F}} = \{\mathcal{V}_f\}_{f \in \mathcal{F}}$, is said to be a *similarity type*. A *state* σ over ν is a function from features to their values, i.e. $\sigma(f) \in \mathcal{V}_f$. A *time domain* $\mathcal{T}$ is a linearly ordered set of *time points*, equipped with operations like e.g. $+$, and relations like $<$. We also have available sets of *constants*, which can be bound to elements of either $\mathcal{F}$, $\mathcal{V}$ or $\mathcal{T}$, respectively. $\square$

Thus, we can use states for modelling aspects (features) of the world over time. We mostly choose $\mathcal{T} = \mathbb{R}$ throughout the paper, which means that we use a continuous time domain. This is not a common choice in this context; mostly a *discrete* time domain is used (e.g. $\mathcal{T} = \mathbb{Z}$), exceptions being e.g. [10, 9, 7] (and maybe [8]), but we take the relatively small amount of work that is needed to include it. We assume that ν and $\mathcal{T}$ are always chosen appropriately for the example at hand.

The world state developing over time is represented by a *development*:

Definition 3.2 A *development* over ν and $\mathcal{T}$ is a function δ mapping time points t to states $\delta(t)$ over ν. Given ν and $\mathcal{T}$, for a development δ, we write $\delta(t, f)$ instead of $\delta(t)(f)$, for convenience. If $\mathcal{V}_f = \{\mathtt{F}, \mathtt{T}\}$, i.e. f is propositional, we write $\delta(t, f)$ instead of $\delta(t, f) = \mathtt{T}$ and $\neg\delta(t, f)$ for $\delta(t, f) = \mathtt{F}$. □

Thus, the reasoning task is to identify what is the set of possible developments, the *scenario* [11], given information about what actions have been performed, what has been observed, and how the world should behave, traditionally stated as formulae in some logic. Then, a property of developments is *entailed* by a scenario if it holds in every development in the scenario.

We represent the execution of actions by features (like in [13]); e.g. an action *shoot* is represented as a propositional feature which is true at time points when the action is invoked, and false otherwise (Sandewall [11] uses actions with duration, so then we get two features: one for the invocation and one for the ending of the action).

Effects of actions during a development of the world are represented by *effect developments* as follows.

Definition 3.3 Let ν and $\mathcal{T}$ be given. An *effect* is a tuple $\langle t, f, v\rangle$, where $t \in \mathcal{T}$ is a time point, $f \in \mathcal{F}$ is a feature and $v \in \mathcal{V}_f$ a feature value. This represents that at t, the feature f should change to the value v from whatever value it had before. An *effect development* is a tuple $\varepsilon = \langle \delta, E, \phi\rangle$, where δ is a development, E is a set of effects, and ϕ is a function mapping constants x to values x in either $\mathcal{T}$, $\mathcal{F}$ or $\mathcal{V}$ depending on x, i.e. $\phi(\mathsf{x}) = x$, representing states over time, what should change, and what constants have what values. □

Note that an effect development decribes effects over time regardless of whether they succeed or not, and is also independent of inertia. A set of effect developments can be used to represent alternatives, as follows.

Definition 3.4 An *effect law* is a set L of effect developments over a fixed ν and $\mathcal{T}$. Also define $\mathcal{E} = \{\langle \delta, E, \phi\rangle | E = \emptyset\}$, i.e. $\mathcal{E}$ is the effect law specifying no effects at all. □

A well-known example illustrates the concept.

Example 3.5 Set

$$Shoot = \{\langle \delta, E, \phi\rangle | E = \{\langle t, alive, \mathtt{F}\rangle | \delta(t, loaded) \wedge \delta(t, shoot)\}\}.$$

Then *Shoot* is an effect law. It is defined to contain exactly those effect developments where the only effects taking place are those affecting the feature *alive* to become false at all time points where the action *shoot* is performed while the feature *loaded* is true. □

Note that the effect of *alive* becoming false does not have to be recorded in δ at all, by e.g. demanding that *alive* really *is* false whenever *shoot* is performed. Of course this is the *desired* result of the *shoot* action, but instead imposing the success of effects as a separate restriction (i.e. modularly), makes it possible to weaken it later on.

The key to compositional specification of actions is that of *combining* simpler effect laws into larger, more complex ones. For this, we specify some useful operations on effect laws:

Definition 3.6 Let L_1 and L_2 be effect laws, and define

$$\begin{array}{lcl} L_1 \oplus L_2 & = & \{\langle \delta, E_1 \cup E_2, \phi\rangle | \langle \delta, E_1, \phi\rangle \in L_1, \langle \delta, E_2, \phi\rangle \in L_2\} \\ L_1 \uplus L_2 & = & L_1 \cup L_2 \cup (L_1 \oplus L_2) \\ L_1 \ominus L_2 & = & \{\langle \delta, E, \phi\rangle | \neg\exists E' : (E' \subseteq E \wedge \langle \delta, E', \phi\rangle \in L_2)\} \\ L_1 \otimes L_2 & = & (L_1 \cup L_2) \ominus (L_1 \oplus L_2) \\ L_1 \rightarrow L_2 & = & (L_2 \cap L_1) \cup (\overline{L_1} \cap \mathcal{E}), \end{array}$$

where in the last operation, we require that the E components of L_1 are completely unrestricted[1]. We also extend the operations $\oplus$, $\uplus$ and $\otimes$ to corresponding operations $\bigoplus \mathcal{L}$, $\biguplus \mathcal{L}$ and $\bigotimes \mathcal{L}$ on *sets* $\mathcal{L}$ of effect laws, but we omit the explicit (quite involved) definitions of these. The operations are named, in the above order: sum, inclusive or, difference, exclusive or and conditionalisation. □

We shall now explain all operations, and exemplify most of them. First the $L_1 \oplus L_2$ operation, yielding a law which prescribes all effects obtained from either of L_1 or L_2, given δ and ϕ.

Example 3.7 Take *Shoot* from Example 3.5, set

$$Load = \{\langle \delta, E, \phi\rangle | E = \{\langle t, loaded, \mathtt{T}\rangle | \delta(t, load)\}\},$$

and combine *Load* and *Shoot* into one effect law L_{YSP} (YSP for the "Yale Shooting Problem" [2]), by

$$L_{\mathrm{YSP}} = \{\langle \delta, E_1 \cup E_2\rangle | \langle \delta, E_1, \phi\rangle \in Load \wedge \langle \delta, E_2, \phi\rangle \in Shoot\},$$

[1] I.e. given a $\langle \delta, E\rangle \in L_1$, for any E', $\langle \delta, E'\rangle \in L_1$.

i.e. $L_{\mathrm{YSP}} = \mathit{Load} \oplus \mathit{Shoot}$. This law describes the intended *effects* of the Yale Shooting Problem (slightly generalised, since any loadings and shootings are allowed). The union $E_1 \cup E_2$ says that effects prescribed by either law, given a δ and ϕ, is prescribed by the sum law. $\Box$

The $\uplus$ operation, which is an "inclusive or" operation, is exemplified by Example 2.1. We do not state it in detail, but note that the definition of $\uplus$ says that the resulting law prescribes effects either by the separate laws, or by the sum law. The $L_1 \ominus L_2$ operation removes from L_1 any effect developments which have a corresponding effect development in L_2, prescribing a *subset* of its effects, i.e. *fewer* effects. Since the $\ominus$ operation is used in $\otimes$, we do not exemplify it separately. Next, the $L_1 \otimes L_2$ operation, which yields a law prescribing effects that are either obtained from L_1 but not L_2, or from L_2 but not L_1, given δ and ϕ.

Example 3.8 We use the first subscenario from Example 2.2, but omit the action feature *toss*. For a time point constant $\mathtt{t}$, define L_1 and L_2 by $L_1 = \{\langle \delta, E, \phi \rangle | E = \{\langle \phi(\mathtt{t}), h_1, \mathtt{T} \rangle\}\}$ and $L_2 = \{\langle \delta, E, \phi \rangle | E = \{\langle \phi(\mathtt{t}), h_2, \mathtt{T} \rangle\}\}$, and form the exclusive disjunction $L = L_1 \otimes L_2$, yielding that every effect development in L changes either of the h_i's, but not both, at the time point $\phi(\mathtt{t})$. $\Box$

Finally, the conditionalisation $L_1 \rightarrow L_2$ yields a law which prescribes for a δ and ϕ the effects in L_2, *provided* that δ and ϕ are allowed by L_1, and no effects otherwise. This means that the effects of L_2 are conditional on L_1.

Example 3.9 This is a small part of an example by Lifschitz [5]. Propositional features sw_1, sw_2 and $light$, denote the status of two switches and whether light is on or not. The intended behaviour is that there is light iff sw_1 and sw_2 have the same truth values. We now wish to model only that at time $\mathtt{t}$, if the switches change to have the same values, then $light$ shall become true. Assume $\mathcal{T} = \mathbb{Z}$, define predicates $On(t, \delta)$ to be true iff $\delta(t, sw_1) = \delta(t, sw_2)$, and $Chg_+(t, \delta)$ iff $\neg On(t-1, \delta) \wedge On(t, \delta)$. Then set $L_1 = \{\langle \delta, E, \phi \rangle | E = \{\langle \phi(\mathtt{t}), light, \mathtt{T} \rangle\}\}$, i.e. that at $\mathtt{t}$, $light$ becomes true regardless of the switches' positions, and $\Delta_1 = \{\langle \delta, E, \phi \rangle | Chg_+(\phi(\mathtt{t}), \delta)\}$, saying that at $\mathtt{t}$, the light should come on. Now, $L = \Delta_1 \rightarrow L_1$ defines effects such that the light should come on at time point $\mathtt{t}$ iff the switches change into the same position, and no other effects. For the lights also going off, we would need a law L_0 to add to L_1, and for the law to hold for *all* time points, we would need to "add" several copies of the law using $\bigoplus$ with the time point $\mathtt{t}$ varying. $\Box$

Yet, this appears to be fairly complicated, but we shall see that it becomes much clearer when we define a logic for defining effect laws.

3.1 Assumptions: Succeeding Effects, Inertia, Action Omniscience

Up to now, we have not required that effects succeed, i.e. that they are recorded in the actual development, or that features do not change unless changed by an effect. Further, we have had no means of specifying when the special features representing *actions* are true (indeed, we shall assume that we know of *all* executed actions). This section shows how to represent such assumptions.

Definition 3.10 Let $\varepsilon = \langle \delta, E, \phi \rangle$ be an effect development. Then ε is said to have *succeeding effects* iff it satisfies $E \subseteq \delta$, or equivalently, setting $SE = \{\langle \delta, E, \phi \rangle | E \subseteq \delta\}$, iff $\varepsilon \in SE$. The set SE is said to be the *property of succeeding effects.* $\Box$

Thus, if L is an effect law, $L \cap SE$ is the subset of L where succeeding effects are required. For the definitions of inertia, we need some notation for partial functions: $f(x) \downarrow$ means that the function f is defined at x, and $f(x) \uparrow$ that f is undefined at x.

Definition 3.11 We need a few auxiliary functions and predicates. First a function which obtains "the last time a feature was changed". Let $\varepsilon = \langle \delta, E, \phi \rangle$ be an effect development. For a time point t and feature f, define

$$PrevEff_\varepsilon(t, f) = sup\{t' | t' < t \wedge \exists v : \langle t', f, v \rangle \in E\},$$

i.e. the supremum of time points before t where the feature f is affected by an effect. Note that $PrevEff$ is undefined iff the set taken supremum of is empty.

Next, we define a predicate checking whether a feature f at time t can be said to have a value that is the result of effects up to t, by the requirement that f's value at t has to occur arbitrarily close to (or equal to) the left of t, while being subject to effects. Thus, for a time point t and feature f, define

$$\begin{aligned}
&IsRes_\varepsilon(t, f) \Leftrightarrow \\
&\quad \forall \epsilon > 0 \exists t' : (t - \epsilon < t' \leq t \wedge \\
&\qquad \delta(t, f) = \delta(t', f) \wedge \exists v : \langle t', f, v \rangle \in E),
\end{aligned}$$

Note that several other values may occur arbitrary close to t, so in that case, an effect law would be nondeterministic.

Finally, for a time point t and feature f, define

$$\begin{aligned}
&Inert_\varepsilon(t, f) \Leftrightarrow \\
&\quad (\neg \exists v : \langle t, f, v \rangle \in E) \wedge \\
&\quad (PrevEff_\varepsilon(t, f) \downarrow \wedge IsRes(PrevEff_\varepsilon(t, f), f) \wedge \\
&\quad\; (\forall t' \in [PrevEff_\varepsilon(t, f), t) : \delta(t', f) = \delta(t, f))) \vee \\
&\quad (PrevEff_\varepsilon(t, f) \uparrow \wedge \forall t' < t : \delta(t', f) = \delta(t, f)).
\end{aligned}$$

i.e. at time point t, the feature f is not affected by effects, and it has the same value as when it was last affected by effects, if it was affected at all; otherwise, its value has been constant for every time point up to t. $\Box$

Note that the definition of inertia for a feature does not require that effects succeed.

We now define the property of *strict inertia* [11], i.e. that the features in some set $\mathcal{I}$ (the *inert* features) never change unless changed by an effect.

Definition 3.12 Let $\varepsilon = \langle \delta, E, \phi \rangle$ be an effect development, and let $\mathcal{I}$ be a set of features. Then ε is said to have *strict inertia* wrt. the set $\mathcal{I}$ of features iff $\forall t \in \mathcal{T}, f \in \mathcal{I} : (Inert_\varepsilon(t, f) \vee \exists v : \langle t, f, v \rangle \in E)$, or equivalently, setting

$$\begin{aligned} SI = \{ & \langle \delta, E, \phi \rangle | \forall t \in \mathcal{T}, f \in \mathcal{I} : \\ & Inert_{\langle \delta, E, \phi \rangle}(t, f) \vee \exists v : \langle t, f, v \rangle \in E \}, \end{aligned}$$

iff $\varepsilon \in SI$. The set SI is said to be *the property of strict inertia*, given $\mathcal{I}$. $\Box$

Thus, intersecting an effect law L with SI removes effect developments which do not have strict inertia, and leaves those where each feature $f \in \mathcal{I}$ satisfies that for each time point t, it is either subject to an effect, or is inert. From here on, assume that $\mathcal{I}$ contains all features except for those denoting *actions*.

What is left for this section is the special treatment of precisely these features, which says that they are true iff an effect forces them to be true; i.e. they immediately have to become false unless stated otherwise. This enables us to specify when actions have been executed, and to assume that they have *not* been executed unless explicitly specified.

Definition 3.13 Let $\varepsilon = \langle \delta, E \rangle$ be an effect development. Then ε is said to have *action omniscience* iff for all actions a, it satisfies that $\forall t \in \mathcal{T} : \delta(t, a) \Leftrightarrow \langle t, a, \mathtt{T} \rangle \in E$, or equivalently, if AO is the set of all developments $\varepsilon = \langle \delta, E \rangle$ satisfying the above requirement for every a, that $\varepsilon \in AO$. This means that features representing actions may only be true iff they are explicitly stated to be true by an effect. The set AO is said to be the *property of action omniscience*. $\Box$

Intersecting with AO imposes action omniscience, as before. We may note that we have just introduced some assumptions that Haugh [3] argues are not necessary to solve the frame problem(s). However, they are *explicitly* imposed, so they can be removed without interfering with other (explicit) assumptions.

Example 3.14 First take L_{YSP} from Example 3.7, and suppose that we wish to specify that at time 2, *load* is performed, and at time 6, *shoot* is performed. This is done by letting the law SCD (for "schedule") be defined by

$$\text{SCD} = \{\langle \delta, E, \phi\rangle | E = \{\langle 2, load, \mathtt{T}\rangle, \langle 6, shoot, \mathtt{T}\rangle\}\}.$$

Now set $L = L_{\mathrm{YSP}} \oplus \text{SCD}$, and impose each of the above assumptions: $L' = L \cap SE \cap SI \cap AO$. Thus L' is the set of effect developments where *load* is performed only at 2, shoot only at 6, these actions succeed, thereby changing *loaded*, and by the inertia assumption, changing *alive* to be false. Still we have the variation that *alive* and *loaded* may have any (fixed) values before 2, but we can require that *alive* is true and *loaded* is false by removing those effect developments from SCD (thus representing *observations*). □

4 Specification Logic

This section presents a logic using the effect law semantics, making the specification of scenarios easier than working directly with the semantics, like we did in Example 3.14.

4.1 Syntax

Definition 4.1 Set $\mathcal{V} = \bigcup_{f \in \mathcal{F}} \mathcal{V}_f$, i.e. all feature values. We assume that $\mathcal{T}$, $\mathcal{F}$ and $\mathcal{V}$ can have fixed ("interpreted") relations R_i and functions f_i on the respective sets, having arities $a(R_i) \geq 0$ and $a(f_i) \geq 0$ respectively. In particular, they are equipped with binary identity relations $=$, and nullary relations $\mathtt{T}$ and $\mathtt{F}$, always being true and false, respectively. The set $\mathcal{R}$ of terms is built up as usual in typed first order logic with only $\mathcal{T}$, $\mathcal{F}$ and $\mathcal{V}$ as types, from functions f_i above, disjoint sets of variables available for each type, and the special function symbol *val* of type $\mathcal{T} \times \mathcal{F} \to \mathcal{V}$.

Now, the set $\mathcal{S}$ of formulae is built up as usual in typed first order logic, from terms in $\mathcal{R}$, relations R_i, the special predicate symbol *Eff* of type $\mathcal{T} \times \mathcal{F} \times \mathcal{V}$, unary connective $\neg$, binary connectives $-$, $\wedge$, $\vee_x$, $\vee$, $\to$, and quantifiers $\forall$, $\exists$ and $\exists!$. Connectives have priority in the above order, except for the quantifiers, whose scopes extend to the right as far as possible. In $\neg\alpha$ and $\alpha \to \beta$, *Eff* may not occur in α.

Sometimes, we restate the formula $val(\mathsf{t}, \mathsf{f}) = \mathsf{v}$ by $Holds(\mathsf{t}, \mathsf{f}, \mathsf{v})$, and when v represents truth values of features, by $Holds(\mathsf{t}, \mathsf{f})$ for $val(\mathsf{t}, \mathsf{f}) = \mathtt{T}$ and $\neg Holds(\mathsf{t}, \mathsf{f})$ for $val(\mathsf{t}, \mathsf{f}) = \mathtt{F}$, whenever that is convenient. For syntactical identity of α and β, we write $\alpha \equiv \beta$. □

The intended meaning of some of the lesser common connectives needs to be explained: $\alpha - \beta$ means (classically) "α and not β", $\alpha \vee_x \beta$ means "α exclusive or β", and $\exists!\mathsf{x}.\alpha$ means the *unique* existence of x such that α is true.

4.2 Semantics

The semantics of the language $\mathcal{S}$ is now presented, but not as usual, by first stating an evaluation function, letting the set of models for a formula α be those where it evaluates to T. Rather, it turns out to be more natural to define the set of models directly, compositionally in terms of the parts of α. This is analogous to what can be done for classical logic, for example in $[\![\alpha \wedge \beta]\!] = [\![\alpha]\!] \cap [\![\beta]\!]$. It might be helpful to have the corresponding equations for first-order logic in mind when reading the following.

Definition 4.2 The semantics for a formula α shall be an effect law, and this is obtained by viewing an effect development $\langle \delta, E, \phi \rangle$ as a first order interpretation, interpreting the function symbol *val* by δ, the predicate symbol *Eff* by E, and all variable symbols x by $\phi(\mathsf{x})$. We denote by $V_\phi^M(\alpha)$ the *classical* evaluation function of α in the interpretation M (effect development) using variables in ϕ, and by $[\![\alpha]\!]$ the set of interpretations where α is true, evaluated using V, i.e. the *classical* model set of α, using only effect developments as models.

Now define the binary relation $=_\mathsf{x}$ on interpretations by $I =_\mathsf{x} I'$ iff I and I' are identical except for what value they assign to x, and the auxiliary function $\mathcal{X}$ for an effect law L, variable x and a value d of x's type by $\mathcal{X}(L, \mathsf{x}, d) = \{I' | I' =_\mathsf{x} I \wedge I \in L \wedge I(\mathsf{x}) = d\}$, i.e. the set of all effect developments obtained from L having d as its value for x, then abstracting away the actual value of x. Also abbreviate x's type by D_x. Now the *specification model set* of a formula α, denoted ${}^s[\![\alpha]\!]$, is defined compositionally, as follows.

- ${}^s[\![R_i(a_1, a_2, \ldots, a_n)]\!] = [\![R_i(a_1, a_2, \ldots, a_n)]\!] \cap \mathcal{E}$, where R_i is an interpreted relation
- ${}^s[\![\mathit{Eff}(\mathsf{t}, \mathsf{f}, \mathsf{v})]\!] = \{M | M(\mathit{Eff}) = \{V_\phi^M(\langle \mathsf{t}, \mathsf{f}, \mathsf{v} \rangle)\}\}$
- ${}^s[\![\neg\alpha]\!] = [\![\neg\alpha]\!] \cap \mathcal{E}$
- ${}^s[\![\alpha \wedge \beta]\!] = {}^s[\![\alpha]\!] \oplus {}^s[\![\beta]\!]$
- ${}^s[\![\alpha \vee \beta]\!] = {}^s[\![\alpha]\!] \uplus {}^s[\![\beta]\!]$
- ${}^s[\![\alpha - \beta]\!] = {}^s[\![\alpha]\!] \ominus {}^s[\![\beta]\!]$
- ${}^s[\![\alpha \vee_x \beta]\!] = {}^s[\![\alpha]\!] \otimes {}^s[\![\beta]\!]$
- ${}^s[\![\alpha \to \beta]\!] = [\![\alpha]\!] \to {}^s[\![\beta]\!]$
- ${}^s[\![\forall \mathsf{x}.\alpha]\!] = \bigoplus_{d \in D_\mathsf{x}} \mathcal{X}({}^s[\![\alpha]\!], \mathsf{x}, d)$
- ${}^s[\![\exists \mathsf{x}.\alpha]\!] = \biguplus_{d \in D_\mathsf{x}} \mathcal{X}({}^s[\![\alpha]\!], \mathsf{x}, d)$

- ${}^{s}[\![\exists!\mathsf{x}.\alpha]\!] = \bigotimes_{d \in D_{\mathsf{X}}} \mathcal{X}({}^{s}[\![\alpha]\!], \mathsf{x}, d)$

□

By the definitions of the effect law operations, we now can express effect laws using the specification logic, and know the result. Note that the semantics of quantifiers can be viewed as "infinite" versions of the connectives $\wedge$, $\vee$ and $\vee_x$, respectively, where the quantified variable varies, corresponding to classical first order semantics, where e.g. $\bigoplus$ would be replaced by $\bigcap$.

4.3 Use of the Syntax

Some examples follow.

Example 4.3 The effect law *Shoot* from Example 3.5 is described by the formula

$$\gamma_s \equiv \forall \mathsf{t}.\mathit{Holds}(\mathsf{t}, \mathit{loaded}) \wedge \mathit{Holds}(\mathsf{t}, \mathit{shoot}) \rightarrow \mathit{Eff}(\mathsf{t}, \mathit{alive}, \mathtt{F}),$$

Load from Example 3.7 by $\gamma_l \equiv \forall \mathsf{t}.\mathit{Holds}(\mathsf{t}, \mathit{load}) \rightarrow \mathit{Eff}(\mathsf{t}, \mathit{loaded}, \mathtt{T})$, and the law L_{YSP} from Example 3.7 by $\gamma_{ysp} \equiv \gamma_l \wedge \gamma_s$. Thus, $L_{\mathrm{YSP}} = {}^{s}[\![\gamma_{ysp}]\!]$. Setting

$$\gamma_{scd} \equiv \mathit{Eff}(2, \mathit{load}, \mathtt{T}) \wedge \mathit{Eff}(6, \mathit{shoot}, \mathtt{T}),$$

we thus get the intended models for the Yale Shooting Problem by

$${}^{s}[\![\gamma_{ysp} \wedge \gamma_{scd}]\!] \cap \mathit{SE} \cap \mathit{SI} \cap \mathit{AO}.$$

□

Now, an example which makes use of the continuous time domain.

Example 4.4 Suppose that we have a ball, which has the property "height", denoted h, a real valued feature representing the ball's height above the ground. We assume that the ball does not bounce when it reaches the ground. The earth's acceleration is denoted g. The syntax for specifying the effect law describing the behaviour of a ball is

$$\forall \mathsf{t}, \mathsf{t}'.(\mathsf{t}' > \mathsf{t} \wedge \mathit{newh}(\mathsf{t}') \geq 0) \rightarrow \mathit{Eff}(\mathsf{t}', h, \mathit{newh}(\mathsf{t}'))$$

where $\mathit{newh}(\mathsf{t}')$ abbreviates the term $\mathit{val}(\mathsf{t}, h) - g(\mathsf{t}' - \mathsf{t})^2/2$. Similar to above, intersecting with the assumptions yield the intended models. □

This is indeed a kind of *ramification*, like that of Reiter's natural events [8] (but more general). Also other kinds of ramification, where causal directions are well-defined and noncircular, can be described.

5 Conclusions

We have defined a compositional semantics for reasoning about action, which solves some of the problems of existing approaches, when dealing with nondeterminism. Furthermore, we manage to strictly separate assumptions about inertia and succeeding actions from the specification of effects, allowing for future generalisations e.g. to handling surprises and failing actions.

Apart from these generalisations, the *qualification problem* and that of *incorrect observations* can be formulated in this framework. Concerning tractability, we have developed an algorithm which reduces every formula of the specification logic into first order logic, relative to the fixed domains. Also, the assumptions of Section 3.1 are first order expressible, with formulae similar to the expressions used in that section, which means that the set of intended models of a scenario is always first order expressible, relative to the fixed domains.

Acknowledgements

Thanks to Erik Sandewall, Patrick Doherty and Marcus Bjäreland for very helpful comments.

References

[1] Peter Grünwald, 'Three characterisations of a causal theory of action: a very preliminary report (draft)'. Available by ftp at `ftp.cwi.nl` as `/pub/pdg/threechar.ps.Z.`, 1995.

[2] Steve Hanks and Drew McDermott, 'Default reasoning, nonmonotonic logics, and the frame problem', in *Proceedings of the National Conference on Artificial Intelligence*, pp. 328–333, (1986).

[3] Brian A. Haugh, 'Omniscience isn't needed to solve the frame problem', *International Journal of Expert Systems*, **3**(3), (1991).

[4] G. Neelakantan Kartha and Vladimir Lifschitz, 'Actions with indirect effects (preliminary report)', in *Proceedings of the 4th International Conference on Principles of Knowledge Representation*, (1994).

[5] Vladimir Lifschitz, 'Frames in the space of situations', *Artificial Intelligence*, **46**, 365–376, (1990).

[6] Fangzhen Lin, 'Embracing causality in specifying the indirect effects of actions', in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1985–1991, (1995).

[7] Tommy Persson and Lennart Staflin, ‘A causation theory for a logic of continuous change.’, in *Proceedings of the European Conference on Artificial Intelligence*, (1990).

[8] R. Reiter, ‘Natural actions, concurrency and continuous time in the situation calculus’, in *Proc. of Common Sense 96: 3rd symposium on Logical Formalizations of Commonsense Reasoning, Stanford, CA, Jan. 6-8*, (1996).

[9] Erik Sandewall, ‘Combining logic and differential equations for describing real-world systems’, in *Proceedings of International Conference on Knowledge Representation, Toronto, Canada*, (1989).

[10] Erik Sandewall, ‘Filter preferential entailment for the logic of action in almost continuous worlds’, in *Proceedings of the International Joint Conference on Artificial Intelligence*, (1993).

[11] Erik Sandewall, *Features and Fluents. The Representation of Knowledge about Dynamical Systems, volume I*, Oxford University Press, 1994. ISBN 0-19-853845-6.

[12] Lynn Andrea Stein and Leora Morgenstern, ‘Motivated action theory: a formal theory of causal reasoning’, *Artificial Intelligence*, **71**, 1–42, (1994).

[13] Michael Thielscher, ‘The logic of dynamic systems’, in *Proceedings of the International Joint Conference on Artificial Intelligence*, (1995).