Preferential Action Semantics

(Preliminary Report)

John-Jules Ch. Meyer and Patrick Doherty

Abstract

In this paper, we propose a new way of considering reasoning about action and change. Rather than placing a preferential structure onto the models of logical theories, we place such a structure directly on the semantics of the actions involved. In this way, we obtain a preferential semantics of actions by means of which we can not only deal with several of the traditional problems in this area such as the frame and ramification problems, but can generalize these solutions to a context which includes both nondeterministic and concurrent actions. In fact, the net result is an integration of semantical and verificational techniques from the paradigm of imperative and concurrent programs in particular, as known from traditional programming, with the AI perspective. In this paper, the main focus is on semantical (i.e. model theoretical) issues rather than providing a logical calculus, which would be the next step in the endeavor.

IDA Technical Report 1996
LiTH-IDA-R-96-24
ISSN-0281-4250

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

Preferential Action Semantics
(Preliminary Report)

John-Jules Ch. Meyer[*]
Intelligent Systems Group
Dept. of Computer Science
Utrecht University
P.O. Box 80089, 3508 TB
Utrecht, The Netherlands
jj@cs.ruu.nl

Patrick Doherty
University of Linköping
Dept. of Computer Science
S-581 83 Linköping, Sweden
patdo@ida.liu.se

Abstract

In this paper, we propose a new way of considering reasoning about action and change. Rather than placing a preferential structure onto the models of logical theories, we place such a structure directly on the semantics of the actions involved. In this way, we obtain a preferential semantics of actions by means of which we can not only deal with several of the traditional problems in this area such as the frame and ramification problems, but can generalize these solutions to a context which includes both nondeterministic and concurrent actions. In fact, the net result is an integration of semantical and verificational techniques from the paradigm of imperative and concurrent programs in particular, as known from traditional programming, with the AI perspective. In this paper, the main focus is on semantical (i.e. model theoretical) issues rather than providing a logical calculus, which would be the next step in the endeavor.

1 Introduction

Reasoning about action and change has long been of special interest to AI and issues of knowledge representation (see [13]). In particular, the issue of representing changes caused by actions in an efficient and economic way without the burden of explicitly specifying what is *not* affected by the actions involved and is left unchanged has been a major issue in this area, since typically this specification is huge and in some cases *a priori* not completely known. In a similar vein, one would also like to avoid explicitly stating all qualifications to actions and all secondary affects of actions. Most of the proposed solutions impose a so-called *law of inertia* on changes caused by actions which states that properties in the world tend to remain the same when actions occur unless this is known to be otherwise. Formally, the inertia assumption in AI has been treated as some kind of default reasoning which in turn has triggered a host of theories about this specific application and defeasible and nonmonotonic theories in general.

The problem that tends to arise with many of the proposed solutions is that application of the inertia assumption is generally too global, or coarse, resulting in unwanted or unintended side effects. One would like to evoke a more local or fine-grained application of inertia to the scenarios at hand and recent proposals tend to support this claim. One explanation for this *coarseness* is that typically one represents an action theory as a set of axioms and then considers a subclass of

[*]This author is partially supported by ESPRIT BRWG project No. 8319 (MODELAGE). This research was initiated during the author's leave to Linköping University (IDA), the hospitality of which is gratefully acknowledged. Moreover, this author wishes to dedicate this paper to the memory of his father B. John Meyer.

the models, the preferred models, as the theories intended meaning. This means that the effects of actions are represented or obtained in a slightly roundabout way: the action theory contains axioms from which the behavior of the actions can be deduced using the preferred models of these axioms which somehow have to capture or respect the law of inertia concerning these actions. In simple situations, this approach works fine, but it is well known that in more complex situations finding the right kinds of preferences on ones models is not only very difficult, but even claimed not to be possible.

Our claim is that this is due to the fact that the instrument of considering preferred models of complete theories is too coarse. The specification of preferred outcomes of actions is a delicate matter depending on the actions (and the environment) at hand, and should be handled at the action semantics level rather than the global logical theory describing the whole system. So, what we will do in this paper is to put preferences at the place they should be put, viz. the semantics of actions. On this level we can more succinctly *fine-tune* these preferences incorporating the mode of inertia that is needed for a particular action given a particular context (environment).

We call this way of assigning meaning to actions *preferential action semantics*, which may be contrasted with traditional preferential semantics, which in contrast can be referred to as preferential *theory* (or *assertion*) semantics. Interestingly, but very naturally, this view will lead us very close to what is studied in the area of so-called concurrency semantics, i.e. that area of computer science where models of concurrent or parallel computations are investigated. We see for instance that in this framework proposals from the AI literature dealing with action and change which use constructs such as occlusion/release ([11], [6]) get a natural interpretation with respect to the aspect of concurrency.

2 Preferential Semantics of Actions

In this section, we define a very simple language of actions[1] with which we illustrate our ideas on preferential semantics of actions. Of course, for representing *real* systems this simple language should be extended, but the current simplification will give the general idea.

We start with the set $\mathcal{FVAR}$ of feature variables and $\mathcal{FVAL}$ of feature values. Elements of $\mathcal{FVAL}$ are typically denoted by the letter d, possibly marked or subscripted.[2] Next, we define a system state σ as a function of feature variables to features values: $\sigma : \mathcal{FVAR} \rightarrow \mathcal{FVAL}$. So, for $x \in \mathcal{FVAR}$, $\sigma(x)$ yields it value. The set of states is denoted by Σ. To denote changes of states we require the concept of a variant of a state. The state $\sigma\{d/x\}$ is defined as the state such that $\sigma\{d/x\}(x) = d$ and $\sigma\{d/y\}(y) = \sigma(y)$ for $y \neq x$.

Let a set $\mathcal{A}$ of atomic actions be fixed. An atomic action $a \in \mathcal{A}$ comes with a signature indicating what variables are *framed*, which of these may nevertheless vary (are *released* from inertia) and which are definitely *set*: $a = a(\text{set}_a, \text{frame}_a, \text{release}_a)$, where $\text{set}_a, \text{frame}_a, \text{release}_a \in \mathcal{FVAR}$. We also define $\text{inert}_a = \text{frame}_a \setminus \text{release}_a$ and $\text{var}_a = \mathcal{FVAR} \setminus (\text{set}_a \cup \text{frame}_a)$.[3] The inert variables are those subject to inertia, so that it is preferred that they retain the same value; the var variables are those not subjected to inertia and are really variables in the true sense of the word. The distinction between var and released variables is a subtle one: typically when describing an action scenario some of the framed variables (which are normally subject to inertia) are temporarily released, while some variables are considered truly variable over the whole scenario. Sandewall [12] describes the three classes of frame-released, frame, and var variables as *occluded*,

[1] Actually, these are action expressions/descriptions rather than actions, but we will use the term rather loosely here.

[2] For convenience, we will assume that all feature variables range over the same set of feature values, mostly the booleans, but of course this restriction can be lifted.

[3] When it is convenient, we may also specify the inert and var variables in an action, such as e.g. $a = a(\text{set}_a, \text{inert}_a, \text{var}_a)$.

remanent, and *dependent*. Kartha and Lifschitz [6] were probably the first to recognize this three-tiered distinction, while Sandewall [10] was the first to use the frame/occluded distinction to deal properly with nondeterministic actions and actions with duration.

Given the set of atomic actions, complex actions can be formed as follows:

$$\alpha = a \mid \text{if } b \text{ then } \alpha_1 \text{ else } \alpha_2 \text{ fi} \mid \alpha_1 \oplus \alpha_2 \mid \alpha_1 + \alpha_2 \mid \alpha_1 \parallel \alpha_2 \mid \text{fail}.$$

Here, $a \in \mathcal{A}$; if b then α_1 else α_2 fi , where b is a boolean test on feature variables, represents a conditional action with the obvious meaning; $\alpha_1 \oplus \alpha_2$ stands for *restricted choice* between actions α_1 and α_2, where the release mechanism is applied to the actions α_1 and α_2 separately; $\alpha_1 + \alpha_2$ stands for an *open* or *liberal choice* between α_1 and α_2, where the release mechanism induced by the two actions α_1 and α_2 is employed for both α_1 and α_2; $\alpha_1 \| \alpha_2$ stands for the *parallel (simultaneous) performance* of both α_1 and α_2; fail denotes the *failing* action, possessing no successor states. The class of all actions is denoted by $\mathcal{A}ct$. We now introduce the class of *preferred actions* (or rather the class of preferred behaviors of actions) denoted by $\mathcal{P}ref\mathcal{A}ct = \{\alpha_\sharp \mid \alpha \in \mathcal{A}ct\}$, where $\alpha_\sharp$ expresses the *preferred* behavior of α.[4]

The formal semantics of actions is given by functions which essentially describe the way actions change states. We define a semantical function $[\![\cdot]\!] : \mathcal{A}ct \to \Sigma \to (2^\Sigma \times 2^\Sigma)$ for $a \in \mathcal{A}ct$, $\sigma \in \Sigma$. $[\![\alpha]\!](\sigma)$ denotes the set of states that computation of action α may result in, together with information about which of these states are preferred. So, $[\![\alpha]\!](\sigma) = (S, S')$, where $S' \subseteq S \subseteq \Sigma$, and S' are the preferred outcome states of α. If $S' = S$, this means that there is no preferred strict subset. In this case, we will just write $[\![\alpha]\!](\sigma) = S$. We will use the convention that when we write $\sigma' \in [\![\alpha]\!](\sigma) = (S, S')$, we refer to the membership of the superstructure S, i.e. $\sigma' \in S$. Thus, effectively, we then forget about the preferred subsets. When we explicitly want to address these, we use the function $\sharp : (2^\Sigma \times 2^\Sigma) \to 2^\Sigma$ to yield the preferred subset $\sharp(S, S') = S'$. $\sharp(S, S')$ is also denoted by $(S, S')_\sharp$.

We allow placing constraints Φ on the set of states, so that effectively, the function $[\![\cdot]\!]$ is constrained: $[\![\cdot]\!] : \mathcal{A}ct \to \Sigma_\Phi \to (2^{\Sigma^\Phi} \times 2^{\Sigma^\Phi})$, where $\Sigma_\Phi = \{\sigma \in \Sigma \mid \sigma \models \Phi\}$.[5]

We are now ready to define the semantics for atomic and complex actions in terms of the functions described above.

<u>Atomic Actions</u>

For atomic action $a = a(\text{set}_a, \text{frame}_a, \text{release}_a)$, we define its semantics as follows. First, we determine the effect of a on the variables in set_a. We assume that this is deterministic; let us denote the (unique) state yielded by this effect by σ_a. We may e.g. write $\text{set}_a = \{+x, -y\}$ when we want to express that x is set to true and y is set to false. For instance, if σ is a state containing boolean information about the feature l ("the gun is loaded or not"), and a is the action $\text{load}(\text{set}_{\text{load}} = \{+l\})$, then $\sigma_{\text{load}} = \sigma\{T/l\}$, representing that the load action sets the variable l to true.

$$[\![a(\text{set}_a, \text{frame}_a, \text{release}_a)]\!](\sigma) = (S, S')$$

where (supposing $\text{frame}_a = \{x_1, x_2, \ldots, x_m\}$, $\text{release}_a = \{x_1, x_2, \ldots, x_n\} \subseteq \text{frame}_a$, so $n \leq m$, and $\text{var}_a = \{y_1, y_2, \ldots, y_k\}$):
$S = \{\sigma_a\{d_1/x_1, d_2/x_2, \ldots, d_m/x_m, d'_1/y_1, d'_2/y_2, \ldots, d_k/y_k\} \in \Sigma_\Phi \mid d_1, d_2, \ldots, d_m, d'_1, d'_2, \ldots, d'_k \in \mathcal{FVAL}\}$ ($= \{\sigma' \in \Sigma_\Phi \mid \sigma'(z) = \sigma_a(z) \text{ for all } z \in \text{set}_a\}$) and
$S' = \{\sigma_a\{d_1/x_1, d_2/x_2, \ldots, d_n/x_n, d'_1/y_1, d'_2/y_2, \ldots, d_k/y_k\} \in \Sigma_\Phi \mid d_1, d_2, \ldots, d_m, d'_1, d'_2, \ldots, d'_k \in \mathcal{FVAL}\}$ ($= \{\sigma' \in \Sigma_\Phi \mid \sigma'(z) = \sigma_a(z) \text{ for all } z \in \text{set}_a \cup \text{inert}_a\}$).
Note that indeed $S' \subseteq S$ ($\in \Sigma_\Phi$).

[4] Note that it is senseless to talk about $(\alpha_\sharp)_\sharp$. This is not allowed by the syntax. We leave the question to future research whether nestings of preference regarding action behavior can be useful in some way.

[5] Constraints will be used to treat the ramification problem in a later section.

Although the definition looks fairly complicated, it simply states formally that the usual semantics of an action $a(\mathrm{set}_a, \mathrm{frame}_a, \mathrm{release}_a)$ consists of those states that apart from the definite effect of the action on the variables in set_a, both var and frame variables may be set to any possible value, whereas the preferred semantics (capturing inertia) keeps the inert variables the same, although the var and release variables are still allowed to vary.

Complex Actions

In the sequel, it will sometimes be convenient to use the notation $\alpha(\mathrm{set}_\alpha = X, \mathrm{frame}_\alpha = Y, \mathrm{release}_\alpha = Z)$, or simply $\alpha(\mathrm{set} = X, \mathrm{frame} = Y, \mathrm{release} = Z)$, or even $\alpha(\mathrm{set}\, X, \mathrm{frame}\, Y, \mathrm{release}\, Z)$, for the action $\alpha(\mathrm{set}_\alpha, \mathrm{frame}_\alpha, \mathrm{release}_\alpha)$, with $\mathrm{set}_\alpha = X$, $\mathrm{frame}_\alpha = Y$, and $\mathrm{release}_\alpha = Z$. In addition, the set-theoretical operators are, when needed, extended to pairs in the obvious way: $(S_1, S_1') \bullet (S_2, S_2') = (S_1 \bullet S_2, S_1' \bullet S_2')$.

The conditional and fail actions are given the following meanings:

$$[\text{if } b \text{ then } \alpha_1 \text{ else } \alpha_2 \text{ fi}](\sigma) = [\alpha_1](\sigma) \text{ if } b(\sigma) = T;\text{ and } [\alpha_2](\sigma) \text{ otherwise.}$$

$$[\text{fail}](\sigma) = (\emptyset, \emptyset).$$

Let's now consider the choice operators. The difference between restricted and liberal choice is illustrated by the following example. Suppose we have the constraint that shower on (o) is equivalent to either a hot shower (h) or a cold shower (c), i.e. $o \leftrightarrow h \vee c$. Let ho stand for the action of putting the hot shower on ($h := T$), and co for the action of putting the cold shower on ($c := T$). In the case where the restricted choice action $\mathsf{ho} \oplus \mathsf{co}$ is performed in a state where $\neg o$ ($= \neg h \wedge \neg c$) holds, we either choose to do ho in this state resulting in a state where $h \wedge o \wedge \neg c$ holds (so inertia is applied to $\neg c$), or co is chosen resulting in a state where $c \wedge o \wedge \neg h$ holds (so inertia is applied to $\neg h$). In contrast, if the liberal choice action $\mathsf{ho} + \mathsf{co}$ is performed in a state where $\neg o$, we just look at the possibilities of doing ho, co, and *possibly both*, resulting in one of the states $\{h \wedge o \wedge \neg c, \neg h \wedge o \wedge c, h \wedge o \wedge c\}$. So one may view this as if every atom o, h, or c is allowed to change value and is not subject to any inertia.

The semantics of the restricted choice operator can be stated as follows. Let the function $\mathrm{Constrain}_\Phi$ be such that it removes all states that do not satisfy the constraints Φ: $\mathrm{Constrain}_\Phi(S) = \{\sigma \in S \mid \sigma \models \Phi\}$. When no confusion arises, we may omit the subscript Φ.

$$\begin{aligned}&[\alpha(\mathrm{set}_\alpha, \mathrm{frame}_\alpha, \mathrm{release}_\alpha) \oplus \beta(\mathrm{set}_\beta, \mathrm{frame}_\beta, \mathrm{release}_\beta)](\sigma) = \\ &\quad \mathrm{Constrain}_\Phi([\alpha(\mathrm{set}_\alpha, \mathrm{frame} = (\mathrm{frame}_\alpha \cup \mathrm{frame}_\beta) \setminus \mathrm{set}_\alpha, \mathrm{release}_\alpha)](\sigma) \cup \\ &\qquad [\beta(\mathrm{set}_\beta, \mathrm{frame} = (\mathrm{frame}_\alpha \cup \mathrm{frame}_\beta) \setminus \mathrm{set}_\beta, \mathrm{release}_\beta)](\sigma)).\end{aligned}$$

The definition states that the restricted choice between α and β regards the actions α and β more or less separately. The only thing that is considered uniformly is the set of frame variables, but the release mechanism works separately for both actions α and β.

The semantics of the liberal choice operator can be stated as follows.

$$\begin{aligned}&[\alpha(\mathrm{set}_\alpha, \mathrm{frame}_\alpha, \mathrm{release}_\alpha) + \beta(\mathrm{set}_\beta, \mathrm{frame}_\beta, \mathrm{release}_\beta)](\sigma) = \\ &\quad \mathrm{Constrain}_\Phi([\alpha(\mathrm{set}_\alpha, \mathrm{frame} = (\mathrm{frame}_\alpha \cup \mathrm{frame}_\beta \cup \mathrm{set}_\beta) \setminus \mathrm{set}_\alpha, \\ &\qquad \mathrm{release} = (\mathrm{release}_\alpha \cup \mathrm{release}_\beta \cup \mathrm{set}_\beta) \setminus \mathrm{set}_\alpha)](\sigma) \cup \\ &\qquad [\beta(\mathrm{set}_\beta, \mathrm{frame} = (\mathrm{frame}_\alpha \cup \mathrm{frame}_\beta \cup \mathrm{set}_\alpha) \setminus \mathrm{set}_\beta, \\ &\qquad \mathrm{release} = (\mathrm{release}_\alpha \cup \mathrm{release}_\beta \cup \mathrm{set}_\alpha \setminus \mathrm{set}_\beta)](\sigma)).\end{aligned}$$

In this case, the situation for the liberal choice operator is considered much more uniformly in the sense that not only the set of frame variables is taken together, but also the release mechanism works in a much more uniform manner. For both actions the sets of release and set variables is added, so that inertia is less potent and more possibility of variability (also with respect to preferred outcomes) is introduced by considering joint effects of the two actions α and β.

The semantics of the parallel operator can be stated as follows.

$$\begin{aligned}
&[\![\alpha(\mathrm{set}_\alpha, \mathrm{frame}_\alpha, \mathrm{release}_\alpha) \parallel \beta(\mathrm{set}_\beta, \mathrm{frame}_\beta, \mathrm{release}_\beta)]\!](\sigma) = \\
&\quad \mathrm{Constrain}_\Phi([\![\alpha(\mathrm{set}_\alpha, \mathrm{frame} = (\mathrm{frame}_\alpha \cup \mathrm{frame}_\beta \cup \mathrm{set}_\beta) \setminus \mathrm{set}_\alpha, \\
&\qquad \mathrm{release} = (\mathrm{release}_\alpha \cup \mathrm{release}_\beta \cup \mathrm{set}_\beta) \setminus \mathrm{set}_\alpha)]\!](\sigma) \cap \\
&\qquad [\![\beta(\mathrm{set}_\beta, \mathrm{frame} = (\mathrm{frame}_\alpha \cup \mathrm{frame}_\beta \cup \mathrm{set}_\alpha) \setminus \mathrm{set}_\beta, \\
&\qquad \mathrm{release} = (\mathrm{release}_\alpha \cup \mathrm{release}_\beta \cup \mathrm{set}_\alpha \setminus \mathrm{set}_\beta)]\!](\sigma)).
\end{aligned}$$

Note the similarity with the liberal choice operator. In fact, the only thing that has changed with respect to the latter is that now *only the joint* effects of both actions are taken into consideration, where the release mechanism for both actions is again taken as liberal as possible allowing for as much interaction as possible.

Finally, we consider the preferred behavior operator $\sharp$:

$$[\![\alpha_\sharp]\!](\sigma) = ([\![\alpha]\!](\sigma))_\sharp.$$

Example

Let us consider the shower example again. The actions ho and co can be described more precisely as $\mathsf{ho}(\mathrm{set}\{+\mathsf{h}\}, \mathrm{frame}\{\mathsf{o}, \mathsf{c}\}, \mathrm{release}\{\mathsf{o}\})$ and $\mathsf{co}(\mathrm{set}\{+\mathsf{c}\}, \mathrm{frame}\{\mathsf{o}, \mathsf{h}\}, \mathrm{release}\{\mathsf{o}\})$. Recall that we have $o \leftrightarrow h \vee c$ as a domain constraint (Φ). Let σ be such that $\sigma = \{F/h, F/c, F/o\}$. Now, $[\![(\mathsf{ho} \oplus \mathsf{co})_\sharp]\!](\sigma)$ becomes
$(\mathrm{Constrain}_\Phi([\![\mathsf{ho}(\mathrm{set}\{+\mathsf{h}\}, \mathrm{frame}\{\mathsf{o}, \mathsf{c}\}, \mathrm{release}\{\mathsf{o}\})]\!](\sigma) \cup [\![\mathsf{co}(\mathrm{set}\{+\mathsf{c}\}, \mathrm{frame}\{\mathsf{o}, \mathsf{h}\}, \mathrm{release}\{\mathsf{o}\})]\!](\sigma)))_\sharp$
$= \{\sigma\{T/h, F/c, T/o\}, \sigma\{F/h, T/c, T/o\}\}$, while $[\![(\mathsf{ho} + \mathsf{co})_\sharp]\!] =$
$(\mathrm{Constrain}_\Phi([\![\mathsf{ho}(\mathrm{set}\{+\mathsf{h}\}, \mathrm{frame} = \mathrm{release} = \{\mathsf{o}, \mathsf{c}\})]\!](\sigma) \cup [\![\beta(\mathrm{set}\{+c\}, \mathrm{frame} = \mathrm{release} = \{o, h\})]\!](\sigma)))_\sharp =$
$\{\sigma\{T/h, F/c, T/o\}, \sigma\{F/h, T/c, T/o\}, \sigma\{T/h, T/c, T/o\}\}$, as expected.

In addition, consider the action $h \parallel c$ in the same setting. Intuitively, one would expect that this action should have the effect of putting the shower on with both cold and hot water. $[\![(\mathsf{ho} \parallel \mathsf{co})_\sharp]\!] = (\mathrm{Constrain}_\Phi([\![\mathsf{ho}(\mathrm{set}\{+\mathsf{h}\}, \mathrm{frame} = \mathrm{release} = \{\mathsf{o}, \mathsf{c}\})]\!](\sigma) \cap [\![\beta(\mathrm{set}\{+c\}, \mathrm{frame} = \mathrm{release} = \{o, h\})]\!](\sigma)))_\sharp$ which is equivalent to $\{\sigma\{T/h, T/c, T/o\}\}$, as desired.

3 Preferential Action Dynamic Logic (PADL)

In order to define a logic for reasoning about actions which includes their preferred interpretations, we simply take the (ordinary) dynamic logic formalism which is well known from the theory of imperative programming ([5]). Formulas in the class $\mathcal{F}orm$ are of the form $[\alpha]\phi$, where $\alpha \in \mathcal{A}ct \cup \mathcal{P}ref\mathcal{A}ct$, $\phi \in \mathcal{F}orm$, closed under the usual classical connectives.

The semantics of formulas is given by the usual Kripke-style semantics. A Kripke model is a structure $M = (\Sigma, \{R_\alpha \mid \alpha \in \mathcal{A}ct \cup \mathcal{P}ref\mathcal{A}ct\})$, where the accessibility relations R_α are given by $R_\alpha(\sigma, \sigma') \Leftrightarrow_{\mathrm{def}} \sigma' \in [\![\alpha]\!](\sigma)$.

Formulas of the form $[\alpha]\phi$ are now interpreted as usual: $M, \sigma \models [\alpha]\phi \Leftrightarrow$ for all $\sigma' : R_\alpha(\sigma, \sigma') \Rightarrow M, \sigma' \models \phi$. The other connectives are dealt with as usual. Note the special case involving formulas with preferred actions where $[\alpha_\sharp]\phi$ is interpreted as: $M, \sigma \models [\alpha_\sharp]\phi \Leftrightarrow$ for all $\sigma' : R_{\alpha_\sharp}(\sigma, \sigma') \Rightarrow M, \sigma' \models \phi \Leftrightarrow$ for all $\sigma' : \sigma' \in [\![\alpha_\sharp]\!](\sigma) \Rightarrow M, \sigma' \models \phi \Leftrightarrow$ for all $\sigma' : \sigma' \in ([\![\alpha]\!](\sigma))_\sharp \Rightarrow M, \sigma' \models \phi$.

Validity in a model, $M \models \phi$, is defined as $M, \sigma \models \phi$ for all σ. Validity of a formula, $\models \phi$, is defined as $M \models \phi$ for all models M.

Some useful validities:
$\models [\alpha](\phi \to \psi) \to ([\alpha]\phi \to \psi)$
$\models [\text{if } b \text{ then } \alpha_1 \text{ else } \alpha_2 \text{ fi}]\phi \leftrightarrow ((b \wedge [\alpha_1]\phi) \vee (\neg b \wedge [\alpha_2]\phi))$
$\models [\alpha]\phi \to [\alpha_\sharp]\phi$
$\models [\alpha]\phi \to [\alpha \parallel \beta]\phi$
$\models [(\alpha + \beta)_\sharp]\phi \leftrightarrow [\alpha_\sharp]\phi \wedge [\beta_\sharp]\phi$
$\models [(\alpha + \beta)_\sharp]\phi \to [(\alpha \oplus \beta)_\sharp]\phi$

Furthermore, as usual in dynamic logic we have that:
$\models \phi \Rightarrow \models [\alpha]\phi$.

However, some notable non-validities are:
$\not\models [\alpha_\sharp]\phi \to [(\alpha \parallel \beta)_\sharp]\phi$
$\not\models [(\alpha \oplus \beta)_\sharp]\phi \to [(\alpha + \beta)_\sharp]\phi$

4 SKIP vs. WAIT: Concurrency

Let us now briefly examine the difference between a wait action in the AI context and a skip action in imperative programming. A strong monotonic inertia assumption is implicitly built into the state transitions of imperative programming where the meaning of the skip action for example is just the identity function; $[\mathsf{skip}] = \lambda\sigma.\sigma$. For the wait action, it also holds that $[\mathsf{wait}] = \lambda\sigma.\sigma$, but in this case, the inertia assumption is weaker in the sense that the action may itself show any behavior , due to additional effects in the environment. Our approach offers the possibility of specifying this weaker notion which will even work properly in the context of unspecified concurrent actions. For example, if $\mathsf{wait} = \mathsf{wait}(\text{set} = \text{frame} = \text{release} = \emptyset)$, $\mathsf{load} = \mathsf{load}(\text{set}\{+\mathsf{l}\})$, and we consider the action $\mathsf{wait} \parallel \mathsf{load}$, we obtain $[\mathsf{wait} \parallel \mathsf{load}](\sigma) = [\mathsf{wait}(\text{set} = \text{frame} = \text{release} = \emptyset) \parallel \mathsf{load}(\text{set}\{+\mathsf{l}\})](\sigma)$ $= [\mathsf{wait}(\text{frame}\{\mathsf{l}\}, \text{release}\{\mathsf{l}\})](\sigma) \cap [\mathsf{load}(\text{set}\{+\mathsf{l}\})](\sigma) = \{\sigma\{T/l\}, \sigma\{F/l\}\} \cap \{\sigma\{T/l\}\} = \{\sigma\{T/l\}\} =$ $[\mathsf{load}](\sigma)$.

More interestingly, if we also consider the propositional fluent a, we see how the release and the law of inertia work together. Suppose $\mathsf{wait} = \mathsf{wait}(\text{frame}\{\mathsf{a}, \mathsf{l}\})$, $\mathsf{load} = \mathsf{load}(\text{set}\{+\mathsf{l}\})$. $[\mathsf{wait} \parallel \mathsf{load}](\sigma) =$ $[\mathsf{wait}(\text{frame}\{\mathsf{a}, \mathsf{l}\}) \parallel \mathsf{load}(\text{set}\{+\mathsf{l}\})](\sigma) = [\mathsf{wait}(\text{frame}\{\mathsf{a}, \mathsf{l}\}, \text{release}\{\mathsf{l}\})](\sigma) \cap \mathsf{load}(\text{set}\{+\mathsf{l}\}\text{frame}\{\mathsf{a}\})](\sigma)$. It follows that $\models (\neg l \wedge a) \to [\mathsf{wait} \parallel \mathsf{load}]\mathsf{l}$, while $\models (\neg l \wedge a) \to [(\mathsf{wait} \parallel \mathsf{load})_\sharp]\mathsf{l} \wedge \mathsf{a}$, as would be expected.

The upshot of all this is that although preferably the wait action has the same effect as the skip action, nethertheless due to the (non-specified) *concurrent* actions that are done in parallel with the wait, and of which we do not have any control, additional effects might occur.

5 Other Examples

We will start with a number of standard examples and move towards larger and more complex examples which combine the frame and ramification problems with concurrent actions.

Yale Shooting Scenario: Initially Fred is alive, then the gun is loaded, we wait for a moment and then shoot. Of course (under reasonable conditions), it is expected that Fred is dead after shooting. In our approach, this example is represented as follows: we have the features loaded (l), alive (a), and the actions $\mathsf{load} = \mathsf{load}(\text{set}\{+\mathsf{l}\}, \text{frame}\{\mathsf{a}\})$, $\mathsf{wait} = \mathsf{wait}(\text{frame}\{\mathsf{a}, \mathsf{l}\})$, and $\mathsf{shoot} =$ $\text{if } \mathsf{l} \text{ then } \mathsf{kill}(\text{set}\{-\mathsf{l}, -\mathsf{a}\}) \text{ else } \mathsf{wait}(\text{frame}\{\mathsf{a}, \mathsf{l}\}) \text{ fi}$. Now we have that $(\neg l \wedge a) \to [\mathsf{load}_\sharp](\mathsf{l} \wedge \mathsf{a})$; $(l \wedge a) \to [\mathsf{wait}_\sharp](\mathsf{l} \wedge \mathsf{a})$; and finally $(l \wedge a) \to [\mathsf{kill}_\sharp]\neg\mathsf{a}$, and hence also $(l \wedge a) \to [\mathsf{shoot}_\sharp]\neg\mathsf{a}$, so that $\models (\neg l \wedge a) \to [\mathsf{load}_\sharp][\mathsf{wait}_\sharp][\mathsf{shoot}_\sharp]\neg\mathsf{a}$.

Russian Turkey Shoot: The scenario is more or less as before, but now the wait action is replaced by a spin action: $\mathsf{spin} = \mathsf{spin}(\mathrm{frame}\{\mathsf{a}\})$, leaving the variable l out of the frame, which may then vary arbitrarily. Clearly, $\not\models (\neg l \wedge a) \rightarrow [\mathsf{load}_\sharp][\mathsf{spin}_\sharp][\mathsf{shoot}_\sharp]\neg\mathsf{a}$, since $\not\models (l \wedge a) \rightarrow [\mathsf{spin}_\sharp]\mathsf{l}$, although it is the case that $\models (l \wedge a) \rightarrow [\mathsf{spin}_\sharp]\mathsf{a}$,

Ramification: The Walking Turkey Shoot. Similar to the Yale Shooting Scenario, but now we also consider the feature walking (w) and the constraint that walking implies alive: $\Phi = \{w \rightarrow a\}$. So now we consider the action
$\mathsf{shoot} = \text{if } \mathsf{l} \text{ then } \mathsf{kill}(\mathrm{set}\{-\mathsf{l}, -\mathsf{a}\}, \mathrm{release}\{\mathsf{w}\}) \text{ else } \mathsf{wait}(\mathrm{frame}\{\mathsf{a}, \mathsf{l}\}) \text{ fi}$, and obtain $\not\models (l \wedge a) \rightarrow [\mathsf{shoot}_\sharp](\neg\mathsf{a} \wedge \neg\mathsf{w})$. In this case, inertia on w is not applied.

We now proceed to some more complicated scenarios.

Jumping into the Lake Example ([1], [4])

Consider the situation in which one jumps into a lake, wearing a hat. Being in the lake (l) implies being wet (w). So we have as a constraint $\Phi = \{l \rightarrow w\}$. If one is initially not in the lake, not wet and wearing a hat, the preferred result using inertia would be that after jumping into the lake, one is in the lake and wet, but no conclusions concerning wearing a hat after the jump can be derived. We do not want to apply inertia to the feature of wearing a hat , since it is conceivable that while jumping, one could lose one's hat. So technically, this means that the feature variable hat-on (h) is left out of the frame. (Another way of representing this, which one might prefer and which will give the same result, is viewing the frame constant over the whole scenario, including h, and then releasing h in the present situation.)

If one is in the lake and wet, we would expect that after getting out of the lake, one is not in the lake, but *still wet* in the resulting state. So, inertia would be applied to the feature wet. Furthermore, we may assume that getting out of the lake is much less violent than jumping into it, so that we may also put h in the frame. Finally, if one is out of the lake and wet, then putting on a hat would typically result in a state where one has a hat on, while remaining out of the lake and wet.

Formally, we can treat this relatively complicated scenario by means of our semantics as follows. Consider the feature variables l (being in the lake), w (being wet), h (wearing a hat), and the constraint $\Phi = \{l \rightarrow w\}$. In addition, we would need three actions.

- jump-into-lake $= \mathsf{jil}(\mathrm{set}\{+\mathsf{l}\}, \mathrm{frame}\{\mathsf{w}\}, \mathrm{release}\{\mathsf{w}\})$, where w must be released in view of the constraint $l \rightarrow w$.
- get-outof-lake $= \mathsf{gol}(\mathrm{set}\{-\mathsf{l}\}, \mathrm{frame}\{\mathsf{w}, \mathsf{h}\})$; although l is set, w is not released, since l is set to false and this does not enforce anything in view of the constraint $l \rightarrow w$.
- put-on-hat $= \mathsf{poh}(\mathrm{set}\{+\mathsf{h}\}, \mathrm{frame}\{\mathsf{l}, \mathsf{w}\},)$.

Now, applying the logic gives the desired results: $(\neg l \wedge \neg w \wedge h) \rightarrow [\mathsf{jil}_\sharp](\mathsf{l} \wedge \mathsf{w})$, and $(\neg l \wedge \neg w \wedge h) \rightarrow [\mathsf{jil}](\mathsf{l} \wedge \mathsf{w})$; $(l \wedge w) \rightarrow [\mathsf{gol}_\sharp](\neg\mathsf{l} \wedge \mathsf{w})$, and $(l \wedge w) \rightarrow [\mathsf{gol}]\neg\mathsf{l}$; $(\neg l \wedge w) \rightarrow [\mathsf{poh}_\sharp](\neg\mathsf{l} \wedge \mathsf{w} \wedge \mathsf{h})$, and $(\neg l \wedge w) \rightarrow [\mathsf{poh}]\mathsf{h}$.

What this example shows is that one still has to choose the signature of actions: what is put in the frame and what is not. This is not done automatically by the framework. We claim this to be an advantage because it provides enormous flexibility in its use, while at the same time it calls for exactness, so that the specifying of agents forces one to specify per action how things should be handled. The law of inertia (applied on non-released frame variables) takes care of the rest, so to speak.

It is important to emphasize that some of the newer approaches for dealing with directed ramification which introduce explicit causal axioms ([7],[14]) essentially encode the same types of behavior, but at the same time rule out similar flexibility in specification of actions. Thielscher ([14]) for example, claims that the frame/released approaches are limited and provides the extended circuit example as a counterexample. One should rather view frame/released approaches as the result of a compilation process which compiles causal dependencies of one form or another. In the extended version of the paper, we will discuss this issue with examples.

Lifting a Bucket of Water

One can also use preferential action semantics in cases where one has certain default behavior of actions *on other grounds than the law of inertia.* Consider the lifting of a bucket filled with water with a left and right handle by means of a robot with two arms. Let lift-left (ll) be the action of the robot's lifting the left handle of the bucket with its left arm and lift-right (lr) be the analogous action of the robot's right arm. Obviously, when only one of the two actions are performed separately, water will be spilled. On the other hand, when the two actions are done concurrently, things go alright and no water is spilled. We place a constraint on the scenario that $\neg s \leftrightarrow (l \leftrightarrow r)$.

Now, we can say that normally when lift-right is performed, water gets spilled. However, in the extraordinary case when lift-right is performed in a context where (coincidentally) lift-left is also performed, water is not spilled. This example can be represented clearly and succinctly with our semantics. One can associate with lift-right the semantics: $[\![\mathsf{lr}(\mathrm{set}\{\mathsf{r}\}, \mathrm{frame}\{\mathsf{l}\})]\!](\sigma) = (\{\sigma\{\mathsf{T/r}\}\{\mathsf{T/s}\}, \sigma\{\mathsf{T/r}\}\{\mathsf{F/s}\}\}, \{\sigma\{\mathsf{T/r}\}\{\mathsf{T/s}\}\})$, expressing that performance of lift-right leads to a state where the right arm is raised (r) and either water gets spilled or not, but that the former is preferred (on other grounds than inertia: note that s is not framed). Analogously, we can define this for lift-left, where instead of the variable r, a variable l is set to indicate the left arm is raised. So, in our dynamic logic, the result is $\models [\mathsf{lr}]\mathsf{r}$ and $\models [\mathsf{ll}]\mathsf{l}$, but $\not\models [\mathsf{lr}]\mathsf{s}$ and $\not\models [\mathsf{ll}]\mathsf{s}$. On the other hand, we do have $\models [\mathsf{lr}_\sharp]\mathsf{s}$ and $\models [\mathsf{ll}_\sharp]\mathsf{s}$. Furthermore, since $[\![\mathsf{ll} \parallel \mathsf{lr}]\!](\sigma) = [\![\mathsf{ll}(\mathrm{set}\{+\mathsf{l}\}, \mathrm{frame} = \mathrm{release} = \{\mathsf{r}\})]\!](\sigma) \cap [\![\mathsf{lr}(\mathrm{set}\{+\mathsf{r}\}, \mathrm{frame} = \mathrm{release} = \{\mathsf{l}\})]\!](\sigma) = \{\sigma\{T/l\}\{T/r\}\{F/s\}, \sigma\{T/l\}\{F/r\}\{T/s\}\} \cap \{\sigma\{T/r\}\{T/l\}\{F/s\}, \sigma\{T/r\}\{F/l\}\{T/s\}\} = \{\sigma\{T/r\}\{T/l\}\{F/s\})$, we also obtain that $\models [\mathsf{ll} \parallel \mathsf{lr}](\mathsf{r} \wedge \mathsf{l} \wedge \neg\mathsf{s})$, as desired.

6 Directions for Future Work

We would like to investigate the possibility of introducing sequences of actions by considering the class $\mathcal{ActSeq}$ given by $\beta = \alpha \mid \beta_1; \beta_2$. This would allow one to write down the outcome of a scenario such as the Yale Shooting problem as: $(\neg l \wedge a) \rightarrow [\mathsf{load}_\sharp; \mathsf{wait}_\sharp; \mathsf{shoot}_\sharp]\neg\mathsf{a}$, instead of having to resort to the (equivalent) slightly roundabout representation $(\neg l \wedge a) \rightarrow [\mathsf{load}_\sharp][\mathsf{wait}_\sharp][\mathsf{shoot}_\sharp]\neg\mathsf{a}$, as we did earlier. Note that by this way of defining action sequences, we (purposely) prohibit considering preferred sequences. Thus, something like $(\beta_1; \beta_2)_\sharp$ would now be ill-formed in our syntax, while $\alpha_{1\sharp}; \alpha_{2\sharp}$ is allowed. It remains subject to further research whether something like $(\beta_1; \beta_2)_\sharp$ could be given a clear-cut semantics and whether it would be a useful construct to have.

Surprises ([10], [11]) can also be expressed in preferential action semantics. A surprise is some outcome of an action which was not expected, so formally we can express this as follows: ϕ is a surprise with respect to action α (denoted $\mathsf{surprise}(\alpha, \phi)$) iff it holds that $[\alpha_\sharp]\neg\phi \wedge \langle\alpha\rangle\phi$. This states that although it is expected that $\neg\phi$ will hold after performing α, ϕ is nevertheless (an implausible but possible) outcome of α. For instance, in a state where Fred is alive (a), it would come as a surprise that after a wait action, he would be *not* alive: $a \rightarrow ([\mathsf{wait}(\mathrm{frame}\{\mathsf{a}\})_\sharp]\mathsf{a} \wedge \langle\mathsf{wait}(\mathrm{frame}\{\mathsf{a}\})\rangle\neg\mathsf{a})$ is indeed true with respect to our semantics.

Other interesting issues to be studied are delayed effects of actions and prediction. It will be interesting to see whether modeling delay by using a wait action with a specific duration *in parallel* with other actions would give adequate results, while prediction seems to be very much related to considering expected results of (longer) chains of actions as compared to chains of preferred actions (as briefly indicated above). Perhaps a notion of *graded typicality* of behavior might be useful in this context. We surmise that by the very nature of the $[\alpha]$ modality (related to weakest preconditions) the framework so far seems to fit for prediction but is not very suitable for postdiction or explanation of scenarios ([11]). Perhaps extending it with the notion of strongest postconditions ([2], [9], [8]) would be helpful here.

7 Related Work

We were much inspired by work by ([9], [8]). In this work the authors also attempted to employ proven verification and correctness methods and logics from imperative programming for reasoning about action and change in AI. In particular Dijkstra's wp-formalism is used. This formalism is based on the notion of weakest preconditions (and strongest postconditions) of actions and is in fact very close to the dynamic logic framework: formulas of the form $[\alpha]\phi$ are actually the same as the wlp (weakest liberal precondition) of action α with respect to postcondition ϕ. In ([9], [8]) a central role is played by the following theorem from Dijkstra and Scholten ([2]) which says that a state $\sigma \models \alpha \wedge \neg wlp(S, \neg\beta)$ iff there is a computation c under control of S starting in a state satisfying α and terminating in a state satisfying β such that σ is the initial state of c.

What all this amounts to is that when in [9], weakest (liberal) preconditions and the above theorem are used, something is stated of the form that after execution of an action α ϕ may possibly be true, which in dynamic logic is expressed as $\langle\alpha\rangle\phi(= \neg[\alpha]\neg\phi)$. Typically, this leads to too weak statements: one does not want to say that there is *some* execution of α that leads to ϕ, but that the set of *all expected* (but of course not *all*) output states satisfy some property. This is exactly what we intend to capture by means of our preferential action semantics. Another aspect that we disagree with , as the reader might suspect from the above, is that [9] uses the skip statement to express the wait action. In our view this is equating *a priori* the action of waiting with its preferred behavior (in view of the law of inertia).

Finally, we mention that the work reported in [3] is similar in spirit to ours. Here also, a distinction between typical (preferred) and possible behavior of actions is made within a dynamic logic setting. Our approach is more concrete in the sense that we directly incorporate aspects of inertia into the semantics, and, moreover, have an explicit preference operator (applied to actions) in the language. This implies that we can also speak about preferred versus possible behavior in the object language. On the other hand, we have not (yet) considered preferred paths of executions of actions as in [3]

References

[1] J. Crawford. Three issues in action. Unpublished note for the 5th Int. Workshop on Nonmonotonic Reasoning, 1994.

[2] E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, 1990.

[3] B. Dunin-Keplicz and A. Radzikowska. Epistemic approach to actions with typical effects. In Chr. Froideveaux and J. Kohlas, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Proc. ECSQARU'95*, Lecture Notes in Artificial Intelligence, pages 180–188. Springer-Verlag, 1995.

[4] E. Giunchiglia and V. Lifschitz. Dependent fluents. In *Proc. IJCAI-95, Montreal*, pages 1964–1969, 1995.

[5] D. Harel. Dynamic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2, pages 496–604. Reidel, Dordrecht, 1994.

[6] G. N. Kartha and V. Lifschitz. Actions with indirect effects (preliminary report). In *Proc. of the 4th Int'l Conf. on Principles of Knowledge Representation and Reasoning, (KR-94)*, pages 341–350, 1994.

[7] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. IJCAI-95, Montreal*, 1995.

[8] W. Łukaszewicz and E. Madalińska-Bugaj. Reasoning about action and change : Actions with abnormal effects. In I. Wachsmuth, C.-R Rollinger, and W. Brauer, editors, *Proc. KI-95: Advances in Artificial Intelligence*, volume 981 of *Lecture Notes in Artificial Intelligence*, pages 209–220. Springer-Verlag, Berlin, 1995.

[9] W. Łukaszewicz and E. Madalińska-Bugaj. Reasoning about action and change using Dijkstra's semantics for programmming languages: Preliminary report. In *Proc. IJCAI-95, Montreal*, pages 1950–1955, 1995.

[10] E. Sandewall. Features and fluents. Technical Report LITH-IDA-R-91-29, Department of Computer and Information Science, Linköping University, 1991.

[11] E. Sandewall. *Features and Fluents: A Systematic Approach to the Representation of Knowledge about Dynamical Systems.* Oxford University Press, 1994.

[12] E. Sandewall. Systematic comparison of approaches to ramification using restricted minimization of change. Technical Report LiTH-IDA-R-95-15, Dept. of Computer and Information Science, Linköping University, May 1995.

[13] E. Sandewall and Y. Shoham. Nonmonotonic temporal reasoning. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Epistemic and Temporal Reasoning*, volume 4 of *Handbook of Artificial Intelligence and Logic Programming.* Oxford University Press, 1994.

[14] M. Thielscher. Computing ramifications by postprocessing. In *Proc. IJCAI-95, Montreal*, pages 1994–2000, 1995.