

Turning an action formalism into a planner : a case study

Joachim Hertzberg, Sylvie Thiébaux

▶ **To cite this version:**

Joachim Hertzberg, Sylvie Thiébaux. Turning an action formalism into a planner : a case study. [Research Report] RR-2036, INRIA. 1993. inria-00074635

HAL Id: inria-00074635
https://inria.hal.science/inria-00074635

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Turning an Action Formalism Into a Planner: A Case Study

Joachim Hertzberg and Sylvie Thiébaux

N° 2036

Septembre 1993

PROGRAMME 3

Intelligence artificielle,
systèmes cognitifs
et interaction homme-machine



ISSN 0249-6399

INRIA
RENNES

Turning an Action Formalism Into a Planner: A Case Study

Joachim Hertzberg* and Sylvie Thiébaux**

Programme 3 — Intelligence artificielle, systèmes cognitifs et interaction homme-machine
Projet Repco

Rapport de recherche n° 2036 — Septembre 1993 — 40 pages

Abstract: The paper describes a case study that explores the idea of building a planner with a neat semantics of the plans it produces, by choosing some action formalism that is "ideal" for the planning application and building the planner accordingly. In general—and particularly so for the action formalism used in this study, which is quite expressive—this strategy is unlikely to yield fast and efficient planners if the formalism is used naïvely. Therefore, we adopt the idea that the planner approximates the theoretically ideal plans, where the approximation gets the closer, the more run time the planner is allowed. As the particular formalism underlying our study allows a significant degree of uncertainty to be modeled and copes with the ramification problem, we end up in a planner that is functionally comparable to modern anytime uncertainty planners, yet based on a neat formal semantics.

(Résumé : tsvp)

*hertz@icsi.berkeley.edu. ICSI, 1947 Center St., Berkeley, CA 94704, USA. On leave from GMD, AI Research Division, Schloss Birlinghoven, 5205 Sankt Augustin 1, Germany.
**thiebaux@irisa.fr.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 38 38 32

D'un formalisme d'actions à un planificateur : une étude de cas

Résumé : Cet article traite d'une étude de cas qui explore l'idée de construire un planificateur avec une sémantique soigneusement décrite des plans qu'il produit, en choisissant un formalisme d'actions idéal pour une application donnée, et en construisant le planificateur en conséquence. En général—et en particulier pour le formalisme d'actions utilisé dans cette étude, qui est assez expressif—il est peu probable que cette stratégie conduise à des planificateurs rapides et performants si le formalisme est utilisé naïvement. En conséquence, nous adoptons l'idée que le planificateur approxime les plans théoriquement idéaux, l'approximation devenant meilleure avec le temps de génération alloué au planificateur. Le formalisme d'actions sous-jacent à notre étude permettant la modélisation d'un degré important d'incertitude et traitant le problème de ramification, nous obtenons un planificateur fonctionnellement comparable aux planificateurs temps-réel traitant l'incertitude, maintenant basé sur une sémantique formelle.

1 Background and Plan of the Paper

There is an increasing amount of work providing logical formalizations of planning systems or of basics of such systems, e.g., [Lifschitz, 1987; Pednault, 1988; Tenenberg, 1991]. The rationale is that formally rigorous descriptions can help understanding the systems, their fundamental procedures, limitations and theoretical complexity—and that formal description can simply serve as a more efficient tool than natural language for communicating a system's essentials, as other sciences have experienced before.

There are in fact two ways to provide a planner with a neat formalization, or semantics. The first is the *ex post* way: The planner comes first, and the attempt to formalize it, later; Lifschitz's [1987] semantics of STRIPS [Fikes and Nilsson, 1971] is the most prominent example, and it also demonstrates that the planner implementors may have used implementation tricks that require a matching formalization to be more intricate than expected. The second is the *ex ante* way: The basic formalization comes first, and the planner is implemented later; an example for that direction is Pednault's [1988] formalization for determining possibly context-dependent action effects using regression, with a planner implementation provided by McDermott [1991].

However, you don't get a planner for free using either way. It is not *planners* that are formalized by an action theory, but, naturally enough, *actions* and how they change world states when applied. The purpose of an action formalism is to specify the reasoning about prerequisites and consequences of actions that an ideally rational agent should perform. Consequently, such formalisms are idealistic in the sense that they, e.g., need not take efficient implementability into account. Planner implementations, on the other hand, must be realistic, having to deal with scruffy things like heuristic search strategies, efficient domain modeling, anytime behavior, or even acceptable graphical user interfaces. It is, hence, understandable that action theoreticians and planner implementors may talk different languages.

So, when we say that a particular action formalism *underlies* a particular planner, we intend to mean only that the planner is, firstly, *correct* with respect to the formalism, i.e., the plans it generates for a planning problem have the property to be executable in every domain correctly modeled by the formalism and achieve the desired goals; we will call this property of *plans* their *correctness* and *completeness.* Secondly, the planner is *complete* with respect to the formalism, i.e., if there is in the formalism a structure (e.g., a sequence) of actions representing a way to solve a planning problem, then the planner will eventually find a corresponding correct and complete plan. As to designing your favorite graphical user interface or employing most tricky search strategies, however, the action formalism gives you and burdens you with full freedom—within the limits of correctness and completeness.

However, requiring correctness and completeness turns out to be overly strong if you want to include the possibility of designing practical planners in the *ex ante* way of construction. Therefore, we want to admit some more liberality and require only *correctness in the limit* for planners, i.e., require that plans delivered be correct and complete, given an arbitrarily high amount of computation time, but may deviate from correctness and completeness if the available time is limited; however, the deviation must in some way be predictable or describable relative to the underlying action formalism. Note that these ideas are not uncommon; they are closely related to, e.g., *anytime* algorithms [Dean and Boddy, 1988] and *bounded optimality* of [Russell and Wefald, 1991].

The following hypothesis underlies the work presented here:

> Building planners by approximating an action formalism in the *ex ante* way just sketched works in principle for any reasonable such formalism and is a generally applicable method for neat planner engineering, (1)

where "reasonable" means in particular implementable. Note that this hypothesis contradicts the view of Russel and Wefald [1991]. They say,

> that existing formal models, by neglecting the fact of limited resources for computation, fail to provide an adequate *theoretical* basis on which to build a science of artificial intelligence. [Russell and Wefald, 1991, p. 10, their emphasis]

One intended side effect of this paper is to demonstrate why we think they are wrong here: It is not the formalism that must take limited resources into account, but the interpretation of the output of a resource-bounded planning system relative to the formalism.

We do *not* believe, let alone assume, that there is *the* one universal action formalism to rely on. The variance in the required expressivity is huge for planners for different application domains involving—or not involving, respectively—numerical time, parallel action execution, incomplete situation descriptions, alternative action effects, or whatsoever. So, the problem is to find a method allowing a planner designer to choose one appropriately expressive formalism and guiding how to build a planner on it that exhibits correctness in the limit.

We know of no such method yet. The purpose of this paper, then, is to describe a case study in *ex ante* planner construction. We start from the possible worlds action formalism by Brewka and Hertzberg [in press] that is briefly described in section 2. Section 3 develops the notion of plans for the actions used and defines the concepts of correctness and completeness of plans relative to the formalism. Section 4 deals with the question of how correctness in the limit can be defined in this framework, and how to obtain it; the key issue for the definition is a function on plans that rates their "degree" of correctness and completeness. Section 5 briefly describes our actual planner implementation PASCALE2; owing to the degree of uncertainty handled by the underlying action formalism (namely, incomplete situation descriptions, context-dependent and alternative action effects) and to the requirements of correctness in the limit, our case study yields as a byproduct an anytime uncertainty planner that may be of interest in itself. Section 6 concludes and sketches some open issues.

Figure 1 summarizes the main steps of our proposed way of turning an action formalism into a planner in general (left column), the respective instances of these steps for the possible worlds formalism in particular (right column), and the respective sections in this paper that deal with the respective steps (middle column).

Single aspects of this work have been reported in [Brewka and Hertzberg, in press; Thiébaux and Hertzberg, 1992; Thiébaux *et al.*, 1993], from which texts this paper borrows occasionally.

2 The Action Formalism

Under the hypothesis (1), we could carry out our case study using any reasonable action formalism as presented, e.g., in [Sandewall, 1991]. As mentioned, the required expressivity of the planner sets lower bounds on the formalism's expressivity. To illustrate that our proposed methodology for planner construction is not restricted to classical planning—and, correspondingly, very restrictive action formalizations—we build a planner able to handle

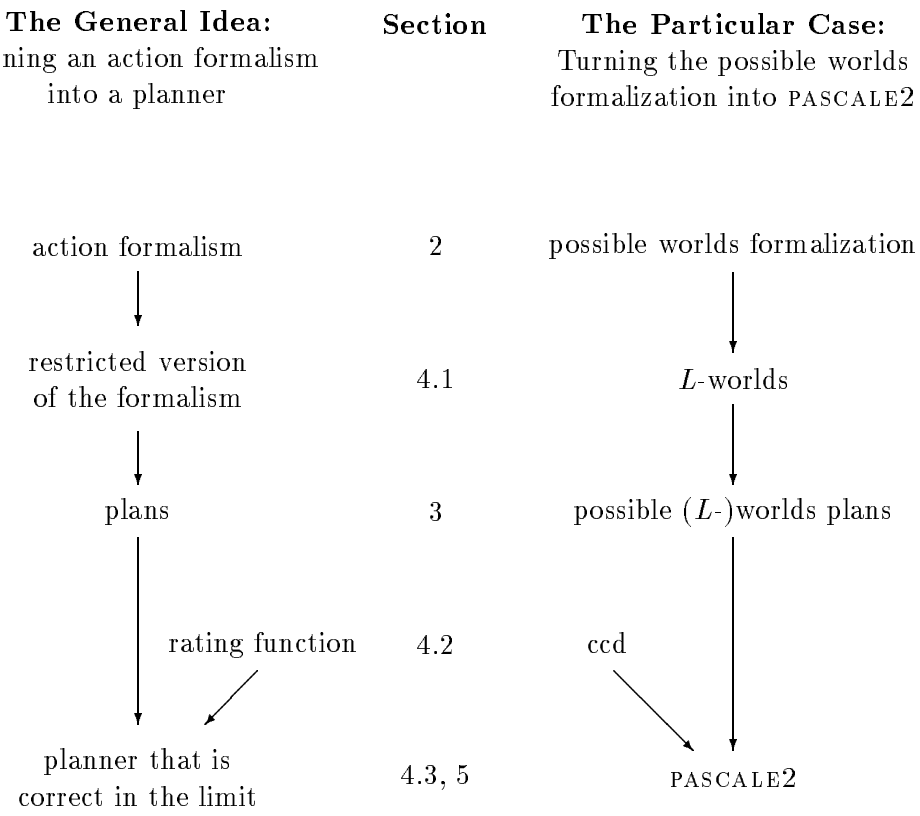


Figure 1: The steps of turning action formalisms into planners, and the structure of the paper.

- underspecified initial situations,
- context-dependency of action effects, and
- alternative effects for an action applied in one context.

In fact, we will see below that considering uncertainty of this sort matches very well the anytime behavior [Dean and Boddy, 1988] that emerges naturally from the requirement of correctness in the limit.

The particular formalism we use is inspired by Ginsberg and Smith's [1988], or rather by the correction of their formalism in [Winslett, 1988]. As it is described in full detail in [Brewka and Hertzberg, in press] and is only an instrument within this paper, our presentation here is sketchy, without further motivation or discussion. Readers familiar with [Brewka and Hertzberg, in press] may safely skip this section or just skim through, to pick up the description of the example domain used throughout the paper.

We assume a first order language $\mathcal{L}$ that represents the application domain, where we assume $\mathcal{L}$ to have a finite Herbrand base.[1] Each Herbrand model, respectively each corresponding conjunction $l_1 \wedge \cdots \wedge l_n$ of ground literals, is called a *world*. We will deliberately switch between set and conjunctive notation for worlds, i.e., we will use the notations $l_1 \wedge \cdots \wedge l_n$ and $\{l_1, \ldots, l_n\}$ interchangeably. Similarly, we interpret sets of worlds as disjunctions of worlds.

[1] The latter assumption is not strictly necessary, but simplifies the presentation.

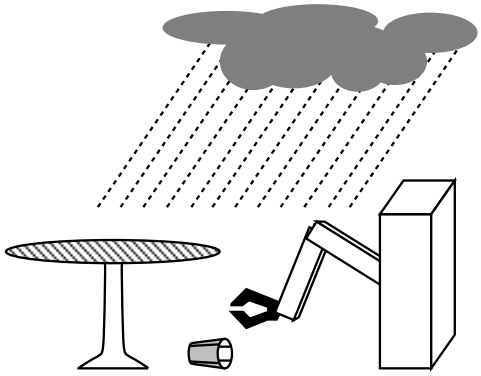


Figure 2: An illustration of the cup domain.

Consider as an example the cup domain, which is inspired by [Chrisman and Simmons, 1991]; Figure 2 is meant to vivify the imagination. The task of a robot is to manipulate a cup from some position, using several actions to be detailed later. The cup can be either on the floor (onfloor) or on a table (ontable). When on the floor, the cup can either stand upright (up), or be tipped forward with its mouth facing the robot (forward), or be tipped backward (back). Experiments take place outside; thus rainy weather (rainy) might affect the robot's performance. The language of the cup domain, $\mathcal{L}_c$, is the first order language induced by the ground atom set $\{\mathsf{onfloor}, \mathsf{ontable}, \mathsf{up}, \mathsf{forward}, \mathsf{back}, \mathsf{rainy}\}$; a world is every set of ground literals made of exactly these ground atoms. (As a notational convention, we will use a subscript c for names referring to constructs in the cup domain.)

A situation is described by a formula $s \in \mathcal{L}$. Note that such a formula does not necessarily describe the real situation in all detail, i.e., there may be many situations—and, correspondingly, multiple worlds—in which some s is true. An example for such a non-unique situation description in the cup domain is

$$s_c = \mathsf{rainy} \wedge (\mathsf{ontable} \vee \mathsf{forward})$$

Knowledge about what is true in all situations is represented in the two sets C and B. C is a set of inference rules over $\mathcal{L}$, called the *causal background knowledge,* and is used to express directed ("causal") relationships. The term Theory(C) denotes the set consisting of implications $f_1 \rightarrow f_2$ for each inference rule $f_1 \Rightarrow f_2$ in C. The *logical background knowledge* B is a set of formulas from $\mathcal{L}$ that are true in all situations; examples for such formulas are general laws, constraints, and formulas introduced for terminology. As a shorthand notation, $K = \text{Theory}(C) \cup B$ denotes the *background knowledge* in sum.

In the cup domain, we assume that bringing about up, forward, or back, respectively, causes the other two to become false and causes the cup to be onfloor (independently of whether or not it was there before); moreover, if all three of up, forward, back are turned false, this causes the cup to be ontable. This is to be expressed in the causal background knowledge C_c. Finally, we define that onfloor if and only if $\neg$ontable, i.e., we exclude any third location by definition; this must be expressed in the logical background knowledge B_c. Hence, B_c, C_c, and K_c are given by

$$B_c \;=\; \left\{\; \mathsf{onfloor} \leftrightarrow \neg\mathsf{ontable} \;\right\}$$

$$C_c = \left\{ \begin{array}{c} \mathsf{up} \Rightarrow \neg\mathsf{forward} \wedge \neg\mathsf{back} \wedge \mathsf{onfloor}, \\ \mathsf{forward} \Rightarrow \neg\mathsf{up} \wedge \neg\mathsf{back} \wedge \mathsf{onfloor}, \\ \mathsf{back} \Rightarrow \neg\mathsf{forward} \wedge \neg\mathsf{up} \wedge \mathsf{onfloor}, \\ \neg\mathsf{up} \wedge \neg\mathsf{forward} \wedge \neg\mathsf{back} \Rightarrow \neg\mathsf{onfloor} \end{array} \right\}$$

$$K_c = \left\{ \begin{array}{c} \mathsf{ontable} \leftrightarrow \neg\mathsf{onfloor}, \\ \mathsf{up} \rightarrow \neg\mathsf{forward} \wedge \neg\mathsf{back} \wedge \mathsf{onfloor}, \\ \mathsf{forward} \rightarrow \neg\mathsf{up} \wedge \neg\mathsf{back} \wedge \mathsf{onfloor}, \\ \mathsf{back} \rightarrow \neg\mathsf{forward} \wedge \neg\mathsf{up} \wedge \mathsf{onfloor}, \\ \neg\mathsf{up} \wedge \neg\mathsf{forward} \wedge \neg\mathsf{back} \rightarrow \neg\mathsf{onfloor} \end{array} \right\}$$

Only worlds consistent with the background knowledge K are possible. That intuition is made precise in the following definition:

Definition 2.1 (Possible worlds in s) *Let $s \in \mathcal{L}$ be a formula and let $K \subset \mathcal{L}$ be a set of formulas. The possible worlds in s with respect to K are all worlds consistent with K and s, i.e., the elements of the set*

$$\mathrm{Poss}_K(s) = \{w \mid w \textit{ is a world and } K \cup \{s\} \not\models \neg w\}$$

We use Poss_K as a shorthand for $\mathrm{Poss}_K(\mathit{true})$, i.e., for the set of all worlds possible with respect to K alone.

In the example of the cup domain with K_c and s_c as defined above, we have

$$\mathrm{Poss}_{K_c}(s_c) = \left\{ \begin{array}{lcl} W_1 & = & \{\mathsf{rainy}, \neg\mathsf{forward}, \neg\mathsf{back}, \neg\mathsf{up}, \neg\mathsf{onfloor}, \mathsf{ontable}\}, \\ W_2 & = & \{\mathsf{rainy}, \mathsf{forward}, \neg\mathsf{back}, \neg\mathsf{up}, \mathsf{onfloor}, \neg\mathsf{ontable}\} \end{array} \right\}$$

where the W_i are names for later reference. Poss_{K_c} consists of W_1, W_2, and the following other worlds, which we present as we will occasionally refer to them:

$$\begin{array}{lcl} W_3 & = & \{\mathsf{rainy}, \neg\mathsf{forward}, \mathsf{back}, \neg\mathsf{up}, \mathsf{onfloor}, \neg\mathsf{ontable}\}, \\ W_4 & = & \{\mathsf{rainy}, \neg\mathsf{forward}, \neg\mathsf{back}, \mathsf{up}, \mathsf{onfloor}, \neg\mathsf{ontable}\}, \\ W_5 & = & \{\neg\mathsf{rainy}, \neg\mathsf{forward}, \neg\mathsf{back}, \neg\mathsf{up}, \neg\mathsf{onfloor}, \mathsf{ontable}\}, \\ W_6 & = & \{\neg\mathsf{rainy}, \mathsf{forward}, \neg\mathsf{back}, \neg\mathsf{up}, \mathsf{onfloor}, \neg\mathsf{ontable}\}, \\ W_7 & = & \{\neg\mathsf{rainy}, \neg\mathsf{forward}, \mathsf{back}, \neg\mathsf{up}, \mathsf{onfloor}, \neg\mathsf{ontable}\}, \\ W_8 & = & \{\neg\mathsf{rainy}, \neg\mathsf{forward}, \neg\mathsf{back}, \mathsf{up}, \mathsf{onfloor}, \neg\mathsf{ontable}\} \end{array}$$

Below, actions will be defined that change situations by making certain postconditions true as a result of their application. In that context, it will be required to express that some formula (describing the original situation) is in some sense *minimally different* from some other formula (describing the resulting situation), where a third formula (the action's postcondition) is true. To make the concept of minimal difference precise, two supplementary definitions are needed, which in turn require some additional terminology.

Let w_1 and w_2 be two possible worlds. $\mathrm{Diff}(w_1, w_2)$ denotes the set of ground literals true in w_2, but not in w_1. In the cup domain, we have, for example,

$$\begin{array}{lcll} \mathrm{Diff}(W_1, W_2) & = & \{\mathsf{forward}, \mathsf{onfloor}, \neg\mathsf{ontable}\} & \text{and} \\ \mathrm{Diff}(W_2, W_1) & = & \{\neg\mathsf{forward}, \neg\mathsf{onfloor}, \mathsf{ontable}\} & \end{array}$$

For a set C of inference rules, the *C-closure* of some set of formulas F is the smallest deductively closed formula set containing F that is also closed under the inference rules of C.

As a shorthand, $F \vdash_C f$ denotes that some formula f is contained in the C-closure of F. For example, the C_c-closure of forward contains $\neg$up $\wedge$ $\neg$back $\wedge$ onfloor, as dictated by the inference rules in C_c, and hence also contains $\neg$up and $\neg$back and onfloor; rainy, e.g., is not included.

Armed with this terminology, one can now define a concept denoting a minimal set of changes transforming a world w_1 into another world w_2, given some background knowledge K. The idea is that the "essential" changes are computed using C, and B is used to determine the additional "trivial" ones, if any.

Definition 2.2 (Causal change set) *Let $K = \text{Theory}(C) \cup B$ be the background knowledge, w_1 and w_2 possible worlds, and $w_2' \subseteq w_2$. A set of formulas S is called a causal change set of (w_1, w_2), iff S is a minimal subset of $\text{Diff}(w_1, w_2)$ such that $S \cup (w_1 \cap w_2) \vdash_C w_2'$ and $w_2' \cup B \vdash w_2$.*

Note that a pair of worlds may have multiple causal change sets. In the cup domain, {forward} is the unique causal change set for (W_1, W_2).[2]

Causal change sets are now used to determine closeness between possible worlds:

Definition 2.3 (Closeness, $\prec_w$) *Let w, w_1 and w_2 be possible worlds. w_1 is closer to w than w_2, denoted $w_1 \prec_w w_2$, iff*

- $\text{Diff}(w, w_1) \subset \text{Diff}(w, w_2)$, *or*
- *every causal change set of (w, w_1) is a subset of a causal change set of (w, w_2), and not vice versa.*

We use the term w-Closest$_K(s)$ as a shorthand notation to denote the set of possible worlds wrt. K that are $\prec_w$-minimal among the worlds satisfying s.

As a cup domain example, consider the possible world W_6. The single causal change set of (W_1, W_6) is $\{\neg\text{rainy}, \text{forward}\}$. Therefore, using the results of previous examples, $W_2 \prec_{W_1} W_6$.

Finally, actions and the result of applying them are defined. The possible worlds formalization adopts the common idea that actions are described by pre and postconditions. Crucially for the formalism, actions can have different effects in different contexts (imagine the action of toggling a lightswitch that has the effect of switching the light on if it was off before, and vice versa); and actions can have alternative effects in the same context (imagine the action of tossing a coin that may result in *heads* or *tails*). The syntactic appearance of an action is then defined as follows:

Definition 2.4 (Action description) *An action description of m contexts is a structure of the form*

$$\begin{array}{lcl} [\quad Pre_1 & | & Post_{1,1}, \ldots, Post_{1,l(1)}; \\ \quad \vdots & & \\ \quad Pre_m & | & Post_{m,1}, \ldots, Post_{m,l(m)} \quad], \end{array}$$

where the preconditions Pre_i and the postconditions $Post_{i,j}$ are arbitrary formulas from $\mathcal{L}$ such that $Pre_1 \vee \cdots \vee Pre_m$ is equivalent to true, and the Pre_i are mutually exclusive.

[2] To see that {forward} is a causal change set, note that

$$\{\text{forward}\} \cup (W_1 \cap W_2) \vdash_{C_c} \{\text{rainy}, \text{forward}, \neg\text{back}, \neg\text{up}, \text{onfloor}\} =: W_2',$$

and that $W_2' \cup B_c \vdash W_2$.

As an example from the cup domain, consider the action *table2up* meant to describe moving the cup from the table to its upright position on the floor. When the cup is originally on the table, the action can produce either the intended effect, i.e., up, or it can fail in the sense that the action results in the cup's position forward (which, by K_c, implies $\neg$up). In all other contexts, the action fails in the sense that nothing is changed. Note that this way of modeling action failure is very generous; in general, failure might result in any sort of damage on the cup's or robot's side or unpredictability of the successor situation. The formalism allows this to be expressed—but we use the generous way throughout the paper for ease of presentation. The action *table2up* is then described by[3]

$$\begin{array}{lcl} \textit{table2up} & = & [\ \ \neg\mathsf{rainy} \wedge \mathsf{ontable} \ \ | \ \ \mathsf{up}, \mathsf{forward}; \\ & & \ \ \ \ \mathsf{rainy} \wedge \mathsf{ontable} \ \ | \ \ \mathsf{up}, \mathsf{forward}; \\ & & \ \ \ \ \neg\mathsf{ontable} \ \ | \ \ \textit{true} \ \ \ \ \ \ \ \ ] \end{array}$$

To fix the meaning of the syntactical action descriptions, the concept of closeness between possible worlds is used. If, for an action α, Pre_i is true in s, the idea is that α results in a set of possible worlds, each of which verify some $Post_{i,j}$, and each of which are closest to s. Formally:

Definition 2.5 (Result of an action in s) *Let $K = \mathrm{Theory}(C) \cup B$ be background knowledge, s a formula, and α an action given by an action description of m contexts. The result of applying α in s under K, denoted $r_K(\alpha, s)$, is the disjunction of all possible worlds w' satisfying the following condition: There are $w \in \mathrm{Poss}_K(S)$, $i \in \{1, \ldots, m\}$, and $j \in \{1, \ldots, l(i)\}$, such that*

- $K \cup \{w\} \models Pre_i$ *and*
- $w' \in w\text{-Closest}_K(Post_{i,j})$, *i.e., w' is $\prec_w$-minimal among the possible worlds satisfying $Post_{i,j}$.*

Recall that we consider a set of possible worlds as identical to their disjunction.

As a cup domain example, let us compute $r_{K_c}(\textit{table2up}, s_c)$. As demonstrated before, $\mathrm{Poss}_{K_c}(s_c)$ consists of the two possible worlds W_1 and W_2. Applying *table2up* in W_2 is uninteresting in the sense that it does not lead to any changes (as $\neg$ontable is true in W_2). Consequently, W_2 is an element of the result.

Then, consider W_1, i.e., context number 2 of *table2up* gets applied as Pre_2 is true. For $Post_{2,2}$, i.e., the formula forward, we did all the necessary calculation in the examples before. The $\prec_{W_1}$-minimal world satisfying forward is good old W_2. That means, applying *table2up* in W_1 may lead to W_2, namely, if $Post_{2,2}$ happens to occur. For $Post_{2,1}$, i.e., the formula up, one can check that the possible world W_4 is $\prec_{W_1}$-minimal among those satisfying up. In sum, $r_{K_c}(\textit{table2up}, s_c) = \{W_2, W_4\}$.

Note that in the description of *table2up*, it is unnecessary to specify that the weather is unaffected and that the cup is not on the table any more. This is an example of the formalism's dealing with the frame and ramification problems.

[3] You may think the two different contexts $\neg$rainy $\wedge$ ontable and rainy $\wedge$ ontable with identical postconditions seem odd. You are right. Normally, one could unite them under the identical precondition ontable. However, we will later, when dealing with limited correctness, inject additional information into the action description, specifying that the action is more likely to fail in rainy weather (as the cup may get slippery, say). So, the distinction simply anticipates later enhancements of our application example.

$$
\begin{array}{lcll}
\mathit{back2up} & = & [\quad \neg\mathsf{rainy} \wedge \mathsf{back} & | \quad \mathsf{up}; \\
 & & \quad\;\; \mathsf{rainy} \wedge \mathsf{back} & | \quad \mathsf{up},\ \mathit{true}; \\
 & & \quad\;\; \neg\mathsf{back} & | \quad \mathit{true} \quad] \\
\mathit{spin} & = & [\quad \mathsf{forward} \vee \mathsf{back} & | \quad \mathsf{forward},\ \mathsf{back}; \\
 & & \quad\;\; \neg(\mathsf{forward} \vee \mathsf{back}) & | \quad \mathit{true} \quad] \\
\mathit{wait} & = & [\quad\;\; \mathsf{rainy} & | \quad \neg\mathsf{rainy},\ \mathit{true}; \\
 & & \quad\;\; \neg\mathsf{rainy} & | \quad \mathsf{rainy},\ \mathit{true} \quad]
\end{array}
$$

Figure 3: Additional actions for the cup domain.

3 Plans

We now turn to the issues of defining plans for the type of actions as used in the possible worlds formalization, and we will define correctness and completeness of plans, and correctness and completeness of planning procedures accordingly. We are still on the theory level here, not yet at implementation: It is the purpose to provide the concepts in terms of which the planner implementation can then be described and evaluated.

We give the definitions for problem description, plan, plan correctness, and plan completeness suitable for the possible worlds formalism here, but it should be emphasized that these notions are essential for building a planner based on *any* action formalism, although they will look different for different formalisms. So, the point here in view of the general methodology for turning action formalisms into planners is: the respective definitions must be provided in a form suitable for the respective formalism. The other notions we give here (planner, planner correctness, and planner completeness) are independent of the possible worlds formalism and may be used literally for building planners on any other one.

We start with the definition of a planning problem description. As this definition is pretty classical, it should be understood without lengthy explanation:

Definition 3.1 (Planning problem description) *Let $\mathcal{L}$ be a first order language. A planning problem description (or problem, for short, if this causes no confusion) is a quadruple $\Psi = (s, g, K, A)$, where*

- *$s \in \mathcal{L}$, the initial situation, is a formula,*
- *$g \in \mathcal{L}$, the goal, is a formula,*
- *$K \subseteq \mathcal{L}$, the background knowledge, is a set of formulas, and*
- *A, the action inventory, is a set of action descriptions.*

Remember that we tacitly assume the background knowledge K to consist of $\mathrm{Theory}(C) \cup B$ for some C, B throughout the paper, even if we do not mention C and B explicitly.

This is the point to present the other cup domain actions to prepare for stating the full action inventory for problems to come. In addition to *table2up*, we have the actions shown in Figure 3, with the following intuition behind:

back2up To move the cup from the back position to the up position. The action is guaranteed to work if it is not rainy; if it *is* rainy, it may succeed or fail (changing nothing); in all contexts where ¬back holds, it fails (again, changing nothing).

spin To (possibly) change the cup from the forward or back position into the forward or back position, where a change may occur from either of the two to either of the two. The action fails, i.e., nothing changes, if ¬(forward ∨ back) holds.

wait To just wait and do nothing. The weather may or may not change from rainy to ¬rainy, or vice versa.[4]

Our favorite cup domain problem for the rest of this text is then

$$\Psi_c = (s_c, \mathsf{up}, K_c, \{table2up, back2up, spin, wait\}),$$

That means, the problem is to get into a situation where up is true, starting from s_c, applying some of the actions, with K_c as the background knowledge.

Let us now turn to defining plans. In the framework of the possible worlds action formalization, plans can obviously not have the structure of classical plans, i.e., a set of actions and a strict linear or non-linear order on this set; this structure is inappropriate if actions, like all of the above, may yield non-unique successor situations. We need something different, then.

Our plan definition is guided by the following idea. Even if a plan cannot be given as an action sequence, it should after all direct what action to execute next, once the executor finds itself in a situation matching a certain possible world. Moreover, the plan should tell which possible worlds are expected to result from executing the action, according to the domain representation. Note that there are only finitely many possible worlds. Hence, a plan is a finite structure, consisting of possible worlds and actions; it directs which action to execute in each possible world expected to emerge; and it tells which possible worlds are expected to result from each such action execution.

We represent a plan as a bipartite directed graph, consisting of T-nodes and W-nodes. Each T-node is meant to represent an action occurrence (or *task*), and each W-node a possible world; each node is labeled with the action whose occurrence it represents, or with the world it represents, respectively.

Definition 3.2 (Plan) *Let $\Psi = (s, g, K, A)$ be a planning problem description, and let start be an action such that start $= [true \mid w_1, \ldots, w_n]$, where $\{w_1, \ldots, w_n\} = \mathrm{Poss}_K(s)$ A plan Π for Ψ is a bipartite directed graph consisting of labeled T- and W-nodes with a unique root, where:*

1. *The root is a T-node called Start; it is labeled with start. Every other T-node is labeled with an element of A.*
2. *Every W-node is labeled with an element of $Poss(K)$. No two W-nodes are labeled with the same world.*

[4] Note that this may be considered ontological cheating. While all other actions' effects can be viewed as *effected* by the respective actions, this is certainly not true for a weather change by waiting. Although we think it can consistently be included, the possible worlds formalization does not provide a means to express external events that may or may not occur independently from executing actions. The only excuse for using this cheated formulation is that it works here.