UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

LEHRSTUHL VIII
KÜNSTLICHE INTELLIGENZ

Turning an Action Formalism Into a Planner—A Case Study

LS–8 Report 8

Joachim Hertzberg **Sylvie Thiébaux**

Dortmund, March 1994

Forschungsberichte des Lehrstuhls VIII (KI),
Fachbereich Informatik
der Universität Dortmund

ISSN 0943-4135

Anforderungen an:

Universität Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

Research Reports of the unit no. VIII (AI)
Computer Science Department
of the University of Dortmund

ISSN 0943-4135

Requests to:

University of Dortmund
Fachbereich Informatik
Lehrstuhl VIII
D-44221 Dortmund

e-mail: reports@ls8.informatik.uni-dortmund.de
ftp: kimo.informatik.uni-dortmund.de:pub/Reports

Turning an Action Formalism Into a Planner— A Case Study

LS–8 Report 8

Joachim Hertzberg **Sylvie Thiébaux**[1]

Dortmund, March 1994

Universität Dortmund
Fachbereich Informatik

[1]IRISA, Campus de Beaulieu, Avenue du Général Leclerc, 35042 Rennes, France

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 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 is based on a neat formal semantics.

This paper is to appear in the *Journal of Logic and Computation,* 1994.

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., [15, 18, 22]. 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 [15] semantics of STRIPS [9] 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 [18] formalization for determining possibly context-dependent action effects using regression, with a planner implementation provided by McDermott [16].

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 [7] and *bounded optimality* of [19].

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 [19]. 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. [19, 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 [3] 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 [3, 23, 24], from which texts this paper borrows occasionally. Note that the formalism from [3] is only used as a tool here. Consequently, discussing its strengths and weaknesses or relating it to comparable formalisms is out of the scope of this paper, and we refer to the original article.

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 [20]. 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

- 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 [7] that emerges naturally from the requirement of correctness in the limit.

The particular formalism we use is inspired by Ginsberg and Smith's [11], or rather by the correction of their formalism in [26]. As it is described in full detail in [3] and is only an instrument within this paper, our presentation here is sketchy, without further motivation or discussion. Readers familiar with [3] may safely skip this section or just skim through, to pick up the description of the example domain used throughout the paper.

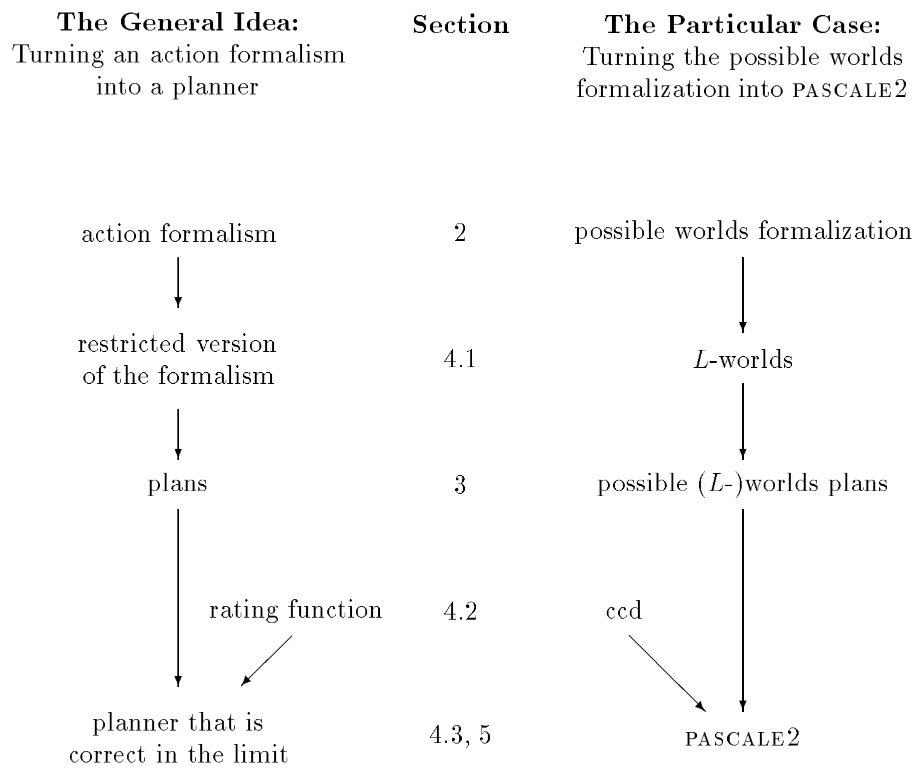


Figure 1: The steps of turning action formalisms into planners, and the structure of the paper.

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.

Consider as an example the cup domain, which is inspired by [6]; 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 ($\mathsf{onfloor}$) or on a table ($\mathsf{ontable}$). When on the floor, the cup can either stand upright (up), or be tipped forward with its mouth facing the robot ($\mathsf{forward}$), 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 any 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

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

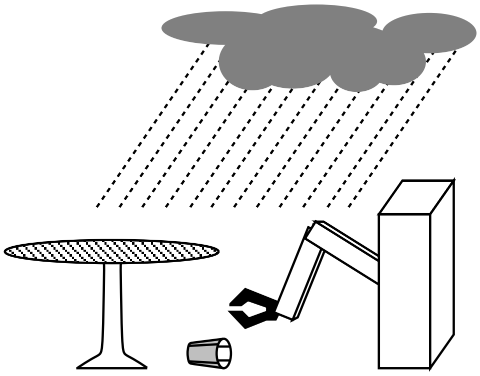


Figure 2: An illustration of the cup domain.

$\mathcal{L}$ 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, $\mathsf{forward}$, or back, respectively, causes the other two to become false and causes the cup to be $\mathsf{onfloor}$ (independently of whether or not it was there before); moreover, if all three of up, $\mathsf{forward}$, back are turned false, this causes the cup to be $\mathsf{ontable}$. This is to be expressed in the causal background knowledge C_c. Finally, we define that $\mathsf{onfloor}$ if and only if $\neg\mathsf{ontable}$, 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

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

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

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

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

Definition 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

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

We use Poss_K as a shorthand for $\text{Poss}_K(\mathit{true})$, 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

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

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:

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

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

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

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 $\mathsf{forward}$ contains $\neg\mathsf{up} \wedge \neg\mathsf{back} \wedge \mathsf{onfloor}$, as dictated by the inference rules in C_c, and hence also contains $\neg\mathsf{up}$ and $\neg\mathsf{back}$ and $\mathsf{onfloor}$; 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 (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, $\{\mathsf{forward}\}$ 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 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 $w\text{-Closest}_K(f)$ as a shorthand notation to denote the set of possible worlds wrt. K that are $\prec_w$-minimal among the worlds satisfying f.

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

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

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

As a cup domain example, consider the possible world W_6. The single causal change set of (W_1, W_6) is $\{\neg\mathsf{rainy}, \mathsf{forward}\}$. 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 4 (Action description) An action description of m contexts is a structure of the form

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

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.

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 $\mathsf{forward}$ (which, by K_c, implies $\neg\mathsf{up}$). 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]

$$\begin{array}{rcrcl} \mathit{table2up} & = & [\quad \neg\mathsf{rainy} \wedge \mathsf{ontable} & \mid & \mathsf{up}, \mathsf{forward}; \\ & & \mathsf{rainy} \wedge \mathsf{ontable} & \mid & \mathsf{up}, \mathsf{forward}; \\ & & \neg\mathsf{ontable} & \mid & \mathit{true} \qquad] \end{array}$$

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 5 (Result of an action in s) Let $K = \text{Theory}(C) \cup B$ 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 set of all possible worlds w' satisfying the following condition: There are $w \in \text{Poss}_K(S)$, $i \in \{1, \ldots, m\}$, 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 $r_{K_c}(\mathit{table2up}, s_c)$. As demonstrated before, $\text{Poss}_{K_c}(s_c)$ 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 $\neg\mathsf{ontable}$ is true in W_2). Consequently, W_2 is an element of the result.

[3] You may think the two different contexts $\neg\mathsf{rainy} \wedge \mathsf{ontable}$ and $\mathsf{rainy} \wedge \mathsf{ontable}$ with identical postconditions seem odd. You are right. Normally, one could unite them under the identical precondition $\mathsf{ontable}$. 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.

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, $r_{K_c}(table2up, s_c) = \{W_2, W_4\}$.

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 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 6 (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 the background knowledge K is assumed to equal 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 $\neg$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 $\neg$(forward $\vee$ back) holds.

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

Figure 3: Additional actions for the cup domain.

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

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

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 7 (Plan) Let $\Psi = (s, g, K, A)$ be a planning problem description, and let *start* be an action such that $\textit{start} = [\textit{true} \mid w_1, \ldots, w_n]$, 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.

3. All leaves are W-nodes.

[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.

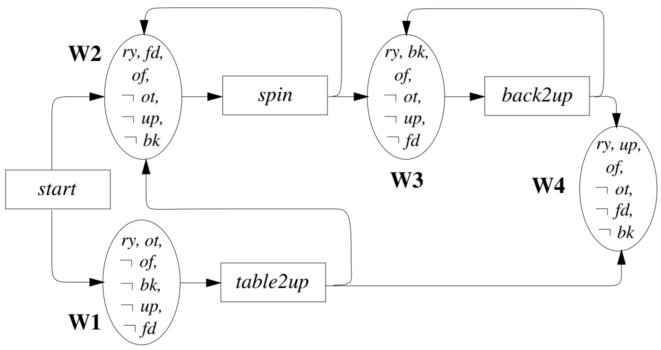


Figure 4: Plan Π_c for Ψ_c. The atoms ry, fd, bk, up, of, ot, respectively, abbreviate rainy, forward, back, up, onfloor, ontable, respectively.

4. T-nodes have only W-nodes as successors.
5. Every non-leaf W-node has exactly one T-node as successor; every non-root T-node has exactly one W-node predecessor.
6. From every node, there is at least one path to a leaf.

In Figure 4, we show a cup domain plan for the example problem Ψ_c defined earlier. T-nodes are represented by boxes, W-nodes by ellipses. Both node types are labeled with the respective corresponding tasks or worlds. The plan is to be interpreted as follows: If the world W_1 is true at the start of the plan execution, then apply *table2up*. As demonstrated earlier, the two worlds W_2 and W_4 may result; consequently, the respective W-nodes are the successors of the respective T-node. Determine which of the possible outcomes of *table2up* is the actual outcome, and proceed until you eventually reach a leaf.

Π_c has one single leaf, namely, the one labeled with W_4. Considering the part of Π_c that consists only of the *Start* node and W_1, W_2, this is also a plan; it contains the two leaves labeled with W_1, W_2.

As these plans are only part of the demonstration substrate in our case study for planner construction, we do not go into detail concerning the underlying execution model and the possibilities to use them for execution monitoring, replanning, or reusing old plans. The interested reader may find hints about this in [23].

The example plan Π_c is rather exceptional, given the simple plan definition that we have: Π_c is meaningful in the sense that it correctly models the postconditions of all actions labeling its T-nodes, as applied in their respective predecessor worlds. The plan definition does not enforce that; it allows for W-nodes that have absolutely nothing to do with the worlds resulting from applying their predecessor T-nodes.

Consequently, we have to define plans that are "meaningful". We do so by defining correctness and completeness for plans relative to the planning problem description Ψ. Note that we do not attempt to formulate correctness with respect to the real world (or some relevant section of it); that would have to be done for Ψ. It could be done along the lines that Lifschitz [15] has drawn for defining the soundness of a STRIPS system, but we simply assume in this text that planning problem descriptions are proven to be sound with respect to the world section they are supposed to model.

We start with correctness. The intuition behind it is to say that if a plan says that something is or may be the case, then it really is or may be the case. In particular, if the plan execution proceeds as a plan tells it, then the plan will rightly predict what may result from applying an action in some situation, every action it directs to execute is in fact executable, and if it directs to stop the plan execution, then a situation will have emerged in which the goal formula is true.

(Remember that all this can, of course, only be guaranteed for a plan, if the planning problem description itself was correct wrt. the real world domain.)

Note that we require only "partial" correctness here in the sense that the goal is true if the plan directs the execution to stop; in general, we cannot guarantee that every plan execution as guided by some plan will eventually lead to such a stop. The reason is that plans may contain loops out of which there is *syntactically* a path to some leaf node (as required in item 6 of definition 7), but we cannot guarantee that they will *in fact* eventually be left. As an example, see the *spin* loop in Π_c in Figure 4: If *spin*ning does in fact always result in forward, then executing the plan will cycle infinitely, if W_2 is reached once. Note, however, that the source of the trouble is the unsoundness of the domain description in this case. For sound descriptions, we have indeed total correctness, guaranteeing that the plan execution will eventually stop.

The correctness definition is given in steps: first for individual nodes, and then for plans.

Definition 8 (Node correctness, plan correctness) Let Π be a plan for the planning problem description $\Psi = (s, g, K, A)$.

Root correctness: *Start* $\in \Pi$ is correct wrt. Ψ iff for every *Start* successor node ω labeled with w in Π: $w \in \mathrm{Poss}_K(s)$

Non-root T-node correctness: Let τ be a non-root T-node in Π, labeled with α, and let ω be its predecessor in Π, labeled with w. τ is correct wrt. Ψ iff for every successor ω' of τ in Π, ω' is labeled with w' such that $w' \in r_K(\alpha, w)$.

Leaf correctness: A leaf $\omega \in \Pi$ labeled with w is correct wrt. Ψ iff $w \cup K \vdash g$.

Plan correctness: Π is correct wrt. Ψ, iff

1. *Start* $\in \Pi$ is correct wrt. Ψ;
2. every non-root T-node in Π is correct wrt. Ψ; and
3. every leaf node in Π is correct wrt. Ψ.

The concept of plan *completeness* is related to plan correctness; the idea is to say: If something may happen at plan execution according to Ψ, then the plan does already respect it. That means in particular, the plan must represent all possible worlds of the initial situation, and it must contain all results of actions that it directs to apply. Note that the issue is *not* that a plan must direct what to do in every possible world in the domain or tell what the result is of applying every action in each of the possible worlds it contains. As a plan is a plan, it deals only with the effects of the particular actions it proposes to apply in the particular possible worlds it contains.

Definition 9 (Node completeness, Plan completeness) Let Π be a plan for the planning problem description $\Psi = (s, g, K, A)$.

Root completeness: *Start* $\in \Pi$ is complete wrt. Ψ iff for all $w \in \mathrm{Poss}_K(s)$ there exists a successor ω of *Start* in Π such that ω is labeled with w.

Non-root T-node completeness: Let τ labeled with α be a non-root T-node in Π, and let ω labeled with w be its predecessor in Π. τ is complete wrt. Ψ iff for all $w' \in r_K(\alpha, w)$, there exists a successor ω' of τ in Π such that ω' is labeled with w'.

Plan completeness: Π is complete wrt. Ψ, iff

1. *Start* $\in \Pi$ is complete wrt. Ψ; and
2. every non-root T-node in Π is complete wrt. Ψ.

Obviously, a correct and complete plan is the ideal plan that one would like a planner to generate, namely, a "solution" to the planning problem, and these two requirements are the background for the vast majority of papers about planning—albeit mostly implicit. The reason to make them explicit here is, of course, that we want to be able to deal with controlled relaxations of plan correctness and completeness later; this will be the key for achieving limited correctness of planners.

The notions of correctness and completeness apply analogously to plan generating procedures, or *planners,* not only to plans. In our context, a planner is an entity that, given a planning problem description, returns plans after a number of computation steps. These steps may be time ticks or numbers of calls to a basic procedure; more generally, the steps can be interpreted as any monotonically increasing time function. For simplicity, we assume a discrete measure over the natural numbers here.

To formally define a planner, we would, strictly speaking, have to specify a language for formulating problem descriptions on that the planner operates. From the theoretical viewpoint we take until now, this is merely a technical matter, which we omit here. We tacitly assume that all problem descriptions are given in some appropriate language, involving the first order language for describing the domain and the language for describing the actions. Moreover, we assume the existence of types natural, problem description, and plan containing objects that are natural numbers, problem descriptions, and plans, respectively. Under the theoretical view that we are still having here, a planner is then somewhat impoverishedly defined as:

Definition 10 (Planner) A planner is a pair $(\mathcal{P}, A_\mathcal{P})$ of functions. $\mathcal{P}$, the *plan generation function,* is of the type

$$\textsf{problem description} \longrightarrow \textsf{natural} \longrightarrow \textsf{plan}$$

$A_\mathcal{P}$, the *availability function,* is a function of type problem description $\longrightarrow$ natural set, such that $\mathcal{P}(\Psi)$ is a total function on $A_\mathcal{P}(\Psi) \subseteq \mathbb{N}$ for every problem Ψ.

Intuitively, $\mathcal{P}$ is the planning procedure itself, and $A_\mathcal{P}$ defines the set of steps at which $\mathcal{P}$ has a plan for a problem available. An obviously interesting special case is a planner for which $A_\mathcal{P} = \mathbb{N}$, i.e., a planner that has an output available after any number of steps—an *any-step planner,* as one could call it. In fact, this is the direct analog to the now-famous anytime planners as first discussed in [7]; if time is considered to be discrete, then the any-step property generalizes the anytime property.

The intuitive idea behind planner correctness or completeness, then, is simple: A planner is *correct* wrt. some criterion to be specified, if and only if all the plans it delivers for a planning problem meet the criterion; and it is *complete,* if and only if it eventually generates all plans for the problem that meet some other to-be-specified criterion.

The obvious question is what these ominous criteria are supposed to be. In general, they can specify everything you like; for example, one could define correctness and completeness of planners wrt. the criterion that plans contain exactly 17 tasks (but are not necessarily correct and complete). The criterion that we will hard-wire into the definitions of both planner correctness and completeness is supposed to be a bit more useful, at least as long as we remain on the theory side: it is—somewhat unsurprisingly—plan correctness and completeness. That means that a correct planner will deliver only correct and complete plans, and that a complete planner will eventually deliver all correct and complete plans for a given problem. The respective definitions are straightforward:

Definition 11 (Planner correctness) A planner $(\mathcal{P}, A_\mathcal{P})$ is correct iff the plan $\mathcal{P}(\Psi)(t)$ is correct and complete wrt. Ψ for all problems Ψ and for all $t \in A_\mathcal{P}(\Psi)$.

Definition 12 (Planner completeness) A planner $(\mathcal{P}, A_\mathcal{P})$ is complete iff, for all problems Ψ and for all plans Π that are correct and complete wrt. Ψ, there exists $t \in A_\mathcal{P}(\Psi)$ such that $\mathcal{P}(\Psi)(t) = \Pi$.

Note that it is very easy to design planners that are *either* correct *or* complete. An example for a trivially correct planner is one that returns no plan at all; all plans it returns are correct and complete—it just returns no plan. An example for a trivially complete planner is one that enumerates all plans (be they correct and complete or not); it will eventually also generate all correct and complete plans. Hence, the interesting matter is to design planners that are correct *and* complete.

At least, that is what pure theory tells. As exposed in the introduction of this paper, the issue here is to develop planners that are correct *in the limit;* for this task, the "ideal" correctness as just defined will just serve as a reference point. Defining this more liberal version of correctness is what we will do in Section 4.

4 Correctness, Ltd.

We now turn to the problem of using the action formalism for planner construction. As said earlier, in view of our goal to achieve planners that are only correct in the limit relative to the formalism, we do not require that such a planner use a "direct" implementation of it. Given that such formalisms may often be designed for criteria other than efficient implementability, this would be unnecessarily restrictive. Instead, one may use a restriction of the formalism, or a formalism implementation that in itself only approximates the pure formalism in the task of, say, determining action effects. It is only the planner whose behavior we constrain: In the limit, it must be correct; and meanwhile, its incorrectness must be describable relative to the ideal, as objectified by the formalism. The formalism is a yardstick, not the Holy Grail.

Consequently, we have to do three things in order to achieve a well-defined concept of limitedly correct planners:

1. *Choose a convenient subset of the formalism.* In case that the formalism is not implemented in full generality, the restriction chosen must be characterized. Independently from that, if the provided implementation just approximates the formalism's results, the approximation behavior must be described. For the particular case of the possible worlds formalization, we describe a restriction on subsets of full worlds that models the full formalism correctly under some restrictions we will also specify; we will not use approximations.

2. *Define a rating for incomplete and incorrect plans.* In case that the planner is supposed to return incorrect and/or incomplete plans (at least after few steps), these must be rated in order to guarantee at least an asymptotic convergence of the planner outputs towards correct and complete plans. For the particular case of possible worlds plans, the rating we describe is based on a notion of utility of nodes, and takes the probability of reaching them into account.

3. *Define correctness in the limit.* Using the previous definitions, this definition of planner behavior proves to be straightforward. Describing the particular case of our possible worlds planner PASCALE2 is postponed until Section 5.

The three parts of this section deal with these three issues in turn; in particular, they exemplify them in terms of the possible worlds action formalization and plans defined earlier.

4.1 Restricting the Formalism

4.1.1 Restricting Formalisms in General

The issue here is, then, to start looking at the underlying action formalism under the view of implementability, and to restrict it accordingly, should this be deemed necessary. In this context, two questions might require clarification.

The first is whether substantially restricting a formalism in a well-defined way or providing a well-defined approximation doesn't in fact yield a new formalism, which could then be directly used as a basis for the planner. In a way, this is true. However, it is a matter of empirical observation (and of the target readership of the respective papers) that formal theories of action and the typically only implicitly specified calculi underlying practical planners are of considerably different nature. We do not think this is necessarily so, but there are good reasons for the empirically observed gap, given the difference of aims in designing a theory and in designing efficient code. This section, then, is to acknowledge that this gap will exist in many, if not all cases, and that it must be bridged for constructing planners in the way we propose here. The restricted version of the formalism or an approximate calculus for it may of course be of its own interest, not only from the practical side, but also theoretically.

The second question is whether action formalizations provide sufficient expressivity for describing a domain in general and its actions in particular, for being suitable as a formal basis for more modern planners. For example, few interesting modern planners will describe actions just by preconditions and postconditions, but there will be time, a difference between preconditions and subgoals, plots, protection intervals, and others, see [12] for an overview. There are two answers to this question. First, even if many existing action formalisms (like the possible worlds formalism that still lacks incorporating, e.g., numerical time, concurrency, or external events) are severely restricted in their expressivity, this need not stay that way; explicitly confronting them with the needs of planning applications may even provide a push for examining greater expressivity. Second, from the viewpoint of formalism, different features of an operator description language may well be mapped onto the same formalism feature. For example, it makes perfect sense to pragmatically differentiate between goals and preconditions in an operator in SIPE [25]; however, when mapping the planner to the theory, they may both be mapped onto preconditions. The heuristic search behavior of the planner takes advantage of the difference, but this behavior is not to be mapped back onto the formalism.

We will now continue presenting the case of our case study, and describe the restriction of the formalism used further on.

4.1.2 Restricting the Possible Worlds Formalism

For the particular planner in this case study, we will go the way of using a mild simplification of the possible worlds formalization that is, however, equivalent to the original under certain restrictions. Of this, we will then use a relatively straightforward implementation.

By definition, there are only finitely many possible worlds. But finitely many can still be a lot, especially as the number of worlds grows exponentially with the cardinality of the finite Herbrand universe of $\mathcal{L}$. Usually, only a fraction of all worlds is possible, but a fraction of an exponentially growing number can still grow exponentially. Consequently, the size of worlds should be something to worry about as we start turning towards building planners. We will sketch a way to both reduce the size (although not the number) of the possible worlds and practically save dealing with the rules in the causal background knowledge C. Part of this material is also drawn from [3], to which we refer to further details.

There are two key ideas to the reduction. The first is to save computation by computing results of an action not for the possible worlds as defined, but for smaller "worlds" based only on a subset L of the Herbrand universe of $\mathcal{L}$, called *L-worlds.* For example, we could choose

$$L_c = \{\mathsf{rainy}, \mathsf{forward}, \mathsf{back}, \mathsf{up}\}$$

in the cup domain. Possible L-worlds can then be defined like possible worlds in Definition 1; just substitute "world" by "L-world". We use the symbol $|_L$ as the notation of the restriction of some set or function to L.

The second key idea is to choose L such that it allows the correct action effects to be computed without using the inference rules of the causal background theory C, but from the formula set K alone. To start with, define the new versions of causal change set, closeness, and $r_K|_L(\alpha, s)$ (i.e., action results restricted on L-worlds), respectively, from the old definitions 2, 3, and 5, respectively, by again substituting "L-world" for "world", replacing every occurence of C by $\emptyset$, and every occurence of B by $B \cup \text{Theory}(C)$. (Note that K does not change, then.) Hence, determining the possible worlds closest to some given world w and verifying some formula f, i.e., $w\text{-Closest}_K(f)$, reduces in its L-restricted form to comparing differences between w and other possible worlds verifying f. To sum up, the above-described changes for the L-world version of the formalisms entail that

- the causal change set definition (Definition 2) is not needed any more
- the L-world version of the closeness definition (Definition 3) says that a possible L-world v_1 is closer to a possible L-world v than a possible L-world v_2 iff $\text{Diff}(v, v_1) \subset \text{Diff}(v, v_2)$.

The definitions of plans, correctness, and completeness then apply analogously to L-worlds.

An obvious requirement for L is that an L-world capture the "essence" of a corresponding "full" world in the sense that it uniquely determine a full world, taking the background knowledge into account. This is made precise in the following definition.

Definition 13 (Spanningness of L) A subset L of ground atomic formulas of $\mathcal{L}$ is called spanning wrt. the background knowledge K, iff for every possible L-world v there is a unique possible world w such that $v \cup K \vdash w$.

In the following, we assume that every L be spanning w.r.t. its respective K. In particular, it can be verified that L_c is spanning wrt. K_c.

In some cases, working with the L-worlds variant of the possible worlds formalism yields results that are logically equivalent to working with the original formalism. To give an example, let us compute the result of applying *table2up* under K_c in s_c, restricted on L_c, i.e., $r_{K_c}|_{L_c}(\mathit{table2up}, s_c)$. The possible L-worlds in s_c are

$$\text{Poss}_{K_c}|_{L_c}(s_c) = \left\{ \begin{array}{lcl} V_1 & = & \{\mathsf{rainy}, \neg\mathsf{forward}, \neg\mathsf{back}, \neg\mathsf{up}\}, \\ V_2 & = & \{\mathsf{rainy}, \mathsf{forward}, \neg\mathsf{back}, \neg\mathsf{up}\} \end{array} \right\}$$

These L-worlds correspond to the worlds W_1, W_2 in the original example using full worlds. Note that, as an effect of L_c's spanningness, the missing literals regarding $\mathsf{ontable}$ and $\mathsf{onfloor}$ can be deduced using K_c. Like in the original example, applying *table2up* to V_2 simply yields V_2, so we restrict our considerations on V_1. Here, the second precondition of *table2up* is true. Hence, we must find $\prec_{V_1}$-minimal possible L-worlds satisfying up and $\mathsf{forward}$, respectively.

Focusing on $\mathsf{forward}$, V_2 is a possible L-world in which it is true; $\text{Diff}(V_1, V_2)$ is $\{\mathsf{forward}\}$; clearly, V_2 is $\prec_{V_1}$-minimal according to the new closeness definition. (Recall that causal change sets are not used here.) Hence, it is an element of the result. Analogously, we obtain the possible L-world $V_4 = \{\mathsf{rainy}, \neg\mathsf{forward}, \neg\mathsf{back}, \mathsf{up}\}$ from the other postcondition formula up. In sum, $r_{K_c}|_{L_c}(\mathit{table2up}, s_c) = \{V_2, V_4\}$.

Now, as $K_c \cup V_2 \vdash W_2$ and $K_c \cup V_4 \vdash W_4$, we happen to find:

$$K_c \cup r_{K_c}(\mathit{table2up}, s_c) \text{ is logically equivalent to } K_c \cup r_{K_c}|_{L_c}(\mathit{table2up}, s_c).$$

This result is not incidental: the structure of the cup domain ensures that the equivalence holds for all its actions and situations. Proving this claim is mostly a technical matter; as it is just of marginal interest for the main point of this paper, we exile its proof to Appendix B.

The point to note is that we have identified a simplification of the possible worlds formalization for which we need to use only L-worlds and can forget about the C-closure.

This simplification can be implemented considerably more efficiently, owing mostly to the decrease in size of the L-worlds. Therefore, the planner we construct operates on L-worlds. In exchange, this planner can safely be applied only to domains and Ls for which using L-worlds is equivalent to using "full" worlds with C-closure. In Appendix B, we give sufficient conditions under which this equivalence holds in a planning domain (these conditions apply to the cup domain).

Restricting the formalism in this way is no theoretical necessity for constructing planners from action formalisms; but we believe it will often be helpful to speed them up. Remember, however, that this restriction is not the only, and probably the less important idea for operating planners under realistic run time constraints—the more important one being correctness in the limit, to be defined in Section 4.3.

4.2 Rating Plans

4.2.1 Defining a Rating Function in General

Having chosen the appropriate restriction of the formalism, i.e., of the possible worlds formalism in our case, the issue is now to define a rating for incorrect and incomplete plans, on which to base later the planner's asymptotic convergence towards delivering correct and complete plans.

The general idea of a plan rating function is very simple. We want to have a function that, given a plan, determines the "degree" of its correctness and completeness, supplementing the sharp notions of correctness and completeness with a gradual valuation for plans that are not correct and complete—i.e., the vast majority of plans that planners practically deal with. Normalizing the values of rating functions to the real interval $[0, 1]$, we get:

Definition 14 (Rating function) Let Ψ be a planning problem description. A rating function for Ψ is a total function ϱ_Ψ mapping plans for Ψ to $[0, 1]$, such that for every plan Π:

$$\varrho_\Psi(\Pi) = 1 \text{ iff } \Pi \text{ is correct and complete wrt. } \Psi$$

For brevity, we again skip defining the language for specifying planning problem descriptions. In effect, we sloppily speak of some ϱ as a *family* of rating functions, to express that ϱ_Ψ is a rating function for every problem Ψ.

Note that we do not require that rating functions are the only functions evaluating the "quality" of plans. In particular, a rating function does not select between different correct and complete plans; measures like plan execution cost, or plan generation time, however, are practically important for a planner to deliver "good" plans, whatever the definition of "good" is in detail. We assume a rating function can be a component of an overall plan evaluation measure, but make no requirements as to whether or not this is the case, whether or not additional plan evaluation functions exist, or how they look like. For defining planner correctness in the limit, rating functions are required, and only these.

An obvious example for a rating function—or even family of rating functions—is the sigmoid function Σ assigning 1 to all correct and complete plans, and 0 to all others. As we will below define a considerably more sophisticated rating function for possible world plans, mapping plans to the whole $[0, 1]$ interval, one might suspect that such a sophisticated function is in some respect generally "better" than Σ. This is wrong. There may be domains where plans that are not correct and complete in the strong sense, are of absolutely no value, making Σ the perfect rating function. The point is that the domain modeler *defines,* of what value a non-correct-and-complete plan is by defining a rating function.

This ends the general part of this Section 4.2. All the rest concerns defining a suitable rating function for the case of the possible worlds plans as defined in Section 3. For defining the function, we pursue the following idea. Given the points at which the possible worlds formalism allows for uncertainty, i.e., for multiple possible worlds of the start situation and different alternative postconditions of actions applied in a possible model, we allow (not force!) the user to inject information about the a-priori and relative probabilities of possible worlds. This information is then used for handling possible worlds by applying results of Nilsson's probabilistic logic [17]. We describe this in Section 4.2.2. In Section 4.2.3, we transfer this probability information to plans; this is done using basic techniques from Markov chain theory and utility theory.[5] In Section 4.2.4, we finally define the particular rating functions on possible worlds plans.

[5] All definitions in the sections 4.2.2 and 4.2.3 will be given for L-worlds. They apply for "full" worlds accordingly.

Let us emphasize that the choice of using probability information to develop a plan rating is special to the case of our case study; it is in no way required by our view of planner construction on some action formalism. Alternative ways to develop such a rating for planners based on the possible worlds formalism in particular might include using possibility theory [8] or some variant of Spohn's [21] model; in general, for planners based on other formalisms, completely different, maybe considerably simpler ways may be appropriate. The point is: one has to choose some such way, and we demonstrate one that is particularly appropriate for the possible worlds formalism and leads to somewhat interesting planners.

And let us express a mild warning, before starting on this way. Remember that the possible worlds action formalization allows coping with missing information, e.g., about the start situation. It is then reasonable to demand that a plan rating cope with the acquisition of additional, uncertainty-reducing information by updating the rating accordingly, i.e., that the rating be dynamic in this respect. This does not require highly sophisticated math in our case, but defining the rating takes more effort than one would expect for, say, some reasonable rating function operating on classical nonlinear plans. The case we study is just not the simplest conceivable one.

4.2.2 Injecting Probability Information into the Possible Worlds Formalism

As for using probability theory to quantify beliefs in our action formalism, we start by assuming that the a-priori probability of some domain facts are given as a set P of probability values for the corresponding sentences in $\mathcal{L}$. The *probabilistic background knowledge*, called P, expresses the constraints, if any, that must hold for the probability distribution on possible L-worlds at any time, in absence of information beyond K. For the cup domain, we have

$$P_c = \{\ p(\mathsf{rainy}) = 0.4,\ \ p(\mathsf{forward} \vee \mathsf{back}) = 0.7,\ \ p(\mathsf{ontable}) = 0.2\ \}$$

expressing, e.g., that, lacking further information, the weather has a 40% chance to be rainy.

Results from Nilsson's probabilistic logic can be used to compute the a-priori probability distribution p over the possible L-worlds space, that strictly reflects K and P, no more, and no less. The main result we use from Nilsson's work is that the probability of a sentence is the sum of the probabilities of the possible L-worlds in this sentence.[6] p is defined as follows:

Definition 15 (A-priori probability distribution) Let K and P be the logical and the probabilistic background knowledge, respectively. The a-priori probability distribution over the possible L-worlds space, noted p, is the probability distribution defined by:

$$p(true) = 1 = \sum_{v \in \ \mathrm{Poss}_K|_L} p(v) \tag{2}$$

$$\forall p(s) \in P \ \ p(s) = \sum_{v \in \ \mathrm{Poss}_K|_L(s)} p(v) \tag{3}$$

for which the entropy, defined as $-\sum_{v \in \ \mathrm{Poss}_K|_L} p(v) \log p(v)$, is maximal.

Equation (2) expresses that a tautology has truth-probability 1, and Equation (3) that p must comply with the probability values given in P. In general, these two equations still induce an infinity of probability distributions. Among them, we select the p with maximal entropy, because this distribution assumes minimal additional information beyond K and P. The probability distribution p for the cup domain is shown in Figure 5 and can be computed following the lines given in [4]. Lacking further knowledge, it constitutes the robot's belief about the current situation of the world.

[6] Nilsson's result applies primarily to "full" possible worlds. However, it is also applicable to L-worlds here, because spanningness ensures that possible L-worlds are mutually exclusive and exhaustive not only with respect to L, but also to $\mathcal{L}$.

	$V \in \mathrm{Poss}_{K_c}\|_{L_c}$	$p(V)$
V_1	{ rainy, ¬forward, ¬back, ¬up }	0.08
V_2	{ rainy, forward, ¬back, ¬up }	0.14
V_3	{ rainy, ¬forward, back, ¬up }	0.14
V_4	{ rainy, ¬forward, ¬back, up }	0.04
V_5	{ ¬rainy, ¬forward, ¬back, ¬up }	0.12
V_6	{ ¬rainy, forward, ¬back, ¬up }	0.21
V_7	{ ¬rainy, ¬forward, back, ¬up }	0.21
V_8	{ ¬rainy, ¬forward, ¬back, up }	0.06

Figure 5: A-priori probability distribution p for the cup example

It is straightforward to revise this probability distribution in order to reflect additional information about the current world situation. If $s \in \mathcal{L}$ describes this situation, our belief that a possible L-world corresponds to the actual situation is revised using Bayesian conditioning, which can be viewed as the probabilistic counterpart of belief revision in the sense of [10]. The revised probability distribution p_s on L-worlds is such that $p_s(v) = p(v|s)$, where $p(v|s)$ denotes the conditional probability of v given s. Using Bayes's theorem, this can easily be shown to be equivalent to the following definition:

Definition 16 (Revised probability distribution) Let p be the a-priori probability distribution over the possible L-worlds space, K the background knowledge, and $s \in \mathcal{L}$ a formula. The revision of p given that s holds, noted p_s, is the probability distribution defined by

$$p_s(v) = \begin{cases} \frac{p(v)}{\sum_{v' \in \ \mathrm{Poss}_K|_L(s)} p(v')} & \text{if } v \in \mathrm{Poss}_K|_L(s) \\ 0 & \text{otherwise.} \end{cases}$$

For example, if the current situation is described by s_c, the real situation of the world corresponds to one of the two possible L-worlds V_1 and V_2 with respective probability:

$$\begin{array}{rcccccc} p_{s_c}(V_1) & = & \frac{p(V_1)}{p(V_1)+p(V_2)} & = & \frac{0.08}{0.22} & \simeq & 0.36 \\ p_{s_c}(V_2) & = & \frac{p(V_2)}{p(V_1)+p(V_2)} & = & \frac{0.14}{0.22} & \simeq & 0.64 \end{array}$$

We now turn to the problem of computing the probabilities of possible L-worlds that result from performing an action. For that purpose, we assume that each action postcondition has associated the probability that this postcondition is achieved when the action is performed. Hence, actions become structures of the form:

$$\begin{array}{lll} [& Pre_1 & | \ (Post_{1,1}, p_{1,1}), \ldots, (Post_{1,l(1)}, p_{1,l(1)}); \\ & \vdots & \\ & Pre_m & | \ (Post_{m,1}, p_{m,1}), \ldots, (Post_{m,l(m)}, p_{m,l(m)}) \quad], \end{array} \tag{4}$$

where $p_{i,j}$ is the probability that executing the action in context i leads to $Post_{i,j}$. For each context i, we furthermore assume that

- the postconditions are mutually exhaustive, i.e., $\sum_{j=1}^{l(i)} p_{i,j} = 1$
- the postconditions are mutually exclusive, i.e., for any two postcondition $Post_{i,j_1}$ and $Post_{i,j_2}$ and for any L-world v where Pre_i holds, we have

$$v\text{-Closest}_K|_L(Post_{i,j_1}) \ \cap \ v\text{-Closest}_K|_L(Post_{i,j_2}) = \emptyset$$

$$
\begin{array}{lclrcll}
\mathit{table2up} & = & [& \neg\mathsf{rainy} \wedge \mathsf{ontable} & \mid & (\mathsf{up}, 0.8),\ (\mathsf{forward}, 0.2); & \\
 & & & \mathsf{rainy} \wedge \mathsf{ontable} & \mid & (\mathsf{up}, 0.6),\ (\mathsf{forward}, 0.4); & \\
 & & & \neg\mathsf{ontable} & \mid & (\mathit{true}, 1) &] \\
\mathit{back2up} & = & [& \neg\mathsf{rainy} \wedge \mathsf{back} & \mid & (\mathsf{up}, 1); & \\
 & & & \mathsf{rainy} \wedge \mathsf{back} & \mid & (\mathsf{up}, 0.8),\ (\mathit{true}, 0.2); & \\
 & & & \neg\mathsf{back} & \mid & (\mathit{true}, 1) &] \\
\mathit{spin} & = & [& \mathsf{forward} \vee \mathsf{back} & \mid & (\mathsf{forward}, 0.5),\ (\mathsf{back}, 0.5); & \\
 & & & \neg(\mathsf{forward} \vee \mathsf{back}) & \mid & (\mathit{true}, 1) &] \\
\mathit{wait} & = & [& \mathsf{rainy} & \mid & (\neg\mathsf{rainy}, 0.1),\ (\mathit{true}, 0.9); & \\
 & & & \neg\mathsf{rainy} & \mid & (\mathsf{rainy}, 0.1),\ (\mathit{true}, 0.9) &]
\end{array}
$$

Figure 6: The cup domain actions with associated postcondition probabilities.

The last condition states that, starting from possible L-world v, the same possible L-world will never be produced via two different postconditions of the action context that holds in v. Both conditions can easily be ensured without loss of generality with respect to the original definition of action descriptions (Definition 4) by adding appropriate contexts. To continue the cup domain example, the postcondition probabilities of its actions are given in Figure 6.

Upon learning that action α is applied in a situation where s holds, our belief about the possible L-worlds space is updated. We compute the probability distribution $p_{(\alpha,s)}$ over the possible L-worlds space resulting from the performance of α in s, using Lewis's imaging [14], which can be viewed as the probabilistic counterpart of updates in the sense of [13].

Definition 17 (Updated probability distribution) Let p be the a-priori probability distribution over the possible L-worlds space, K the background knowledge, $s \in \mathcal{L}$ a formula, and let p_s be the revision of p given that s holds. Let α be an action as in (4) with exhaustive and mutually exclusive postconditions. The update of p_s given that α is applied in s, noted $p_{(\alpha,s)}$, is the probability distribution defined by

$$
p_{(\alpha,s)}(v') = \sum_{v \in \mathrm{Poss}_K|_L(s)} \left(p_s(v) \sum_{j=1}^{l(i)} p(v' \mid v\text{-}\mathrm{Closest}_K|_L(\mathit{Post}_{i,j}))\ p_{i,j} \right),
$$

where $K \cup v \vdash \mathit{Pre}_i$, and $p(v' \mid v\text{-}\mathrm{Closest}_K|_L(\mathit{Post}_{i,j}))$ is the conditional probability of v', given that the postcondition $\mathit{Post}_{i,j}$ is achieved by applying α in v.

The value of $p(v' \mid v\text{-}\mathrm{Closest}_K|_L(\mathit{Post}_{i,j}))$ can be computed as the revision of $p(v')$, given that $v\text{-}\mathrm{Closest}_K|_L(\mathit{Post}_{i,j})$—considered as a disjunction of conjunctions—holds. Since all actions in the cup domain have the property of yielding exactly one L-world per effect alternative in every precondition context,[7] $v\text{-}\mathrm{Closest}_K|_L(\mathit{Post}_{i,j})$ is always a singleton, and hence this probability is either 0 or 1. Furthermore, since the postconditions of the same context are mutually exclusive and exhaustive, each i will have exactly one $\mathit{Post}_{i,j}$, for which this probability is 1. Thus, things are simple for the cup domain. When applying, e.g., *table2up* in s_c, the situation corresponds to

[7] In Appendix B, we show that this property is a consequence of spanningess of and alternative-wise result uniqueness as defined in Definition 25.

one of the possible L-worlds V_4 or V_2 with the respective probabilities

$$\begin{array}{rcl} p_{(table2up,s_c)}(V_4) & = & p_{s_c}(V_1) \times (1 \times 0.6 + 0 \times 0.4) + p_{s_c}(V_2) \times 0 \times 1 \\ & \simeq & 0.6 \times 0.36 \simeq 0.22, \\ p_{(table2up,s_c)}(V_2) & = & p_{s_c}(V_1) \times (0 \times 0.6 + 1 \times 0.4) + p_{s_c}(V_2) \times 1 \times 1 \\ & \simeq & 0.4 \times 0.36 + 1 \times 0.64 \simeq 0.78. \end{array}$$

4.2.3 Reckoning Probability Information in Possible Worlds Plans

Using the notions just described, we can use the probability information provided by the user for computing the updated probability distributions $p_{(\alpha,s)}$, telling which possible worlds result from applying α in s with which probability. The question we tackle now is, how to assemble these distributions to determine the average number of times an action will be performed when executing a plan, or the probability to end up in one of its leaf nodes.

The basic tools we use here are Markov chains. For those readers unfamiliar with Markov chain theory, we give a brief introduction in Appendix A, which they are advised to consult before continuing with this section. All other readers should run into no problems from just skipping Appendix A.

The basic idea of mapping plans to Markov chains is to associate a plan with an absorbing Markov chain whose state set includes the plan's T-nodes set. Transient states of the chain correspond to the T-nodes in the plan. Its absorbing states are interpreted to denote that the plan execution is over, which may occur in two cases:

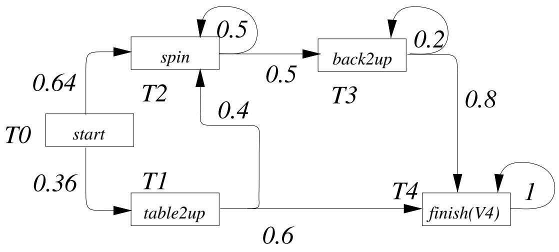
- the current world situation is represented by a leaf W-node of the plan, i.e., the plan does not prescribe to apply any subsequent action in the current situation, either because the corresponding W-node is incorrect, or because the situation it represents matches the goal;
- the current world situation is among those that should have been expected as a result of a given action, but is not represented as a W-node successor of a T-node labeled with this action in the plan, i.e, this T-node is incomplete.

To cope with these cases, we introduce two types of absorbing Markov states, additionally to states representing T-nodes. *Finish* states are labeled with dummy actions applied in L-worlds corresponding to W-node leaves of the plan, and *Inc* states are labeled with dummy actions that are applied when the current world situation is not represented as a W-node that a complete plan would have to include. The probability law of the Markov chain follows the probabilities provided by the action formalism. Before giving the constructive definition how to generate the associated Markov chain of a plan, let us give an example.

Figure 7 shows the Markov chain associated with the L-worlds version of the correct and complete plan Π_c from Figure 4,[8] and its transition matrix. In Π_c, there is a transition from the state labeled with *start* to the T-node labeled with *spin* through the L-world V_2. Since $p_{s_c}(V_2) = 0.64$, there is a transition in the Markov chain from the state labeled with *start* to the state labeled with *spin*, with probability 0.64. The unique leaf of Π_c is labeled with V_4. Hence, we have one unique *Finish* state in the Markov chain. Since V_4 is obtained from *back2up* applied in V_3 and from *table2up* applied in V_1, and since $p_{(back2up,V_3)}(V_4) = 0.8$ and $p_{(table2up,V_1)}(V_4) = 0.6$, we have two transitions leading to the state labeled with *finish*(V_4) in the chain: one with probability 0.8 from the state labeled with *back2up*, and one with probability 0.6 from the state labeled with *table2up*.

Before giving an example of an incorrect and incomplete plan, let us first formally define the chain associated with an arbitrary plan.

[8] L-worlds version means: It is essentially the same plan like Π_c, only with its possible worlds W_i exchanged by its corresponding possible L-worlds V_i.



$$
\begin{array}{c}
\\ T_0 \\ T_1 \\ T_2 \\ T_3 \\ \hline T_4
\end{array}
\left(
\begin{array}{ccccc}
T_0 & T_1 & T_2 & T_3 & \multicolumn{1}{|c}{T_4} \\
0 & 0.36 & 0.64 & 0 & \multicolumn{1}{|c}{0} \\
0 & 0 & 0.4 & 0 & \multicolumn{1}{|c}{0.6} \\
0 & 0 & 0.5 & 0.5 & \multicolumn{1}{|c}{0} \\
0 & 0 & 0 & 0.2 & \multicolumn{1}{|c}{0.8} \\ \hline
0 & 0 & 0 & 0 & \multicolumn{1}{|c}{1}
\end{array}
\right)
$$

Figure 7: The Markov chain associated with the L-worlds version of Π_c, and its transition matrix. 0 probability transitions are omitted in the graphical representation.

Definition 18 (Markov chain associated with a plan) Let $\Psi = (s, g, K, A)$ be a planning problem description, P the associated probabilistic background knowledge, and Π a plan for Ψ. For any T-node τ in Π and any non-root T-node τ' in Π, let

$$
\begin{aligned}
Succ(\tau, \Pi) &= \{v \mid \text{there exists a successor } \nu \text{ of } \tau \text{ in } \Pi \text{ labeled with } v\}) \\
Pre(\tau', \Pi) &= \text{the label of the predecessor of } \tau' \text{ in } \Pi
\end{aligned}
$$

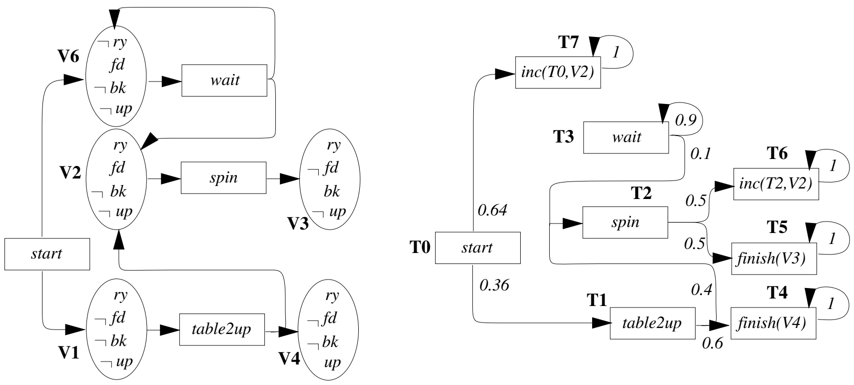
Let furthermore $\mathcal{T}(\Pi)$ be the set of T-nodes in Π, and let $\mathcal{T}'(\Pi)$ be defined as follows[9]

$$
\begin{array}{rcll}
\mathcal{T}'(\Pi) & = & \{ \ Finish(v) & \mid \ v \text{ is the label of a leaf of } \Pi \quad \} \\
& \cup & \{ \ Inc(Start, v) & \mid \ v \in \mathrm{Poss}_K|_L(s) \setminus Succ(Start, \Pi) \quad \} \\
& \cup & \{ \ Inc(\tau, v) & \mid \ \tau \in \Pi \text{ is a non-root } T\text{-node labeled with } \alpha \\
& & & \quad \text{and } v \in r_K|_L(\alpha, Pre(\tau, \Pi)) \setminus Succ(\tau, \Pi) \quad \}
\end{array}
$$

The Markov chain associated with Π, noted by the triple $\langle \{X_t, t = 0, 1, \ldots\}, \mathcal{T}(\Pi) \cup \mathcal{T}'(\Pi), \pi \rangle$, is the family $\{X_t, t = 0, 1, \ldots\}$ of random variables taking values in the set of states $\mathcal{T}(\Pi) \cup \mathcal{T}'(\Pi)$, such that the conditional probability distribution π of X_{t+1} is defined as:

$$
\pi(X_{t+1} = \tau' \mid X_t = \tau) =
\begin{cases}
p_s(Pre(\tau', \Pi)) & \text{for } \tau = Start \text{ and } \tau' \in \mathcal{T}(\Pi) \\
p_s(v) & \text{for } \tau = Start \text{ and } \tau' = Finish(v) \\
p_s(v) & \text{for } \tau = Start,\ \tau' = Inc(Start, v) \\
p_{(\alpha, Pre(\tau,\Pi))}(Pre(\tau', \Pi)) & \text{for } \tau \in \mathcal{T}(\Pi) \text{ labeled with } \alpha \text{ and } \tau' \in \mathcal{T}(\Pi) \\
p_{(\alpha, Pre(\tau,\Pi))}(v) & \text{for } \tau \in \mathcal{T}(\Pi) \text{ labeled with } \alpha \text{ and } \tau' = Finish(v) \\
p_{(\alpha, Pre(\tau,\Pi))}(v) & \text{for } \tau \in \mathcal{T}(\Pi) \text{ labeled with } \alpha \text{ and } \tau' = Inc(\tau, v) \\
1 & \text{for } \tau \in \mathcal{T}'(\Pi) \text{ and } \tau' = \tau \\
0 & \text{otherwise}
\end{cases}
$$

[9]For any $Finish(v)$, resp. $Inc(\tau, v)$ Markov state, we assume that there exists a corresponding action called $finish(v)$, resp. $inc(\tau, v)$, which is defined in the analogous way used for $start$ in definition 7.



$$
\begin{array}{c}
 \\ T_0 \\ T_1 \\ T_2 \\ T_3 \\ \hline T_4 \\ T_5 \\ T_6 \\ T_7
\end{array}
\begin{array}{c}
\begin{array}{cccc|cccc}
T_0 & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7
\end{array}\\
\left(\begin{array}{cccc|cccc}
0 & 0.36 & 0 & 0 & 0 & 0 & 0 & 0.64 \\
0 & 0 & 0.4 & 0 & 0.6 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\
0 & 0 & 0.1 & 0.9 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)
\end{array}
$$

Figure 8: Incorrect and incomplete plan Π'_c and its associated Markov chain with its transition matrix

Figure 8 shows the incorrect and incomplete plan Π'_c for our cup example, and its associated Markov chain. The root T-node in Π'_c is incorrect, because it has a W-node labeled with V_6 as successor, and V_6 is not a possible L-world in s_c. Hence, the transition probability from *Start* to the state labeled with *wait* (the successor of the T-node labeled with V_6 in Π'_c) is $p_{s_c}(V_6) = 0$. *Start* is also incomplete: it should have a W-node labeled with V_2 as successor. Hence, there is a transition from *Start* to the state labeled with $inc(Start, V_2)$ in the Markov chain, with probability $p_{s_c}(V_2) = 0.64$. The same case arises with *spin*, which should also result in V_2. The two leaves of Π'_c being respectively labeled with V_3 and V_4—the former leaf being incorrect—the two respective *Finish* absorbing states are inserted in the Markov chain.

4.2.4 Defining the Rating Function for Possible Worlds Plans

The sections 4.2.2 and 4.2.3 deliver the tools for defining for possible worlds plans what this section 4.2 is all about in general: a rating function on plans. In the light of the intended use of the rating for defining correctness in the limit, we define it as (and call it) the *correctness and completeness degree (*ccd*)* of a plan. Like the proper notions of plan correctness and completeness, ccd is defined inductively, starting at the nodes, and generalizing to plans then.

According to Definitions 8 and 9, there are three types of nodes of interest when defining correctness and completeness of plans. These nodes are *Start*, the other T-nodes of the plan, and the W-node leaves of the plan, to which we want to assign a degree of correctness and completeness. The respective ccd function is specified in Definition 19. Before presenting it, we comment on the respective definition steps and motivate our choice. We give the comments for L-worlds, but they apply analogously to full worlds.

T-node correctness degree. Definition 8 says a T-node is correct iff it has no "wrong" W-node successor. This means that if $Succ(\tau, \Pi)$ is the set of possible L-worlds labeling the W-nodes successors of T-node τ in plan Π, then $Succ(\tau, \Pi)$ must be included in the set of possible L-worlds that are expected as labels of successors of τ. These latter L-worlds are in $\mathrm{Poss}_K|_L(s)$ for the $Start$ node, and in $r_K|_L(\alpha, v)$ for a non-root T-node labeled with α and whose W-node predecessor is labeled with v. The correctness degree for T-nodes is then chosen to be inversely proportional to the number of "wrong" L-worlds in $Succ(\tau, \Pi)$. Having $\setminus$ denote set difference, the degree is more exactly:

$$\begin{array}{ll} \frac{1}{card(Succ(Start,\Pi)\setminus \mathrm{Poss}_K|_L(s))+1} & \text{for } Start \\[2ex] \frac{1}{card(Succ(\tau,\Pi)\setminus r_K|_L(\alpha,v))+1} & \text{for non-root } T\text{-nodes} \end{array}$$

Leaf W-node correctness. Definition 8 says a leaf is correct iff it matches the goal g of the problem, where "matching" means that the goal is derivable from the respective possible L-world. This could be translated directly into the measure of leaf W-node correctness by saying that leaves matching the goal get a 1, and all others 0.

We want to allow for a bit more flexibility. We assume the user has to define a utility function on L-worlds that is goal-oriented in the sense that it yields 1 iff the goal is true in some L-world. More precisely, we assume there is a function $u : \mathrm{Poss}_K|_L \times \mathcal{L} \to [0, 1]$ such that

$$u(v, g) \begin{cases} = 1 & \text{for } v \text{ and } g \text{ s.t. } K \cup v \vdash g \\ < 1 & \text{otherwise} \end{cases}$$

Given such a goal-oriented utility function u, the correctness degree for a leaf labeled with v is $u(v, g)$.

T-node completeness degree. Definition 9 says a T-node is complete iff it has at least all the "right" W-nodes successors. This means that if $Succ(\tau, \Pi)$ is the set of possible L-worlds labeling the W-nodes successors of T-node τ in plan Π, then $Succ(\tau, \Pi)$ must be a superset of the set of possible L-worlds that are expected as labels of successors of τ. We could then proceed in analogy to T-node correctness to measure the extent to which this property holds. However, there exists a more accurate measure here, because the L-worlds in $Succ(\tau, \Pi)$ can be considered as random variables. Thus, we choose to measure the T-node completeness degree as the probability of $Succ(\tau, \Pi)$, given that the initial situation s holds for $\tau = Start$, or, respectively, given that α is applied in v, for a non-root T-node τ labeled with α whose predecessor is labeled with v. Hence, the degree is

$$\begin{cases} p_s(Succ(Start, \Pi)) & \text{for } Start \\ p_{(\alpha,v)}(Succ(\tau, \Pi)) & \text{for non-root } T\text{-nodes} \end{cases}$$

There is no completeness degree for plan leaves, since completeness does not depend on them, see Definition 9.

Node correctness and completeness degree. We choose to define the degree of correctness *and* completeness of each T-node as the product of the two respective degrees, and that of a leaf as its correctness degree.

Plan correctness and completeness degree. Definitions 8 and 9 say a plan is correct and complete iff all its nodes of interest are correct and complete. We then choose to measure the extent to which a plan is correct and complete, as the average degree of correctness and completeness per node encountered during its execution. As the formal definition of the correctness and completeness degree of a plan looks a bit lengthy, we don't repeat it here from the following Definition 19. It involves essentially two terms, averaging over the ccd value of, first, all T-nodes, and, second, all leaves, via the *Finish* states of the Markov chain.[10] These two terms are then weighed and added, where the weight factors must sum to 1 and reflect the relative importance given to final execution states and to non-final ones.

The following definition summarizes the above informal explanations.

Definition 19 (Node and Plan Correctness and Completeness Degrees)
Let Π be a plan for the planning problem description $\Psi = (s, g, K, A)$, P the probabilistic background knowledge, and u the goal-oriented utility function. For any task τ in Π, let $Succ(\tau, \Pi)$ be defined as follows:

$$Succ(\tau, \Pi) = \{v \mid \text{there exists a successor } \nu \text{ of } \tau \text{ in } \Pi \text{ labeled with } v\}$$

Degree of root correctness and completeness: The degree of correctness and completeness of *Start* in Π wrt. Ψ is

$$\mathrm{ccd}_\Psi(Start, \Pi) = \frac{p_s(Succ(Start, \Pi))}{card(Succ(Start, \Pi) \setminus \mathrm{Poss}_K|_L(s)) + 1}$$

Degree of non-root T-node correctness and completeness: Let τ be a non-root T-node in Π labeled with α and let ν labeled with v be its predecessor in Π. The degree of correctness and completeness of τ in Π wrt. Ψ is

$$\mathrm{ccd}_\Psi(\tau, \Pi) = \frac{p_{(\alpha,v)}(Succ(\tau, \Pi))}{card(Succ(\tau, \Pi) \setminus r_K|_L(\alpha, v)) + 1}$$

Degree of leaf correctness and completeness: Let ν labeled with v be a leaf of Π. The degree of correctness and completeness of ν in Π wrt. Ψ is

$$\mathrm{ccd}_\Psi(\nu, \Pi) = u(v, g)$$

Degree of plan correctness and completeness: Let the Markov chain associated with Π be $\langle \{X_t, t = 0, 1, \ldots\}, \mathcal{T} \cup \mathcal{T}', \pi \rangle$. The degree of correctness and completeness of Π wrt. Ψ is

$$\mathrm{ccd}_\Psi(\Pi) = c_1 \cdot ccd_{T-node} + c_2 \cdot ccd_{leaf},$$

where c_1, c_2 are positive real constants such that $c_1 + c_2 = 1$, and

$$ccd_{T-node} = \frac{\sum_{\tau \in \mathcal{T}} \left(\mathrm{ccd}_\Psi(\tau, \Pi) \cdot \sum_{t=0}^{\infty} \pi(X_t = \tau \mid X_0 = Start) \right)}{\sum_{\tau \in \mathcal{T}} \sum_{t=0}^{\infty} \pi(X_t = \tau \mid X_0 = Start)}$$

$$ccd_{leaf} = \sum_{Finish(\nu) \in \mathcal{T}'} \left(\mathrm{ccd}_\Psi(\nu, \Pi) \cdot \lim_{t \to \infty} \pi(X_t = Finish(\nu) \mid X_0 = Start) \right)$$

Fortunately, the correctness and completeness degree for a plan can be computed from the fundamental matrix of its associated Markov chain, and from the matrix representing the transitions from the transient states to the absorbing states of this chain, as stated in the following proposition. To see the claimed equality, one has simply to substitute the elements of the respective vectors for their values given in appendix A (for N_0 and $(N \times R)_0$), and in the proposition itself (for U, U' and $\vec{1}$); therefore, we give no formal proof here.

[10] Definition 19 does not treat *Inc* states in a special way. In effect, the ccd rating does not properly reward if the goal is accidentally achieved by an *Inc* node. As the planner itself or its rating is not the central topic of this paper, we allow for this slight sloppyness, which simplifies matters.

Proposition 20 (Computation of ccd**)** Let Π be a plan for the planning problem description $\Psi = (s, g, K, A)$, P the probabilistic background knowledge, and u the utility function. Let furthermore N, resp. R be the fundamental matrix, resp. the matrix representing the transitions from transient states to absorbing states, of the Markov chain associated with Π. The following equation is correct:

$$\mathrm{ccd}_\Psi(\Pi) = c_1 \frac{(N_0 \times U)}{(N_0 \times \vec{1})} + c_2 (N \times R)_0 \times U',$$

where

- N_0 is the first row (i.e., the row corresponding to *Start*) of the matrix N of the chain,
- $(N \times R)_0$ is the first row of the matrix $N \times R$,
- U is a column vector containing the degrees of completeness and correctness of the T-nodes of the plan,
- $\vec{1}$ is a column vector of the same dimension as U, consisting only of 1's, and
- U' is the column vector containing the degree of correctness and completeness of the W-nodes corresponding to the *Finish* states of the chain, and 0 for the *Inc* states.

As an immediate consequence of Proposition 20, we see that ccd_Ψ maps plans to $[0, 1]$ and that $\mathrm{ccd}_\Psi(\Pi) = 1$ iff Π is a correct and complete plan. Hence, we can state

Corollary 21 ccd_Ψ is a rating function.

Finally, let us give an example and compute $ccd_{\Psi_c}(\Pi'_c)$, where Ψ_c and P_c are defined as before. Define $u_c(v, g) = 1$ iff $K_c \cup v \vdash g$, and 0 otherwise; and let $c_1 = c_2 = 0.5$. We have:

$$N_0 = \begin{array}{c} \begin{array}{cccc} T_0 & T_1 & T_2 & T_3 \end{array} \\ \left(\begin{array}{cccc} 1 & 0.36 & 0.144 & 0 \end{array} \right) \end{array} \quad (N \times R)_0 = \begin{array}{c} \begin{array}{cccc} T_4 & T_5 & T_6 & T_7 \end{array} \\ \left(\begin{array}{cccc} 0.216 & 0.072 & 0.072 & 0.64 \end{array} \right) \end{array}$$

$$\begin{array}{rcl} \mathrm{ccd}_{\Psi_c}(T_0, \Pi'_c) & = & 0.36/2 = 0.18 \\ \mathrm{ccd}_{\Psi_c}(T_1, \Pi'_c) & = & 1/1 = 1 \\ \mathrm{ccd}_{\Psi_c}(T_2, \Pi'_c) & = & 0.5/1 = 0.5 \\ \mathrm{ccd}_{\Psi_c}(T_3, \Pi'_c) & = & 1/1 = 1 \\ \mathrm{ccd}_{\Psi_c}(V_3, \Pi'_c) & = & 0 \\ \mathrm{ccd}_{\Psi_c}(V_4, \Pi'_c) & = & 1 \end{array}$$

$$U = \begin{array}{c} T_0 \\ T_1 \\ T_2 \\ T_3 \end{array} \begin{pmatrix} 0.18 \\ 1 \\ 0.5 \\ 1 \end{pmatrix} \qquad U' = \begin{array}{c} T_4 \\ T_5 \\ T_6 \\ T_7 \end{array} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Therefore,

$$\mathrm{ccd}_{\Psi_c}(\Pi'_c) = 0.5 \cdot \frac{0.18 + 0.36 + 0.072}{1 + 0.36 + 0.144} \; + \; 0.5 \cdot 0.216 \; \simeq \; 0.31$$

As said previously, it is reasonable to demand that the rating of a plan be updatable during the execution of this plan, according to additional information that becomes available. First, this can be useful for an executor. If several plans are available for solving a problem, it can begin the execution with the plan with the best a-priori rating, but interrupt this execution and switch to another plan whenever this one becomes of higher rating. Second, this can be useful for on-line planners, which incrementally improve the correctness and completeness of some part of a plan during its execution. Based on the material just presented, it is straightforward to define a continually updating version of ccd. Assume the current state of the plan execution (i.e., the currently executed task) is τ'. We have to substitute in Definition 19 every occurrence of $X_0 = Start$ with $X_0 = \tau'$; moreover, the computation of the rating's update must use the N and $(N \times R)$ matrices' rows corresponding to τ'. Further detail concerning this issue lies outside of this paper's scope.

4.3 Defining Correctness, Ltd.

We finally use the rating function to define limited correctness, as announced several times above. We do not consider the special case of possible worlds planning here; our planner PASCALE2, which shows correctness in the limit with respect to the ccd function, is discussed in the subsequent Section 5.

The notion of correctness in the limit is based on the following idea. Standard notions of planner correctness and completeness (as given in Definitions 11 and 12) require that a planner generate all correct and complete plans for a problem, and only these. Practically, that is too restrictive. In particular, as long as nothing is assumed about preferences among correct and complete plans (like plan cost or plan generation time, as briefly addressed at the beginning of Section 4.2), it is even worthless to require generating more than one such plan: one is as good as all the others then. Furthermore, if the number of computation steps available to generate plans is small, then even incomplete and incorrect plans may be of interest. Using a rating function for measuring the "degree" of plan correctness and completeness, we consequently require that the plans a planner delivers get more correct and more complete as run time increases, and that a correct and complete plan is returned eventually at some step t. Of course, we do not require that a limitedly correct planner is really given as much as t steps, nor do we assume that it stops running then: it may continue, searching for plans that are not only correct and complete, but also maximize some other plan quality function.

Definition 22 (Planner correctness in the limit) Let ϱ be a family of rating functions. A planner $(\mathcal{P}, A_{\mathcal{P}})$ is correct in the limit wrt. ϱ, iff for all problems Ψ:

1. if there is a plan that is correct and complete wrt. Ψ, then the planner will eventually deliver a correct and complete plan, i.e., then there exists $t \in A_{\mathcal{P}}(\Psi)$ such that $\varrho_\Psi(\mathcal{P}(\Psi)(t)) = 1$, and
2. the ratings of the plans that the planner returns increase monotonically as planning steps increase, i.e., for all $t_1, t_2 \in A_{\mathcal{P}}(\Psi)$, if $t_2 \geq t_1$ then $\varrho_\Psi(\mathcal{P}(\Psi)(t_2)) \geq \varrho_\Psi(\mathcal{P}(\Psi)(t_1))$

We will use the terms *limited correctness, limitedly correct,* resp., as short forms of *correctness in the limit,* resp., *correct in the limit.*

Comparing this definition with the one of "pure" planner correctness (Definition 11), a remarkable theoretical difference is that limited correctness requires that at least one correct and complete plan be returned, whereas the normal correctness does not. This may be interpreted as a flavor of "completeness in the limit" in this concept of limited correctness. The definition guarantees, on the other hand, that a limitedly correct planner actually turns to a correct planner after the magic number of t steps, for which there is no analogy with respect to completeness. Even with this idea in mind, the monotonicity condition is more restrictive than might be considered necessary: another reasonable variant of limited correctness could require planner correctness after the first correct and complete plan has been delivered, but not insist on monotonicity.

However, the very idea of planner correctness in the limit makes sense only if plan correctness and completeness is the dominant factor in an overall measure of plan quality; and if this is the case, then it seems reasonable to require a non-arbitrary, somewhat predictable behavior of the planner output with respect to this factor. There may be additional criteria to discriminate different correct and complete plans, measured by particular other quality sub-functions. We neither require nor exclude these.

Let us state some observations concerning the definition; all these observations follow directly from the respective other definitions, which is why we do not formally prove them.

- While the definition of a rating function ensures that they all rate $\mathcal{P}(\Psi)(t)$ with 1, i.e., item (1) is independent of rating functions, it is not required that a plan Π_1 rated higher than Π_2 by some rating function ϱ, is also rated higher than Π_2 by some other rating function ϱ'. Consequently, a planner that is limitedly correct wrt. ϱ need not be so wrt. ϱ', because it may violate the monotonicity condition (2) for ϱ'.

- All correct planners that eventually return plans for all solvable problems, i.e., $A_{\mathcal{P}}(\Psi) \neq \emptyset$ for all solvable Ψ, are limitedly correct wrt. to all rating functions—as should intuitively be expected. Consequently, all correct and complete planners are limitedly correct wrt. to all rating functions.

- Incorrect, incomplete, and both incorrect and incomplete planners may be limitedly correct wrt. some rating function.

The latter item is of most importance here. For good reasons, practical planners tend to be highly incorrect and incomplete. The notion of correctness in the limit allows them to keep this salvaging property, yet being put into a clear relationship to an arbitrarily neatly or idealistically defined action calculus.

Of course, being able to establish this relationship does not come for free. Even assuming that a limitedly correct planner is built in the *ex ante* fashion (i.e., have the action calculus first, and build the planner then), it still requires designing an appropriate rating function and building the planner accordingly. However, we are convinced that this effort is worth being taken, if the goal is to build planners delivering plans with a clear semantics, yet operating under realistic run time constraints.

5 PASCALE2—An Anytime Uncertainty Planner

To complete the description of the case of our case study, it remains to sketch the planner PASCALE2 we have constructed; in particular, we have to show that it is limitedly correct wrt. the ccd rating function. We summarize what we had to implement, sketch how we did it, draw the conclusions from this sketch, and present an example. Note that PASCALE2 is not described here as the objective of our work—in which case its description would have been more detailed—but just as the byproduct or example of our view of planner construction. Aspects of it or its predecessor are presented in more depth elsewhere [23, 24].

The work described in the previous sections yields precise functional specifications for the following planner components:

1. the possible worlds formalism for L-worlds, including the $r|_L$ function,

2. the computation of the probability distributions over the space of possible L-worlds,

3. the manipulation of possible worlds plans, and

4. the plan rating function ccd, including generation and handling of the Markov chains associated with plans.

Implementing these components is far from trivial, at least if their run time is something to worry about. Note in particular that computing the a-priori probability distribution p over the set of possible L-worlds requires computing all possible L-worlds; this can be done prior to planning and once for each new domain, but it is inherently costly. Given the possible L-worlds, computing an approximation of p is easy and relatively efficient following the lines of [4]. Concerning the computation of the possible L-worlds resulting from the performance of an action, [5] describes an algorithm for doing that relatively efficiently, which is, however, not yet used in PASCALE2.

An important point for the following description is that PASCALE2 calculates and inserts into the plan the exact result of applying an action in some possible L-world. As a consequence, all T-nodes in a PASCALE2-plan are correct and complete, all PASCALE2-plans are complete, and only their leaves may be incorrect. This, in turn, simplifies the calculation of the ccd value of the plans, since the first term in the calculation (cf. Proposition 20) always equals the c_1 constant.

Even if the issue of how to implement the above components is crucial for obtaining a usable planner, we sacrifice a more thorough description for saving space and refer to the given references. The PASCALE2 component that must be discussed, however, is:

5. the search strategy for traversing the space of possible L-world plans.

This strategy is not constructively specified by the formalism; it is just constrained if the planner is supposed to be correct in the limit. PASCALE2 is not correct and not complete, and we have mentioned above (4.3) that the order in which plans are generated is crucial for such planners' property of being limitedly correct, because of the monotonicity requirement for the plan rating.

Figure 9 specifies a somewhat simplified version of PASCALE2's core in pseudo-code and informally describes its top-level sub-procedures. Obviously, it performs a best-first backtracking search through a search space consisting of plans. Note that this procedure practically excludes completeness of the planner using it, as it will discard, among others, all correct and complete plans whose quality is not higher than the one of an already found correct and complete plan. While this is the case for PASCALE2, it is in no way necessary. Variants of the other well-known basic search strategies could also have been used for this basic procedure.

The step size (in the sense of the planner definition 10) is an iteration through the main **while**-loop. The procedure is trivially any-step for this step definition, i.e., it can return a plan after any step, namely, best-plan. (We will show a protocol of the respective best-plans for the example problem Ψ_c below.) Owing to the use of the global variable best-plan, the procedure can trivially be made interruptible at any *time* after the initialization phase, provided that the assignment to this variable within the **while** loop is atomic.

The measure used to order plans is given by the sub-procedure quality(Ψ, Π). The value it returns can be identical to $\mathrm{ccd}_\Psi(\Pi)$, but we do not require that this be the case. In general, one might use more sophisticated quality functions that take, e.g., operator cost, execution time, or plan generation time into account, maybe even weighing it with the probability data provided by the Markov chain. PASCALE2 allows for the flexibility of using such a quality function. However, to be correct in the limit, it suffices that the planning procedure use a quality function that is order-preserving wrt. ccd in the sense of the following lemma. Formally proving this lemma would be technically effortful, which is why we don't do it here; the intuitive argument, however, should be obvious, given the search strategy defined by the off-line-plan procedure and the role played by the quality procedure.

Lemma 23 The procedure off-line-plan in Figure 9 yields a limitedly correct planner, if for all problems Ψ its sub-procedure quality(Ψ, Π) is order-preserving wrt. ccd_Ψ in the sense that

1. if quality$(\Psi, \Pi_1) >$ quality(Ψ, Π_2), then $\mathrm{ccd}_\Psi(\Pi_1) \geq \mathrm{ccd}_\Psi(\Pi_2)$, and
2. if there is a correct and complete plan for Ψ, then there is a plan Π such that $\mathrm{ccd}_\Psi(\Pi) = 1$, and for all Π': if $\mathrm{ccd}_\Psi(\Pi') < 1$, then quality$(\Psi, \Pi) >$ quality(Ψ, Π').

To give an impression of PASCALE2's behavior, we show its results for the cup domain problem Ψ_c. We choose quality$(\Psi, \Pi) = \mathrm{ccd}_\Psi(\Pi)$, i.e., correctness in the limit is guaranteed. Figure 10 shows how the quality of the available best plan monotonically increases over run time, ending up at 1 after 3.6 seconds; the respective plan Π_c'' is shown in Figure 11.

Looking at figure 11, one can see how PASCALE2 expands the current plan in depth first manner, expanding the most probable outcome of the last inserted task first. Consequently, a plan that is

```
function off-line-plan (problem,max-step) =
best-plan := empty-plan(problem);
search-space := [ best-plan ];
current-step := 0;
while search-space ≠ [] and current-step < max-step
        current-step := current-step + 1;
        current-plan := head(search-space);
        if      quality(problem,current-plan) > quality(problem,best-plan)
        then best-plan := current-plan fi;
        T := last-inserted-task(current-plan);
        L := select-maximal-probable(current-plan,T,problem);
        if      L ≠ nil
        then successors := order(expand(current-plan,L,problem));
                search-space := append(successors,tail(search-space)) fi
end;
return best-plan
```

empty-plan(Ψ) builds a plan embryo containing the *Start* T-node and the possible models of the initial situation of problem Ψ.

quality(Ψ, Π) returns a quality value of plan Π for the problem Ψ.

last-inserted-task(Π) returns the lastly inserted T-node in plan Π.

select-maximal-probable(Π, τ, Ψ) returns the leaf of plan Π that is maximally probable to be reached from T-node τ, unless one of the leaves that can be reached from τ matches the goal of problem Ψ. In that case, returns the leaf that is maximally probable to be reached from *Start* and does not match this goal. If there is none, then returns **nil**.

expand(Π, l, Ψ) returns the list of valid plans resulting from the expansion of plan Π at leaf l, using the available actions of problem Ψ. Returns **nil** for correct and complete plans.

order(Λ) orders the list of plans Λ in decreasing order of the quality(Ψ, Π) procedure.

Figure 9: PASCALE2's core procedure (simplified), and its top-level sub-procedures. It is assumed that the utility of all T-nodes equals 1.

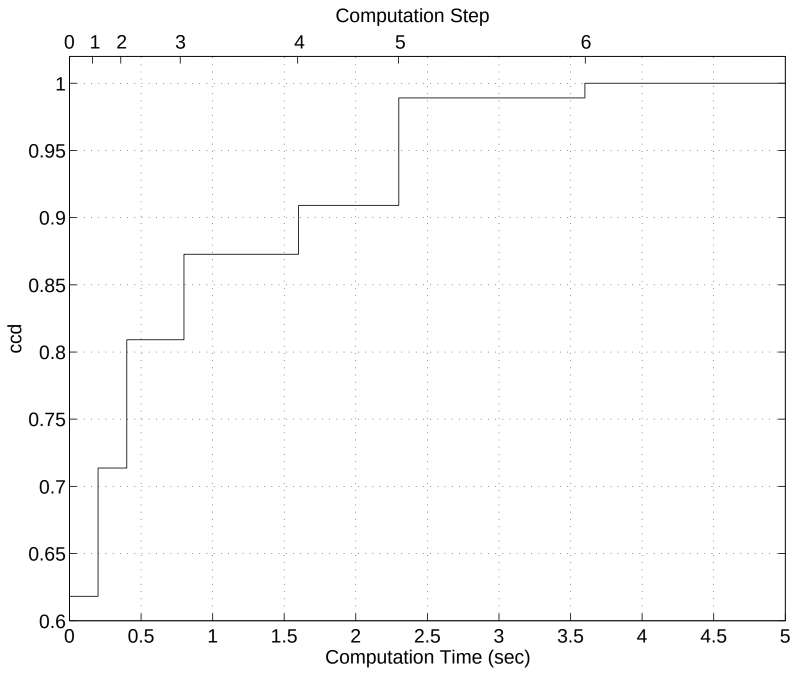


Figure 10: Plan quality of best-plan as a function of computation time on Ψ_c. The computation steps $0, \ldots, 6$ refer to the development of the plan Π_c'' in Figure 11.

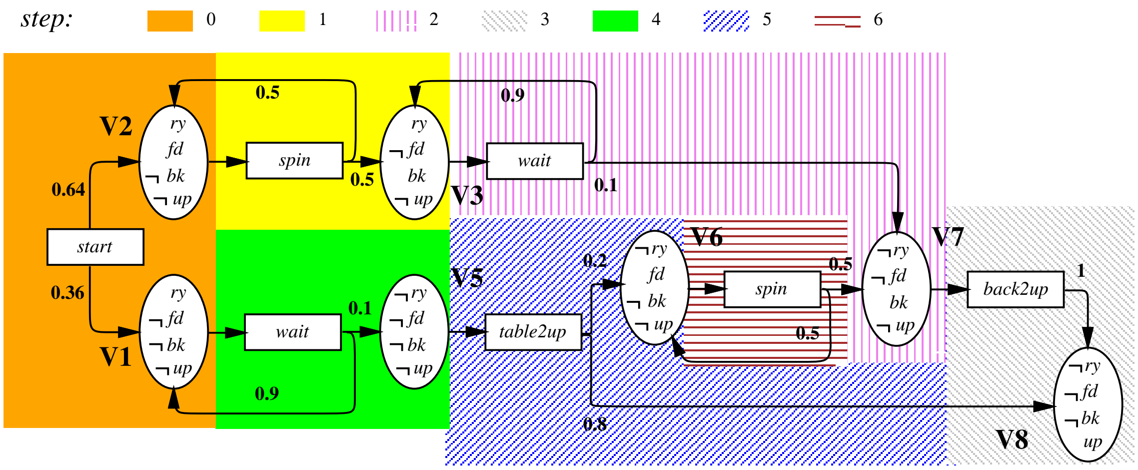


Figure 11: The plan Π_c'' for solving Ψ_c. Different background textures are signalling the number of steps after which the respective plan part was found.

not yet correct and complete in general may well correctly and completely describe what happens most probably if its execution is started, although it is not yet finished. This behavior of PASCALE2 is the clue to the rapid increase of the ccd value at the beginning of the run time. (after 0.8 seconds, ccd already has a value of 0.87.)

Since quality equals ccd for this experiment, there is no strictly better plan than Π''_c. As the criterion for returning other plans is a proper increase in quality, PASCALE2 does not return any other correct and complete plans, e.g., the Π_c plan discussed earlier. Changing the comparison criterion between the quality of the current plan and the best-plan to $\geq$ would lead to returning additional correct and complete plans. Note that the goal-oriented utility function for this example was specially designed for the didactic purpose of yielding a relatively slow convergence towards a correct and complete plan; using different such functions, PASCALE2 would have generated other plans—in most cases considerably faster than here.

PASCALE2 is implemented in Standard ML on a SUN Sparc IPX workstation.

6 Conclusion and Open Issues

The purpose of an *action formalism* is to specify the reasoning about prerequisites and consequences of actions that an ideally rational agent should perform. A *planner* has to do reasoning about prerequisites and consequences of actions, but if it insists on doing it in an idealistic way, then it is likely to fail on most realistic usage constraints for practical applications. Consequently, it seems the camp of interesting planners and the camp of interesting action formalisms are way apart.

Our idea to build a bridge between the two camps is to interpret the reasoning behavior of planners as an approximation of the ideal reasoning as specified by the formalism. If a planner is correct in the limit and is given enough time, then it is guaranteed to deliver plans up to the ideal as set by the action formalism; given less time, the degrees of incorrectness and incompleteness of plans found until then can be quantified by the plan rating, providing an exact measure of the deviation relative to the ideal. Designing planners in this spirit of *ex ante* helps solve the often-deplored problem of lack of formal background for plans and planners, and it provides a handle for planner designers to making use of work from the camp of action formalisms.

The case study we have provided along with the generic piers for building such bridges demonstrates that they cannot only be built between highly restrictive formalisms and very classical planners. Moreover, the study is a written testimony for the fact that we do not claim turning an action formalism into a planner is always straightforward. Maybe there is a straightforward planner for every action formalism (one for the possible (L-)worlds formalism was the forerunner of PASCALE2, called PASCALE [23]), but that is not our point. Starting from requirements for a particular planner (like expressivity of the action language, anytime behavior, or quality features for plans), we assume that the action formalism to lay under this particular planner will nearly never be available off-the-shelf. The effort for restricting the formalism, designing plans, defining the rating function, and finally building the planner will still be considerable. But we think the potential result is worth a try.

The study also exemplifies or reveals a number of open issues in the general idea of building planners *ex ante*. Let us address some of them, starting with special issues and turning more and more general. We will not discuss PASCALE2 in particular: it was just used as the experiment in this study and is more thoroughly discussed elsewhere [24].

Alternative definitions of correctness in the limit. While notions like limited correctness of planners and correctness and completeness of plans will play key rolls in every method for turning action formalisms into planners, they may well be defined differently, reflecting different assumptions or domain characteristics. Let us first consider limited correctness.

Definition 22 requires monotonicity of the rating function ϱ over time. However, other versions of limited correctness can be thought of. First, monotonicity of ϱ might be considered unnecessary during the phase where no correct and complete plan has been found yet. This would enable the planner to choose more freely which plan to consider currently best. However, revisiting Figure 10, PASCALE2's rating converges smoothly (modulo the inevitable discontinuity) towards 1 for Ψ_c and the parameter setting chosen; such a smooth convergence would at least be discouraged by skipping the monotonicity requirement. Second—just pushing further the argument for freedom of choice—even after having found a correct and complete plan, the planner might be allowed to return plans that are incorrect and incomplete, leaving as the only constraint on limited completeness that the planner return a correct and complete plan sometime.

The point is: the monotonicity requirement does no harm; in particular it does in no way (as one might suspect) force the planner into local plan optima from which there is no escape. However, it restricts the choice of possible rating functions. It is not yet determined whether this particular restriction is in general wise or obstructive.

Plan quality vs. rating functions. Closely related with the previous point, it is a crucial question, what the determining factor is for considering plans "good" or "bad". If plan correctness and completeness is not this factor, then the monotonicity requirement for rating functions will most probably be troublesome. We assume there may exist domains in which this is the case. Consequently, it remains an open question to define a more general version of limited correctness that constrains the planner behavior, yet allows an arbitrary quality function to guide the planner behavior—or at least a quality function that is less restricted than by the order-preservingness conditions in Lemma 23. In PASCALE2, we allow for using much more general quality definitions than those yielding limited correctness, involving, e.g., operator cost. This very fact shows that we see the practical necessity to do something about a more generous combination between a strict rating function and a general quality measure.

The value of computation. Russell and Wefald [19] advocate the idea to meta-reason during planning, to determine whether or not acting based on the plan developed so far pays more than trying to enhance the plan quality by further planning. The idea is to define the expected utility of developing the recent plan and weigh it against the expected utility of acting right away.

This idea does not depend on whether or not the planner is based on some action formalism. However, it nicely suits our particular blueprint for basing planners on action formalisms using planner correctness in the limit. In particular, expected plan rating increase can be a major factor in determining the expected utility of further developing the plan. It is a point for future work to explicitly include the possibility for the meta-reasoning view into this blueprint.

Action formalisms to base planners on. In the introduction of this paper, we have stated a basic hypothesis for our work (see the paragraph labeled with (1)), involving the requirement that the formalism to lay under the *ex ante* planner construction be "reasonable". We still have to explain what this means.

In fact, we do not see any theoretical constraints on the formalism. In particular, it need not be decidable or even *itself* be in any way efficiently implementable, because we have assumed that a planner implementation use a restricted version of the formalism or an implementation that approximates the real results of the formalism itself. In the latter case, the formalism is of course only the approximative formal basis of the plans that the planner generates, which might be a drawback for the overall goal of the *ex ante* planner construction of obtaining plans and planner behavior with a clear semantics.

Consequentially, the only—inherently fuzzy—requirement on the formalism itself is that it allow to define a restricted version or approximation that both makes sense and is efficiently implementable. The latter point is crucial if the goal is building practical planners, as reasoning in its action formalism lies at the heart of every planner. We have no recipe or theory for telling which formalism allows for defining a reasonable restriction, or how to obtain it.

General applicability of the approach. We have presented our work as a case study for our supposedly more general method of *ex ante* planner construction. The crucial question is whether this method is really generally applicable—be it in precisely the way we have described here, or in some variation regarding the steps, notions, and definitions we have used. This question cannot yet be answered conclusively.

In particular, while we see a fair amount of work that is closely related to single aspects of ours, we are not aware of other work under the same or a comparable perspective. To repeat pointers to related work from above, there is related work

- about anytime generation of plans [7]—but it does not ground the plan quality in the firm semantics provided by an action formalism;
- about giving plans a formal semantics [15]—but it is *ex post,* hence does not help the planner designers, and does not deal with limited correctness;
- about implementing given formal action or planner models [2, 16]—but it strives for implementing a planner that "directly" uses the formalism, which is instructive for understanding the formalism and the particular planning aspect it focuses on, but is a dead end with respect to building practical planners based on more expressive formalisms.

Consequently, answering the question whether our proposed method of planner construction is generally applicable, would amount to generalizing from one complex experience, which, as common sense tells, is better avoided. However, both this experience and theoretical analysis did not reveal any very basic obstacles, and our first results reported here appear sufficiently promising that our plans for further work include studying further cases, i.e., attempting to turn other action formalisms into other planners, and study the general pattern behind.

Acknowledgements

Joachim Hertzberg is on leave from the German National Research Center of Computer Science (GMD), AI Research Division, Schloss Birlinghoven, 53757 Sankt Augustin, Germany. The first version of this paper was written while he was a guest researcher at the International Computer Science Institute (ICSI), Berkeley, Ca, USA.

Thanks to Thierry Montaut for useful discussions. Thanks to Jerry Feldman and Angi Voß for comments on a previous version of this paper.

References

[1] C. Bäckström and E. Sandewall (eds.). *Current Trends in AI Planning. EWSP'93 – 2nd European Workshop on Planning.* IOS Press, Amsterdam, 1994.

[2] S. Biundo. Present-day deductive planning. In *[1]*, pages 1–5, 1993.

[3] G. Brewka and J. Hertzberg. How to do things with worlds: On formalizing actions and plans. *J. Logic and Computation*, 3(5):517–532, 1993.

[4] P. Cheeseman. A method of computing generalized bayesian probability values for expert systems. In *Proc. IJCAI-83*, pages 198–202, 1983.

[5] T. Chou and M. Winslett. Immortal: a model-based belief revision system. In *Proc. KR-91*, pages 99–109, 1991.

[6] L. Chrisman and R. Simmons. Sensible planning: Focusing perceptual attention. In *Proc. AAAI-91*, pages 756–761, 1991.

[7] T.L. Dean and M. Boddy. An analysis of time-dependent planning. In *Proc. AAAI-88*, pages 49–54, 1988.

[8] D. Dubois and H. Prade. Belief revisions and updates in numerical formalisms. an overview, with new results for the possibilistic framework. In *Proc. IJCAI-93*, pages 620–625, 1993.

[9] R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to theorem proving in problem solving. *J. Art. Intell.*, 2:189–208, 1971.

[10] P. Gärdenfors. *Knowledge in Flux.* MIT Press, Cambridge, MA, 1988.

[11] M.L. Ginsberg and D.E. Smith. Reasoning about action I: A possible worlds approach. *J. Art. Intell.*, 35:165–195, 1988.

[12] J. Hendler, A. Tate, and M. Drummond. AI planning: Systems and techniques. *AI Magazine*, 11(2):61–77, 1990.

[13] H. Katsuno and A.O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. KR-91*, pages 387–394, 1991.

[14] D.K. Lewis. Probabilities of conditionals and conditionals probabilities. *The Philosophical Rev.*, 85:297–315, 1976.

[15] V. Lifschitz. On the semantics of STRIPS. In M.P. Georgeff and A.L. Lansky, editors, *Proc. 1986 Workshop Reasoning about Actions and Plans, Timberline, OR*, pages 1–9, Los Altos, 1987. Morgan Kaufmann.

[16] D. McDermott. Regression planning. *Int. J. Intell. Syst.*, 6(4):357–416, 1991.

[17] N.J. Nilsson. Probabilistic logic. *J. Art. Intell.*, 28(1):71–87, 1986.

[18] E.P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *J. Comp. Intell.*, 4:356–372, 1988.

[19] S. Russell and E. Wefald. *Do the Right Thing. Studies in Limited Rationality.* MIT Press, Cambridge, Massachusetts, 1991.

[20] E. Sandewall. Features and fluents. Technical Report LiTH-IDA-R-91-29, Dept. of Computer and Information Sciences, Linköping University, 1991.

[21] W. Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In W.L. Harper and B.L. Skyrms, editors, *Causation in Decision, Belief Change, and Statistics*, pages 105–134. Reidel, 1987.

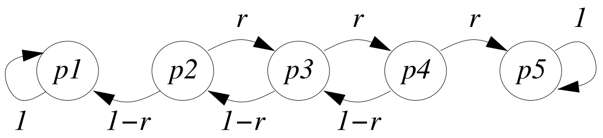
[22] J.D. Tenenberg. Abstraction in planning. In J. Allen, H. Kautz, R. Pelavin, and J. Tenenberg, editors, *Reasoning about Plans*, chapter 4, pages 213–283. Morgan Kaufmann, 1991.

[23] S. Thiébaux and J. Hertzberg. A semi-reactive planner based on a possible models action formalization. In J. Hendler, editor, *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS92)*, pages 228–235, San Mateo, CA, 1992. Morgan Kaufmann.

[24] S. Thiébaux, J. Hertzberg, W. Shoaff, and M. Schneider. A stochastic model of actions and plans for anytime planning under uncertainty. *Int. J. of Intelligent Systems*, 1994. In press; short version in [1].

[25] D. Wilkins. *Practical Planning. Extending the Classical AI Planning Paradigm.* Morgan Kaufmann, San Mateo, CA, 1988.

[26] M. Winslett. Reasoning about action using a possible models approach. In *Proc. AAAI-88*, pages 89–93, 1988.



$$\begin{array}{c|c}
 & \begin{array}{ccc|cc} p_2 & p_3 & p_4 & p_1 & p_5 \end{array} \\
\begin{array}{c} p_2 \\ p_3 \\ p_4 \\ \hline p_1 \\ p_5 \end{array} & \left(\begin{array}{ccc|cc} 0 & r & 0 & 1-r & 0 \\ 1-r & 0 & r & 0 & 0 \\ 0 & 1-r & 0 & 0 & r \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right)
\end{array}$$

Figure 12: A Markov chain and its transition matrix.

A Markov Chain Basics

A Markov chain is a stochastic process whose probabilistic transition through a set of states at some instant $t+1$ depends only on the state at the preceding instant t, and not on the states the chain passed through prior to time t. If, furthermore, the transition probabilities do not depend on t but remain stationary over time, the chain is said to be *stationary*. Stationary chains can by definition be represented by a single transition matrix relating the probability of the succeeding state to the current state.

Definition 24 (Markov chain, transition matrix) A Markov chain is a family of random variables $\{X_t, t = 0, 1 \ldots\}$ taking values in a set of states S, such that the conditional probability distribution π of the state at time $t+1$ has the following property for all $t \in \{0, 1, \ldots\}$ and all $s_0, \ldots, s_{t+1} \in S$:

$$\pi(X_{t+1} = s_{t+1} \mid X_t = s_t, \ldots, X_0 = s_0) = \pi(X_{t+1} = s_{t+1} \mid X_t = s_t)$$

A Markov chain is stationary if and only if for all $t \in \{0, 1, \ldots\}$ and all $s_i, s_j \in S$:

$$\pi(X_{t+1} = s_j \mid X_t = s_i) = m_{ij}$$

for some constant $m_{ij} \in [0, 1]$. The transition matrix for a stationary Markov chain is the matrix with entries m_{ij}.

A classical example of a process that can be modeled as a stationary Markov chain is a random walk. A particle moves between five points $p_1, \ldots, p_5$ on a line. At each step, it can go from one point to the right with probability r, and from one point to the left with probability $1-r$. It moves until it reaches one of the two boundaries of the line, and remains at this boundary. The corresponding stationary chain and its transition matrix are shown in Figure 12.

We are interested here in a special type of stationary chain called *absorbing chain*. An absorbing chain is a stationary chain with two types of states: transient states, which can be left on at least one path that never returns, and absorbing states, which cannot be left once entered. It is easy to see that the random walk is an absorbing chain. p_1 and p_5 are absorbing, while the other states are transient. The transition matrix of an absorbing chain can be divided into 4 submatrices as shown below, where the submatrix Q denotes transitions from transient to transient states, R denotes transitions from transient to absorbing states, I is the identity matrix, and O consists only of 0's.

$$\begin{array}{c|c}
 & \begin{array}{c|c} trans. & abs. \end{array} \\
\begin{array}{r} trans. \\ \hline abs. \end{array} & \left(\begin{array}{c|c} Q & R \\ \hline O & I \end{array}\right)
\end{array}$$

These submatrices can be used to compute quantitative information about the process modeled as an absorbing chain. The matrix $I - Q$ always has an inverse N, called the *fundamental matrix,* where $N = (I-Q)^{-1} = \sum_{t=0}^{\infty} Q^t$. The definition of N implies that its ij^{th} element is the average number of times transient state s_j will be entered before an absorbing state is reached, given that we are in transient state s_i. That is, for two transient states s_i and s_j:

$$N_{i,j} = \sum_{t=0}^{\infty} \pi(X_t = s_j | X_0 = s_i)$$

Furthermore, the ij^{th} element of matrix $N \times R$ is the probability of reaching absorbing state s_j, given that we are in transient state s_i. That is, for transient state s_i and absorbing state s_j:

$$(N \times R)_{i,j} = \lim_{t \to \infty} \pi(X_t = s_j | X_0 = s_i)$$

Note that this probability of reaching an absorbing state can also be viewed as the average number of times the absorbing state will be entered before the process becomes stable.

B The Equivalence Between Using Worlds and L-worlds

In Section 4.1.2, we postponed the proof of the claim that using full worlds and using L_c-worlds is equivalent in the cup domain. This is the place to deliver it.

What we are heading for is a statement of the form

$$K_c \cup r_{K_c}(\alpha, s) \text{ is logically equivalent to } K_c \cup r_{K_c}|_{L_c}(\alpha, s)$$

for every formula $s \in \mathcal{L}_c$ and every cup domain action α. There are different ways to get there. The tedious one is to check it for all actions and all sets of possible worlds. Instead, we state two more generally applicable conditions, the conjunction of which is sufficient for the equivalence to hold. These two conditions are true for the cup domain.[11]

The first condition needed is spanningness, as stated in Definition 13. For the cup domain, it can be easily verified that L_c is spanning wrt. K_c.

The second condition requires each postcondition $Post_{i,j}$ of α to be sufficiently precise, so as to ensure, for any possible world w verifying Pre_i, the uniqueness of the possible world having the minimal difference with w, among the possible worlds verifiying $Post_{i,j}$. This property, which we call alternative-wise result uniqueness, is formally defined as follows.

Definition 25 (Alternative-wise result uniqueness) Let α be an action as in Definition 4. α yields alternative-wise unique results iff for every possible world w and every precondition Pre_i of α:

> if $w \cup K \models Pre_i$ and $j \in \{1, \ldots, l(i)\}$
> then there is exactly one possible world w' such that $\mathrm{Diff}(w, w')$ is (set inclusion) minimal among the possible worlds satisfying $Post_{i,j}$.

Again, it can be easily verified that all actions in the cup domain are alternative-wise result unique. In fact, this property already got used in Section 4.2.2 (see Footnote 7).

The required proposition, then, is the following. As stated above, its antecedent is true for the cup domain; hence it implies the required equivalence.

Proposition 26 (Equivalence Theorem) Let α be an action, L a subset of the ground atoms of $\mathcal{L}$, and K the background knowledge.

If L is spanning w.r.t. K and if α yields alternative-wise unique results, then $K \cup r_K(\alpha, s)$ is logically equivalent to $K \cup r_K|_L(\alpha, s)$ for every formula $s \in \mathcal{L}$.

[11] Note that [3] contains a similar proposition stating the reqiured equivalence (Proposition 15). However, it is not applicable here because the independence property is not true for L_c.

Proof Let w be a possible world, $w|_L$ its corresponding possible L-world, Pre_i a precondition of α, such that $K \cup w \models Pre_i$, and let $Post_{i,j}$ ($j \in \{1, \dots l(i)\}$) be a postcondition of α. The equivalence is demonstrated in three steps.

- By alternative-wise result uniqueness, there is a unique possible world w' among those verifying $Post_{i,j}$ such that for every possible world $w'' \neq w'$ verifying $Post_{i,j}$ $\text{Diff}(w, w') \subset \text{Diff}(w, w'')$. This implies that w' is the unique possible world verifying $Post_{i,j}$ such that for every possible world $w'' \neq w'$ verifying $Post_{i,j}$:

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

  Hence, according to the closeness definition on possible worlds (Definition 3)

  $$w\text{-Closest}_K(Post_{i,j}) = \{w'\} \tag{5}$$

- By definition of L-worlds, w implies $w|_L$, and, as L is spanning, $w|_L \cup K$ implies $w \cup K$. Hence $w|_L \cup K$ is logically equivalent to $w \cup K$, and

  $$w|_L \cup K \models Pre_i \;\text{ iff }\; w \cup K \models Pre_i \tag{6}$$

  This means that the same branch of α gets applied in a possible world and in its corresponding possible L-world.

- Let us prove that:

  $$w|_L\text{-Closest}_K|_L(Post_{i,j}) = \{w'|_L\} \tag{7}$$

  where $w'|_L$ is the possible L-world corresponding to w'.

  First, recall that $w|_L\text{-Closest}_K|_L(Post_{i,j}) = \{w'|_L\}$ iff $w'|_L$ verifies $Post_{i,j}$ (i.e., $w'|_L \cup K \models Post_{i,j}$), and for every possible L-world $w''|_L \neq w'|_L$ verifying $Post_{i,j}$, we have $\text{Diff}(w|_L, w'|_L) \subset \text{Diff}(w|_L, w''|_L)$.

  By alternative-wise result uniqueness, we know that for every possible world $w'' \neq w'$ verifying $Post_{i,j}$, we have $\text{Diff}(w, w') \subset \text{Diff}(w, w'')$. By spanningness, there must be X, Y that are sets of ground literals of $\mathcal{L}$ not in L, such that

  $$\text{Diff}(w|_L, w'|_L) \cup X \subset \text{Diff}(w|_L, w''|_L) \cup Y$$

  where $w'|_L$ and $w''|_L$, by an argument similar to that of (6), both statisfy $Post_{i,j}$.

  Since X and $\text{Diff}(w|_L, w'|_L)$, as well as Y and $\text{Diff}(w|_L, w''|_L)$ are disjoint sets, it must be the case that

  - $\text{Diff}(w|_L, w'|_L) = \text{Diff}(w|_L, w''|_L)$ and $X \subset Y$, or that
  - $\text{Diff}(w|_L, w'|_L) \subset \text{Diff}(w|_L, w''|_L)$ and $X \subseteq Y$

  However, the first case cannot occur: it implies that $w'|_L = w''|_L$, which by spaningness implies that $w' = w''$, contradicting the hypothesis. Hence we have shown that for every possible L-world $w''|_L \neq w'|_L$ satisfying $Post_{i,j}$ it is true that $\text{Diff}(w|_L, w'|_L) \subset \text{Diff}(w|_L, w''|_L)$. Thus, we have proved (7).

- From (5), (7), and spaningness, we have

  $$K \cup w|_L\text{-Closest}_K|_L(Post_{i,j}) \;\text{ is logically equivalent to }\; K \cup w\text{-Closest}_K(Post_{i,j}),$$

which, using (6), transfers from single worlds and postconditions to arbitrary formulas and the result functions r_K and $r_K|_L$, respectively. As w, α, Pre_i were chosen arbitrarily, the claimed equivalence follows.