Turning an Action Formalism Into a Planner—Essentials of a Case Study

Joachim Hertzberg[1] and Sylvie Thiébaux[2]

[1] Dortmund University, Informatik VIII, 44221 Dortmund, Germany, and GMD, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
[2] IRISA, Campus de Beaulieu, 35042 Rennes, France

Abstract. We describe notions that are useful for building planners whose reasoning about action matches a given formal calculus for such reasoning. As a planner has to reason under resource constraints, one may require that its plans are just approximately correct wrt. the action formalism. To nonetheless ground its practical reasoning formally, we develop the notion of limited correctness of a planner wrt. an action formalism.

Even if we were to grant that most of the advances in AI are inspired by experimentalists and that formalists serve mainly to "tidy up", it is nevertheless our opinion that the important new results in AI will be achieved by those researchers whose experiments are launched from the high platform of solid theory. — Genesereth & Nilsson [6, p. viii]

1 Background

The purpose of an *action formalism* is to specify the reasoning about prerequisites and consequences of actions that an ideally rational agent should perform; among the many examples for such formalisms are the situation calculus [9], Pednault's [11] formalization for determining possibly context-dependent effects, and the possible-models-based formalism suitable for context-dependent and non-deterministic effects under incomplete information described in [2], which underlies our study. A *planner* has to do reasoning about prerequisites and consequences of actions, but if it insists on doing it in an ideally rational way, then it is likely to fail on most realistic usage constraints for practical applications. Consequently, a typical planner either does only trivial reasoning about action, or a neat formal description of its reasoning about action yields surprise [8], or it just has no neat description. In sum, it seems as if the camp of interesting planners and the camp of interesting action formalisms are way apart. However, this gap should be bridged, because not bridging it would imply that planning has to reinvent much of the work already done in action formalisms.

A natural idea for bridging the gap is to design planners whose plans are only *approximately* correct wrt. to a given action formalism. However, it is not completely clear what a useful approximation should be here: plausible alternatives are probabilistic approximations in analogy to PAC (probably approximately correct) learning [14], and run-time dependent approximations that, in the limit, converge to correct reasoning. For this paper, we chose the second alternative, because it combines nicely with viewing planning as an anytime [3] activity, which, in turn, seems plausible under incomplete information and non-deterministic action effects as formalized in [2].

This paper summarizes a case study in which we have developed the planner PASCALE2, proceeding along the lines just sketched; the study is described comprehensively in a full paper [7]. For lack of space, we cannot present in any interesting detail particularities of the formalism that we used for the study, of the corresponding plan format, or of the resulting planner; the reader is referred to the full paper. We do present here the more general notions of building a planner starting from some action formalism. Note, however, that this is based on a generalization from one single complex case; so it may turn out that even if our notions and procedure were completely adequate, other formalisms should be approximated differently in planners.

Section 2 introduces the basic notions of action formalisms, planning problems, plans, and planners generating these plans. Moreover, the section introduces the notion of "strict" correctness and completeness of planners wrt. an action formalism. Section 3 defines and discusses limited planner correctness, the central concept for approximating the reasoning of an action formalism. Section 4 points to related work and concludes.

2 Action Formalisms, Problems, Plans, and Planners

To describe how to build a planner on the basis of an action formalism, we have to state more clearly what an action formalism and a planner are in the first place. As we do not want to go into the details of one particular action formalism here, we will only state their general aspects that are necessary for this paper. In particular, we do not really define what plans and planners are, but we only state requirements that we assume every concrete such object to fulfill; realistic plans and planners will include many more substructures or procedures than we address here, but the claim is that, e.g., any planner having at least those components that we require a planner to have, can be looked at in the way we describe.

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 different application domains, involving or not involving, resp., incompleteness of information, time, non-deterministic action effects, or other features. Different action formalisms intended for domains of different characteristics should mirror these differences, and so will the respective planners and their related concepts like planning problem and plan. Let us start with characterizing action formalisms.

At the level of abstraction required here, an action formalism is a formal apparatus that takes the description of a state of affairs in the world and the description of an action, and yields a description of (possible) *resulting* states of affairs. An example is the situation calculus a world state is described by a set of first order formulas, an action is represented by a syntactical function called operator, and the action effects are given by another (unique) world state where the resulting changes are axiomatized by first order formulas, too. Note that formalisms admitting for multiple alternative action effects may yield non-unique successor states: every formalism capable of expressing the action of tossing a coin is an example.

Practically, one might wish to require that the formalism be correct in the sense that, given an intuitively "right" description of the previous state and a "right" description of an action, it yields a "right" (set of) successor state description(s). However, this is no issue here: in the following, we assume that all action formalisms are correct in this sense. For simplicity, we restrict the language for describing world states to first order logic. Moreover, we assume a language for describing actions, every individual such description being called an *operator.* In the example of situation calculus, this action language is a first order language. An action formalism, then, consists of a language for describing situation, a language for describing actions, and a mapping that axiomatizes the result of action applications subject to the following constraints:

Requirement 1 (Result) *Given a first order language $\mathcal{L}$ and a language $\mathcal{A}$ for describing actions (each such description called an* operator*), the result of an operator is described by a function $r : \mathcal{L} \times \mathcal{A} \rightarrow 2^{\mathcal{L}}$.*

Turning now to the planning side of the story, the "classical" definition of a planning problem, and, accordingly, of a plan for solving it, requires that a problem consist of start state, goal description, and a set of operators. We slightly generalize this view:

Requirement 2 (Planning problem description) *Let $\mathcal{L}$ be a first order language and $\mathcal{A}$ an operator language. A planning problem description over $\mathcal{L}$ and $\mathcal{A}$ (or problem, for short) is a quadruple $\langle s, g, K, O \rangle$, where $s \in \mathcal{L}$ describes the initial situation (possibly incompletely), $g \in \mathcal{L}$ describes the goal, $K \subseteq \mathcal{L}$ is the background knowledge, and $O \in \mathcal{A}$, the operator inventory, is a set of operators.*

It is useful to identify background knowledge as that part of the domain description which does not change by any application of actions; examples are causal laws or descriptions of terminological equivalences.

It sometimes helps to include an initial plan in a problem description that may be used as the starting point of a planner, where some planning techniques might start with a plan dummy consisting of pseudo-operators representing descriptions of the initial and final states, and other others might start with a plan skeleton or a finished library plan that have to be properly instantiated or modified. Trying to keep separated the notions of problem and plan, we do not follow this pragmatically useful idea. Remember that this and the following requirements are just meant to specify constraints for the respective notions that get used later in this text; concrete theories or implementations may add other components as they need.

Classically, a plan is a set of occurrences of operator instances with a (possibly partial) order imposed on them, and we will keep these requirements. Note, however, that we do not require that the order be interpreted in the usual, "classical" STRIPS [5] way or that actions have unique effects: in the plan format in [7], we allow,e.g., for actions with multiple alternative effects. The additional requirement has to do with how to determine operator results in a plan.

Classical planning assumes the result of an operator to be independent from the situation in which it is applied, facilitating to determine these results looking at the operator alone. In general, this is too simple, and we need a more elaborated means to tell what the effects or set of possible effects are of an operator in the plan. How this is determined has to do with the underlying time model (are operator executions allowed to overlap? Is the effect of overlapping action execution different from the union of operator effects?), with the plan representation (are effects explicitly specified in the plan?), and possibly other factors. On the other hand, there *must* be some such means, because it must be possible to determine whether some operator in the plan has its preconditions satisfied and whether the goal description is fulfilled. Consequently, we require that there be some function that, given a plan and one of its operators, tells what this operator effects in; this is described by a set of formulas, corresponding to possible effect alternatives yielded by the operator itself or by different contexts it can be applied in. This leads to the following requirements for plans:

Requirement 3 (Plan) *Let $\Psi = \langle s, g, K, O \rangle$ be a problem over $\mathcal{L}$ and $\mathcal{A}$. A plan for Ψ is a triple $\langle \mathcal{I}, \Omega, \prec \rangle$, where*

- *Ω is the set of operator occurrences, i.e., elements of Ω are elements of O, but identical operators may occur more than once and are distinguishable;*
- *$\mathcal{I}$ is the initial node, i.e., a pseudo- (non-executable) operator meant to (possibly incompletely) describe the initial state. For notational convenience, we require that $\mathcal{I} \in \Omega$.*
- *$\prec$ is the operator order, i.e., an ordering relation on Ω, such that $\mathcal{I}$ is the unique $\prec$-smallest element.*

Let plan *and* $\mathsf{operator}$ *denote the types of plans and operators in plans, resp.*

There is a function r_p : $\mathsf{plan} \times \mathsf{operator} \longrightarrow 2^{\mathcal{L}}$ returning the result of an operator in a plan.[3]

Note that the operator ordering may be interpreted differently, depending on the time model that is used.

These requirements fulfilled, we can now define the notions of (strict) correctness and completeness of a *plan* wrt. an action formalism and a problem description Ψ. The intuition behind plan *correctness* is to say that if a plan Π 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 Π, using r_p, will rightly predict what may result, according to the action formalism's r, from applying an action; 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

[3] For notational convenience, we identify the type plan with the *set* of plans here; analogously for operators.

which Ψ's goal formula is true. (All this can, of course, only be guaranteed for a plan, if the planning problem description itself was intuitively correct wrt. the real world domain).

The intuition behind plan *completeness* is that if something may happen at plan execution according to Ψ and the action formalism, then the plan does already respect it. In particular, the plan start must subsume the initial situation, and the plan must contain all results of actions that it directs to apply. Both plan correctness and completeness can be defined rigorously, but we do not do so here for lack of space. In the full paper [7], we have given the respective definitions for the particular type of plans used there.

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 on planning—albeit mostly implicit. We make them explicit here in order to deal with controlled relaxations of them; this will be the key for achieving limited correctness of planners in the next section.

The notions of correctness and completeness apply analogously to *planners,* not only to plans. In our context, a planner is an entity that, given a planning problem description, returns a plan after a number of computation steps. These steps may be time ticks, plan expansion steps or more generally, any monotonically increasing time function. For simplicity, we assume a discrete measure over the natural numbers here. Moreover, we assume the existence of types natural and problem description containing objects that are natural numbers and problem descriptions, respectively. Under the theoretical view that we are having here, a planner is then constrained as follows:

Requirement 4 (Planner) *A planner is a pair* $\langle \mathcal{P}, A_\mathcal{P} \rangle$ *of functions, where* $\mathcal{P}$: problem description $\rightarrow$ natural $\rightarrow$ plan *is the* plan generation function *and* $A_\mathcal{P}$: problem description $\rightarrow$ natural set*, the* availability function, *is a total function 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 [3].

The intuitive idea behind planner correctness or completeness, then, is simple: A planner is *correct* wrt. some criterion to be specified, if all the plans it delivers for a planning problem meet this criterion; and it is *complete,* 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. The criterion that we will hard-wire into the definitions of both planner correctness and completeness is supposed to be 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.

Definition 1 (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 2 (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 $t \in A_{\mathcal{P}}(\Psi)$ s.t. $\mathcal{P}(\Psi)(t) = \Pi$.

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. An example for a trivially complete planner is one that enumerates all plans: 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 our reference point. Defining this more liberal version of correctness is what we will do now.

3 Correctness, Ltd.

In view of our goal to achieve planners that are only correct in the limit relative to an action formalism, we allow, but do not require that such a planner use a "direct" implementation of the formalism. Instead, one may use a restriction, or a formalism implementation that in itself only approximates the formalism's result function r. 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. However, we cannot exemplify this any further here, given that this paper is formalism-independent; [7, Sec. 4.1] gives an example.

The main tool for expressing limited correctness is a rating function for incomplete and incorrect plans 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 3 (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: $\varrho_\Psi(\Pi) = 1$ iff Π is correct and complete wrt. Ψ

In the rest of the paper, we sloppily speak of some ϱ as a *family* of rating functions for a problem *domain* with potentially many problem descriptions, to express that ϱ_Ψ is a rating function for every problem Ψ in the domain.

The notion of correctness in the limit is based on the following idea. Standard notions of planner correctness and completeness (Definitions 1 and 2) 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, it is even worthless to generate more than one such plan: one is as good as all the others. 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 (1) the plans delivered get more correct and more complete as run time increases, and that (2) a correct and complete plan is returned eventually at some step t. Of course, we do not assume that a limitedly correct planner is given as much as t steps, nor 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 4 (Planner correctness in the limit). Let ϱ be a rating function family. A planner $\langle \mathcal{P}, A_{\mathcal{P}} \rangle$ is limitedly correct wrt. ϱ, iff for all problems Ψ:

1. for all $t_1, t_2 \in A_{\mathcal{P}}(\Psi)$, if $t_2 \geq t_1$ then $\varrho_\Psi(\mathcal{P}(\Psi)(t_2)) \geq \varrho_\Psi(\mathcal{P}(\Psi)(t_1))$, and
2. if there is a correct and complete plan wrt. Ψ, then there exists $t \in A_{\mathcal{P}}(\Psi)$ s.t. $\varrho_\Psi(\mathcal{P}(\Psi)(t)) = 1$

Comparing this definition with the one of "strict" planner correctness (Definition 1), a remarkable theoretical difference is that limited correctness requires that at least one correct and complete plan be returned, whereas strict 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 (item 1) 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 given this, it seems reasonable to require a non-arbitrary, 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 other quality sub-functions. We neither require nor exclude these.

Note that 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 intuitively expected. Consequently, all correct and complete planners are limitedly correct wrt. to all rating functions. Note further that incorrect, incomplete, and both incorrect and incomplete planners may be limitedly correct wrt. some rating function. This 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 formalism. Of course, being able to establish this relationship does not come for free. 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.

4 Related Work and Conclusion

So what have we achieved? There are two ideas behind this work, which are basically independent, but fit together well: First, the output of a planner should be describable relative to some rigorously defined action formalism; second, approximating the formalism is sufficient for this describability, given realistic constraints on a planner's resources. Both these ideas are not new, but combining them seems uncommon. Let us briefly put them into the perspective of the literature and then conclude.

There is an increasing amount of work providing logical formalizations of planning systems or of their basics, e.g., [8, 11]. The rationale is that formally rigorous descriptions can help understand 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 two ways to provide a planner with a neat formalization. The first is the *ex post* way: The planner comes first, and the attempt to formalize it, later; Lifschitz's [8] semantics of STRIPS [5] is the most prominent example, and it also demonstrates that the planner implementors may have used 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 [11] formalization for determining possibly context-dependent (i.e., conditional) action effects using regression, with a planner implementation provided by McDermott [10].

The *ex post* strategy—while perfectly adequate when the field of planning was in its infancy—does not appear wise for planning as a maturing engineering sub-field: It gives you no guidance as to from which theory platform to start with your action representation and calculus.[4] If STRIPS operators were an adequate action representation for all interesting application domains, then there would be no problem, [8] telling you all formal background you need; but most planning researchers seem to agree that they are not. Consequently, we need a blueprint for using a given operator formalism in a to-be-designed planner.

A somewhat traditional method for going this *ex ante* way is to do *deductive planning* [1]. If you can afford to use a planner that is correct and complete in the sense of Definitions 1 and 2 (or in some other "strict" sense), and if you have a suitable deductive theorem prover at hand, then this is probably the method of choice. However, assuming that these conditions are not universally true, the

[4] Obviously, it gives you no guidance either as to how to write your planner, e.g., which search strategy and heuristics to employ. But this is a different point. Here, we have to do with the *representation* aspects.

blueprint must look different, and we have proposed to base it on the idea of a planner *approximating* the action formalism in the sense of Definition 4. Within AI research, this has its roots back in Simon's [13] idea of *bounded rationality;* more recently, Russell's group has worked on *limited rationality* (e.g., [12]). Another related idea is anytime planners [3], focusing on the requirement that planning has to be a timely activity. Our contribution is to connect the ideas of reasoning within limited resources and of grounding a planner's reasoning about action in a pre-specified action formalism.

Space not permitting to describe more than just the most general framework for this connection, let us finally sketch at least the most characteristic features of the planner PASCALE2 that we have developed according to the notions just introduced [7]. Starting point is the action formalism described in [2] that allows

- information about the world to be incomplete in the sense that it may be only known that $p \vee q$ is the case (for arbitrary formulas p, q) or that it is not known whether p or $\neg p$ is true;
- actions to have alternative effects, the minimalistic standard example being the action of tossing a coin whose effect is either Heads or Tails; and
- actions to have context dependent effects, the standard example being to toggle a light switch, resulting in lights On if Off before, and vice versa.

Moreover, [7] allows probability information to be added to the domain representation, describing, e.g., that, lacking other information, the light is Off with a 40% chance, or that tossing an (unfair) coin yields Heads with a 70% chance.

Plans matching this formalism must be highly conditional, and the effects of operators not unique, so classical nonlinear plans are inappropriate. Assuming that the language $\mathcal{L}$ for describing the domain is finite, the respective plan format is much like a finite non-deterministic automaton: The nodes correspond to finite descriptions of world states; every such state has an operator associated; and the arcs correspond to the possible outcomes of applying this operator. Additionally, the probability information allows arcs to be labeled with transition probabilities. Executing such a plan means to react with the specified action, once the plan executor finds itself in a world state matching a specified description.

In sum, PASCALE2 has many features of modern decision-theoretic anytime planners much like,e.g., the one described in [4], yet is based on the safe platform of a well-understood action formalism. The probabilistic information makes it particularly intuitive to define a rating function using decision-theory: the degree of *plan* correctness and completeness is the expected cumulative value of the *operators* in that plan (this value expressing the overlap between the r and r_p functions). Just to give a flavor of the behavior of a limitedly correct planner, Figure 1 shows the values of PASCALE2's rating function (called CCD) over run time on a demo problem explained in [7].

References

1. S. Biundo. Present-day deductive planning. In C. Bäckström and E. Sandewall,

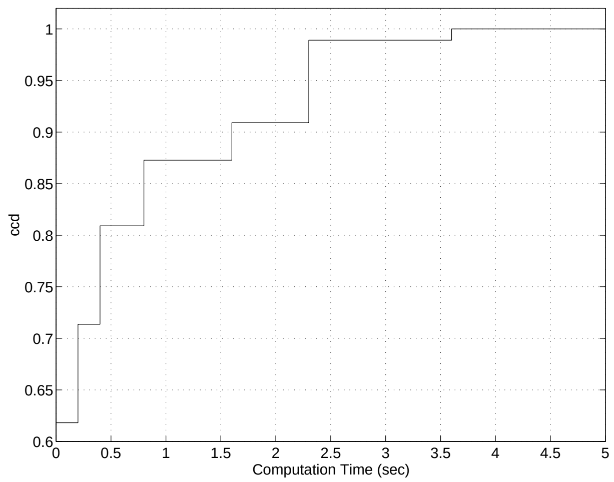

Fig. 1. Plan quality as a function of computation time on a demo problem [7].

editors, *Current Trends in AI Planning. EWSP'93 – 2nd European Workshop on Planning*, pages 1–5. IOS Press, 1994.

2. G. Brewka and J. Hertzberg. How to do things with worlds: On formalizing actions and plans. *J. Logic and Computation*, 3(5), 1993.
3. T.L. Dean and M. Boddy. An analysis of time-dependent planning. In *Proc. AAAI-88*, pages 49–54, 1988.
4. T.L. Dean, L. Kaelbling, J. Kirman, and A. Nicholson. Planning with deadlines in stochastic domains. In *Proc. AAAI-93*, pages 574–579, 1993.
5. R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to theorem proving in problem solving. *J. Art. Intell.*, 2:189–208, 1971.
6. M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Palo Alto, CA, 1988.
7. J. Hertzberg and S. Thiébaux. Turning an action formalism into a planner—a case study. *J. Logic and Computation (Sp. Is. Actions and Processes)*, 4(5), Oct. 1994.
8. V. Lifschitz. On the semantics of STRIPS. In M.P. Georgeff and A.L. Lansky, editors, *Proc. 1986 Workshop Reasoning about Actions and Plans*, pages 1–9, Los Altos, 1987. Morgan Kaufmann.
9. J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–507, 1969.
10. D. McDermott. Regression planning. *Int. J. Intell. Syst.*, 6(4):357–416, 1991.
11. E.P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *J. Computational Intelligence*, 4:356–372, 1988.
12. S. Russell and E. Wefald. *Do the Right Thing. Studies in Limited Rationality.* MIT Press, Cambridge, Massachusetts, 1991.
13. H. A. Simon. *Administrative Behavior. A Study of Decision-Making Processes in Administrative Organizations.* Free Press, New York, 3 edition, 1976. Original: New York (Macmillan) 1947.
14. L. Valiant. A theory of the learnable. *C. ACM*, 27:1134–1142, 1984.