Abstract Operators, Indeterminate Effects, and the Magic Predicate

Fangzhen Lin
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
email: fl@ai.toronto.edu

Abstract

To make the task of planning more manageable, an effective technique is to use abstract operators. However, abstract operators often have indeterminate effects, and most current planning algorithms can only deal with actions with only determinate effects. Instead of trying to extend planning algorithms to deal with indeterminate actions, we introduce a special "magic" predicate and reduce indeterminate actions to determinate ones.

1 Introduction

To make the task of planning more manageable, an effective technique is to use abstract operators.

Consider the 8 puzzle (Figure 1). One way to solve it is to introduce the following 8 abstract operators (Korf [5]): For $0 \leq i \leq 7$, if all numbers $< i$ are already in their goal positions, then M_i will move i to its goal position, and leave all numbers $< i$ where they were. For instance, M_0 moves 0 to its goal position, and has indeterminate effects on all other numbers. If the numbers 0 and 1 are already in their respective goal positions, then M_2 moves 2 to its goal position, leaves the

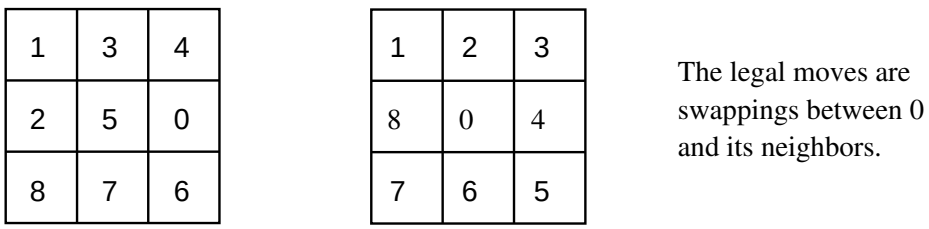

Figure 1: The 8 puzzle

positions for the numbers 0 and 1 unchanged, and has indeterminate effects on all other numbers. Notice that if all numbers < 7 are already in their respective goal positions, then M_7 will move 7 to its goal position without violating the already achieved goal positions for all the smaller numbers. This means that M_7 will move 8 to its goal position as well. With these 8 abstract operators, a solution to the puzzle becomes apparent: perform in sequence the abstract actions $M_0, ..., M_7$.

Of course, we do not really solve the puzzle until we have specified how the abstract operators are realized in terms of the available primitive actions. One can think of at least two ways how this may be done. One is to ask the user to provide explicit definitions of the abstract operators. In this case, abstract operators are macros[1] or more generally, programs constructed from the available primitive actions. The other is to treat an abstract operators as a run time dynamic goal to achieve. For instance, to realize M_0, the planner generates the goal of achieving 0 in its goal position. The sequence of primitive actions the planner comes up with for this goal will then be the realization of this abstract operator.

In both cases, however, the planner needs to know the effects of the available abstract operators. For instance, given a goal to achieve, it needs to know which, if any, of the available abstract operators will bring about the goal. In the first case, i.e. when the abstract operators are explicitly defined by the user, the effects of abstract operators can either be inferred from their definitions or provided by the user. Although the former is preferable, the latter is perhaps more realistic, given the complexity of formal reasoning about programs, in particular when there are loops and recursion. In the second case, i.e. when the planner is required to come up with the definitions of the abstract operators at run time, it is necessary that the user provide formal specifications of the effects of these abstract operators.

The problem with formally specifying the effects of abstract operators, however, is that these effects are often indeterminate. For instance, the description of M_0 leaves open how the positions of the numbers greater than 0 are affected by the abstract operator.

There have been some proposals in the literature for specifying the effects of indeterminate actions (e.g. Barel [1], Boutilier and Friedman [2], Kartha and Lifschitz [4], and Zhang [15]). But none of them seem suitable for our purpose. We are particularly interested in backward chaining, goal-directed planners.[2] Implicitly or explicitly, these planners rely on *regression* [13]. For instance, after deciding to use M_7 to move 7 to its goal position, the planner needs to know the conditions under which M_7 will have this effect. The planner also needs to know if, and under what conditions, the operator M_7 will have effects on other outstanding goals. Answering these questions amounts to computing the regression of a formula through the operator. To our knowledge, none of the existing proposals provide ways to compute regression through an indeterminate operator.

The approach we'll take in this paper is based on Vladimir Lifschitz's idea[3] of using a special predicate to reduce indeterminate actions to determinate ones. Briefly, if the fluent p is an

[1] This is how the above 8 abstract operators are defined in [5] for the 8 puzzle.

[2] Most classical AI planners (e.g. [3, 10]) are of this type.

[3] Private Communication, 1992. Ray Reiter has let us know that Charles Elkan has also proposed a similar idea in 1992.

indeterminate effect of the action a under the context γ, then we write

$$Poss(a, s) \wedge \gamma \wedge Magic(a, p, s) \supset Holds(p, do(a, s)),$$

where $Magic$ is a special fluent, and $Magic(a, p, s)$ denotes the unknown conditions under which the action a will cause the fluent p to become true. As we shall see, this approach appears providing satisfactory solutions to some difficult examples involving the indeterminate effects of actions. Also, by reducing indeterminate actions to determinate ones, solutions to the frame, the ramification, and the qualification problems developed for determinate actions can be inherited for indeterminate ones. More importantly, well known techniques such as regression can now be extended to indeterminate actions.

2 The Situation Calculus

We shall reify fluents, and treat them as first-order objects. As usual, $Holds(p, s)$ denotes that the fluent p holds in the situation s. If F is a fluent constant, then we'll use $F(s)$ as a shorthand for $Holds(F, s)$. In addition to $Holds$, we have the following special predicates:

- $Poss(a, s)$ – the action a is executable in s.
- $Magic(a, p, s)$ – the special predicate used to denote the unknown conditions about the situation s under which the action a will make p true, provided p is an indeterminate effect of the action a.
- $s < s'$ – the situation s' is reachable from s by performing an executable sequence of actions.

In the following, we call a formula $\varphi(s)$ a *simple state formula about* s if in it, s is the only, if any, situation term in φ, and $Holds$ and $Magic$ are the only, if any, situation dependent predicates.

2.1 Basic Action Theories

With determinate actions, given a domain of interest, our goal is to generate a *basic action theory* [9, 12]:

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0},$$

where

- Σ is the set of *foundational axioms* [9, 14] for the situation calculus.
- $\mathcal{D}_{ap}$ is a set of *action precondition axioms*, one for each action, of the form $Poss(A, s) \equiv \Phi(s)$, where $\Phi(s)$ is a simple state formula about s, and does not mention $Magic$.

- $\mathcal{D}_{ss}$ is a set of successor state axioms, one for each fluent, of the form

$$Poss(a, s) \supset F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s), \tag{1}$$

  where Φ_F is a simple state formula about s, and does not mention $Magic$.
- $\mathcal{D}_{una}$ is the set of unique names axioms about actions and fluents.
- $\mathcal{D}_{S_0}$ is a set of simple state formulas about S_0.

Since we'll be reducing indeterminate actions to determinate ones using the $Magic$ predicates, our concern is still with basic action theories. However, for indeterminate actions, we shall allow the formula Φ_F in the successor state axiom (1) for F to include the $Magic$ predicate, and in addition to the above components, we have one more set $\mathcal{D}_{magic}$ to include constraints about the $Magic$ predicate, since we will not have a successor state axiom for it.

3 Axiomatizing Indeterminate Effects of Actions

We illustrate with the coin tossing example. Tossing a coin causes it to land on one of its two sides. However, we mortal men normally don't know enough to predict accurately which side the coin will land on. Assuming, which we always can, that $Magic(head(x), toss(x), s)$ denotes the unknown conditions under which tossing the coin x will cause it to land on its head, we have the following effect axiom:

$$Poss(toss(x), s) \wedge Magic(toss(x), head(x), s) \supset head(x, do(toss(x), s)). \tag{2}$$

Since $toss(x)$ will either cause x to land on its head, or cause x to not land on its head (on its tail), we have

$$Poss(toss(x), s) \wedge \neg Magic(toss(x), head(x), s) \supset \neg head(x, do(toss(x), s)). \tag{3}$$

Similarly, we have:

$$Poss(toss(x), s) \wedge Magic(toss(x), tail(x), s) \supset tail(x, do(toss(x), s)), \tag{4}$$

$$Poss(toss(x), s) \wedge \neg Magic(toss(x), tail(x), s) \supset \neg tail(x, do(toss(x), s)). \tag{5}$$

Now since $tail$ is an antonym of $head$, we have the following constraint about $Magic$:

$$[Magic(toss(x), head(x), s) \vee Magic(toss(x), tail(x), s)] \wedge \neg[Magic(toss(x), head(x), s) \wedge Magic(toss(x), tail(x), s)]. \tag{6}$$

Suppose now that (2) and (3) are the only effect axioms, and there are no domain constraints, then we can generate the following successor state axioms for $head$ and $tail$ (see Reiter [13]):

$$\begin{aligned} Poss(a, s) \supset \{head(x, do(a, s)) \equiv [a = toss(x) \wedge Magic(toss(x), head(x), s)] \vee \\ [head(x, s) \wedge \neg(a = toss(x) \wedge \neg Magic(toss(x), head(x), s))]\}, \\ Poss(a, s) \supset \{tail(x, do(a, s)) \equiv [a = toss(x) \wedge Magic(toss(x), tail(x), s)] \vee \\ [tail(x, s) \wedge \neg(a = toss(x) \wedge \neg Magic(toss(x), tail(x), s))]\}. \end{aligned}$$

Except for mentioning the $Magic$ predicate, these are just like ordinary successor state axioms for actions with only determinate effects. They entail the above effect axioms, and also the following frame axiom:

$$Poss(a, s) \wedge a \neq toss(x) \supset head(x, do(a, s)) \equiv head(x, s).$$

It also follows from the successor state axioms that

$$Poss(toss(x), s) \supset head(x, do(toss(x), s)) \equiv Magic(toss(x), head(x), s),$$

and similarly for $tail$. Thus, together with the constraint (6) on $Magic$, we have

$$\begin{aligned} Poss(a, s) \supset \{&[head(x, do(toss(x), s)) \vee tail(x, do(toss(x), s))] \wedge \\ &\neg[head(x, do(toss(x), s)) \wedge tail(x, do(toss(x), s))]\}. \end{aligned}$$

However, there will be no successor state axiom for the special predicate $Magic$. The only constraint on it will be (6), i.e. $\mathcal{D}_{magic} = \{(6)\}$. So, in planning, if say $Magic(toss(x), head(x), s)$ is an outstanding subgoal, then we know that the whole goal will be unachievable. We'll say more about this later, in Section 6

3.1 Exclusive Indeterminate Effects

The indeterminate effects of $toss(x)$ are exclusive in the sense that exactly one of $\{head(x), tail(x)\}$ will be caused to be true.

Generally, if the action A has *exclusive indeterminate effects* on $F_1, \cdots, F_n$, i.e. causes exactly one of them to be true and the rest of them to be false, under the context γ, then we'll have the following effect axioms:

$$\begin{aligned} Poss(A, s) \wedge \gamma \wedge Magic(A, F_1, s) &\supset F_1(do(A, s)), \\ Poss(A, s) \wedge \gamma \wedge \neg Magic(A, F_1, s) &\supset \neg F_1(do(A, s)), \\ &\vdots \\ Poss(A, s) \wedge \gamma \wedge Magic(A, F_n, s) &\supset F_n(do(A, s)), \\ Poss(A, s) \wedge \gamma \wedge \neg Magic(A, F_n, s) &\supset \neg F_n(do(A, s)), \end{aligned} \tag{7}$$

and the following constraints:

$$\gamma \supset [Magic(A, F_1, s) \dot{\vee} \cdots \dot{\vee} Magic(A, F_n, s)], \tag{8}$$

where $\dot{\vee}$ is the exclusive or operator, and is defined as:

$$\varphi_1 \dot{\vee} \cdots \dot{\vee} \varphi_k \equiv (\varphi_1 \vee \cdots \vee \varphi_k) \wedge \bigwedge_{1 \leq i \neq j \leq k} \neg(\varphi_i \wedge \varphi_j).$$

The number of indeterminate effects need not be finite. If, under the context γ, the action A has exclusive indeterminate effects on $F(x)$ for those x that satisfy φ, then we have the following effect axioms

$$Poss(A, s) \wedge \gamma \wedge \varphi(x) \wedge Magic(A, F(x), s) \supset F(x, do(A, s)),$$
$$Poss(A, s) \wedge \gamma \wedge \varphi(x) \wedge \neg Magic(A, F(x), s) \supset \neg F(x, do(A, s)),$$

and the constraint:

$$\gamma \wedge (\exists x)\varphi(x) \supset \{(\exists! x).\varphi(x) \wedge Magic(A, F(x), s)\},$$

where $(\exists! x)$ means that there is a unique x. For instance, picking a ball from a bag causes one to hold one and only one of the balls in the bag: let γ be $true$, $\varphi(x)$ be $in\text{-}bag(x, s)$, and $F(x)$ be $holding(x)$.

3.2 Inclusive Indeterminate Effetcs

Consider Reiter's example of dropping a pin on a checkerboard.[4] This is an action with *inclusive indeterminate effects*: the pin may land on a white square, a black square, or both. Inclusive or has traditionally been problematic for nonmonotonic logics. For instance, circumscribing P in $a \neq b \wedge (P(a) \vee P(b))$ will turn "$\vee$" into an exclusive or. Perhaps for the same reason, current proposals for formalizing indeterminate actions either cannot handle Reiter's example or require many more axioms than that for exclusive indeterminate effects.

As it turns out, representing inclusive indeterminate effects is easier than exclusive ones in our framework, for the reason that logical or in classical logic is inclusive.

Generally, if the action A has inclusive indeterminate effects on $F_1, \cdots, F_n$ under the context γ, then we still have the indeterminate effect axioms (7), but instead of the constraint (8), we have

$$\gamma \supset [Magic(A, F_1, s) \vee \cdots \vee Magic(A, F_n, s)].$$

For instance, for Reiter's example, we have the following effect axioms:

$$Poss(drop, s) \wedge Magic(drop, white, s) \supset white(do(drop, s)),$$
$$Poss(drop, s) \wedge \neg Magic(drop, white, s) \supset \neg white(do(drop, s)),$$
$$Poss(drop, s) \wedge Magic(drop, black, s) \supset black(do(drop, s)),$$
$$Poss(drop, s) \wedge \neg Magic(drop, black, s) \supset \neg black(do(drop, s)),$$

and the following constraint on $Magic$:

$$Magic(drop, white, s) \vee Magic(drop, black, s). \tag{9}$$

[4] Private communication

4 Reiter's Consistency Conditions

Intuitively, $Magic(a, p, s)$ denotes the *unknown* conditions under which the action a will cause p to be true. An interesting issue arises when some of the unknown conditions become known.

Consider again the coin tossing example. Suppose we know that if a coin is unfair, then tossing it will always cause it to land on its head:

$$Poss(toss(x), s) \supset unfair(x) \supset head(x, do(toss(x), s)). \tag{10}$$

Now from (10) and (3), we have:

$$Poss(toss(x), s) \supset \neg[unfair(x) \wedge \neg Magic(toss(x), head(x), s)]. \tag{11}$$

This axiom has the same form as that for action qualification axioms such as

$$Poss(toss(x), s) \supset coin(x) \wedge holding(x, s).$$

However, it is wrong to include $Magic$ in action precondition axioms such as

$$Poss(toss(x), s) \equiv \neg[unfair(x) \wedge \neg Magic(toss(x), head(x), s)] \wedge coin(x) \wedge holding(x, s).$$

This is because, intuitively, $Poss(a, s)$ is the condition under which the action a is executable, and the common context for all the effects of the action a. So this condition should not refer to specific contexts like $Magic(toss(x), head(x), s)$ that are for particular effects like $head(x)$. For instance, if we also know that tossing a coin will cause it to land on the floor:

$$Poss(toss(x), s) \supset on\text{-}floor(x, do(toss(x), s)),$$

then it is both ridiculous and unfeasible that in order to conclude $on\text{-}floor(x, do(toss(x), S_0))$, we'd have to check certain conditions involving $Magic(toss(x), head(x), s)$!

Generally, we'll have this problem when the *consistency conditions* (Reiter [12]) for effect axioms are violated: Two effect axioms of the forms

$$Poss(a, s) \wedge \gamma^+ \supset F(\vec{x}, do(a, s)),$$
$$Poss(a, s) \wedge \gamma^- \supset \neg F(\vec{x}, do(a, s))$$

are said to satisfy the consistency condition iff

$$\mathcal{D}_{una} \models (\forall \vec{\xi}) \neg(\gamma^+ \wedge \gamma^-),$$

where $\vec{\xi}$ is the tuple of the free variables in $\gamma^+ \wedge \gamma^-$. In the case of determinate actions, it may be reasonable to assume, as Reiter does in [12], that the users are responsible for maintaining this condition. But in our case, it seems intuitive that the solution is to amend indeterminate effect axioms. In our coin tossing example, this means that we should revise the axiom (3) into:

$$Poss(toss(x), s) \wedge \neg unfair(x) \wedge \neg Magic(toss(x), head(x), s) \supset$$
$$\neg head(x, do(toss(x), s)).$$

According to McCarthy's principle of elaboration tolerance [11], it is desirable to write indeterminate effect axioms in a form that is general enough so that such revision would not be necessary. One way of doing this will be to introduce two special predicates ϵ^+ and ϵ^-, and instead of (2) and (3) say, write:

$$Poss(toss(x), s) \wedge Magic(toss(x), head(x), s) \wedge \epsilon^+(toss(x), head(x), s) \supset head(x, do(toss(x), s)),$$
$$Poss(toss(x), s) \wedge \neg Magic(toss(x), head(x), s) \wedge \epsilon^-(toss(x), head(x), s) \supset \neg head(x, do(toss(x), s)),$$

together with appropriate second-order axioms axiomatizing that ϵ^+ and ϵ^- are *weakest* relations such that all consistency conditions are satisfied:

$$\mathcal{D}_{una} \models \neg[Magic(toss(x), head(x), s) \wedge \epsilon^+(toss(x), head(x), s) \wedge \neg Magic(toss(x), head(x), s) \wedge \epsilon^-(toss(x), head(x), s)],$$
$$\mathcal{D}_{una} \models \neg[unfair(x) \wedge \neg Magic(toss(x), head(x), s) \wedge \epsilon^-(toss(x), head(x), s)].$$

Due to the technical complexity of this approach and the space limitation, we shall not pursue this any further.

5 Some Properties of Basic Action Theories

To recast, a basic action theory for actions with possibly indeterminate effects is

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{magic} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}.$$

Normally, $\mathcal{D}_{ap}$ and $\mathcal{D}_{S_0}$ will not mention $Magic$, although they may in principle.

In the rest of this section, we shall assume that $\mathcal{D}_{magic}$ is a set of axioms of the form: $(\forall s).\varphi(s)$, where $\varphi(s)$ is a simple state formula about s, and does not mention $Holds$. In other words, the only situation dependent predicate in φ is $Magic$.

Roughly, this restriction on $\mathcal{D}_{magic}$ means that the indeterminate effects of actions are context free, and are not indexed by a fluent such as $in\text{-}bag(x, s)$ in the picking a ball from a bag example in Section 3.1.

With this restriction on axioms in $\mathcal{D}_{magic}$, many useful theorems about basic action theories for determinate actions can be generalized. For instance, we have:

Proposition 5.1 *For any formula $\phi(S_0)$ that is otherwise a simple state formula about S_0 except that it may contain $Magic(a, p, S)$ for any state term S, $\mathcal{D} \models \phi(S_0)$ iff*

$$\mathcal{D}_{una} \cup \mathcal{D}_{magic} \cup \mathcal{D}_{S_0} \models \phi(S_0).$$

This means that, by using regression, as done in (Reiter [13]), temporal projections of the form $G(do(\Gamma, S_0))$ can be reduced to logical entailments from the initial state database, the unique names axioms, and the constraints about $Magic$.

Proposition 5.2 *Suppose that $\mathcal{D}_{S_0}$ does not mention $Magic$. If $\varphi(s)$ is a simple state formula about s that does not mention $Holds$, i.e. can only mention $Magic$, then the following assertions are equivalent:*

1. $\mathcal{D} \models (\exists s)\varphi(s)$.
2. $\mathcal{D} \models (\forall s)\varphi(s)$.
3. $\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{magic} \models (\forall s)\varphi(s)$.
4. $\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{magic}^{S_0} \models \varphi(S_0)$, *where* $\mathcal{D}_{magic}^{S_0}$ *is the result of replacing every axiom* $(\forall s)\phi(s)$ *in it by* $\phi(S_0)$.

As is often the case, $\mathcal{D}_{magic}$ will not contain equality over action and fluent terms, so $\mathcal{D}_{una}$ can be dropped. Furthermore, $\mathcal{D}_{S_0}$ can often be divided into two sets: one mentions only situation independent predicates and functions, the other about the initial truth values of the fluents. So checking if a $Magic$ goal is achievable or not will reduce to checking logical entailments of $\mathcal{D}_{magic}^{S_0}$ together with the situation independent facts in $\mathcal{D}_{S_0}$.

An interesting consequence of Proposition 5.2 is that if $\mathcal{D}_{S_0}$ does not mention $Magic$, then an atomic $Magic$ goal $Magic(A, F, s)$ is achievable iff it is true in every situation. This means that if $Magic(A, F, s)$ is achievable, then F will not be a genuine indeterminate effect of A, for

$$Poss(A, s) \wedge \gamma \wedge Magic(A, F, s) \supset F(do(A, s))$$

will then be equivalent to

$$Poss(A, s) \wedge \gamma \supset F(do(A, s)),$$

that is, there is no need for the $Magic$ predicate here. So proper use of the $Magic$ predicate means that none of the atomic $Magic$ goals are achievable.

6 Applications in Planning

As we mentioned in the introduction, our motivation for introducing the $Magic$ predicate, and reducing indeterminate effects into determinate ones is to be able to formally specifying the effects, which are often indeterminate, of abstract operators so that planning techniques developed for determinate actions can be used for planning with abstract operators as well.

Consider again the 8 puzzle. Suppose the fluent $in(x, y)$ denotes that the number x is in the square y, the number y's goal position. Then we can specify, for instance, M_2 by the following action precondition axiom:

$$Poss(M_2, s) \equiv (\forall x)(0 \leq x < 2 \supset in(x, x, s)),$$

the following effect axioms:

$$Poss(M_2, s) \supset in(2, 2, do(M_2, s)),$$
$$Poss(M_2, s) \wedge Magic(M_2, in(x, y), s) \wedge 2 < x, y \leq 8 \supset in(x, y, do(M_2, s)),$$
$$Poss(M_2, s) \wedge \neg Magic(M_2, in(x, y), s) \wedge 2 < x, y \leq 8 \supset \neg in(x, y, do(M_2, s)),$$

and the following constraints on $Magic$:

$$2 < x \leq 8 \supset Magic(M_2, in(x,3), s) \dot{\vee} \cdots \dot{\vee} Magic(M_2, in(x,8), s),$$
$$Magic(M_2, in(x,y), s) \wedge Magic(M_2, in(x',y), s) \supset x = x'. \quad (12)$$

The last constraint is necessary because each square can hold only one number. Other abstract operators can be similarly specified.

Now suppose that the goal is $in(1,1,s) \wedge in(2,2,s)$. Achieving it using the abstract operator M_2 will reduce (regress) it to the goal $Poss(M_2, s) \wedge in(1,1,s)$, which is equivalent to $in(0,0,s) \wedge in(1,1,s)$. However, if the goal is $in(3,3,s)$, then using M_2 will be a bad idea, because the regressed goal will now be $Poss(M_2, s) \wedge Magic(M_2, in(3,3), s)$, an impossible one. Following this line of reasoning, we see that the only way of achieving the goal $(\forall x).0 \leq x \leq 8 \supset in(x,x,s)$ using the abstract operators is to perform in sequence $M_0, ..., M_7$.

Although the unachievability of $Magic(M_2, in(3,3), s)$ can be formally proved using Proposition 5.2, in practice, a good heuristic would be that whenever the planner encounters an atomic $Magic$ goal, it will assume that this goal is not achievable. This, at the very least, will not sacrifice the soundness of the planner, and is exactly right when $Magic$ is properly used as we discussed following Proposition 5.2.

7 General Discussions, Concluding Remarks, and Future Work

- All our examples so far have the property that if $Magic(a, p, s)$ holds, then a causes p to be true; otherwise, a causes p to be false. Generally, this need not be the case. For instance, exploding a bomb causes some of the nearby people to die:

  $$Poss(explode\text{-}bomb, s) \wedge nearby(x, s) \wedge Magic(explode\text{-}bomb, dead(x), s) \supset dead(x, do(explode\text{-}bomb, s)),$$

  with the following constraint on $Magic$:

  $$(\exists x) nearby(x, s) \supset (\exists x)[nearby(x, s) \wedge Magic(explode\text{-}bomb, dead(x), s)].$$

  But exploding a bomb will not cause some dead people to become alive.

- We have been writing constraints about $Magic$ directly. However, it seems that many such constraints can be generated from constraints and effect axioms about ordinary fluents. For instance, the constraint (6) in the coin tossing example seems to come from the following indeterminate effect axiom:

  $$Poss(toss(x), s) \supset head(x, do(toss(x), s)) \dot{\vee} tail(x, do(toss(x), s)),$$

  while the constraint (9) in Reiter's checkerboard example seems to come from:

  $$Poss(drop, s) \supset white(do(drop, s)) \vee black(do(drop, s)).$$

As we said, the constraint (12) for M_2 in the 8 puzzle comes from the following constraint:

$$in(x, y, s) \wedge in(x', y, s) \supset x = x'.$$

An important advantage of doing this, i.e. deriving constraints about magic from ordinary axioms, is that we'll then be able to hide *the special magic predicate from the users, who are only allowed to write effect axioms and domain constraints using the ordinary fluents.* Making this idea precise is currently our top priority.

- Ramification. One advantage of reducing indeterminate effects into determinate ones is that solutions to the ramification problem developed for deterministic actions can be inherited. For instance, using the $Caused$ predicate in (Lin [8]), we can write constraints such as

$$called\text{-}head(x, s) \wedge head(x, s) \supset Caused(happy, true, s),$$
$$head(x, s) \vee tail(x, s) \supset caused(on\text{-}floor, true, s),$$

  and use the techniques in [8] to automatically generate appropriate successor state axioms such as

$$Poss(a, s) \supset happy(do(a, s)) \equiv$$
$$(\exists x).a = toss(x) \wedge called\text{-}head(x, s) \wedge Magic(toss(x), head(x), s) \vee ...$$
$$Poss(a, s) \supset on\text{-}floor(x, do(a, s)) \equiv a = toss(x) \vee ...$$

- It is interesting to compare our approach with GOLOG's [6]. For instance, in GOLOG, the abstract operator M_6 may be specified as: $(\pi x_7, x_8)\ M_6(x_7, x_8)$, where $M_6(x_7, x_8)$ is like M_6 except that the numbers 7 and 8 will be moved to x_7 and x_8, respectively. The difficulty with this approach from a planning point of view is that the planner will now have to deal with $M_6(x_7, x_8)$, and the subtle issues of when to instantiate the variables. Of course, this is not the intended use of GOLOG. In fact, one motivation behind GOLOG is to bypass the issue of planning by providing users with high level programming constructs.
- Implementation. We have implemented a planner that can take in formal specifications of abstract operators, and realize them at run time [7]. Currently, there is no special mechanism for processing $Magic$ goals. But it should not be hard to add one.

Acknowledgements:

Thanks to Vladimir Lifschitz for sharing with me his ideas on formalizing indeterminate effects of actions, on which the special magic predicate in this paper is based. I have also benefited tremendously from discussions with Ray Reiter on related issues over a long period of time. I also thank Ray for his thought provoking comments on an earlier draft of this paper. This research was supported by grants from the National Science and Engineering Research Council of Canada, the Institute for Robotics and Intelligent Systems of Canada, and the Information Technology Research Center of Ontario.

References

[1] C. Barel. Reasoning about actions: nondeterministic effects, constraints, and qualification. In *Proc. of IJCAI–95*, pages 2017–2023, 1995.

[2] C. Boutilier and N. Friedman. Nondeterministic actions and the frame problem. In *Proc. of the AAAI Spring Symposium on Extending Theories of Action*, pages 39–44, 1995.

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

[4] G. N. Kartha and V. Lifschitz. Action with indirect effects (preliminary report). In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 341–350, 1994.

[5] R. E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[6] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. Submitted.

[7] F. Lin. A language for planning. In preparation.

[8] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of IJCAI–95*, pages 1985–1993, 1995.

[9] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 4(5):655–678, 1994.

[10] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI–91)*, pages 634–639, 1991.

[11] J. McCarthy. Mathematical logic in Artificial Intelligence. *Daedalus*, pages 297–311, 1988.

[12] R. Reiter. On specifying database updates. *Journal of Logic Programming*. To appear.

[13] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, CA, 1991.

[14] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[15] Y. Zhang. *Reasoning About Persistence: A Unified Principle for State Change*. PhD thesis, Department of Computer Science, Sydney University, Sydney, Australia, 1994.