7.4. A BACKWARD PRODUCTION SYSTEM

7.4.1. DEVELOPMENT OF THE B-RULES

In order to construct robot plans in an efficient fashion, we often want to work backward from a goal expression to an initial state description, rather than vice versa. Such a system starts with a goal description (again a conjunction of literals) as its global database and applies B-rules to this database to produce subgoal descriptions. It successfully terminates when it produces a subgoal description that is matched by the facts in the initial state description.

Our first step in designing a backward production system is to specify a set of B-rules that transform goal expressions into subgoal expressions. One strategy is to use B-rules that are based on the F-rules that we have just discussed. A B-rule that transforms a goal G into a subgoal G' is logically based on the corresponding F-rule that when applied to a state description matching G' produces a state description matching G.

We know that the application of an F-rule to any state description produces a state description that matches the add list literals. Therefore, if a goal expression contains a literal, L, that unifies with one of the literals in the add list of an F-rule, then we know that if we produce a state description that matches appropriate instances of the preconditions of that F-rule, the F-rule can be applied to produce a state description matching L. Thus, the subgoal expression produced by a backward application of an F-rule must certainly contain instances of the preconditions of that F-rule. But if the goal expression contains other literals (besides L), then the subgoal expression must also contain other literals, which after application of the F-rule, become those other literals (i.e., other than L) in the goal expression.

7.4.2. REGRESSION

To formalize what we have just stated, suppose that we have a goal given by a conjunction of literals $[L \wedge G1 \wedge \ldots \wedge GN]$ and that we want to use some F-rule (backward) to produce a subgoal expression. Suppose an F-rule with precondition formula, P, and add formula, A, contains a literal L' in A that unifies with L, with most general unifier u. Application of u to the components of the F-rule creates an instance of the F-rule. Certainly the literals in Pu are a subset of the literals of the subgoal that we seek. We must also include the expressions $G1', \ldots, GN'$ in the complete subgoal. The expressions $G1', \ldots, GN'$ must be such that the application of the instance of the F-rule to any state description matching these expressions produces a state description matching $G1, \ldots, GN$. Each Gi' is called the *regression* of Gi through the instance of the F-rule. The process of obtaining Gi' from Gi is called *regression*.

For F-rules specified in the simple STRIPS-form, the regression procedure is quite easily described for ground instances of rules. (A *ground instance* of an F-rule is an instance in which all of the literals in the precondition formula, the delete list, and the add formula are ground

literals.) Let $R[Q;Fu]$ be the regression of a literal Q through a ground instance Fu of an F-rule with precondition, P, delete list, D, and add list, A. Then,

if Qu is a literal in Au,

$$R[Q;Fu] = T \text{ (True)}$$

else, if Qu is a literal in Du,

$$R[Q;Fu] = F \text{ (False)}$$

else, $R[Q;Fu] = Qu$

In simpler terms, Q regressed through an F-rule is trivially T if Q is one of the add literals, it is trivially F if Q is one of the deleted literals; otherwise, it is Q itself.

Regressing expressions through incompletely instantiated F-rules is slightly more complicated. We describe how we deal with incompletely instantiated F-rules by some examples. Suppose the F-rule is **unstack**, given earlier and repeated here:

unstack(x,y)
P & D: $HANDEMPTY$, $CLEAR(x)$, $ON(x,y)$
A: $HOLDING(x)$, $CLEAR(y)$

In particular, suppose we are considering the instance **unstack**(B,y), perhaps because our goal is to produce $HOLDING(B)$. This instance is not fully instantiated. If we were to regress $HOLDING(B)$ through this F-rule instance, we would obtain T, as expected. (The literal $HOLDING(B)$ is unconditionally true in the state resulting after applying the F-rule.) If we were to regress $HANDEMPTY$ through this F-rule instance, we would obtain F. (The literal $HANDEMPTY$ can never be true *immediately* after applying **unstack**.) If we were to regress $ONTABLE(C)$, we would obtain $ONTABLE(C)$. (The literal $ONTABLE(C)$ is unaffected by the F-rule.)

Suppose we attempt to regress $CLEAR(C)$ through this incompletely instantiated instance of the F-rule. Note that if y were equal to C, $CLEAR(C)$ would regress to T; otherwise, it would simply regress to

$CLEAR(C)$. We could summarize this result by saying that $CLEAR(C)$ regresses to the disjunction $(y = C) \vee CLEAR(C)$. (In order for $CLEAR(C)$ to hold after applying any instance of **unstack**(B,y), either y must be equal to C or $CLEAR(C)$ had to have held before applying the F-rule.) Unfortunately, to accept a disjunctive subgoal expression would violate our restrictions on the allowed forms of goal expressions. Instead, when such a case arises, we produce two alternative subgoal expressions. In the present example, one subgoal expression would contain the precondition of **unstack**(B,C), and the other would contain the uninstantiated precondition of **unstack**(B,y) conjoined with the literal $\sim(y = C)$.

A related complication occurs when we regress an expression matching an incompletely instantiated literal in the delete list. Suppose, for example that we want to regress $CLEAR(C)$ through **unstack**(x,B). If x were equal to C, then $CLEAR(C)$ would regress to F; otherwise, it would regress to $CLEAR(C)$. We could summarize this result by saying that $CLEAR(C)$ regressed to

$$[(x = C) \Rightarrow F] \wedge [\sim(x = C) \Rightarrow CLEAR(C)] .$$

As a goal, this expression is equivalent to the conjunction $[\sim(x = C) \wedge CLEAR(C)]$.

The reader might ask what would happen if we were to regress $CLEAR(B)$ through **unstack**(B,y). In our example, we would obtain T for the case $y = B$. But $y = B$ corresponds to the instance **unstack**(B,B), which really ought to be impossible because its precondition involves $ON(B,B)$. Our simple example would be made more realistic by adding the precondition $\sim(x = y)$ to **unstack**(x,y).

In summary, a STRIPS-form F-rule can be used as a B-rule in the following manner. The applicability condition of the B-rule is that the goal expression contain a literal that unifies with one of the literals in the add list of the F-rule. The subgoal expression is created by regressing the other (the nonmatched) literals in the goal expression through the match instance of the F-rule and conjoining these and the match instance of the precondition formula of the F-rule.

Let's consider a few more examples to illustrate the regression process. Suppose our goal expression is $[ON(A,B) \wedge ON(B,C)]$. Referring to the F-rules given earlier, there are two ways in which **stack**(x,y) can be

used on this expression as a B-rule. The mgu's for these two cases are $\{A/x, B/y\}$ and $\{B/x, C/y\}$. Let's consider the first of these. The subgoal description is constructed as follows:

(1) Regress the (unmatched) expression $ON(B,C)$ through **stack**(A,B) yielding $ON(B,C)$.

(2) Add the expressions $HOLDING(A)$, $CLEAR(B)$ to yield, finally, the subgoal $[ON(B,C) \wedge HOLDING(A) \wedge CLEAR(B)]$.

Another example illustrates how subgoals having existentially quantified variables are created. Suppose our goal expression is $CLEAR(A)$. Two F-rules have $CLEAR$ on their add list. Let's consider **unstack**(x,y). As a B-rule, the mgu is $\{A/y\}$, and the subgoal expression created is $[HANDEMPTY \wedge CLEAR(x) \wedge ON(x,A)]$. In this expression, the variable x is interpreted as existentially quantified. That is, if we can produce a state in which there is a block that is on A and whose top is clear, we can apply the F-rule, **unstack**, to this state to achieve a state that matches the goal expression, $CLEAR(A)$.

A final example illustrates how we might generate "impossible" subgoal descriptions. Suppose we attempt to apply the B-rule version of **unstack** to the goal expression $[CLEAR(A) \wedge HANDEMPTY]$. The mgu is $\{A/y\}$. The regression of $HANDEMPTY$ through **unstack**(x,A) is F. Since no conjunction containing F can be achieved, we see that the application of this B-rule has created an impossible subgoal. [That is, there is no state from which the application of an instance of **unstack**(x,A) produces a state matching $CLEAR(A) \wedge HANDEMPTY$.]

Impossible goal states might be detected in other ways also. In general, we could use some sort of theorem prover to attempt to deduce a contradiction. If a goal expression is contradictory, it cannot be achieved. Checking for the consistency of goals is important in order to avoid wasting effort attempting to achieve those that are impossible.

Sometimes the mgu of a match between a literal on the add list of an F-rule and a goal literal does not further instantiate the F-rule. Suppose, for example, that we want to use the STRIPS rule **unstack**(u,C) as a B-rule applied to the goal $[CLEAR(x) \wedge ONTABLE(x)]$. The mgu is $\{C/x\}$. Now, even though this substitution does not further instantiate

unstack(u,C), the substitution is used in the regression process. When $ONTABLE(x)$ is regressed through this instance of **unstack**(u,C), we obtain $ONTABLE(C)$.

7.4.3. AN EXAMPLE SOLUTION

Let us show how a backward production system, using the STRIPS rules given earlier, might achieve the goal:

$$[ON(A,B) \wedge ON(B,C)].$$

In this particular example, the subgoal space generated by applying all applicable B-rules is larger than the state space that we produced using F-rules. Many of the subgoal descriptions, however, are "impossible," that is, either they contain F explicitly or rather straightforward theorem proving would reveal their impossibility. Pruning impossible subgoals greatly reduces the subgoal space.

In Figure 7.5 we show the results of applying some B-rules to our example goal. (The tail of each B-rule arc is adjacent to that goal literal used to match a literal in the add list of the rule.) Note in Figure 7.5 that when **unstack** was matched against $CLEAR(B)$, it was not fully instantiated. As we discussed earlier, if a possible instantiation allows a literal in the add list of the rule to match a literal in the goal expression, we make this instantiation explicit by creating a separate subgoal node using it.

All but one of the tip nodes in this figure can be pruned. The tip nodes marked "*" all represent impossible goals. That is, no state description can possibly match these goals. In one of them, for example, we must achieve the conjunct $[HOLDING(B) \wedge ON(A,B)]$, an obvious impossibility. We assume that our backward reasoning system has some sort of mechanism for detecting such unachievable goals.

The tip node marked "**" can be viewed as a further specification of the original goal (that is, it contains all of the literals in the original goal plus some additional ones.) Heuristically, we might prune (or at least delay expansion of) this subgoal node, because it is probably harder to achieve than the original goal. Also, this subgoal is one of those produced by matching $CLEAR(B)$ against the add list of a rule. Since $CLEAR(B)$ is already true in the initial state, there are heuristic grounds against

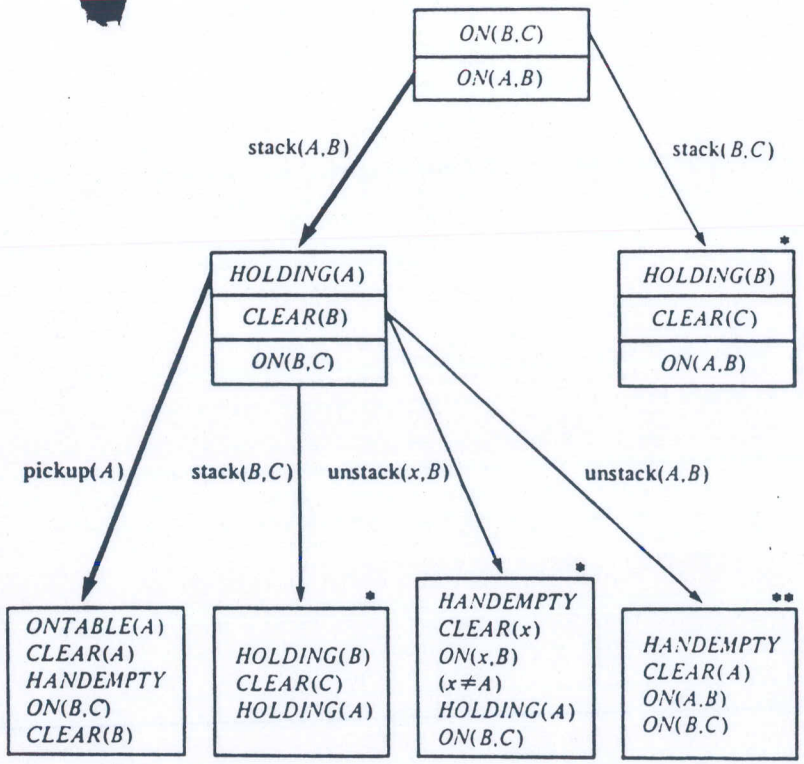

Fig. 7.5 Part of the backward (goal) search graph for a robot problem.

attempting to achieve it when it occurs in subgoal descriptions. (Sometimes, of course, goal literals that already match literals in the initial state might get deleted by early F-rules in the plan and need to be reachieved by later F-rules. Thus, this heuristic is not always reliable.)

The pruning operations leave just one subgoal node. The immediate successors of this subgoal are shown numbered in Figure 7.6. In this figure, nodes 1 and 6 contain conditions on the value of the variable x. (Conditions like these are inserted by the regression process when the delete list of the rule contains literals that might match regressed literals.) Both nodes 1 and 6 can be pruned in any case, because they contain the literal F, which makes them impossible to achieve. Note also that node 2 is impossible to achieve because of the conjunction $HOLDING(B) \wedge ON(B,C)$. Node 4 is identical to one of its ancestors (in Figure 7.5), so it can be pruned also. (If a subgoal description is merely implied by one of its ancestors instead of being identical to one of them,

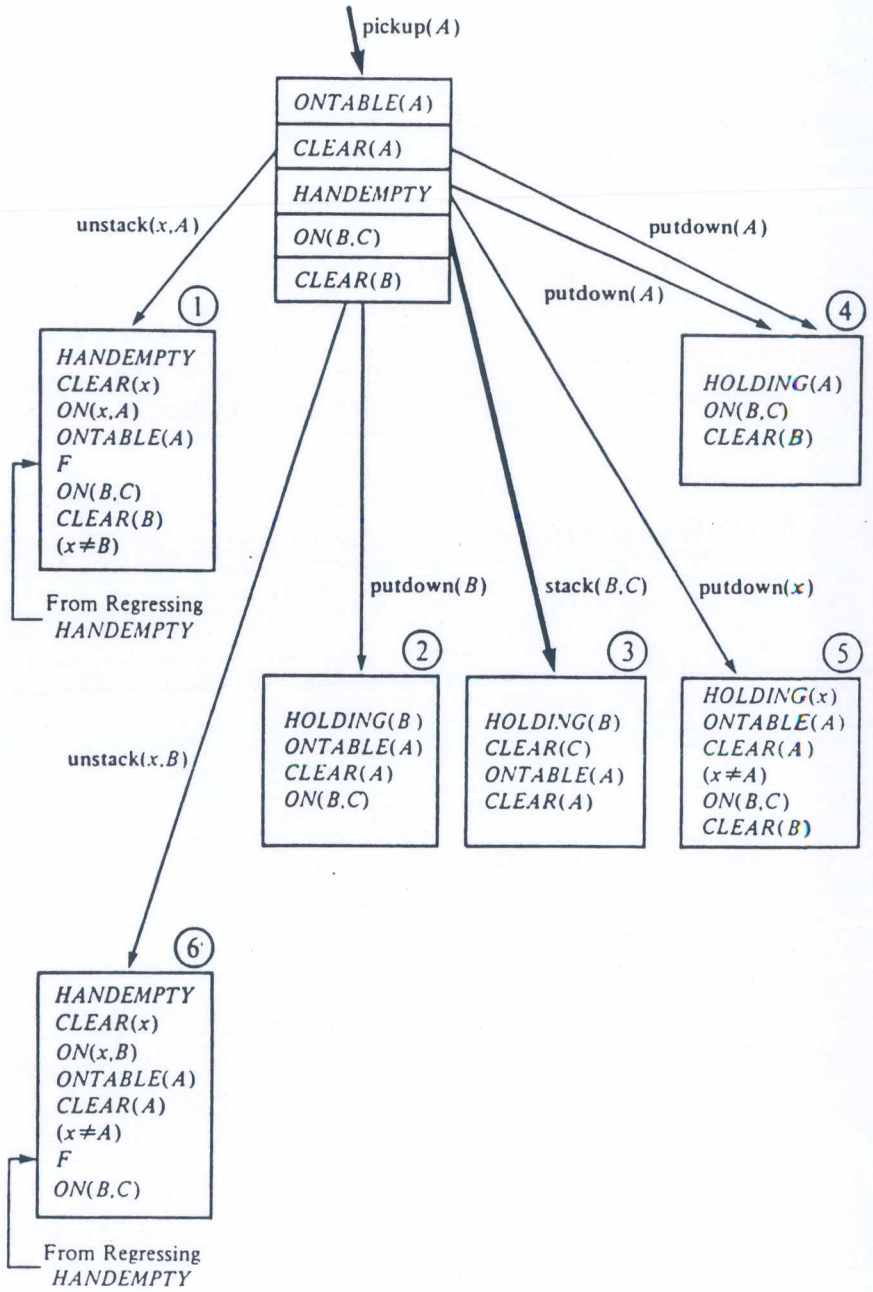

Fig. 7.6 Continuation of the backward search graph.

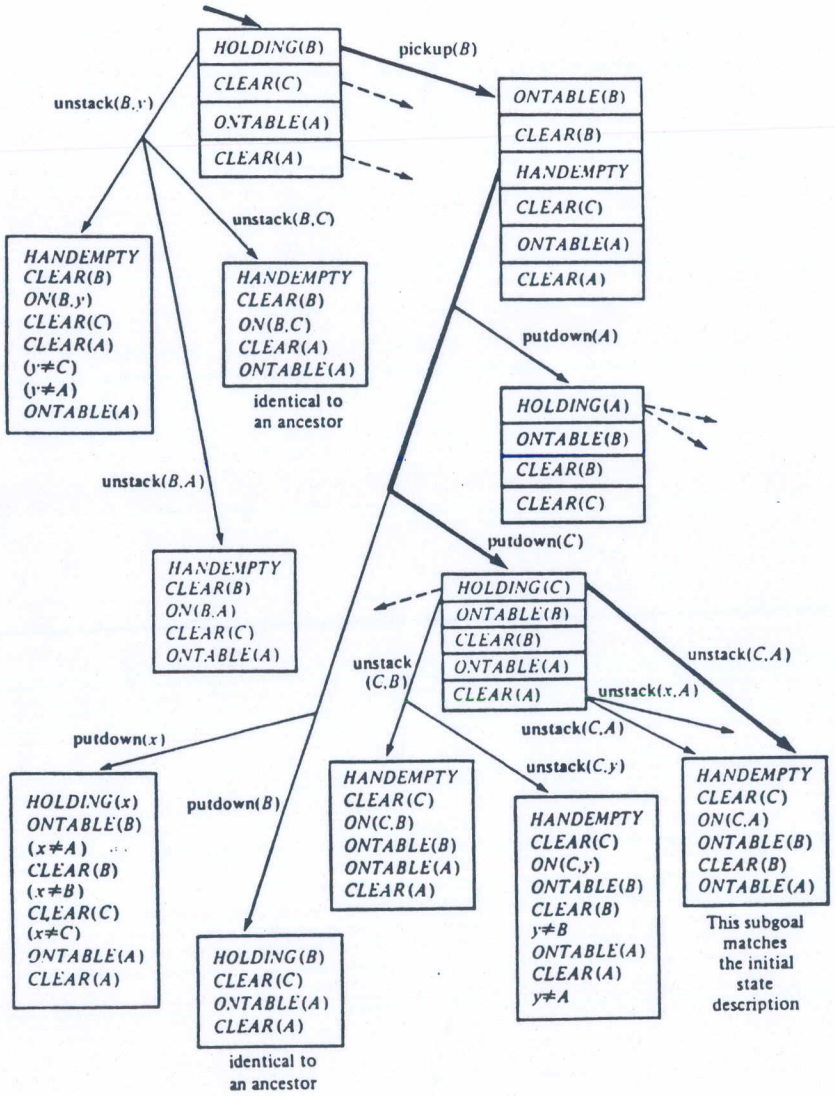

Fig. 7.7 Conclusion of the backward search graph.

we cannot, in general, prune it. Some of the successors generated by the ancestor might have been impossible because literals in the ancestor, but not in the subgoal node, might have regressed to *F*.)

These pruning operations leave us only nodes 5 and 3. Let's examine node 5 for a moment. Here we have an existential variable in the goal description. Since the only possible instances that can be substituted for x (namely, B and C in this case) lead to impossible goals, we are justified in pruning node 5 also.

In Figure 7.7 we show part of the goal space below node 3, the sole surviving tip node from Figure 7.6. This part of the space is a bit more branched than before, but we soon find a solution. (That is, we produce a subgoal description that matches the initial state description.) If we follow the B-rule arcs back to the top goal (along the darkened branches), we see that the following sequence of F-rules solves our problem: {**unstack**(C,A), **putdown**(C), **pickup**(B), **stack**(B,C), **pickup**(A), **stack**(A,B)}.