Goal Interactions and Plan Quality*

M. Alicia Pérez
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
aperez@cs.cmu.edu

Manuela M. Veloso
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
veloso@cs.cmu.edu

Abstract

Goals rarely occur in isolation. Efficient planning for conjunctive goals is therefore of central interest. We identify a particular class of conjunctive goal interactions, which cannot be directly inferred from the domain knowledge, as they are not explicitly represented in the preconditions and effects of the domain operators. We call such goal interactions, *quality goal interactions*, for they relate to issues on plan quality rather than to the common plan achievement issues. We illustrate the discussion with examples and briefly present the method we are developing for acquiring control knowledge to direct the planner to consider the quality-goal interactions.

1 What Are Goal Interactions

Planning goals rarely occur in isolation. A planner must be capable of taking into account the interactions between conjunctive goals in order to produce a plan to solve the problem. There have been many research efforts addressing the issue of planning for conjunctive goals, focusing on a variety of aspects, including analyzing the complexity of this planning problem [Sussman, 1975, Chapman, 1987], designing appropriate planning algorithms [Sacerdoti, 1977, Tate, 1977, Drummond and Currie, 1987], categorizing different types of goal interactions [Wilensky, 1983], and learning control knowledge to efficiently handle the search for the interactions [Minton, 1988, Etzioni, 1990]. Our work, though built upon this previous work, goes beyond it as we aim at identifying goal interactions directly related to the quality of the plans produced.

From a practical implementation point of view we distinguish two categories of goal interactions, *explicit goal interactions* and *quality goal interactions*.

1.1 Explicit goal interactions

We include in this category the goal interactions that are explicitly represented as part of the domain knowledge in terms of preconditions and effects of the operators. A plan exhibits a goal interaction of this type when there is a goal in the plan that has been negated by a previous step in the plan [Minton, 1988, Etzioni, 1990]. These goal interactions enforce particular goal orderings in order that the planner may be able to produce a solution to the problem.

In a typical example of a two-goal interaction, after one of the goals has been achieved, it is deleted by an operator that works towards achieving the other goal. [1]

Goal interactions in this category is the well-known Sussman's anomaly in the blocksworld [Sussman, 1975]. Consider also another illustrative example in a transportation domain. In this domain, packages are to be moved among different cities. Packages are carried within the same city in trucks and across cities in airplanes. Trucks and airplanes may have limited capacity. At each city there are several locations, e.g. post offices (po) and airports (ap). A package `P1` is at the Pittsburgh airport. There is only one airplane, `A1`, available also at the Pittsburgh airport. The goal consists of having both the airplane and the package at the Boston airport, and is represented by the conjunction `(and (at-airplane A1 bos-airport) (at-object P1 bos-airport)`. If the goal `(at-airplane A1 bos-airport)` is addressed first, and A1 flies from Pittsburgh to Boston, there is no way to achieve the second goal without first flying back A1 to Pitts-

*This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The first author is supported by a scholarship from the Ministerio de Educación y Ciencia of Spain. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government or the Spanish Government.

[1] Other goal interactions in this category may be beneficial to the planning process, when solving one goal makes a second goal easier to achieve. This is generally termed goal concord and opportunistic planning takes advantage of these situations [Converse and Hammond, 1992].

burgh. The resulting plan involves flying A1 back and forth unnecessarily. It is conceivable to design an algorithm that fixes this kind of plans by removing unnecessary operations that reachieve the clobbered goals [Rich and Knight, 1991].

In some problems, these interactions are unavoidable and the planning system must find a solution that minimizes their effects. When this happens, search time is typically reduced and better solutions tend to be found. These solutions are generally shorter in length, and more direct [Minton, 1988, Ryu and Irani, 1992, Veloso, 1992].

In least-commitment planners the critics take care of these goal interactions by establishing ordering constraints among the conflicting goals. In the case of PRODIGY, a casual-commitment planner, goal preference control knowledge is automatically acquired to deal effectively (in the sense of problem solving effort) with this kind of goal interactions by different machine learning approaches, namely explanation-based learning [Minton, 1988], static analysis [Etzioni, 1990], or derivational analogy [Veloso, 1992].

A particular problem may have many different solutions. These solutions may differ in the set of operators in the plan. If the ordering constraints between achieving two goals are explicit in the domain representation, then all the solutions to a particular problem will have the two goals interacting. On the other hand the dependencies may be the result of a particular problem solving path explored. In this case for some solutions the goals may interact and for some others they may not.

1.2 Quality goal interactions

To illustrate this difference and to motivate the quality goal interactions, we further discuss different plans with ordering constraints that are or are not explicit in the domain. In the *one-way rocket* domain [Veloso, 1989], the goals of moving two objects to a different location interact, because the rocket can only move once. This is an interaction that is represented in the domain definition as the moving operator explicitly deletes the location of the rocket. The *machine-shop scheduling* domain [Minton *et al.*, 1989] also constraints that holes in parts must be drilled before parts are polished, as the drilling operator deletes the shining effect. In this domain, the goals of polishing and making a hole in a part interact again due to the domain definition.

However, in this same machine-shop scheduling domain, when two identical machines are available to achieve two identical goals, these goals may interact, if the problem solver chooses to use just one machine to achieve both goals, as it will have to wait for the machine to be idle. If the problem solver uses the two machines instead of just one, then the goals do not interact in this particular solution.

There is a variety of equivalent examples in the logistics transportation domain. In general it is not clear what use of resources is overall the best. For example, in the logistics transportation domain, suppose that the problem solver assumes that the same truck (or airplane) must be used when moving objects from the same location into the same (or close) destiny. In this case the goals of moving the objects interact. But if different carriers are chosen, there is not such interaction. Note that the problem can become quite complicated as the domain considers other types of constraints, such as capacity for the carriers, size of the objects to be transported, distances between locations that dictate the type of carrier to use, and so on.

These interactions are related to plan quality as the use of resources dictates the interaction between the goals. The control knowledge that guides the planner to solve these interactions is harder to learn automatically, as the domain theory does not encode these quality criteria. [Pérez, 1992] is a current research effort on learning control knowledge to improve the quality of the plans generated by the problem solver.

There are a few other planners that analyze the relationship between goal interactions and plan quality. Wilensky's planner [Wilensky, 1983] takes advantage of this relationship. He makes an analysis of the different types of goal interactions and develops meta-planning mechanisms that deal with them. When a goal overlap, or positive goal interaction between a planner's goals, occurs, his planner is able to carry out an action that is in the service of a number of goals at once. This might involve executing a single plan that simultaneously fulfills several goals, achieving a goal that serves more than one purpose, or employing a plan that is worthwhile only when a sufficient number of similar goals is involved. In Wilensky's system then a goal overlap situation provides an opportunity to achieve goals more economically than they could be achieved otherwise, and the planner prefers efficient plans over inefficient ones. This principle would appear to be the underlying justification for a number of processes incorporated in other planning systems. For example, several of NOAH's critics [Sacerdoti, 1977] including "use existing objects," "eliminate redundant preconditions," and "optimize disjuncts" are motivated by this idea and correspond to particular kinds of goal overlap situations. SteppingStone [Ruby and Kibler, 1990] heuristically decomposes a problem into simpler subproblems, and then learns to deal with the interactions that arise between the subproblems. The system allows hard-constraints, that must be met and usually outline key aspects of the problem, and soft constraints that measure the quality of a solution, and are usually real-valued. The system learns to optimize soft constraints as well as solve hard constraints.

Some existing domain-independent planning systems solve multiple-goal problems by developing separate plans for the individual goals, combining these plans to form a naive plan for the conjoined goal, and then performing optimizations to yield a better combined plan [Nau *et al.*, 1990]. However they restrict the types of goal interactions that may happen. In this context, the quality of a plan is only considered as far as dealing with and taking as much advantage as possible of goal interactions. A similar mechanism is also used by some domain-dependent planners [Hayes, 1990, Nau, 1987].

In our approach the control knowledge that detects quality goal interactions and deals with them is created by a semi-automated knowledge acquisition system. It is extracted from a domain expert incrementally as the planner sees new interesting problems in the domain. This knowledge refers to concrete quality metrics, and is about why a solution is better than other, and how search can be guided to improve solution quality. We do not claim that the knowledge acquired will necessarily guide the planner to find optimal solutions, but that the quality of the plans will incrementally improve with experience by interaction with the domain expert.

2 Example of quality goal interactions

In this section we present a complete planning example where there are goal interactions not explicitly represented in the domain knowledge. In the process planning phase of production manufacturing, plan quality is crucial in order to minimize both resource consumption and execution time. The goal of process planning is to produce plans for machining parts given their specifications. Such planning requires taking into account both technological and economical considerations, for instance "it may be advantageous to execute several cuts on the same machine with the same fixing to reduce the time spent setting up the work on the machines", or "if a hole H_1 opens into another hole H_2, then one is recommended machining H_2 before H_1 in order to avoid the risk of damaging the drill" [Descotte and Latombe, 1985]. Most of these considerations are not pure constraints but only preferences when compromises are necessary. They often represent both the experience and the know-how of engineers, so they may differ from one company to the other.

Let us look at a concrete example on the relationship between goal interactions and the quality of plans in this domain. The domain concentrates on the machining, joining, and finishing steps of production manufacturing [Gil, 1991]. The goal is to produce one or more parts according to certain specifications. In order to perform an operation on a part, the part has to be secured to the machine table with a holding device, and in many cases the part has to be clean and without burrs from preceding operations. The appropriate tool has to be selected and installed in the machine as well.

As an example, Figure 1 shows a machine set up to drill a hole in a part. Figure 2 sketches graphically the steps to produce a reamed hole. Before performing each of these steps, the appropriate tool has to be set in the machine spindle, namely a spot-drill, a high-helix-drill, and a reamer. Then some holding device (a vise in the example) has to be put on the machine, and the part has to be held by the holding device.

Suppose the planner has to build a plan to have a part with *two* reamed holes on one of its sides. If the planner works on making each hole separately, it will obtain a solution, sketched in Figure 3(a) (the operators to hold the part are

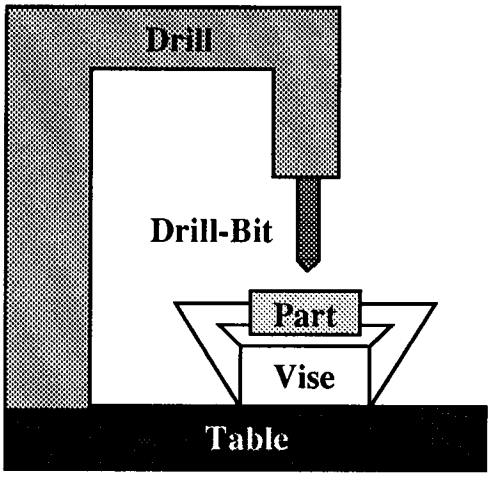

Figure 1: An Example of Set Up in the Machining Domain (from [Joseph, 1992]). In this example the holding device is a vise, the machine a drilling machine, and the tool a drill-bit.

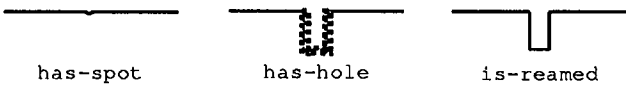

Figure 2: The Steps to Make a Reamed Hole: first a spot hole is drilled on the part, then the hole itself is made, and finally the hole is reamed. For each of these the operations the appropriate tool (respectively a spot drill, some appropriate drill bit, or a reamer) has to be installed in the drilling machine.

omitted). This solution is not the shortest one (and in this domain a shorter solution may mean a faster and cheaper way to produce a large number of parts). Some steps may be eliminated by reordering the operations. Both holes, and spot holes for that matter, have to be in the same side and may be made with the same tools. Therefore once we have set the appropriate tool in the drill spindle and held the part on the machine table, the operations corresponding to both holes can be performed consecutively. Figure 3(b) shows a better solution to the problem. The same tool set up is used for drilling the two spot holes, then it is changed to drill both holes, and finally the reamer is set once to ream both holes.

In this example the planner obtains the better solution by interleaving the problem goals. In PRODIGY this decision may be encoded in the form of search control rules of Figure 4.

Note that if the goal interactions are not considered the planner still constructs a valid solution. It is only because of quality considerations that the interactions occur as the same set-up is used for achieving the goals for both holes. These interactions are not explicitly represented in the domain specification.

Some of the interactions between goals are due to the use of a state-space planner, as the operator ordering in the final

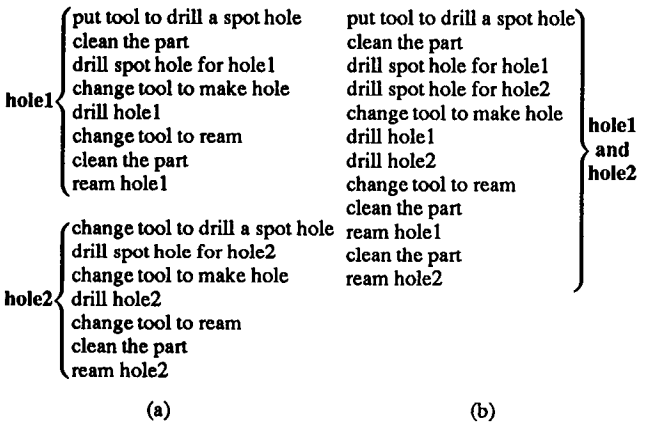

Figure 3: Two Plans of Different Quality to Make Two Reamed Holes on a Part. Some steps in solution (a) may be eliminated by reordering the operations, since once the corresponding tool is set, the operation may be performed for both holes consecutively. Solution (b) captures such improvement by interleaving the operations on hole1 and hole2.

plan is tied to the goal ordering during problem solving. By using a plan-space planner, in which actions can be inserted anywhere in the plan, some of these problems may go away. However there is still the issue of which is the appropriate control knowledge, heuristics or critics, to select the best place to insert actions into the plan.

3 Acquiring Control Knowledge to Cover Quality Goal Interactions

Most of the work to date on automated control-knowledge acquisition for planning systems has been aimed to improve the *efficiency of planning*; this work has been termed speed-up learning. These systems have been successful in dealing effectively (in the sense of problem solving effort) with the first type of goal interactions by analyzing the domain knowledge. However they have not taken into account considerations of plan quality. [Pérez, 1992] proposes to focus on the acquisition of control knowledge to guide the planner towards better solutions, i.e. to improve the *quality of plans* produced by the planner. We discuss here briefly this approach.

Human experts gather knowledge for producing better plans through experience. Here "better" is defined in a context-sensitive manner as a combination of plan-quality factors such as those listed extensively in [Pérez, 1992]. It is precisely this experiential knowledge that we seek to capture from a combination of system planning experience and interaction with the expert.

We are considering situations where the expert, or a domain-dependent evaluation function, provides feedback on the quality of the plans from one or several examples of plans produced by the planner. This feedback may be localized or

```
(control-rule MAKE-ANOTHER-SPOT
;;if the tool and part are in place, make a second spot hole
  (if
   (and
    (candidate-goal (has-spot <part> <hole2> <side> <x2> <y2>))
    (known (holding <machine> <holding-device> <part> <side>))
    (known (holding-tool <machine> <tool>))
    (type-of-object <tool> SPOT-DRILL)))
  (then prefer goal
        (has-spot <part> <hole2> <side> <x2> <y2>) <other>))

(control-rule MAKE-ANOTHER-HOLE
;;if the tool and part are in place, make a second hole
  (if
   (and
    (candidate-goal
       (has-hole <part> <hole> <side> <depth> <diam> <x> <y>))
    (known (holding <machine> <holding-device> <part> <side>))
    (known (holding-tool <machine> <tool>))
    (type-of-object <tool> DRILL-BIT)
    (' (type-of-object <tool> SPOT-DRILL))))
  (then prefer goal
        (has-hole <part> <hole> <side> <depth> <diam> <x> <y>)
        <other>))
```

Figure 4: Search control rules to deal with quality-related goal interactions.

global. One situation may correspond to the learning of local search control rules and the other may require the storage of the episodic case corresponding to the good plan. The planner would then be able to either use a localized search control rule when the same situation occurs, or replay a past good plan when a globally similar situation is perceived.

Let us look back to the example of Section 2. The solution in Figure 3(a) was obtained by PRODIGY without any knowledge about goal ordering as by default it achieves the goals in the order in which they were given. As we discussed this is not the best solution. We have implemented a prototype to acquire control knowledge to improve solution quality. It relies on the expert pointing that the planner's solution is not good enough, and providing modifications to that solution, or a complete solution. In this example some steps may be eliminated by reordering the operations. Then the system is able to determine the points during search in which different decisions should have been made in order to obtain the better solution, and prompts the acquisition of the relevant control knowledge.

4 Conclusion

We discussed planning goal interactions that are related to the quality of the plans produced by the planner. We differentiate these goal interactions from the ones explicitly represented in the domain specification. We illustrated the difference between these two categories of goal interactions with several examples.

We finished the discussion by briefly outlining a method to interact with an expert on the quality of the plans produced and to automatically translate the expert's feedback into knowledge that the planner can use in subsequent planning situations. The control knowledge acquired, either in the form of localized search control rules or episodic global cases for replay, enables the planner to handle the qual-

ity goal interactions and ultimately produce plans of better quality.

References

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–378, 1987.

[Converse and Hammond, 1992] Timothy M. Converse and Kristian J. Hammond. Learning to satisfy conjunctive goals. In D. Sleeman and P. Edwards, editors, *Machine Learning: Proceedings of the Ninth International Conference (ML92)*, pages 117–122. Morgan Kaufmann, San Mateo, CA., 1992.

[Descotte and Latombe, 1985] Yannick Descotte and Jean-Claude Latombe. Making compromises among antagonist constraints in a planner. *Artificial Intelligence*, 27:183–217, 1985.

[Drummond and Currie, 1987] Mark Drummond and Ken Currie. Goal ordering in partially ordered plans. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 960–965, Detroit, MI, 1987.

[Etzioni, 1990] O. Etzioni. *A Structural Theory of Explanation-Based Learning*. PhD thesis, Carnegie Mellon University, School of Computer Science, 1990. Also appeared as Technical Report CMU-CS-90-185.

[Gil, 1991] Yolanda Gil. A specification of process planning for PRODIGY. Technical Report CMU-CS-91-179, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 1991.

[Hayes, 1990] Caroline Hayes. *Machining Planning: A Model; of an Expert Level Planning Process*. PhD thesis, The Robotics Institute, Carnegie Mellon University, December 1990.

[Joseph, 1992] Robert L. Joseph. *Graphical Knowledge Acquisition for Visually-Oriented Planning Domains*. PhD thesis, Carnegie Mellon University, School of Computer Science, August 1992.

[Minton *et al.*, 1989] Steven Minton, Craig A. Knoblock, Daniel R. Kuokka, Yolanda Gil, Robert L. Joseph, and Jaime G. Carbonell. PRODIGY2.0: The manual and tutorial. Technical Report CMU-CS-89-146, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1989.

[Minton, 1988] S. Minton. *Learning Effective Search Control Knowledge: An Explanation-based Approach*. PhD thesis, Carnegie Mellon University, School of Computer Science, 1988. Also appeared as Technical Report CMU-CS-88-133.

[Nau *et al.*, 1990] Dana S. Nau, Qiang Yang, and James Hendler. Optimization of multiple-goal plans with limited interaction. In *Proceedings of the Darpa Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 160–165, San Diego, CA, November 1990.

[Nau, 1987] Dana S. Nau. Automated process planning using hierarchical abstraction. *Texas Instruments Technical Journal*, Winter:39–46, 1987.

[Pérez, 1992] M. Alicia Pérez. Semi-automated acquisition of control knowledge to improve the quality of plans. Thesis proposal, School of Computer Science, Carnegie Mellon University, 1992.

[Rich and Knight, 1991] E. Rich and K. Knight. *Artificial Intelligence*. Mc Graw Hill, New York, second edition, 1991.

[Ruby and Kibler, 1990] David Ruby and Dennis Kibler. Learning steppinstones for problem solving. In *Proceedings of the Darpa Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 366–373, San Diego, CA, November 1990.

[Ryu and Irani, 1992] Kwang R. Ryu and Keki B. Irani. Learning from goal interactions in planning: Goal stack analysis and generalization. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 401–407, San Jose, CA, July 1992. AAAI Press/The MIT Press.

[Sacerdoti, 1977] Earl Sacerdoti. *A Structure for Plans and Behavior*. Elsevier, North Holland, New York, 1977.

[Sussman, 1975] Gerald J. Sussman. *A Computer Model of Skill Acquisition*. American Elsevier, New York, 1975. Also available as technical report AI-TR-297, Artificial Intelligence Laboratory, MIT, 1975.

[Tate, 1977] Austin Tate. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 888–900, Cambridge, MA, 1977.

[Veloso, 1989] M. Veloso. Nonlinear problem solving using intelligent casual-commitment. Technical Report CMU-CS-89-210, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1989.

[Veloso, 1992] Manuela M. Veloso. *Learning by Analogical Reasoning in General Problem Solving*. PhD thesis, Carnegie Mellon University, School of Computer Science, August 1992. Available as technical report CMU-CS-92-174.

[Wilensky, 1983] Robert Wilensky. *Planning and Understanding*. Addison-Wesley, Reading, MA, 1983.