

Combining Backward-Chaining with Forward-Chaining AI Search

Eric Parker

eric@semsyn.com

SemSyn is an automatic plan synthesis algorithm that endeavors to fulfill the requirements of flexible, industrial-strength, next-generation AI planning. Historically, AI planning systems have not been viewed as practical because users have had to be skilled artificial intelligence practitioners. This is in part due to the fact that systems built to solve large-scale, real-world problems traditionally rely on optimisation and/or heuristic procedures. Further difficulties with such systems are that optimisation procedures are usually tailored around specific types of problems, and that heuristic procedures are not guaranteed to find a solution. The former approach produces planning systems that are inflexible. The latter approach produces planning systems that are unsuitable for industrial settings that require critical systems.

SemSyn, on the other hand, performs an exhaustive search, thereby retaining both completeness and flexibility. The algorithm combines well-known forward-chaining search (FCS) and backward-chaining search (BCS) strategies from the AI literature (e.g. [5]). That is, the children-generation function of FCS consists of producing the domain actions that are applicable in the current *state*, while the children-generation function of BCS consists of producing the domain actions that are applicable to the current set of *subgoals*. Intuitively, combining the approaches seems to be the right move, since a desirable outcome is that some *subgoals* (namely, the top-level goals) are satisfied in some *state*. In any case, FCS and BCS separately share the common fate of combinatorial explosion, and SemSyn hopes to play the strengths of one against the weaknesses of the other (in the spirit of [4]). This is done by using a generalised BCS to compute the causal link information, and by using the FCS states to impose a total order on (some subset of) the causal links. The causal links computation must be efficient enough so as not to outweigh the benefit of their use.

Classical Backward-Chaining Search

SemSyn implements BCS in a breadth-first manner and employs a sideways-information-passing technique that provides an upper bound on the number of actions at each level of the search. The SemSyn approach can be better understood in relation to the classical BCS approach. The root of the classical BCS search tree consists of the top-

level problem goals. The root's children are those domain actions that both achieve some top-level goal and don't delete any of the top-level goals. Domain actions that meet these requirements are said to be applicable to the top-level goals, or more generally, they are said to be applicable to the current set of subgoals. The current set of subgoals for each child is computed from its parent's subgoals by regressing the parent subgoals through the child [7]. The children-generation function is then re-applied to each of these nodes to produce the root's grandchildren, and so on. When BCS is implemented in a breadth-first manner it builds action sequences of increasing length, which provides the opportunity to apply sideways-information-passing techniques [1].

Generalised Backward-Chaining Search

SemSyn's backward-chaining search (SBCS) differs from BCS in two important ways: 1. Instead of having a single root, the root level of the SBCS search tree (in fact, a graph) has one node for each top-level goal. The current set of subgoals for each of these "root" nodes consists only of the node's top-level goal. Put differently, SBCS builds partial plans, whereas BCS builds plans. Since partial plans are, hopefully, shorter than plans, the total amount of work is sometimes reduced. 2. SBCS tries to pass information between partial plans of equal length. The strategy relies on the fact that the same domain action can be applied to more than one set of subgoals at each level of the graph. For every level L of the graph, and for every domain action, if the action is applicable to n subgoal sets of L , then create one child having two sets of subgoals. One set of subgoals, *u-subgoals*, is the union of all of the n subgoal sets, and the other set of subgoals, *x-subgoals*, is the intersection of all of the n subgoal sets. Note that since all of the n subgoal sets are computed by regression through the same action, *x-subgoals* may only be empty for a domain action that has no precondition. Note also that not considering secondary preconditions during the regression may lead to incompleteness. For example, this occurs when an action A with no precondition and a single conditional effect E has an instantiated predicate as the antecedent of E . In this case, A is functionally

equivalent to an action B , where the precondition of B is the antecedent of E , and B 's effect is the consequent of E .

Next, we generalise what it means for a domain action to be applicable to a set of subgoals, since we now have a double of subgoals. A domain action is applicable to a subgoal double (u -subgoals, x -subgoals) if it both achieves some subgoal in u -subgoals and doesn't delete any subgoal in x -subgoals. Because of the generalization it is possible to generate more children from a particular node than the usual way, but the generalisation also has the special property that it puts an upper bound on the number of children generated for a particular level. In the worst case, each level of the graph contains no more nodes than there are domain actions (in the spirit of [2]). In and of itself, the generalisation of subgoal sets is admittedly naïve. However, on the whole, it is instructive to try to convince oneself that the SBCS graph contains all of the causal links, and that no solutions will be lost.

Goal-Directed Forward-Chaining Search

SemSyn's forward-chaining search (SFCS) is relegated to the task of searching the SBCS causal links, in effect assembling partial plans into plans. The children-generation function of SFCS differs from that of FCS in that the candidates are not chosen from the entire set of domain actions, but rather are constrained to be only those domain actions appearing in an appropriate causal link entry. In particular, if none of the domain actions achieve any of the top-level goals, then SFCS will terminate immediately without generating any children, whereas FCS in the worst case degenerates into a blind enumeration of all action sequences possible from the initial situation. SFCS alone has no more pruning ability than FCS. The research effort thus far has just been to integrate BCS and FCS, and to evaluate the usefulness of doing so. It is hoped that SemSyn will eventually provide a useful tool for further study.

Putting It All Together

Finally, next-generation planning systems will need to interact with their human users. This was one of the driving motivations for the research community's move from an automatic to an automated paradigm. We posit that automatic algorithms can still be useful as sub-modules to the more encompassing automated systems. Moreover, SBCS and SFCS have human-understandable and intuitive children-generation functions, as do BCS and FCS, so it becomes alluring the possibility to allow the user to view and to manipulate the search's internal data structures - they are simply plan fragments! It is our thesis that user's of automatic planning systems are freed from planning concerns, and are able to fully concentrate on the domain modeling aspects of their applications.

The initial testing phase is being carried out in cooperation with SemSyn's participation in the 4th International

Planning Competition (IPC), hosted at the 2004 International Conference on Automated Planning and Scheduling. The IPC series has developed a formidable testbed, and a rigorous evaluation of the results is forthcoming.

SemSyn's preliminary results appear satisfactory insofar as it is able to solve problems from a variety of domains. However, the algorithm has trouble with domains that have relatively little variation in the domain operators. This is because the traditional wisdom of the research culture is to design a sequence of problems of increasing difficulty in an artificial way, by increasing the number of actions that can be instantiated from a few operators, i.e. by increasing the number of predicates the operators have at their disposal. Conversely, in SemSyn's view, the predicates are akin to database tuples. This means that it is the user's responsibility to model the domain in such a way as the predicate space can be efficiently explored. Indeed, it is possible to write database queries that don't terminate, yet people routinely use Database Management Systems as an integral part of their overall information systems. Analogously, SemSyn's goal is to separate the planning aspects from the domain modeling activities, and to devote its effort to the task of planning - that is, the efficient construction of plans based upon knowledge encoded in the domain operators themselves, regardless of instantiated actions.

References

1. Beeri, C. and Ramakrishnan, R. 1991, "On The Power of Magic". In *Journal of Logic Programming*, 10(3):255-299.
2. Blum, A. and Furst, M.L. 1995, "Fast Planning Through Planning Graph Analysis". In *Proc. International Joint Conference on Artificial Intelligence*, pp. 1636-1642.
3. Fikes, R.E. and Nilsson, N.J. 1971, "STRIPS: a new approach to the application of theorem proving to problem solving". *Artificial Intelligence*, 2(3-4):189-208.
4. Fuchs, D. and Fuchs, M. 1999, "Cooperation between Top-Down and Bottom-Up Theorem Provers". *Journal of Artificial Intelligence Research*, 10:169-198.
5. Kambhampati, S. 1997, "Refinement Planning as a Unifying Framework for Plan Synthesis". *AI Magazine*, 18(2):67-97.
6. Lifschitz, V. 1986, "On the semantics of STRIPS". In *Proc. Reasoning about Actions and Plans*, pp. 1-9. Morgan Kaufmann.
7. McDermott, D. 1996, "The Current State of AI Planning Research". Invited paper, 9th Intl. Conf. on Industrial and Engineering Applications of AI and Expert Systems, pp. 25-34.