Learning rewrite rules to improve plan quality

Muhammad Afzal Upal
Department of Computing Science
University of Alberta
upal@cs.ualberta.ca

Considerable planning and learning research has been devoted to the problem of learning domain specific search control rules to improve planning efficiency. There have also been a few attempts to learn search control rules that improve plan quality but such efforts have been limited to state-space planners. The reason being that most of the newer planning approaches are based on plan refinement. In such planners, information about the current state of the world that is required to evaluate a complex quality metric is simply not available during planning. An alternative technique is *planning by rewriting* that suggests first generating an initial plan using a refinement planner and then using a set of rewrite-rules to transform it into a higher quality plan (Ambite & Knoblock 1997). Unlike the search control rules that are defined on the space of partial plans, rewrite rules are defined on the space of complete plans. This paper presents a system called REWRITE that automatically learns rewrite rules.

REWRITE has three main components. The first is a partial-order causal-link planner (POP). The second component does the analytic work of identifying the *replacing* and *to-be-replaced* action sequences. The third component is a case library of plan-rewrite rules.

The input to REWRITE's analytic component is (a) a problem described by an initial state and goals (b) the plan and planning trace produced by the partial order planner for this problem, and (c) a "better plan" for the same problem. The better plan is the one that has a higher quality rating than the one produced by the underlying partial-order planner, as per the quality function that assesses how resources are impacted by each plan. This better plan might be provided by some oracle, by a user, or by some other planner.

REWRITE's analytic component first reconstructs a set of causal link relationships between the steps in the better plan and a set of required ordering constraints. The second step is to retrace POP's planning-trace, looking for plan-refinement decisions that added a constraint that is not present in the better plan's constraint set. We call such a decision point a *conflicting choice point*. Each conflicting choice point indicates a gap in REWRITE's knowledge and hence a possible opportunity to learn. For any conflicting choice point, there are two different plan-refinement decision sequences that can be applied to a partial plan: the one added by the default POP algorithm, and the other inferred from the better-quality plan. The application of one set of plan-refinement decisions leads to a higher quality plan and the other to a lower quality plan. However, all of the "downstream" planning decisions may not be relevant to resolving the flaw at the conflicting choice point. The rest of the better-plan's trace and the rest of the worse-plan's trace are then examined, with the goal of labeling a subsequent plan-refinement decision q relevant if (a) there exists a causal-link $q \xrightarrow{c} p$ such that p is a relevant action, or (b) q binds an uninstantiated variable of a relevant open-condition.

Once both the better plan's relevant decisions and the worse plan's relevant decisions have been identified, REWRITE computes (a) the actions that are added by the worse plan's relevant decision sequence. These become the action sequence to-be-replaced; (b) The actions that are added by the better plan's relevant decision sequence. These become the replacing action sequence; (c) The preconditions and effects of the replacing and the to-be-replaced action sequence. REWRITE then stores this information as a rule. Currently, REWRITE uses a speed-up partial order planning system DerPOP to generate an initial plan P along with the casual-link and ordering constraints (Upal & Elio 1999). All rules whose to-be-replaced action sequence S_1 is a subset of P are then retrieved and applied and the highest quality plan thus produced is returned.

Preliminary evaluation results indicate that the overhead for applying these techniques to identify, store and then use planning-rewrite rules is not large. The benefit is that rewrite rules do not have to be hand-coded.

References

Ambite, J., and Knoblock, C. 1997. Planning by rewriting. In *Proc. of AAAI-97.*

Upal, M. A., and Elio, R. 1999. Learning rationales to improve plan quality. In *Proc. of FLAIRS-99.*