Flexibly Interleaving Processes

Erica Melis[⋆] Carsten Ullrich

Universität des Saarlandes, FB Informatik
D-66041 Saarbrücken, Germany
melis@cs.uni-sb.de

Abstract. We discuss several problems of analogy-driven proof plan construction which prevent a solution for more difficult target problems or make a solution very expensive. Some of these problems are due to the previously assumed fixed order of matching, reformulation, and replay in case-based reasoning and from a too restricted combination of planning from first principles with the analogy process. In order to overcome these problems we suggest to interleave matching and replay as well as case-based planning with planning from first principles.
Secondly, the restricted mixture of case-based planning and planning from first principles in previous systems is generalised to intelligently employing different planning strategies with the objective to solve more problems at all and to solve problems more efficiently.

1 Introduction

The common CBR cycle [1] consists of a sequence of the subprocesses matching + reformulation/ retrieval, replay, adaptation, and storage. However, empirical psychological results, e.g. in [6], question this fixed sequence of subprocesses of analogical reasoning.

Secondly, previous CBR systems focus on a single problem solving strategy only. This might be justified if no domain knowledge is available that could be used to solve a problem from first principles. In case-based planning (CBP), however, knowledge such as planning operators is available while control knowledge may not. In several computational case-based planning systems (including analogy-driven proof plan construction) one kind of combining case-based planning with planning from first principles is realized in which planning from first principles is employed to close open subgoals remaining after the derivational replay of a source plan or to close subgoals remaining after the transfer of abstract steps, see [3]. That is, these systems use planning from first principles for the adaptation of retrieved and replayed source plans.

Empirical evidence shows, however, that analogical reasoning can more generally be combined with standard problem solving. Psychologically this fact has been experimentally verified, e.g., in physics problem solving [18]. In physics problem solving VanLehn and Jones found that poor problem human solvers

[⋆] This work was supported by the Deutsche Forschungsgemeinschaft, SFB 378

© Springer-Verlag Berlin Heidelberg 1999

use analogy instead of problem solving from first principles even when this is not most effective, whereas good problem solvers use analogy to learn control knowledge that is missing and to fill other knowledge gaps. Nelson, Thagard, and Hardy [16] discuss the need for a unified theory of analogy, rule-based reasoning, and explanation.

For computational case-based planning we shall question both, the previous fixed sequence of subprocesses of case-based reasoning and the restricted combination of case-based planning with planning from first principles. In particular, we demonstrate how some problems that prevent analogy-driven proof plan construction from replaying proofs that are considered analogous by mathematicians or from being efficient can be solved by interleaving several subprocesses of analogy. Furthermore, we shall show that we need more flexibility in choosing between case-based planning and planning from first principles than that resulting from closing remaining subgoals by sourceplanning from first principles.

In the remainder, we first briefly review the original analogy-driven proof plan construction. Then we discuss the problems of this and similar approaches to case-based planning and then describe solutions to these problems. Finally, we discuss a generalization to multi-strategy planning.

2 Analogy-Driven Proof Plan Construction

Analogy-driven proof plan construction was first introduced in [11] and realized in the proof planner *CLAM*[14]. Compared with previous approaches to theorem proving by analogy, its main novelties are the analogical transfer at the abstract level of proof *plans* and the need for *reformulations* of proof plans that go beyond symbol mapping. Often, these reformulations are a prerequisite for successfully second-order[1] matching a source with a target theorem that do not match in the first place. The original analogy-driven proof plan construction, as realized on top of the proof planner *CLAM* and in the ΩMEGA system [2], is outlined in Table 1.

To start with, repair-matching tries to second-order match and when no proper match is found, an appropriate reformulation is applied to the problem and then matching is tried again. When a repair-match of the problems is obtained, the found reformulations are realized.

That is, the reformulations do not only change the theorem and proof assumptions but also the proof plan. For instance, Add-Argument may duplicate whole subplans[2] or adding an antecedent may introduce an additional subgoal into the plan.

Some of the frequently needed reformulations are

– symbol-to-symbol and symbol-to-term mapping,

[1] As opposed to first-order matching, second-order matching may match function and relation variables.

[2] Actually, the routine of the original Add-Argument reformulation that checks and changes operators and subgoals of the source plan is quite complicated, since it has to traverse and analyse the whole plan and to predict local changes.

```
input: source plan, source theorem and assumptions, target theorem and assumptions
output: (partial) target plan

Repair-match: Attempt to second-order match source and target problems triggers
reformulations of the source plan.
Reformulation: source plan ← reformulated source plan.

Replay: until source plan exhausted do
    Get next operator M from source plan.
    Check M's justifications.
    if justifications hold, then transfer M to target,
    else choose suitable action.
Plan: Plan from first principles for remaining open goals.
```

Table 1. Outline of analogy-driven proof plan construction

– swapping function arguments,
– duplicating arguments of functions,
– adding an antecedent to a formula,
– freezing function arguments to constants,
– abstractions, and more complex reformulations such as
– adding/removing final or initial plan segments.

The first five reformulations have been realized in the implementations described in [14] and [12]. For instance, in ABALONE [14] the reformulation Condt is applied when a source theorem Th_s is repair-matched with a target theorem $C \rightarrow Th_t$, where Th_s and Th_t match. Condt introduces C as a subgoal and closes a certain plan branch, if C is disproved.

The last kind of reformulations correspond to Carbonell's T(ransformation)-operators final-segment concatenation and initial-segment concatenation [5], where a final plan segment reduces the theorem to a (set of) subgoals and an initial segment transforms assumptions to other proof assumptions. Consider, for instance, the source theorem $Th_s : \ldots \forall x. x \in setA \rightarrow x \in setB$ and the target theorem $Th_t : \ldots \rightarrow setA' \subseteq setB'$. Initially, Th_s does not match with Th_t. Only when the operator `ApplyDefinition` applies the definition of $\subseteq$ to Th_t, then the resulting subgoal matches Th_s. Of course, the additional segments can be longer than one step and more diverse than just the operator `ApplyDefinition`.

2.1 Problems with this Analogy

When experimenting with analogy-driven proof plan construction on increasingly difficult problems a mathematician would consider analogous, we discovered several problems some of which are discussed below. Essentially, the discussed problems fall into four classes: (1) How to recognize the transfer of a source *sub*plan, (2) how to realize segment concatenations efficiently, (3) when to execute plan reformulations, and (4) on which goals to replay which operators?

1. The first class includes the following problems
   (a) **Subplans.** More often than not, a whole source plan cannot be transferred analogically but a *sub*plan is transferable. For example, if the source plan proves the theorem $A \wedge B$ and the target theorem B' matches with B, then only the subplan satisfying B should be transferred. Therefore, it is desirable to replay a subplan only. How do we know which subplan?
   (b) **Irrelevant matches.** Matching proof assumptions that do not belong to the replayed part of the source proof causes an unnecessary overhead.
2. **Final- and initial segment concatenation.** The search space for a sequence of reformulations that yields final- or initial-segment concatenation is potentially infinite, because potentially any sequence of plan operators has to be considered (such as `ApplyDefinition`, `Normalize`, etc). An efficient choice of a sequence of reformulations would require a severe restriction of the operators that can possibly be added and a restriction to short segments. This is comparable with Kolbe and Walther's 'matching modulo evaluation' in [10] that allows to add one application of a definition in case a symbol mapping reformulation does not yield a match between source and target theorem. In domains with a limited choice of operators as equality proofs, this approach may be adequate as the search space is small. In more complex proofs as planned with the ΩMEGA system, this restriction is too strict.
   An alternative way to realize certain segment concatenations is to prove the implication $(Th_s \rightarrow Th_t)$ – as Koehler proposed in [8] – and, if successful, add the resulting proof as an initial segment of the target plan (and similarly for the proof assumptions, add a subplan as a final segment). This alternative, however, excludes a combination with other reformulations, even with symbol or term mapping because the implication to be proved is fixed and therefore, a transfer of the plan is only possible when the source and target problem share the same predicates.
3. The third class of problems concerns the decision as to when and how reformulations should be applied.
   (a) **Reformulating before the replay?**
   When a reformulation is applied *before* the replay as realized previously, it may become pretty difficult to reformulate the source plan appropriately, because many local situations have to be distinguished. For instance, an additional conjunct C in a definition causes different changes depending on whether the definition is applied in forward planning or in backward planning. In forward planning, an `AndE` operator is introduced and the assumption C is added, whereas in backward planning the operator `AndI` is introduced and the goal C is added. Moreover the changes depend on the position at which C occurs in the target formula. For instance, if C is an additional conjunct hidden in a subformula of a definition as, e.g., in $(F \equiv A \wedge C \rightarrow B)$, the reformulation should affect those parts of the source plan that deal with establishing A.
   (b) **Inserting an operator or reformulating.** It is difficult to decide in advance whether the same (source) operator should be applied in

the target again or whether the plan has to be reformulated first. For instance, an operator might be applicable to a conjunctive goal as well as to a goal that is not a conjunction and in this case we do not need to apply the reformulation AddConjunct.

(c) **Origins of symbols.** In case a symbol- or term mapping is needed to repair-match the source and the target theorem, the mapping cannot generally be applied to subgoals and operators. For instance, the operator `ComplexEstimate` that is used for planning proofs of limit theorems applies, among others, the Triangle Inequality $|A + B| \leq |A| + |B|$ that contains the symbol $+$. Now if the source theorem is LIM+: $\lim_{x \to a}(f(x) + g(x)) = L_1 + L_2$ and the target theorem LIM* is $\lim_{x \to a}(f(x) * g(x)) = L_1 * L_2$, a mapping $+ \mapsto *$ would be triggered. How does this mapping affect the occurrence, change, or replacement of $+$ in a proof plan that contains `ComplexEstimate`?

As a partial solution, function- and relation symbols could be indexed. Running the source plan (again) with these indices would indicate, where which occurrence of a symbol in subgoals originates from. This indexing is realized in [9] and [14]. It may be reasonable for small proofs but produces a considerable overhead which often is unnecessary. Moreover, such a discrimination of symbols does not tell anything about the symbols occurring in operators.

(d) **Plan reformulation.** A reformulation that changes the content of operators does not fit our general philosophy, *not* to act at the low level of a logical calculus but at the plan-level in case-based proof planning. Hence, in [14] reformulations of a proof plan introduce, delete, or replace operators and subgoals (as opposed to the reformulations originally experimented with in the *Ω*MEGA system).

4. **Corresponding goals.** Since an operator specifies a program that produces a – not necessarily fixed – sequence of inference steps, more often than not the target subgoals/assumptions to which the next operator should be applied, are not known in advance but only after the actual application of the preceding operator. Even the number of subgoals that an operator produces may depend on the planning context. Hence, one cannot decide in advance (before any operator is replayed), on which goal to replay an operator. Suppose, in the source the operator O_1 is applied to the goal g_{s1} and O_1 was replayed on a target goal g_{t1} and produced the subgoals $g_{t11} \ldots g_{t1n}$. If the next operator O_2 was applied to one of the source subgoals produced by O_1, then – because of the goal-dependency justification – it has to be replayed on one of the g_{t1i}. Hence, we have to find the corresponding subgoals. To see that this really might be a problem, consider the following example. The operator `ModusPonensBackward` is applicable to any goal. Hence, choosing the wrong goal will most likely create a false analogical transfer.

3 Solving the Problems

In this section we describe our solutions to the above mentioned problems. Table 2 gives an outline of the revised analogy-driven proof plan construction.

```
input: source plan, source theorem and assumptions, target theorem and assumptions
output: (partial) target plan

Repair-match or plan:
    until
        a cheap second-order repair-match between target theorem and a
        source (sub)goal $g_s$ down to depth $d$ is found

    do plan from first principles for target goal.
$M$ := operator satisfying $g_s$
Interleaved match/replay:
until source plan exhausted do

    Repair-Match:
    Find target goal $g_t$ corresponding to $g_s$ by extending the repair-match.
    ReplayRef:
    if $M$ applicable then apply $M$ to $g_t$ and
    advance source plan ($M$ := next operator, $g_s$ := goal satisfied by $M$)
    else apply reformulation triggered by repair-match or by the failed application
    conditions.

Plan: Plan from first principles for remaining open goals.
```

Table 2. Outline of the revised analogy-driven proof plan construction

In the **repair-match or plan** cycle, we establish a connection between the target theorem and a source (sub)goal. First, we try to repair-match the target theorem with the source theorem. In case no collection of *simple* reformulations[3] of the source goal matches the target theorem, we try to match with subgoals down to depth d (parameter) in the source plan until a simple repair-match with a subgoal can be established. If no cheap repair-match is found, we plan one step from first principles and restart the **repair-match or plan** cycle. The planning step yields a step in a final plan segment.

In the **match/replay** cycle, we try to replay an operator M from the source plan. In order to apply M in a correct way (problem 4), we first find the target goal corresponding to the currently treated g_s by repair-matching the g_s against

[3] Simple reformulations are: symbol mapping, term mapping, permutation of arguments, freezing arguments, duplication of arguments, adding conjuncts, adding antecedents.

those target goals that satisfy the same goal dependencies as g_s. If M is applicable, M is replayed in the target. Otherwise, a reformulation triggered by the repair-match or by the failed application conditions of the operator to be replayed is applied in a way corresponding to the local situation. For instance, the reformulation AddConjunct triggered by repair-matching source and target goals may result in additionally applying the operator `AndIntro` and in introducing a new subgoal.

Eventually, when the source plan is exhausted, the remaining open goals are tackled by planning from first principles.

More often than not – but depending on the generality of the operators – the application conditions of an operator still hold when the repair-match consists of swapping arguments, symbol mapping, or term mapping. Then, no proper plan change is necessary at all and the repair-match is used only to specify the goal correspondences.

Reformulations represent heuristics on how to resolve critical differences between goals or assumptionsof source and target and on how to change plans when a replay is not possible in the first place. Since reformulations change a plan by adding, removing, and replacing operators rather than by changing the (code of the) operators directly (see problem 3d), an interesting conclusion is that domain-dependent knowledge has to be used in order to determine by which operator to replace a source operator in order to be applicable to the reformulated goal. For instance, in proving theorems for real numbers that involve estimations, the correspondence between a source goal $a < b$ and a target goal $a > b$ requires to replace a $\texttt{Solve}_<$ operator by a $\texttt{Solve}_>$ operator.

3.1 What do the Solutions look like?

How does this new analogy-procedure solve the above mentioned problems?

Problem 1a. Going down the source plan for repair-matching means to restrict the potential replay to *sub*plans. For example, let $Th_t : A \wedge B \rightarrow C$ be the target theorem, $Th_s : F$ the source theorem and $g_{s_1} : A' \wedge B' \rightarrow C'$ a source subgoal of depth one which results from applying the operator `ApplyDefinition` on Th_s. To find the (sub)plan to be transfered, we start by repair-matching Th_t and Th_s. As no cheap repair-match is found, we continue by stepwise going down the source plan to match the subgoal g_{s_1} with Th_t. Here, a cheap repair-match is possible. As a result the subplan whose root node is g_{s_1} is replayed analogically as illustrated in Figure 1.

Problem 1b is solved by matching assumptions when needed only. It does not make much sense to try to match every source assumption with a target assumption *before* the replay of the proof plan because not every source assumption belongs to the partial plan that is actually replayed. Therefore, this matching takes place within the **interleaved match/replay** routine.

Problem 2 is solved by the planning part in the **repair-match or plan** cycle which produces a plan segment as shown in Figure 2. Consider the following example: let be $Th_s : A \wedge B \rightarrow C$ be the source theorem and $Th_t : F$ the target

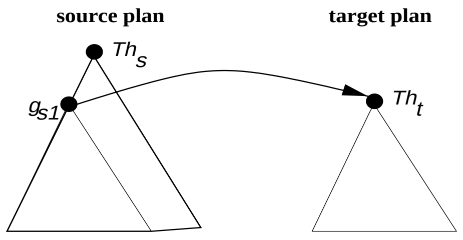

Fig. 1. Finding a subplan for the replay

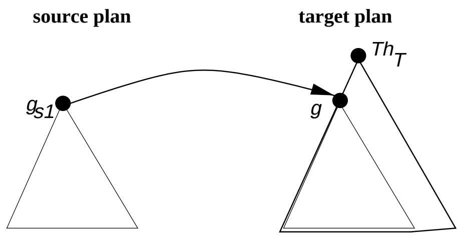

Fig. 2. Planning produces segment concatenation

theorem. No cheap repair-match is found (even when looking at the subgoals), so we plan from first principles in the target. This suggests the application of the operator `ApplyDefinition` which has the subgoal $g_{t_1} : A' \wedge B' \to C'$. Now, repeating the cycle, we find a cheap repair-match between Th_s and the newly created g_{t_1}. This yields a final-segment concatenation. An initial-segment concatenation is naturally produced by planning for remaining goals including those suggested as new target lemmata by certain reformulations.

Problem 3. By delaying the application of a reformulation, we have exact information about the locally needed reformulations (e.g., is an operator applicable without any reformulation of the plan as asked in problem 3b) including the (local) effects of a reformulation that are clearly understood (thus solving problem 3a). For instance, let's assume that the last operator we replayed produced the target subgoals $g_{t_1} : A' \wedge B'$ and $g_{t_2} : C' \vee D'$. The next operator to be replayed is `ApplyDefinition`, which was applied in the source plan on $g_{s_1} : A$. Repair-matching g_{s_1} with g_{t_1} and g_{t_2} returns the cheapest match $m : g_{s_1} \mapsto g_{t_1}$ even though g_{t_1} contains an additional conjunct. Now the application of the operator `ApplyDefinition` is tried. If it is not applicable due to failing application conditions, then the failure is analyzed. The failure analysis might find that the cause is the additional conjunct B'. This would trigger a reformulation that locally introduces the additional planning step `AndIntro` with the subgoals $g_{t_3} : A'$ and $g_{t_4} : B'$. On $g_{t_3} : A'$ the replay of `ApplyDefinition` is finally possible.

Problem 4 is solved by the **interleave match/replay** cycle because at that point in the analogy procedure the set of target subgoals is known that contains the goal that corresponds to a given source subgoal.
To summarize, interleaving matching and analogical replay with planning from first principles can help to reduce prohibitively large search spaces when searching for reformulations and their possible effects on a proof plan. Interleaving processes in analogy-driven prof plan construction allows for simpler reformulations at the plan-level and for a more tractable matching. The interleaving results in a more flexible analogy that is easier to implement and more powerful.

An example for a proof plan produced by our case-based planning is a partial plan for LIM*, i.e., the goal
$\lim_{x \to a} f(x) = L_1 \wedge \lim_{x \to a} g(x) = L_2 \to \lim_{x \to a} f(x) * g(x) = L_1 * L_2$, shown in Figure 3, where light circles represent nodes in the proof plan with closed subgoals and operators, darker circles represent open goals, the triangles represent hypotheses and proof assumptions and squares are coreferences). The source plan is one of

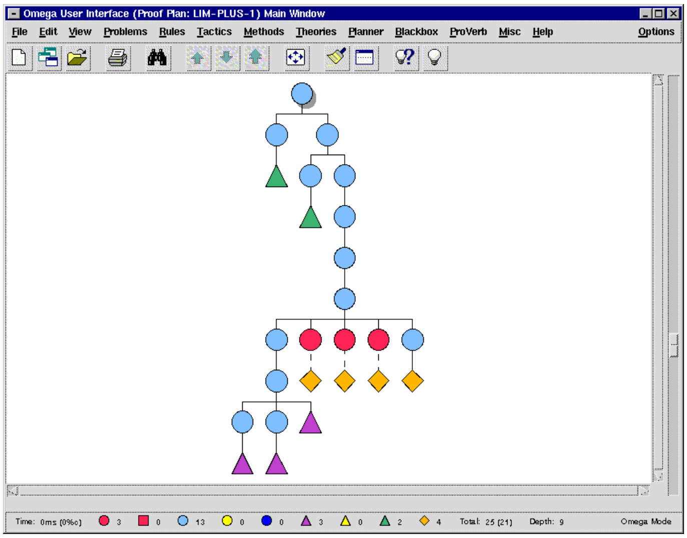

Fig. 3. Screenshot from planning LIM* by analogy to LIM+ in OMEGA (open goals are darker)

LIM+ defined by the assumptions $\lim_{x \to a} f(x) = L_1$, $\lim_{x \to a} g(x) = L_2$ and the goal $\lim_{x \to a} f(x) + g(x) = L_1 + L_2$. For LIM*, first some planning from first principles introduces `Normalize` operators and then case-based planning replays operators of the plan of LIM+ as long as possible. Then planning from first principles satisfies the remaining open goals.

4 Flexibly Planning with Multiple Strategies

In the above procedure, the subprocesses of analogy, planning, match, reformulation, and replay are interleaved when needed. Similarly, we may ask: Why should we solve a planning problem *only* by analogy in the first place rather than calling the analogy by need only? In particular, it is conceivable that different planning operations can be interleaved when necessary or when efficient. For instance, instead of encoding a fixed sequence of planning from first principle, expansion of operators, and instantiation of meta-variables, an intelligent interleaving of these planning operations is useful.

From a case-based reasoning perspective, we propose a flexible mixture of case-based planning with planning from first principles. This is in accordance with the psychological results cited in the introduction. Further psychological evidence suggests that the unreflected analogical transfer of *all* steps of a plan, independent on how simple the subproblem is, is a rather novice-like behaviour [18].

In order to describe our multi-strategy approach, first we define what a planning strategy is based on the refinement operations defined in [7].

Definition 1. Any refinement or modification operation on partial plans is called a *planning strategy*. A refinement operation refines a partial plan π by adding steps and or constraints to π and thereby reduces the set of potential solutions.

In fact, case-based planning can be considered a particular refinement operation. It adds steps and order constraints to a partial plan. Similarly, plan-space or state-space planning from first principles, the expansion of a complex operators [17], difference-reduction planning [4], and the instantiation of meta-variables can be considered planning (refinement) strategies; they go, however, beyond the scope of this paper.

The case-based planning strategy suggests the next planning operator rather than searching for it. Thereby this strategy may reduce the search in planning. However, it causes an additional overhead for matching, retrieval, and adaptation. Hence, only *intelligently* chosing case-based planning or planning from first principles can reduce the overall effort in planning.

Hence, if only one operator is applicable in a plan node after the control-rules have been evaluated (i.e., no branching occurs), then it does not make sense to choose case-based planning because it causes more overhead than planning from first principles. Therefore, case-based planning pays and is chosen in particular, when control knowledge is absent or rare.

We designed a meta-planner that has several strategies at its disposal [13] in order to employ different planning strategies in a flexible way. In the ΩMEGA system [2], we just implemented a first prototype of the multi-strategy planner that has an analogy strategy as one of several planning strategies. The meta-planner decides which of several strategies to use in a particular situation. The meta-planner does not cause much overhead other than evaluating control heuristics.

For instance, for planning LIM*, first some default planning from first principles is performed. That is, the applications of operators such as `Quantifier-Elimination` and `Normalization` are planned essentially without any search. A branching of the search space occurs for the first time when the `Focus` operator is to be applied. The reason is that `Focus` can be applied to any existing subformula of the assumption. If no control knowledge supports the choice of the subformula, the number of subformulae is the branching factor. Therefore, the case-based planning strategy is called and it replays the `Focus` operator with a similar instantiation of the subformula to be focused on in LIM+ and then `Unwraphyp` and `ComplexEstimate` operators are replayed, etc. Certain subgoals cannot be closed by analogical replay but are satisfied by the strategy planning from first principles. After completing the top-level plan of LIM*, the instantiation strategy is applied for particular meta-variables. Then the (complex) operators, indicated by the light circles in Figure 3, can be expanded to subplans by the expansion strategy such that eventually the fully expanded plan has about 420 steps. The remaining meta-variables are instantiated only when certain operators are expanded.

Qualitatively, a flexible choice of strategies allows more problems to be solved without changing the code of the planner or of strategies such as case-based planning. Moreover, new strategies can be integrated more easily into planning.

Quantitatively, i.e., in order to yield a more efficient planning process, the decision which strategy to choose in a particular planning situation needs to address the well known utility problem discussed, e.g., in [15]. So, the meta-planner should decide to call case-based planning only when the analogy-drivenproof plan construction is superior to planning from first principles in terms of estimated costs. Currently, the meta-planner aims at reducing the costs only because other measures such as the quality of proof plans are even more difficult to evaluate. In particular, the estimation $cost_{fp} > cost_{cbp}$, where fp means planning from first principles and cbp means case-based planning, may be computed based on

$$cost_{fp} \approx n \cdot (mc + ac) \cdot b^{(l-1)}, \quad cost_{cbp} \approx (rmc + rc + ac) \cdot l, \tag{1}$$

where n is the number of alternatives in the current planning situation *after* the evaluation of control knowledge, mc is the average match costs of a single operator, ac is the average cost of an operator application, b is the average branching factor, l is the expected length of the remaining plan, rmc is the average cost of repair matching, and rc is the average cost of a reformulation.

If $cost_{fp} > cost_{cbp}$ then case-based planning is preferred, oherwise planning from first principles. This means that if the saved search overweights the overhead of the case-based planning strategy caused by the matching and reformulation, then choosing case-baed planning is superior to planning from first principles.

The estimation of $cost_{fp}$ is related to the number n of alternative operators remaining *after* the evaluation of control-rules. That is, the available control knowledge decreases the estimated costs for planning from first principles considerably. Therefore, the better the control knowledge, the more likely will the choice of standard planning be as opposed to case-based planning.

A lesson for case-based reasoning in general is its controlled use, i.e. case-based reasoning should be chosen only when it is superior to other problem solving strategies, i.e., when other strategies cannot find a solution or when analogy has lower estimated costs than other working strategies. Of course, control heuristics are necessary, e.g. heuristics that evaluate the complexity of the plan reformulations and compare this with the branching of the search space.

5 Conclusion and Future Work

We presented a more flexible analogy procedure that can interleave planning from first principles with repair-matching and that interleaves repair-matching with replay in case-based proof planning. The greater flexibility leads to a better performance and allows to solve more problems with our analogy-driven proof plan construction.

The comparison with our previous procedure is explained in detail above. Compared with Veloso's case-based planning [19] our procedure has reformulations and the repair-match rather than a simple symbol match, Veloso's match does not allow to match one symbol of the source to several symbols in the target, and our analogy can plan from first principles not just for initial-segment but also for final-segment concatenations. Furthermore, the proof planning domain is more complicated than the typical planing domains.

We have just implemented an even more radical interplay of case-based planning with other planning strategies that allows to invoke case-based planning or planning from first principles dependent on the planning situation. A meta-planner chooses the strategies according to strategic control knowledge. Although some strategic knowledge is encoded into the estimation (1), we expect it to be a crude estimation. Therefore, we plan to rather learn the strategic control decisions.

References

1. A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7:39–59, 1994.
2. C. Benzmueller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, A. Meier, E. Melis, W. Schaarschmidt, J. Siekmann, and V. Sorge. OMEGA: Towards a mathematical assistant. In W. McCune, editor, *Proceedings 14th International Conference on Automated Deduction (CADE-14)*, pages 252–255, Townsville, 1997. Springer.
3. R. Bergmann, H. Munoz-Avila, M.M. Veloso, and E. Melis. Case-based reasoning applied to planning. In M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess, editors, *Case-Based Reasoning Technology from Foundations to Applications*, volume 1400 of *Lecture Notes on Artificial Intelligence (LNAI)*, pages 169–200. Springer, 1998.
4. A. Bundy, Stevens A, F. Van Harmelen, A. Ireland, and A. Smaill. A heuristic for guiding inductive proofs. *Artificial Intelligence*, 63:185–253, 1993.

5. J.G. Carbonell. Learning by analogy: Formulating and generalizing plans from past experience. In R.S. Michalsky, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 137–162. Tioga, Palo Alto, 1983.
6. K.J. Holyoak, L.R. Novick, and E.R. Melz. Component processes in analogical transfer: Mapping, pattern completion, and adaptation. In K.J. Holyoak and J.A. Barnden, editors, *Advances in Connectionist and Neural Computation Theory*, volume 2, pages 113–180. Ablex Publishing, 1994.
7. S. Kambhampati and B. Srivastava. Universal classical planner: An algorithm for unifying state-space and plan-space planning. In M. Ghallab and A. Milani, editors, *New Directions in AI Planning*, pages 61–78. IOS Press, Amsterdam, Oxford, 1996.
8. J. Koehler. Planning from second principles. *Artificial Intelligence*, 87, 1996.
9. Th. Kolbe and Ch. Walther. Reusing proofs. In *Proceedings of 11th European Conference on Artificial Intelligence (ECAI-94)*, Amsterdam, 1994.
10. Th. Kolbe and Ch. Walther. Second-order matching modulo evaluation – a technique for reusing proofs. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, 1995. Morgan Kaufmann.
11. E. Melis. A model of analogy-driven proof-plan construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 182–189, Montreal, 1995.
12. E. Melis. The Heine-Borel challenge problem: In honor of Woody Bledsoe. *Journal of Automated Reasoning*, 20(3):255–282, 1998.
13. E. Melis. Proof planning with multiple strategies. In *CADE-15 workshop: Strategies in Automated Deduction*, 1998.
14. E. Melis and J. Whittle. Analogy in inductive theorem proving. *Journal of Automated Reasoning*, 22:2 117–147, 1999.
15. S. Minton. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–391, 1990.
16. G. Nelson, P. Thagard, and S. Hardy. Integrating analogy with rules and explanations. In *Advances in Connectionist and Neural Computation, Analogical Connections*, volume 2, pages 181–206. Ablex, 1994.
17. A. Tate. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 888–893. Morgan Kaufmann, 1977.
18. K. VanLehn and R.M. Jones. Better learners use analogical problem solving sparingly. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 338–345, Amherst, MA, 1993. Morgan Kaufmann.
19. M.M. Veloso. Planning and Learning by Analogical Reasoning. Springer Verlag, Berlin, New York,x 1994.