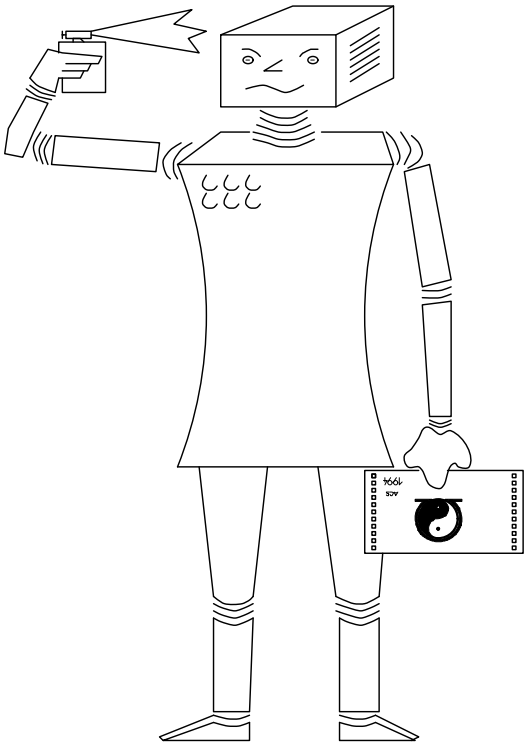

Analogy Makes Proofs Feasible

Erica Melis and Manuela Veloso

Published as: Proceedings of the AAAI -94 workshop on CBR, pages 13-17

Analogy Makes Proofs Feasible

Erica Melis* and **Manuela Veloso**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
email: {melis,mmv}@cs.cmu.edu

Abstract

Many mathematical proofs are hard to generate for humans and even harder for automated theorem provers. Classical techniques of automated theorem proving involve the application of basic rules, of built-in special procedures, or of tactics. Melis (Melis 1993) introduced a new method for analogical reasoning in automated theorem proving. In this paper we show how the derivational analogy replay method is related and extended to encompass analogy-driven proof plan construction. The method is evaluated by showing the proof plan generation of the Pumping Lemma for context free languages derived by analogy with the proof plan of the Pumping Lemma for regular languages. This is an impressive evaluation test for the analogical reasoning method applied to automated theorem proving, as the automated proof of this Pumping Lemma is beyond the capabilities of any of the current automated theorem provers.

1 Introduction

Automated theorem proving systems have attained a remarkable strength when it comes to pure deductive search (McCune 1990). They are, however, still weak with respect to long range planning or other global search and control issues, as needed in complex proofs. In order to attack complex mathematical problems we need to combine the strength of traditional automated theorem provers with human-like capabilities.

Mathematicians have clearly recognized and emphasized the power of analogical reasoning in mathematical problem solving (Hadamard 1945; Polya 1954; 1957; van der Waerden 1964). Hence, integrating analogy into theorem provers is one of the challenging problems in automated theorem proving (Bledsoe 1986; Wos 1988).

Melis analyzed several empirical sources (Melis 1994b) on how mathematicians prove theorems. As a result, she identified the use of the following reasoning strategies: proof planning with partially instantiated methods, structuring of proofs, analogy embedded in proof planning, analogical transfer of partial subproofs, and analogical transfer of reformulated subproofs and methods. This set of strategies is closely related to the analogical reasoning paradigm used in other areas of AI, such as planning (Hammond 1986; Kambhampati & Hendler 1992; ?; Veloso 1992).

In this paper, we explain how to incorporate analogical reasoning capabilities into automated theorem proving systems using this planning perspective. The evaluation of the method needs to be done by concrete identification of mathematical proofs that can be generated by analogy with other proofs. In (Melis 1993) simpler examples are given for the application of the method. Here we have chosen a difficult example in order to show the relevance of the technique that helps to prove analogous theorems with less effort or to prove them automatically at all. The paper shows how to derive by analogy the proof of the Pumping Lemma for context free languages (CFL) from the proof of the Pumping Lemma for regular languages (REGULAR). The reader unfamiliar with the formalism of mathematical proofs may often refer to the figures as an additional source of evidence and refer to the formalism only as a last resource.

The paper is organized in four sections. Section 2 briefly introduces the procedure developed for automated theorem proving by analogy. Section 3 presents the example and Section 4 draws conclusions on this work.

2 Analogy Procedure

Theorem proving by analogy consists of finding a proof for a target problem on the basis of a given proof of a source problem, which is similar to the target problem (See Figure 1).

Traditionally, the analogy between the source proof and the target proof was realized by establishing a mapping from the primitive symbols of the source theorem onto the symbols of the target theorem, and by extending this map such that it provides a proof of the target theorem when applied to the single steps of the source proof. Here the primitive symbols are those symbols of the signature in which the theorems are expressed. In other words, traditional approaches (Kling

*On leave from University of Saarbrücken, Germany. This work was supported by the Max Kade Foundation.

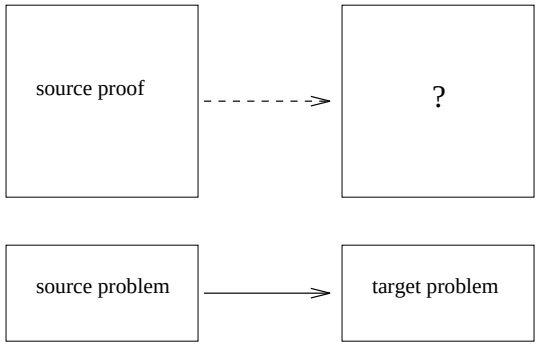

Figure 1: Analogy in theorem proving

1971; Munyer 1981; Owen 1990) are centered around symbol mapping and the transfer of single proof steps.

The analysis of empirical sources, as performed in (Melis 1994b), however, suggests the following features of analogy in theorem proving:

- representation of analogy within a proof planning framework,
- analogical transfer of proof ideas, subproofs, or methods found by decomposition,
- analogical transfer based on reformulation, including abstraction, rather than just by symbol mapping.

The complete general analogical cycle involves the generation, storage, retrieval, and adaptation of past similar solutions. In this paper we address only the adaptation phase and show how it relates to the problem of replaying the derivation of a solution for a planning problem. We **extend** the original analogical replay of (Veloso 1992) to encompass forward chaining in addition to backward chaining. Proofs are therefore bidirectionally constructed as a combination of inferences originated in the assumptions and deductions required to reach the goal. We further include the ability to reformulate and restructure the operators, goals, and assumptions of the source case. Table 1 outlines the analogical reasoning replay cycle when applied to theorem proving. (Note that in this paper we do not specify the methods of reformulation and restructuring. The interested reader is referred to the reference (Melis 1993).)

We now illustrate this procedure applied to a specific complex proof.

3 Example: Constructing a Proof Plan for the Pumping Lemma for CFL

The Pumping Lemmata in general state that every sufficiently long word in a language L contains some subwords which can be repeated ("pumped") any number of times, and the resulting word will still be in L. The general idea underlying a proof for the Pumping Lemma is to find two nodes of the parse tree of a word,

1. Terminate if the goal statement is satisfied in the current state.
2. If the source case is exhausted, then base-level plan for the remaining goals.
3. Get a new action from the source case and apply to it the current set of reformulations. The action is either an assumption that was used in *forward* chaining, or a goal that was used in *backward* chaining.
4. Let p be the goal expanded or the assumption used in the source case.
5. Let R be the reformulation of p, such that p matches a current target goal or assumption. Add R to the set of reformulations for future use.
6. Link the new p node to the source case. Advance case.
7. Select the operator chosen in the case and apply reformulations.
8. Restructure the operator if necessary.
9. Check if the operator choice is verifiable. If it is not, then
   - Select another operator by base-level planning.
   - Go to step 1.
10. Add to the set of pending goals all the preconditions of the operator that are not satisfied in the current state. Add to the set of assumptions the effects of operators applied forward.
11. Link the new operator node to the source case. Advance case.
12. Go to step 1.

Table 1: Outline of the extended analogical replay cycle.

which are labelled with the same variable and to repeat the subword(s) that are located between them.

Proving the Pumping Lemma for context free languages automatically is still far beyond the capabilities of automated theorem provers (such as OTTER (McCune 1990))[1]. The proof has too many assumptions and is very complex[2], and hence, the search space for the proof is enormous. It is even hard for humans to write a correct logical proof. Theorem proving by analogy as shown below, however, generates a proof of CFL from a proof of REGULAR. More precisely, starting with a proof plan of the Pumping Lemma for regular languages, we obtain a proof plan for CFL by analogy-driven proof plan construction. This construction requires two other *small* subproofs to be found directly by a theorem prover.

[1] A lot of user guidance is crucial for any complex proof. E.g., for Nqthm (Boyer & Moore 1988), user guidance in form of lemmas is crucial (personal communication with Bob Boyer). Shankar's proof of Gödel's theorem has 1741 lemmas and the file to generate this proof is more than 1 Mbyte.

[2] According to the third, fourth, and sixth complexity criterion in (Boyer & Moore 1988).

We do not focus on initial generation of proof plans here, but start already with a source proof plan. The operators of this plan closely correspond to the steps of the proof in (Hopcroft & Ullman 1979). The example reveals the importance of automated decomposition of proofs into appropriate subproofs (see (Melis 1994a)) because decomposition yields subproofs that can be transferred analogically for the target proof, or which can be proved directly.

In order to speak about formal proofs of the Pumping Lemmata, we need some formal notation. First we provide the notation so that the reader can refer to it when reading the technical details of the proofs.

Notation

In general, a logical proof is represented as a tree of problems such that the leaves are the *assumptions* of the proof, the root is the theorem to be proven, and each subgoal (node) is derived by logic calculus rules from its immediate predecessors. Notice that the proof arguments for the Pumping lemma in textbooks are often easier to understand, since they operate with pictures while a logical proof uses only formulas.

Let a *language* L be a tuple $L = (V, T, p, S)$, where V is a set of non-terminal symbols, T is a set of terminal symbols, p is a set of production rules, and S is the start symbol. Derivations in L are denoted by $\stackrel{*}{\Rightarrow}$. We assume that the reader is familiar with regular and context free languages, the Chomsky normal form for CFL, etc (see (Hopcroft & Ullman 1979)).

$|z|$ denotes the length of the word z. xy is an abbreviation for the concatenation $x \circ y$ of words x and y. The parse tree $tree(w)$ represents the derivation of a word w using the rules of p. The root of $tree(w)$ is labeled with S. $path$ denotes a path in $tree(w)$ starting in S. The function $l(node)$ returns the label of $node$ which is a variable or terminal symbol used in the derivation at that step in $tree(w)$. Notice that different nodes may have the same label. The function $yield$ maps nodes of $tree(w)$ to words (see (Hopcroft & Ullman 1979)). The function $depth$ returns the depth of a node in the tree. The function $sw(N_i, N_j)$ is only defined for N_i, N_j with $depth(N_i) < depth(N_j)$ and both N_i and N_j belonging to the same path in $tree(w)$, and returns the subword of $yield(N_i)$ which precedes $yield(N_j)$ in $yield(N_i)$[3]. The function $swp(N_i, N_j)$ returns the pair of subwords of $yield(N_i)$ which consists of the subword preceding $yield(N_j)$ and the subword succeeding $yield(N_j)$ in $yield(N_i)$[4].

A Proof Plan of REGULAR

We first give the source theorem, the proof assumptions, and subgoals of the source proof plan according to a straightforward logical reconstruction of the mathematical proof in (Hopcroft & Ullman 1979).

[3] The proof of REGULAR uses this function.

[4] The proof of CFL uses this function.

Theorem (REGULAR): Let L be a regular language. Then there is a constant k (dependent only on L), such that if $w \in L$ and $|w| \geq k$, then there are $u, v, x \in T^*$, such that $w = uvx$, $|uv| \leq k$, $|v| \geq 1$, and $\forall i(uv^i x \in L)$.

For a proof of REGULAR it suffices to show (according to (Lewis & Papadimitriou 1981)) that:

1. $\exists k \forall w\ (|w| > k \rightarrow \exists u, v, x, A\ (u, v, x \in T^* \wedge A \in V \wedge S \stackrel{*}{\Rightarrow} uA \wedge A \stackrel{*}{\Rightarrow} vA \wedge A \stackrel{*}{\Rightarrow} x \wedge uvx = w \wedge |v| \geq 1))$.

We consider eleven assumptions, which are used in the proof[5]; A11 is named L11 because it is a lemma (subgoal), the proof of which is not decomposed further. L11 is derived from the features of the parse tree for words w of a regular language L.

A0: $\forall x(x \neq e \rightarrow |x| \neq 1$
A1: $yield(S) = w$
A2: $\mathbf{regular}(w) \rightarrow \forall X, N, \mathbf{a}, c\ (\mathbf{sw}(X, N) = \mathbf{a} \wedge yield(N) = c \rightarrow yield(X) = \mathbf{a} \circ c)$
A3: $\forall N, x\ (N \in tree(w) \wedge yield(N) = x \rightarrow l(N) \stackrel{*}{\Rightarrow} x)$
A4: $\mathbf{regular}(w) \rightarrow \forall X, N_j, \mathbf{v}\ (\mathbf{sw}(X, N_j) = \mathbf{v} \rightarrow l(X) \stackrel{*}{\Rightarrow} \mathbf{v} \circ l(N_j))$
A5: $\forall N_i \exists path\ (N_i \in tree(w) \rightarrow S, N_i \in path)$
A6: $\forall N_i \exists x\ (N_i \in tree(w) \rightarrow yield(N_i) = x \wedge x \in T^*)$
A7: $\forall N_i, N_j \forall path\ (N_i, N_i \in path \wedge N_i \neq N_j \rightarrow \mathbf{sw}(N_i, N_j) \neq \mathbf{e})$
A8: $\forall N_i, N_j, \forall path \exists \mathbf{v}\ (N_i, N_j \in path \wedge depth(N_i) < depth(N_j) \wedge \neg leaf(N_i) \wedge \neg leaf(N_j) \rightarrow \mathbf{sw}(N_i, N_j) = \mathbf{v} \wedge \mathbf{v} \in T^*)$
A9: For any sequence σ of length n and any function l from this sequence to a set of labels smaller than n there exist $e_1, e_2 \in \sigma$ with $l(e_1) = l(e_2)$
A10: $|V| = k$, for a constant k and number $|V|$ of elements in V.
L11: $\exists path(length(path) > n$, where n is a parameter and preconditions $|w| > n + 1 \ldots$.

The meaning of each of these assumptions follows directly from the notions introduced above. For instance, A3 expresses formally that for each node N in $tree(w)$, the yield of N can be derived (in L) from the label of N.

Figure 2 sketches the proof plan of REGULAR, where the numbers A.. refer to the assumptions above. In order to produce new conclusions from assumptions or previously derived facts, we apply operators. The figure points to where the operators are applied but we do not show here their contents. These operators are complex and correspond to subproofs with preconditions and effects. The subgoals encountered are numbered and correspond to the following enumeration[6]:

1. $\exists k \forall w(|w| > k \rightarrow \exists \mathbf{u}, \mathbf{v}, x, A(\mathbf{u}, \mathbf{v}, x \in T^* \wedge A \in V \wedge S \stackrel{*}{\Rightarrow} \mathbf{u}A \wedge A \stackrel{*}{\Rightarrow} \mathbf{v}A \wedge A \stackrel{*}{\Rightarrow} x \wedge \mathbf{uv}x = w \wedge |\mathbf{v}| \neq 1)$

[5] Bold font indicates the pieces of the proof subject to adaptation later on.

[6] Again, bold symbols will be subject to change in the adaptation.

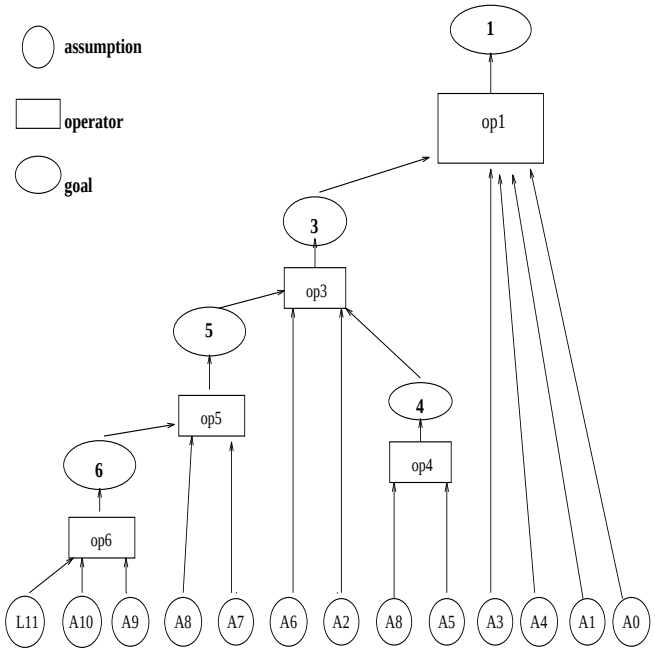

Figure 2: Proof plan of REGULAR

3. $\exists k \forall w \exists N_i, N_j, \mathbf{u}, \mathbf{v}, x(|w| > k \rightarrow N_i, N_j \in tree(w) \wedge \mathbf{sw}(N_i, N_j) = \mathbf{v} \wedge l(N_i) = l(N_j) \wedge yield(N_j) = x \wedge \mathbf{sw}(S, N_i) = \mathbf{u} \wedge yield(S) = \mathbf{u} \circ \mathbf{v} \circ x \wedge \mathbf{v} \neq \mathbf{e} \wedge \mathbf{u}, \mathbf{v}, x \in T^*)$
4. $\forall w, N_i \exists \mathbf{u}(sw(S, N_i) = \mathbf{u} \wedge \mathbf{u} \in T^*)$
5. $\exists k \forall w \exists N_i, N_j, \mathbf{v}(|w| > k \rightarrow N_i, N_j \in tree(w) \wedge \mathbf{v} \in T^* \wedge l(N_i) = l(N_j) \wedge \mathbf{sw}(N_i, N_j) = \mathbf{v} \wedge \mathbf{v} \neq \mathbf{e})$
6. $\exists k \forall w \exists N_i, N_j \exists path(|w| > k \rightarrow N_i, N_j \in tree(w) \wedge N_i, N_j \in path \wedge depth(N_i) < depth(N_j) \wedge l(N_i) = l(N_j) \wedge N_i, N_j \notin leaves \wedge N_i, N_j \neq root)$

The subgoal 6 expresses, for instance, that for words longer than a certain k there are two nodes, which are not leave nodes and not root nodes, that are labelled same.

The Target Theorem CFL

Now we give the target theorem (for CFL) and its assumptions indicating differences to the assumptions of REGULAR by bold font.

Theorem (CFL) (see (Hopcroft & Ullman 1979)): Let L be any context free language. Then there is a constant k (depending only on L) such that if $w \in L$ and $|w| \geq k$, then there are $u, y, v, z, x \in T^*$, such that $w = uvxzy$, $|uxy| \leq k$, $|vz| \geq 1$, and $\forall i(uv^i xz^i y \in L)$.

Again, it suffices to show

1'.: $\exists k \forall w(w \in L \wedge |w| > k \rightarrow \exists u, y, v, z, x, A(u, y, v, z, x \in T^* \wedge A \in V \wedge S \stackrel{*}{\Rightarrow} uAy \wedge A \stackrel{*}{\Rightarrow} vAz \wedge A \stackrel{*}{\Rightarrow} x \wedge uvxzy = w \wedge |vz| \geq 1))$.

Assumptions, that can be used in the proof, are

A0: $\forall x(x \neq e \rightarrow |x| \neq 1$
A1: $yield(S) = w$
A2": $\mathbf{cf}(w) \rightarrow \forall X, N, \mathbf{a}, \mathbf{b}, c\ (\mathbf{swp}(X, N) = (\mathbf{ab}) \wedge yield(N) = c \rightarrow yield(X) = \mathbf{a} \circ c \circ \mathbf{b})$
A3: $\forall N, x\ (N \in tree(w) \wedge yield(N) = x \rightarrow l(N) \stackrel{*}{\Rightarrow} x)$
A4": $cf(w) \rightarrow \forall X, N_j, \mathbf{v}, \mathbf{z}\ (\mathbf{swp}(X, N_j) = (\mathbf{v}, \mathbf{z}) \rightarrow l(X) \stackrel{*}{\Rightarrow} \mathbf{v} \circ l(N_j) \circ \mathbf{z})$
A5: $\forall N_i \exists path\ (N_i \in tree(w) \rightarrow S, N_i \in path)$
A6: $\forall N_i \exists x\ (N_i \in tree(w) \rightarrow yield(N_i) = x \wedge x \in T^*)$
A7': $\forall N_i, N_j \forall path\ (N_i, N_i \in path \wedge N_i \neq N_j \rightarrow \mathbf{swp}(N_i, N_j) \neq (\mathbf{e}, \mathbf{e}))$
A8': $\forall N_i, N_j \forall path \exists \mathbf{v}, \mathbf{z}\ (N_i, N_j \in path \wedge depth(N_i) < depth(N_j) \wedge \neg(leaf(N_i) \wedge \neg leaf(N_j)) \rightarrow \mathbf{swp}(N_i, N_j) = (\mathbf{v}, \mathbf{z}) \wedge v, z \in T^*)$
A9: For any sequence σ of length n and any function l from this sequence to a set of labels smaller than n there exist $e_1, e_2 \in \sigma$ with $l(e_1) = l(e_2)$
A10: $|V| = k$, for a constant k.
In the CFL case we do not know about a lemma associated with L11.

Further assumptions about parse trees of words in context free languages will be necessary to establish that under certain conditions $\exists path(length(path(w)) > n)$ for a parameter n.

Analogy-Driven Plan Construction

According to the analogy procedure in section 2, the source proof plan is changed in order to obtain a correct target proof plan, i.e., one with verified operators, i.e., correct subproofs, and no open goals. We show the instantiated procedure adapting the proof of REGULAR to the new proof of CFL, based on the reformulated assumptions as shown above.

1. Terminate if the goal statement is satisfied in the current state.
2. If the source case is exhausted, then base-level plan for the remaining goal L11.
3. Get a new action from the source case and apply to it the current set of reformulations.
4. Let p be the goal expanded or the assumption used in the source case.
5. Get the reformulation R of p. Associating corresponding assumptions and lemmata by the user improves the efficiency of finding reformulations. The following term mappings are found to establish these matches:

M1 The mapping M1 transfers A2, A4, A7, and A8 to A2', A4', A7', and A8' by replacing sw by swp, replacing each term t that occurs in as a rhs of an equation where the lhs is $sw(\ldots)$ by a pair of terms t_1, t_2 of the same type (e.g., $sw(N_i, N_j) = v \Rightarrow swp(N_i, N_j) = (v, z)$), and replacing $t \in X$ by $t_1, t_2 \in X$. M1 is a common generalizing reformulation.

M2 The additional mapping M2 transfers A2' and A4' to A2" and A4" by replacing terms $(t_1, t_2) \circ t$ by $t_1 \circ t \circ t_2$.

M3 For A2" and A4" we need the additional mapping $regular \Rightarrow cf$ in order to match their preconditions.

A0, A1, A3, A5, A6, A9, and A10 stay unchanged.
These reformulations can be user-supplied or even found automatically.
Add M1, M2, and M3 respectively to the set of reformulations.

6. Link the new p node to the source case. Advance case.
7. Get the relevant operator from the proof plan of REGULAR and apply the reformulations.
8. Restructuring splits the reformulated op1 into op1' and op2'. The effect of op2' is 2':
$\exists k \forall w (w \in L \wedge |w| > k \rightarrow \exists u, y, v, z, x, A (u, y, v, z, x \in T^* \wedge A \in V \wedge S \stackrel{*}{\Rightarrow} uAy \wedge A \stackrel{*}{\Rightarrow} vAz \wedge A \stackrel{*}{\Rightarrow} x \wedge uvxzy = w \wedge (v, z) \neq (e, e))$.
from op1 in the source plan.
9. Check if the operator is verifiable: op1' cannot be verified, since A0 cannot be applied to $(v, z) \neq (e, e)$.
   - Find another subproof for the goal 1', taking 2' as a precondition. (Using A0, but additionally using the pair axiom $(u, z) \neq (e, e) \rightarrow u \neq e \vee z \neq e$, a tiny subproof can be found by a theorem prover.)
   - Go to step 1.
10. Add to the set of pending goals all the preconditions of the operator that are not satisfied in the current state. Add to the set of assumptions the effects of operators applied forward.
11. Link the new operator node to the source case. Advance case.
12. Go to step 1.

Figure 3 shows the resulting proof plan of CFL, where we marked the nodes that are new and reformulated with respect to the proof plan of REGULAR, shown in figure 2.

Introducing new subproofs and replacing assumptions by small subproofs is typical in theorem proving by analogy. The complexity of the proofs to be done from scratch using analogy, is usually smaller by orders of magnitude compared with the whole proof.

4 Conclusions

We have shown how the analogical replay of Veloso (Veloso 1992) is extended by forward and backward chaining and by reformulating and restructuring goals, assumptions, and operators. The examined analogy-driven proof plan construction has actually shown how important it is to incorporate the restructuring of proofs into theorem proving by analogy since this technique yields small subproofs which, if not transferred by analogy, can be proven automatically.

Our method can be used for analogy-driven theorem proving and is not restricted to the particular example (see (Melis 1993)). The evaluation of our method is qualitative: The example presented is a *real* one, showing a *real* impact on automated theorem proving: Proving the Pumping Lemma for context free languages automatically is still beyond the capabilities of the current automated theorem provers. However, by analogy-driven proof-plan construction from the proof

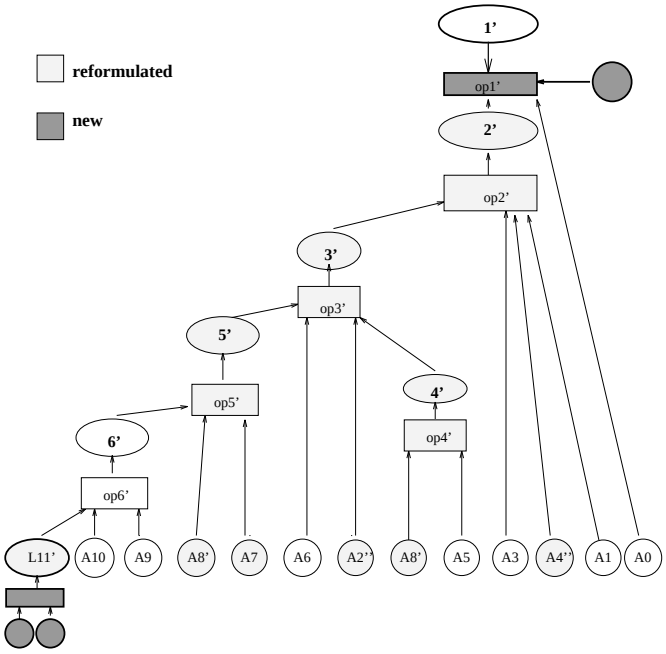

Figure 3: Proof plan of CFL

plan of the Pumping Lemma for regular languages this target proof became tractable, since only small new subproofs are to prove automatically from the target assumptions. Of course, a limitation of the method is the need of a source proof plan. For a discussion of the appropriateness of our method see (Melis 1993; 1994b). To determine the class of problems that can be solved easier or at all by this method is an open research problem.

References

Bledsoe, W. 1986. The use of analogy in automatic proof discovery. Tech.Rep. AI-158-86, Microelectronics and Computer Technology Corporation, Austin, TX.

Boyer, R., and Moore, J. 1988. *A Computational Logic Handbook*. San Diego: Academic Press.

Hadamard, J. 1945. *The Psychology of Invention in the Mathematical Field*. Princeton: Princeton Univ. Press.

Hammond, K. J. 1986. *Case-based Planning: An Integrated Theory of Planning, Learning and Memory*. Ph.D. Dissertation, Yale University.

Hopcroft, J., and Ullman, J. 1979. *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley.

Kambhampati, S., and Hendler, J. A. 1992. A validation based theory of plan modification and reuse. *Artificial Intelligence* 55(2-3):193–258.

Kling, R. 1971. A paradigm for reasoning by analogy. *Artificial Intelligence* 2:147–178.

Lewis, H., and Papadimitriou, C. 1981. *Elements of the Theory of Computation*. Prentice Hall.

McCune, W. 1990. Otter 2.0 users guide. Technical Report ANL-90/9, Argonne National Laboratory, Maths and CS Division, Argonne, Illinois.

Melis, E. 1993. Change of representation in theorem proving by analogy. SEKI-Report SR-93-07, Universität des Saarlandes, Saarbrücken.

Melis, E. 1994a. Decomposition techniques and their applications. In *Proc. AIMSA'94*, 15–25.

Melis, E. 1994b. How mathematicians prove theorems. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, 624–628.

Munyer, J. 1981. *Analogy as a Means of Discovery in Problem Solving and Learning.* Ph.D. Dissertation, University of California, Santa Cruz.

Owen, S. 1990. *Analogy for Automated Reasoning.* Academic Press.

Polya, G. 1954. *Mathematics and Plausible Reasoning.* NJ: Princeton University Press.

Polya, G. 1957. *How to Solve it.* New York: 2nd ed. Doubleday.

van der Waerden, B. 1964. Wie der Beweis der Vermutung von Baudet gefunden wurde. *Abh.Math.Sem.Univ.Hamburg* 28.

Veloso, M. 1992. *Learning by Analogical Reasoning in General Problem Solving.* Ph.D. Dissertation, Carnegie Mellon University, CMU, Pittsburgh, USA. CMU-CS-92-174.

Wos, L. 1988. *Automated Reasoning: 33 Basic Research Problems.* Englewood Cliffs: Prentice-Hall.