Patching Proofs for Reuse (Extended Abstract)

Thomas Kolbe and Christoph Walther

FB Informatik, TH Darmstadt, Alexanderstr. 10, D-64283 Darmstadt, Germany.
Email: {kolbe|walther}@inferenzsysteme.informatik.th-darmstadt.de

1 Introduction

We investigate the application of machine learning paradigms [2, 4, 3] in automated reasoning for improving a theorem prover by reusing previously computed *proofs* [7]. Assume that we have already computed a proof P of a conjecture

$$\varphi := \big(\forall u\ \mathsf{plus}(\mathsf{sum}(x), \mathsf{sum}(u)) \equiv \mathsf{sum}(\mathsf{append}(x, u))\big) \\ \rightarrow \mathsf{plus}(\mathsf{sum}(\mathsf{add}(n, x)), \mathsf{sum}(y)) \equiv \mathsf{sum}(\mathsf{append}(\mathsf{add}(n, x), y))$$

from a set of axioms AX. The schematic conjecture $\Phi := \mathrm{H} \rightarrow \mathrm{C}\ :=$

$$\big(\forall u\ F(G(x), G(u)) \equiv G(H(x, u))\big) \rightarrow F(G(D(n, x)), G(y)) \equiv G(H(D(n, x), y))$$

is obtained from φ via the generalization $\{\mathsf{plus} \mapsto F, \mathsf{sum} \mapsto G, \mathsf{append} \mapsto H, \mathsf{add} \mapsto D\}$ of function *symbols* $\mathsf{plus}, \mathsf{sum}, ...$ to function *variables* $F, G, ...$ In the same way a *schematic catch*, i.e. a set of schematic axioms $AX' = \{(1), (2), (3)\}$ is obtained from AX where e.g. (1) stems from the axiom $\mathsf{sum}(\mathsf{add}(n, x)) \equiv \mathsf{plus}(n, \mathsf{sum}(x))$. The generalization of P finally yields a schematic proof P' of Φ in which the schematic conclusion C is modified in a backward chaining style:

$$G(D(n, x)) \equiv F(n, G(x)) \quad (1)$$
$$H(D(n, x), y) \equiv D(n, H(x, y)) \quad (2)$$
$$F(F(x, y), z) \equiv F(x, F(y, z)) \quad (3)$$

$F(G(D(n, x)), G(y))$	$\equiv$	$G(H(D(n, x), y))$	C
$F(\underline{F(n, G(x))}, G(y))$	$\equiv$	$G(H(D(n, x), y))$	Replace (1)
$F(F(n, G(x)), G(y))$	$\equiv$	$G(\underline{D(n, H(x, y))})$	Replace (2)
$F(F(n, G(x)), G(y))$	$\equiv$	$\underline{F(n, G(H(x, y)))}$	Replace (1)
$F(F(n, G(x)), G(y))$	$\equiv$	$F(n, \underline{F(G(x), G(y))})$	Replace (H)
$\underline{F(n, F(G(x), G(y)))}$	$\equiv$	$F(n, F(G(x), G(y)))$	Replace (3)
	$\underline{\mathrm{TRUE}}$		Reflexivity

The key idea of our reuse procedure is to *instantiate* such a schematic proof with a second-order substitution π obtained by matching Φ with a new conjecture ψ which is (formally) similar to φ, i.e. $\psi = \pi(\Phi)$. As long as the matcher π only replaces function variables with function symbols, the instantiated schematic proof $\pi(P')$ is a proof of ψ from the axioms $\pi(AX')$ because the structure of P' is preserved. However, the success of the method is limited by such a restriction. Therefore function variables are also replaced using general second-order substitutions[1] like $\pi := \{F/w_2,\ G/\mathsf{minus}(w_1, w_1),\ H/\mathsf{plus}(w_1, w_2),\ D/\mathsf{succ}(w_2)\}$ obtained by matching Φ with the new conjecture $\psi := \pi(\Phi) = \pi(\mathrm{H} \rightarrow \mathrm{C}) =$

[1] A second-order substitution replaces a n-ary function variable V with a (first-order)

$$\begin{aligned}&(\forall u\ \mathsf{minus}(u,u) \equiv \mathsf{minus}(\mathsf{plus}(x,u),\mathsf{plus}(x,u)))\\ &\to \mathsf{minus}(y,y) \equiv \mathsf{minus}(\mathsf{plus}(\mathsf{succ}(x),y),\mathsf{plus}(\mathsf{succ}(x),y)).\end{aligned}$$

AX' is instantiated yielding the set of axioms $\pi(AX') = \{\pi(1), \pi(2), \pi(3)\}$:

$$\begin{aligned}\mathsf{minus}(\mathsf{succ}(x),\mathsf{succ}(x)) &\equiv \mathsf{minus}(x,x) && \pi(1)\\ \mathsf{plus}(\mathsf{succ}(x),y) &\equiv \mathsf{succ}(\mathsf{plus}(x,y)) && \pi(2)\\ z &\equiv z && \pi(3)\end{aligned}$$

If the proof P shall be reused for proving ψ from the set of axioms $\pi(AX')$ by instantiating the schematic proof P' with π, we obtain $\pi(P')$ as

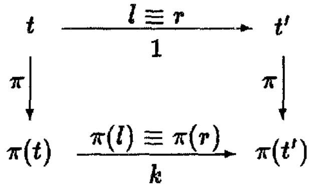
$\mathsf{minus}(y,y) \equiv \mathsf{minus}(\mathsf{plus}(\mathsf{succ}(x),y),\mathsf{plus}(\mathsf{succ}(x),y))$		π(C)
$\mathsf{minus}(y,y) \equiv \mathsf{minus}(\mathsf{plus}(\mathsf{succ}(x),y),\mathsf{plus}(\mathsf{succ}(x),y))$		Replace ($\pi(1)$)
$\mathsf{minus}(y,y) \equiv \mathsf{minus}(\underline{\mathsf{succ}(\mathsf{plus}(x,y))},\underline{\mathsf{succ}(\mathsf{plus}(x,y))})$		Replace ($\pi(2)$)
$\mathsf{minus}(y,y) \equiv \underline{\mathsf{minus}(\mathsf{plus}(x,y),\mathsf{plus}(x,y))}$		Replace ($\pi(1)$)
$\mathsf{minus}(y,y) \equiv \underline{\mathsf{minus}(y,y)}$		Replace (π(H))
$\underline{\mathsf{minus}(y,y)} \equiv \mathsf{minus}(y,y)$		Replace ($\pi(3)$)
TRUE		Reflexivity

But $\pi(P')$ is *not* a proof: Although each statement is implied by the statement in the line below, the *justifications* of the inference steps are not valid. E.g. the first *replace*($\pi(1)$)-step is illegal because the *position* of the replacement (the former first argument of F) does not exist in π(C). Also the *replace*($\pi(2)$)-step is illegal, as it actually consists of *two* replacements which have to be performed separately at different positions. Finally, the *replace*($\pi(3)$)-step is redundant and should be omitted. Thus $\pi(P')$ has to be *patched* for obtaining a proof of ψ.

Such a machine-found proof can be processed subsequently, e.g. by translating it into natural language to obtain a proof similar to those found in mathematical textbooks [5]. Furthermore proofs can be worked up for planning or synthesis tasks if plans or programs should be extracted form proofs [1]. These applications require a *specific* proof, i.e. it is not enough to know that *some* proof exists.

2 An Algorithm for Patching Proofs

We first illustrate the patching of a single replacement step: Let t be a schematic term (containing function variables) which can be modified by *one* replacement step with a certain schematic equation $l \equiv r$ at a certain position p (i.e. $t|_p = l$) yielding another schematic term $t' = t[p \leftarrow r]$ as the result. The function call *patch_positions*(t, p, π) yields for an arbitrary second-order substitution π a list of positions $[p_1, ..., p_k]$ such that the instance $\pi(t)$ can be modified by a (possibly empty) *sequence* of k replacement steps with the instantiated equation $\pi(l) \equiv \pi(r)$ at the positions $p_1, ..., p_k$ such that the instance $\pi(t')$ is obtained.



term where special *argument variables* $w_1, \ldots, w_n$ serve as the formal parameters of V. For instance π replaces the binary function variable D with the function symbol succ, where the first argument w_1 of D is ignored.

function *patch_positions* $(t,\ p,\ \pi)$: *list of positions in* $\pi(t)$
if $p = \epsilon$ **then return** $[\epsilon]$
else let $p =: ip'$; $t =: X(t_1, ..., t_n)$; $[p_1, ..., p_k] := patch\_positions(t_i, p', \pi)$
  if $X \in \mathrm{dom}(\pi)$ **then** $s := \pi(X)$; $[q_1, ..., q_m] := \{q \in \mathrm{Pos}(s) \mid s|_q = w_i\}$
    return $[q_1p_1, ..., q_1p_k, ..., q_mp_1, ..., q_mp_k]$
  else return $[ip_1, ..., ip_k]$ **fi fi**

Theorem 1. [6] *Let* t, l, r *be schematic terms,* p *a position in* t *and* π *a second-order substitution. If* $t|_p = l$ *then the call* $patch\_positions(t, p, \pi)$ *terminates yielding a list of positions* $[p_1, ..., p_k]$ *in* $\pi(t)$ *such that for* $i, j \in \{1, ..., k\}$
1) if $i \neq j$ *then there is no* $p \in \mathbb{N}^*$ *such that* $p_i = p_jp$ *or* $p_j = p_ip$,
2) $\pi(t)|_{p_j} = \pi(l)$ *and* $\pi(t)[p_1, ..., p_k \leftarrow \pi(r)] = \pi(t[p \leftarrow r])$.

The goal of a (schematic) proof is a so-called *sequent* $H \rightarrow C$ with a conjunction H of hypotheses each of which is of the form $\forall u^*\ t_1 \equiv t_2$ and a conclusion C of the form $s_1 \equiv s_2$. A *proof* of $H \rightarrow C$ (from a set of axioms AX) is a list $[S_0, j_1, S_1, j_2, \ldots, S_n]$ of sequents S_i (with $S_0 = H \rightarrow C$) and *justifications* j_i, where the latter contain the information how the next sequent is derived. A proof is constructed by applying the following inference rules,[2] where σ is a first-order substitution, p is a position in C and $m \in \{\text{"}AX\text{"}, \text{"}H\text{"}\}$:

Reflexivity $$\frac{}{[H \rightarrow t \equiv t]}$$

Replacement $$\frac{[H \rightarrow C[p \leftarrow \sigma(r)] \mid L]}{[H \rightarrow C[p \leftarrow \sigma(l)],\ \langle p, \sigma, u^*, l, r, m\rangle,\ H \rightarrow C[p \leftarrow \sigma(r)] \mid L]}$$

if either $\forall u^*\ l \equiv r\ \in AX$ and $m = \text{"}AX\text{"}$
or $\forall u^*\ l \equiv r\ \in H$, $\mathrm{dom}(\sigma) \subseteq u^*$ and $m = \text{"}H\text{"}$.

function *patch_proof* $(P',\ \pi)$: *proof*
if $P' = [H \rightarrow C]$ **then return** $[\pi(H) \rightarrow \pi(C)]$
else let $P' =: [H \rightarrow C,\ \langle p, \sigma, u^*, l, r, m\rangle,\ H \rightarrow C' \mid L]$
  $P_\pi := patch\_proof([H \rightarrow C' \mid L],\ \pi)$
  if $\pi(C) \neq \pi(C')$ **then** $[p_1, ..., p_k] := patch\_positions(C, p, \pi)$
    $\sigma_\pi := \{v/\pi(\sigma(v)) \mid v \in \mathrm{dom}(\sigma)\}$; $C_k := \pi(C')$
    for $j := k$ **downto** 1 **do** $C_{j-1} := C_j[p_j \leftarrow \sigma_\pi(\pi(l))]$
      $P_\pi := [\pi(H) \rightarrow C_{j-1},\ \langle p_j, \sigma_\pi, u^*, \pi(l), \pi(r), m\rangle \mid P_\pi]$ **od fi**
  return P_π **fi**

In a replacement step an *instance* $\sigma(l) \equiv \sigma(r)$ of an equation $l \equiv r$ is applied, but in the patched proof only (instances of) the equation $\pi(l) \equiv \pi(r)$ are available. However, we can use the first-order substitution $\sigma_\pi := \{v/\pi(\sigma(v)) \mid v \in \mathrm{dom}(\sigma)\}$ in *patch_proof* because $\pi(\sigma(u)) = \sigma_\pi(\pi(u))$ holds for each (schematic) term u.

Now we can compute $P_\pi := patch\_proof(P', \pi)$ to obtain a patched proof for

[2] Proofs can be extended to deal with arbitrary formulas instead of equations only if we define further inference rules. Then H may also contain additional conditions.

the conjecture $\psi = \pi(\mathrm{H}) \rightarrow \pi(\mathrm{C})$ from Section 1:

$$\begin{array}{lll}
\mathsf{minus}(y,y) \equiv \mathsf{minus}(\mathsf{plus}(\mathsf{succ}(x),y),\mathsf{plus}(\mathsf{succ}(x),y)) & & \pi(\mathrm{C}) \\
\mathsf{minus}(y,y) \equiv \mathsf{minus}(\underline{\mathsf{succ}(\mathsf{plus}(x,y))},\mathsf{plus}(\mathsf{succ}(x),y)) & & \text{Replace } (\pi(2)) \\
\mathsf{minus}(y,y) \equiv \mathsf{minus}(\mathsf{succ}(\mathsf{plus}(x,y)),\underline{\mathsf{succ}(\mathsf{plus}(x,y))}) & & \text{Replace } (\pi(2)) \\
\mathsf{minus}(y,y) \equiv \underline{\mathsf{minus}(\mathsf{plus}(x,y),\mathsf{plus}(x,y))} & & \text{Replace } (\pi(1)) \\
\mathsf{minus}(y,y) \equiv \underline{\mathsf{minus}(y,y)} & & \text{Replace } (\pi(\mathrm{H})) \\
\underline{\mathbf{TRUE}} & & \text{Reflexivity}
\end{array}$$

Compared to the schematic proof P from Section 1, the first *replace*(1)-step is eliminated while the *replace*(2)-step is doubled. The test $\pi(C) \neq \pi(C')$ in *patch_proof* is merely an optimization to avoid redundant steps like *replace*($\pi(3)$), cf. Section 1.

Theorem 2. [6] *Let P' be a proof of the sequent $H \rightarrow C$ from the set of axioms AX. Then for each second-order substitution π, the call patch_proof(P', π) terminates and yields a proof P_π of $\pi(H) \rightarrow \pi(C)$ from $\pi(AX)$.*

Summing up, we have presented an algorithm that constructs a proof for the instantiated conjecture from a schematic proof of a schematic conjecture and a second-order substitution. This allows us to exploit the full flexibility of second-order instantiations for the reuse procedure developed in [7]. Thus more conjectures are (formally) similar than by just instantiating function variables with function symbols, i.e. the applicability of a schematic catch is increased. Furthermore the obtained proofs may be more flexible, i.e. the reusability of a schematic catch is increased.

Acknowledgements. This work was supported under grants no. Wa652/4-1,2,3 by the DFG within the focus program "Deduktion". We thank Jürgen Brauburger, Stefan Gerberding, Jürgen Giesl and Martin Protzen for helpful comments and discussions.

References

1. J. L. Bates and R. L. Constable. Proofs as Programs. *ACM Transactions on Programming Languages and Systems*, 7(1):113–136, 1985.
2. T. Ellman. Explanation-Based Learning: A Survey of Programs and Perspectives. *ACM Computing Surveys*, 21(2):163–221, 1989.
3. F. Giunchiglia and T. Walsh. A Theory of Abstraction. *Artificial Intelligence*, 57:323–389, 1992.
4. R. P. Hall. Computational Approaches to Analogical Reasoning: A Comparative Analysis. *Artificial Intelligence*, 39:39–120, 1989.
5. X. Huang. PROVERB: A System Explanining Machine-Found Proofs. In *Proc. of 16th Annual Conference of the Cognitive Science Society*, Atlanta, Georgia, 1994.
6. T. Kolbe and C. Walther. Patching proofs for reuse. Technical report, Technische Hochschule Darmstadt, 1994.
7. T. Kolbe and C. Walther. Reusing Proofs. In A. Cohn, editor, *Proceedings of the 11th European Conference on Artificial Intelligence, Amsterdam*, pages 80–84. John Wiley & Sons, Ltd., 1994.