HOL proofs of two theorems about unification

Marek Materzok

22 November 2007

Useful theorems

Values constructed by different constructors are different.

```
# let term_distinct = prove_constructors_distinct term_rec
val term_distinct : thm =
  |- (!a a'. ~(Variable a = Constant a')) /\
     (!a a0' a1'. ~(Variable a = Apply a0' a1')) /\
     (!a a0' a1'. ~(Constant a = Apply a0' a1'))
```

Constructors are injective.

```
# let term_injective = prove_constructors_injective term_re
val term_injective : thm =
  |- (!a a'. Variable a = Variable a' <=> a = a') /\
     (!a a'. Constant a = Constant a' <=> a = a') /\
     (!a0 a1 a0' a1'. Apply a0 a1 = Apply a0' a1' <=>
                      a0 = a0' /\ a1 = a1')
```

Useful theorems

One can do case analysis on constructors. (like induction, but without inductive hypotheses – simpler)

```
# let term_cases = prove_cases_thm term_ind;;
val term_cases : thm =
  |- !x. (?a. x = Variable a)
         (?a. x = Constant a)
         (?a0 a1. x = Apply a0 a1)
```

Theorems for handling packed substitutions:

```
let subst_cases = prove_cases_thm subst_ind;;
val subst_cases : thm = |- !x. ?a. x = Makesubst a
let subst_injective = prove_constructors_injective subst_re
val subst_injective : thm =
    |- !a a'. Makesubst a = Makesubst a' <=> a = a'
```

First theorem

Theorem

Let t be a term such that $x \in FV(t)$ and $t \neq x$. Then the equation $x = t$ has no unifiers.

Proof.

Let w be a weight function defined as follows:

$$\begin{aligned} w(x) &= 1 \\ w(c) &= 1 \\ w(t_1 t_2) &= w(t_1) + w(t_2) + 1 \end{aligned}$$

We see that $t = t_1 t_2$ for some terms t_1 and t_2, so $w(x) < w(t)$. Assume that σ is an unifier of $x = t$. Then we have $w(x\sigma) < w(t\sigma)$, so $x\sigma \neq t\sigma$, which is a contradiction. $\square$

Technique: simplifying assumptions

Sometimes rewriting allows to get simpler, more useful assumptions. The `DISCH_TAC` tactic is a shorthand for `DISCH_THEN ASSUME_TAC`, so it's easy to do some rewriting on an assumption.

```
Goal: `~(?subst. isunifier subst
    (Addequation Emptysystem (Variable n) (Apply a0 a1)))`
e (DISCH_THEN (CHOOSE_TAC o
    REWRITE_RULE [unifierdef;appltermdef]));;
Added assumption: `applterm subst (Variable n) =
    Apply (applterm subst a0) (applterm subst a1)`
```

Technique: subgoals

Subgoals allow to do forward reasoning easily. Proven subgoal is added to the list of assumptions.

```
e (SUBGOAL_TAC ""
    `~(termweight (applterm subst (Variable n)) =
       termweight (applterm subst (Apply a0 a1)))`
   [ASM_MESON_TAC[13;15;ARITH_RULE `a<b ==> ~(a=b)`]]);;
```

Lemma 1

Lemma

For all t, $w(t) > 0$.

Proof.

Straightforward case analysis. □

```
g `!t. termweight t > 0`;;
e GEN_TAC;;
e (STRUCT_CASES_TAC (SPEC `t:term` term_cases));;
e (REWRITE_TAC [termweightdef] THEN ARITH_TAC);;
e (REWRITE_TAC [termweightdef] THEN ARITH_TAC);;
e (REWRITE_TAC [termweightdef] THEN ARITH_TAC);;
let l5 = top_thm();;
```

Technique: proving arithmetic inequalities

Tactic ARITH_TAC attempts to prove a true sentence about natural numbers. The sentence may have the form of an implication – the prover will then use the left hand side as an assumption.

For example, the following sentence can be proved with ARITH_TAC:

```
‘termweight (applterm s a1) >= termweight a1
 ==> termweight (applterm s a0) >=
     termweight (applterm s (Variable n)) +
        termweight a0 - 1
 ==> termweight (applterm s a0) +
        termweight (applterm s a1) + 1 >=
     termweight (applterm s (Variable n)) +
     (termweight a0 + termweight a1 + 1) - 1‘
```

Lemma 2

Lemma

For all substitutions σ and terms t, $w(t\sigma) >= w(t)$.

Proof.

Proof by structural induction over t.

- $t = x$. By definition $w(x) = 1$, by Lemma 1 $w(x\sigma) > 0$, so $w(x\sigma) >= w(x)$.
- $t = c$. We have $w(c\sigma) = w(c)$.
- $t = t_1 t_2$. We have $w(t\sigma) = w(t_1\sigma) + w(t_2\sigma) + 1 >= w(t_1) + w(t_2) + 1 = w(t)$.

$\square$

Technique: undischarge

Because `ARITH_TAC` can't use the assumption list, assumptions required for proving the goal must be moved to the goal. That's exactly what `UNDISCH_TAC` does.

```
Goal: `termweight (applterm s (Variable a)) >=
       termweight (Variable a)`
e (ASSUME_TAC (SPEC `applterm s (Variable a)` l1));;
e (UNDISCH_TAC
    `termweight (applterm s (Variable a)) > 0`);;
Goal: `termweight (applterm s (Variable a)) > 0
   ==> termweight (applterm s (Variable a)) >=
       termweight (Variable a)`
```

Lemma 3

Lemma

Let t be a term. Then $w(t) > 1$ iff there are terms t_1, t_2 such that $t = t_1 t_2$.

Proof.

Simple case analysis.

- $t = x$ or $t = c$. Then both left and right side is false.
- $t = t_1 t_2$. By Lemma 1 and the definition of w, left side is true, right side is of course also true.

$\square$

Technique: case analysis for types

With a cases theorem about some type, one can split the goal to several subgoals.

```
val term_cases : thm =
  |- !x. (?a. x = Variable a)
         (?a. x = Constant a)
         (?a0 a1. x = Apply a0 a1)
Goal: ‘termweight t > 1 <=> (?t1 t2. t = Apply t1 t2)‘
e (STRUCT_CASES_TAC (SPEC ‘t:term‘ term_cases));;
We have three new goals now.
```

Lemma 4

Lemma

If $x \in FV(t)$, then for all substitutions σ we have $w(t\sigma) \geq w(x\sigma) + w(t) - 1$.

Proof.

Induction over t.

- $t = y$. Because $x \in FV(y)$, $x = y$, so trivial.
- $t = c$. Both sides are equal to 1.
- $t = t_1 t_2$. Suppose, without loss of generality, that $x \in FV(t_1)$. So we have $w(t_1\sigma) \geq w(x\sigma) + w(t_1) - 1$. By Lemma 2 we have $w(t_2\sigma) \geq w(t_2)$. Thus,

$$\begin{aligned} w((t_1 t_2)\sigma) &= w(t_1\sigma) + w(t_2\sigma) + 1 \\ &\geq w(x\sigma) + w(t_1) - 1 + w(t_2) \\ &= w(x\sigma) + w(t) - 1 \end{aligned}$$

$\square$

Technique: case analysis

Using a disjunction one can create new goals identical to current goal, but with different assumptions.
Here I use `SUBGOAL_THEN`, which allows to do something else with a subgoal than assuming it.

```
e (SUBGOAL_THEN
    `varoccursinterm n a0  varoccursinterm n a1`
    DISJ_CASES_TAC);;
e (ASM_MESON_TAC[varoccursintermdef]);;
We now have two goals.
```

Lemma 5

Lemma

Let $x \in FV(t)$ *and* $w(t) > 1$*. Then* $w(x\sigma) < w(t\sigma)$*.*

Proof.

From Lemma 4 we have $w(t\sigma) \geq w(x\sigma) + w(t) - 1$. So, because $w(t) > 1$, $w(t\sigma) > w(x\sigma)$. $\square$

Second theorem

Theorem

Let t be a term such that $x \notin FV(t)$. Then $\sigma = [x/t]$ is the mgu of $x = t$.

Proof.

Obviously, σ is an unifier of $x = t$. Let τ be any unifier of $x = t$. Let τ' be a substitution such that $x\tau' = x$ and for all $y \neq x$ $y\tau' = y\tau$. I'll show that $\sigma\tau' = \tau$.
Let y be a variable different than x. Then obviously $y\sigma\tau' = y\tau' = y\tau$. It remains to show that $x\sigma\tau' = x\tau$. By definition of σ we have $x\sigma\tau' = t\tau'$. Because $x \notin FV(t)$, we have $t\tau' = t\tau$, and because τ is a unifier of $x = t$, then $t\tau = x\tau$, which finishes the proof. □

Technique: using tautologies

Sometimes some tautology is needed to push the proof forward. Tautologies can be easily proven with `TAUT`.

```
e (DISJ_CASES_TAC (TAUT `n'=n:num \/ ~(n'=n:num)`));;
```

Lemma 6

Lemma

Suppose that $x \notin FV(t)$. Then $t[x/u] = t$ for all u.

Proof.

Structural induction over t.

- $t = y$. Because $x \notin y$ we have $x \neq y$, so $y[x/u] = y$.
- $t = c$. Trivial.
- $t = t_1 t_2$. Then $x \notin t_1$ and $x \notin t_2$, so $(t_1 t_2)[x/u] = t_1[x/u]t_2[x/u] = t_1 t_2$.

$\square$

Lemma 7

Lemma

Let t be a term such that $x \notin FV(t)$. Then $\sigma = [x/t]$ is an unifier of $x = t$.

Proof.

Lemma 6. $\square$

Lemma 8

Lemma

Assume that $x \notin t$. Let σ be any substitution, let τ be a substitution such that $x\tau = x$ and $y\tau = y\sigma$ for $y \neq x$. Then $t\sigma = t\tau$.

Proof.

Structural induction over t.

- $t = y$. Because $x \notin y$ we have $x \neq y$. So $y\sigma = y\tau$ by definition of τ.
- $t = c$. Trivial.
- $t = t_1 t_2$. Easy.

$\square$