Implicational Rewriting Tactics

Vincent Aravantinos, Sofiene Tahar

vincent.aravantinos@fortiss.org
http://www.fortiss.org/en

fortiss
Munich, Germany

tahar@ece.concordia.ca
http://hvg.ece.concordia.ca

Concordia University
Hardware Verification Group
Faculty of Engineering and Computer Science
Montréal, Canada

July 17th, 2014

Outline

This work in one slide

- Overall objective: new tactics to increase automation
- More precisely: we identify some situations where we usually need to introduce a subgoal manually
- And we define tactics to automatize this subgoal introduction
- Main benefit: time saved

A concrete example

Objective:

- Prove Cauchy-Schwarz inequality
- For any complex inner-space
- In HOL Light

Initial goal (Cauchy-Schwarz)

$$\begin{aligned}&\forall s\ \langle.|.\rangle\ x\ y.\\ &\quad \mathit{isInnerSpace}\ (s, \langle.|.\rangle)\ \wedge x \in s \wedge y \in s\\ &\qquad \Rightarrow \mathit{norm}(\langle x|y\rangle)^2\\ &\qquad\quad \leq \mathit{realOfComplex}(\langle x|x\rangle).\mathit{realOfComplex}(\langle y|y\rangle)\end{aligned}$$

A concrete example

Proof without our tactics (1/3):

```
e(REPEAT STRIP_TAC
THEN Pa.SUBGOAL_THEN
  "norm (inprod x y) pow 2 <=
    real_of_complex (inprod x x * (inprod (inprod x y / inprod x x % x)
      (inprod x y / inprod x x % x) + inprod (y - inprod x y
      / inprod x x % x) (y - inprod x y / inprod x x % x)))"
      ASSUME_TAC
  THENL [
    Pa.SUBGOAL_THEN
      "norm (inprod x y) pow 2 <=
        real_of_complex (Cx (norm (inprod x y) pow 2) * (inprod x x * inprod x x)
          * Cx (inv (norm (inprod x x) pow 2)) + inprod x x
            * inprod (y - (inprod x y * inv (inprod x x)) % x) (y - (inprod x y * inv (inprod x x)) % x))"
    ASSUME_TAC
    THENL [
      Pa.SUBGOAL_THEN
      "norm (inprod x y) pow 2 = norm (inprod x y) pow 2
        * norm (inprod x x) pow 2 * inv (norm (inprod x x) pow 2)"
      ASSUME_TAC
      THENL [
        Pa.ASM_CASES_TAC "(norm (inprod x x) pow 2 = &0)"
        THENL [
          POP_ASSUM MP_TAC THEN ASM_SIMP_TAC[REAL_MUL_LZERO;REAL_MUL_RZERO]
          THEN SIMP_TAC[REAL_POW_EQ_0]
          THEN ASM_MESON_TAC[COMPLEX_NORM_ZERO;INSPACE_ZERO_EQ;INSPACE_LZERO];
          ASM_SIMP_TAC[REAL_MUL_RINV;REAL_MUL_RID];
        ];
        Pa.SUBGOAL_THEN
          "real (Cx (norm (inprod x y) pow 2)) /\ real (inprod x x)
            /\ real (Cx (inv (norm (inprod x x) pow 2)))
```

A concrete example

Proof without our tactics (2/3):

```
          /\ real (Cx (norm (inv (inprod x x) pow 2)))
          /\ real (inprod (y - (inprod x y * inv
          (inprod x x)) % x) (y - (inprod x y * inv (inprod x x)) % x))
          /\ real (Cx (norm (inprod x y) pow 2) * (inprod x x * inprod x x) *
            Cx (inv (norm (inprod x x) pow 2)))
          /\ real (inprod x x * inprod (y - (inprod x y * inv (inprod x x)) % x)
            (y - (inprod x y * inv (inprod x x)) % x))
          /\ real_of_complex (inprod x x) = norm (inprod x x)
          /\ real_of_complex (inprod (y - (inprod x y * inv (inprod x x)) % x)
            (y - (inprod x y * inv (inprod x x)) % x))
            = norm (inprod (y - (inprod x y * inv (inprod x x)) % x)
              (y - (inprod x y * inv (inprod x x)) % x))"
        STRIP_ASSUME_TAC
      THENL [
        ASM_MESON_TAC[REAL_CX;INSPACE_SELF_REAL;CFUN_SUBSPACE_SUB;
          CFUN_SUBSPACE_SMUL;INSPACE_IS_SUBSPACE;REAL_MUL;INSPACE_SELF_NORM];
        ASM_SIMP_TAC[REAL_OF_COMPLEX_ADD;REAL_OF_COMPLEX_MUL;REAL_MUL;REAL_OF_COMPLEX_CX;GSYM REAL_POW_2
        THEN MATCH_MP_TAC (REAL_ARITH `x = y /\ &0 <= z ==> x <= y + z`)
        THEN ASSUM_LIST (REWRITE_TAC o map GSYM)
        THEN MATCH_MP_TAC REAL_LE_MUL THEN ASM_REWRITE_TAC[NORM_POS_LE]
      ];
    ];
    ASM_REWRITE_TAC[complex_div;COMPLEX_ADD_LDISTRIB]
    THEN Pa.SUBGOAL_THEN "(inprod x y * inv (inprod x x)) % x IN s" ASSUME_TAC
    THENL [
      ASM_MESON_TAC[INSPACE_IS_SUBSPACE;CFUN_SUBSPACE_SMUL];
      Pa.SUBGOAL_THEN "inprod ((inprod x y * inv (inprod x x)) % x) ((inprod x y * inv (inprod x x)) % x
        = cnj (inprod x y * inv (inprod x x)) * (inprod x y * inv (inprod x x)) * inprod x x"
        (fun x -> REWRITE_TAC[x])
      THENL [
        ASM_MESON_TAC[INSPACE_LSMUL;INSPACE_RSMUL];
```

A concrete example

Proof without our tactics (3/3):

```
        ASM_REWRITE_TAC[CNJ_MUL;CNJ_INV;
          Pa.COMPLEX_FIELD "x * (cnj y * inv z) * (y * inv x) * x = (y * cnj y) * (x * x) * (inv (x * z
          COMPLEX_MUL_CNJ;GSYM CX_POW;GSYM CX_INV]
      ];
    ];
  ];
  POP_ASSUM MP_TAC
  THEN Pa.SUBGOAL_THEN
    "(inprod x y / inprod x x % x) IN s
      /\ (y - inprod x y / inprod x x % x) IN s
      /\ are_orthogonal (s,inprod) (inprod x y / inprod x x % x) (y - inprod x y / inprod x x % x)"
    STRIP_ASSUME_TAC
  THENL [
    ASM_MESON_TAC[INSPACE_IS_SUBSPACE;CFUN_SUBSPACE_SMUL;CFUN_SUBSPACE_SUB;
      REWRITE_RULE[LET_DEFS]ARE_ORTHOGONAL_DECOMPOSITION;ARE_ORTHOGONAL_LSCALAR];
    ASM_MESON_TAC[];
  ]
];
```

→ Proof is 3 slides long

→ Most goals are *not* meaningful

A concrete example

Proof with our tactics:

```
(IMP_REWRITE_TAC[GSYM REAL_OF_COMPLEX_MUL;INSPACE_SELF_REAL]
THEN TARGET_REWRITE_TAC[REWRITE_RULE[LET_DEFS] ARE_ORTHOGONAL_DECOMPOSITION]
  ARE_ORTHOGONAL_INSPACE_SELF_ADD
THEN SEQ_IMP_REWRITE_TAC[REWRITE_RULE[LET_DEFS] ARE_ORTHOGONAL_DECOMPOSITION;
  ARE_ORTHOGONAL_LSCALAR;CFUN_SUBSPACE_SUB;INSPACE_RSMUL;CFUN_SUBSPACE_SMUL;
  INSPACE_IS_SUBSPACE;INSPACE_LSMUL;COMPLEX_ADD_LDISTRIB]
THEN REWRITE_TAC[complex_div;CNJ_MUL;CNJ_INV;COMPLEX_MUL_CNJ;GSYM CX_POW;
  Pa.COMPLEX_FIELD "x*(y*inv x)*(z*inv t)*x = (y*z)*((x*x)*inv(x*t))";
  GSYM CX_INV]
THEN IMP_REWRITE_TAC[REAL_OF_COMPLEX_ADD;REAL_CX;REAL_OF_COMPLEX_MUL;
  REAL_OF_COMPLEX_CX;REAL_MUL;INSPACE_SELF_REAL;CFUN_SUBSPACE_SUB;
  CFUN_SUBSPACE_SMUL;INSPACE_IS_SUBSPACE;GSYM INSPACE_SELF_NORM;
  GSYM REAL_POW_2;REAL_ARITH `x = y /\ &0 <= z ==> x <= y + z`;REAL_LE_MUL;
  NORM_POS_LE]
THEN CASE_REWRITE_TAC REAL_MUL_RINV
THEN IMP_REWRITE_TAC[REAL_MUL_RID;REAL_MUL_LZERO;REAL_MUL_RZERO;
  REAL_POW_EQ_0;COMPLEX_NORM_ZERO;INSPACE_ZERO_EQ;INSPACE_LZERO]);;
```

→ Much shorter

→ Get rids of subgoals previously manually provided

Note:

- We do not want to get rid of *meaningful* subgoals.
- Just the ones introduced by the user because the thm prover does not provide him/her with any other mean to go on with

Outline

Overview

Description by refinement from usual rewriting to implicational rewriting:

1. Usual rewriting
2. Conditional rewriting
3. Dependent rewriting
4. Implicational rewriting

Usual rewriting

Given:

- A goal g
- A theorem of the form $\vdash l = r$ such that g contains $l\sigma$ for some substitution σ

(let's focus on simple cases: one match only, r variables appear in l variables. . .)

Generate:

- A goal g' s.t. $l\sigma$ is turned into $r\sigma$

Problem:

- Many theorems actually have the form $\vdash P \Rightarrow l = r$

$\Rightarrow$ Conditional rewriting

Usual rewriting

Given:

- A goal g
- A theorem of the form $\vdash l = r$ such that g contains $l\sigma$ for some substitution σ

(let's focus on simple cases: one match only, r variables appear in l variables. . .)

Generate:

- A goal g' s.t. $l\sigma$ is turned into $r\sigma$

Problem:

- Many theorems actually have the form $\vdash P \Rightarrow l = r$

$\Rightarrow$ Conditional rewriting

Conditional rewriting

Given:

- A goal g
- A theorem of the form $P \Rightarrow l = r$ such that g contains $l\sigma$ for some substitution σ

Generate:

- A goal g' s.t. $l\sigma$ is turned into $r\sigma$
- If the simplifier manages to prove $P\sigma$

Problem: Often the simplifier cannot prove $P\sigma$

- Either because $P\sigma$ too complex
- Or because simplifier missing theorems
- Or because $P\sigma$ is simply false

$\rightarrow$ then problem = no feedback (tactic fails or does not progress)

$\rightarrow$ time-consuming to find out the required theorems or that the condition cannot be proved

$\Rightarrow$ Dependent rewriting

Conditional rewriting

Given:

- A goal g
- A theorem of the form $P \Rightarrow l = r$ such that g contains $l\sigma$ for some substitution σ

Generate:

- A goal g' s.t. $l\sigma$ is turned into $r\sigma$
- If the simplifier manages to prove $P\sigma$

Problem: Often the simplifier cannot prove $P\sigma$

- Either because $P\sigma$ too complex
- Or because simplifier missing theorems
- Or because $P\sigma$ is simply false

$\rightarrow$ then problem = no feedback (tactic fails or does not progress)

$\rightarrow$ time-consuming to find out the required theorems or that the condition cannot be proved

$\Rightarrow$ Dependent rewriting

Dependent rewriting

(naming used after Homeier in HOL4 - used in many places, never really named)

Given: (same input as conditional rewriting)

- A goal g
- A theorem of the form $P \Rightarrow l = r$ such that g contains $l\sigma$ for some substitution σ

Generate:

- A goal $P\sigma \wedge g'$ s.t. $l\sigma$ is turned into $r\sigma$ in g'
- Variant: generate g' and $P\sigma$ as two separate goals

Still some problems. Example:

- Goal: $\forall x.(\textit{big term implying } x \neq 0) \Rightarrow \frac{x}{x} * y = y$
- Apply dependent rewriting with $\vdash x \neq 0 \Rightarrow \frac{x}{x} = 1$
- New goal: $x \neq 0 \wedge \forall x.(\textit{big term impl. } x \neq 0) \Rightarrow 1 * y = y$
  $\rightarrow$ the big term cannot be used to get rid of $x \neq 0$

$\Rightarrow$ Implicational rewriting

Dependent rewriting

(naming used after Homeier in HOL4 - used in many places, never really named)

Given: (same input as conditional rewriting)

- A goal g
- A theorem of the form $P \Rightarrow l = r$ such that g contains $l\sigma$ for some substitution σ

Generate:

- A goal $P\sigma \wedge g'$ s.t. $l\sigma$ is turned into $r\sigma$ in g'
- Variant: generate g' and $P\sigma$ as two separate goals

Still some problems. Example:

- Goal: $\forall x.(big\ \ term\ \ implying\ \ x \neq 0) \Rightarrow \frac{x}{x} * y = y$
- Apply dependent rewriting with $\vdash x \neq 0 \Rightarrow \frac{x}{x} = 1$
- New goal: $x \neq 0 \wedge \forall x.(big\ \ term\ \ impl.\ x \neq 0) \Rightarrow 1 * y = y$
  $\rightarrow$ the big term cannot be used to get rid of $x \neq 0$

$\Rightarrow$ Implicational rewriting

Implicational rewriting

Given: (same input as conditional rewriting)

- A goal g
- A theorem of the form $P \Rightarrow l = r$ such that g contains $l\sigma$ for some substitution σ

Generate:

- A goal g' s.t. $l\sigma$ is turned into $r\sigma$
- And containing $P\sigma$ *as close as possible* to the atom containing $l\sigma$

Just a detail, but surprisingly makes this thing much more useful

Note: A little bit of care on how to add $P\sigma$:

- $P\sigma$ added as a *conjunct* in positive positions
- As a *premise* in negative positions

Implicational rewriting: Real-life example (1/4)

Excerpt from the Cauchy-Schwarz example:

Goal

$$\forall s\ \langle .|. \rangle\ x\ y. isInnerSpace\ (s, \langle .|. \rangle) \wedge x \in s \wedge y \in s$$
$$\Rightarrow norm\ (\langle x|y \rangle)^2 \leq \mathbf{realOfComplex}$$
$$(norm\ (\langle x|y \rangle)^2.(\langle x|x \rangle.\langle x|x \rangle).\frac{1}{norm(\langle x|x \rangle)^2} +$$
$$\langle x|x \rangle. \left\langle y - (\langle x|y \rangle.\frac{1}{\langle x|x \rangle})\%x \middle| y - (\langle x|y \rangle.\frac{1}{\langle x|x \rangle})\%x \right\rangle)$$

We want to use the theorem:

Theorem

$$\vdash \forall u\ v.\ real\ u \wedge real\ v$$
$$\Rightarrow \mathbf{realOfComplex}\ (u+v) = realOfComplex\ u + realOfComplex\ v$$

$\rightarrow$ Conditional rewriting requires identifying the instantiations of u, v and the corresponding theorems to get rid of the condition

$\rightarrow$ time-consuming

$\Rightarrow$ natural to want the machine do it for us

Implicational rewriting: Real-life example (2/4)

Dependent rewriting is not enough. Yields:

Goal obtained by dependent rewriting

$$\mathit{real}\ (\mathit{norm}\ (\langle x|y\rangle)^2.(\langle x|x\rangle.\langle x|x\rangle).\frac{1}{\mathit{norm}(\langle x|x\rangle)^2})$$
$$\wedge\ \mathit{real}\ (\langle x|x\rangle.\left\langle (y-(\langle x|y\rangle.\frac{1}{\langle x|x\rangle})\%x) \middle| (y-(\langle x|y\rangle.\frac{1}{\langle x|x\rangle})\%x)\right\rangle)$$
$$\wedge\forall s\ \langle .|.\rangle\ x\ y.\mathit{isInnerSpace}\ (s,\langle .|.\rangle) \wedge x\in s \wedge y \in s$$
$$\Rightarrow \mathit{norm}\ (\langle x|y\rangle)^2 \leq \mathit{realOfComplex}\ (\mathit{norm}\ (\langle x|y\rangle)^2.(\langle x|x\rangle.\langle x|x\rangle).\frac{1}{\mathit{norm}(\langle x|x\rangle)^2})$$
$$+\mathit{realOfComplex}\ (\langle x|x\rangle.\left\langle (y-(\langle x|y\rangle.\frac{1}{\langle x|x\rangle})\%x) \middle| (y-(\langle x|y\rangle.\frac{1}{\langle x|x\rangle})\%x)\right\rangle)$$

$\rightarrow$ instantiations automatically found

$\rightarrow$ but new goal not provable

(x, y, s, $\langle .|.\rangle$, are out of scope)

Implicational rewriting: Real-life example (3/4)

There is a solution allowing to still use dep. rewriting:

- Start over
- Discharge $\mathit{isInnerSpace}\ (s, \langle .|. \rangle) \wedge x \in s \wedge y \in s$
- Now only, apply dependent rewriting
- Use the premise (now in the assumptions) to prove $P\sigma$

$\rightarrow$ but not satisfying:

1. *First and foremost, these steps can be automatized (and are not mathematically meaningful)*
   $\rightarrow$ So if it can be automatized why not doing it?
2. *Discharging often leads several goals*
   $\rightarrow$ Breaks compositionality
3. *Different behavior than usual/conditional rewriting*

Implicational rewriting: Real-life example (3/4)

There is a solution allowing to still use dep. rewriting:

- Start over
- Discharge *isInnerSpace* $(s, \langle . | . \rangle) \wedge x \in s \wedge y \in s$
- Now only, apply dependent rewriting
- Use the premise (now in the assumptions) to prove $P\sigma$

$\rightarrow$ but not satisfying:

1. *First and foremost, these steps can be automatized (and are not mathematically meaningful)*
   $\rightarrow$ So if it can be automatized why not doing it?
2. *Discharging often leads several goals*
   $\rightarrow$ Breaks compositionality
3. *Different behavior than usual/conditional rewriting*

Implicational rewriting: Real-life example (4/4)

On the other hand, implicational rewriting yields:

Goal obtained by implicational rewriting

$$\forall s\ \langle.|.\rangle\ x\ y.isInnerSpace\ (s, \langle.|.\rangle) \wedge x \in s \wedge y \in s$$
$$\Rightarrow real\ (norm\ (\langle x|y\rangle)^2.(\langle x|x\rangle.\langle x|x\rangle).\frac{1}{norm(\langle x|x\rangle)^2})$$
$$\wedge\ real\ (\langle x|x\rangle.\left\langle (y - (\langle x|y\rangle.\frac{1}{\langle x|x\rangle})\%x) \middle| y - (\langle x|y\rangle.\frac{1}{\langle x|x\rangle})\%x \right\rangle)$$
$$\wedge\ norm\ (\langle x|y\rangle)^2 \le realOfComplex$$
$$(norm\ (\langle x|y\rangle)^2.(\langle x|x\rangle.\langle x|x\rangle).\frac{1}{norm(\langle x|x\rangle)^2})$$
$$+realOfComplex\ (\langle x|x\rangle.$$
$$\left\langle (y - (\langle x|y\rangle.\frac{1}{\langle x|x\rangle})\%x) \middle| (y - (\langle x|y\rangle.\frac{1}{\langle x|x\rangle})\%x) \right\rangle)$$

→ instantiations automatically found

→ and new goal still provable

Summing up: *automatized, compositionality improved, similar behavior as rewriting*

Outline

Motivation

Problem with implicational rewriting:

- Sometimes works too well (quoting Marco Maggesi)
  $\rightarrow$ diverges or rewrites too many things
- Usual solution in HOL{4-Light}: "ONCE" rewrite
  (applies the rewrite once; in parallel positions)
- Usual further problems: still rewrites too many
- Usual solutions: better ways to select what is rewritten
  (manually giving positions, patterns, . . .)
- But still done manually: time-consuming

$\rightarrow$ Target rewriting

Why does that actually happen?

Why do we actually want to apply a rewrite in a particular position/a given number of times?

My answer/feeling:

- Because we think one step further
- We know that the next step of the proof applies to a particular pattern only
- So we want to massage the goal so as to obtain this pattern

Target rewriting

What if we have a tactic which takes two theorems:

1. one to be used for massaging
2. the other to be used for the "next step"

And the tactic does the job of using the massaging theorem on the goal so that the next step can apply

→ Target rewriting

Concrete example: problems without target rewringt

Back again to Cauchy-Schwarz:

Initial goal

$$\forall s\ \langle .|. \rangle\ x\ y.\ \mathit{isInnerSpace}\ (s, \langle .|. \rangle) \wedge x \in s \wedge y \in s$$
$$\Rightarrow \mathit{norm}(\langle x|y \rangle)^2 \leq \mathit{realOfComplex}\ (\langle x|x \rangle . \langle y|y \rangle)$$

And we want to use the following decomposition theorem:

Theorem (Decomposition)

$$\forall s\ \langle .|. \rangle\ u\ v.\ \mathit{isInnerSpace}\ (s, \langle .|. \rangle) \wedge u \in s \wedge v \in s$$
$$\Rightarrow u = \frac{\langle v|u \rangle}{\langle v|v \rangle} v + u - \frac{\langle v|u \rangle}{\langle v|v \rangle} v\ \wedge\ v \perp (u - \frac{\langle v|u \rangle}{\langle v|v \rangle} v)$$

In order to obtain:

Expected goal

$$\forall s\ \langle .|. \rangle\ u\ v.\ \mathit{isInnerSpace}\ (s, \langle .|. \rangle) \wedge u \in s \wedge v \in s$$
$$\Rightarrow \mathit{norm}(\langle x|y \rangle)^2 \leq \mathit{realOfComplex}\ ($$
$$\langle x|x \rangle . \left\langle \frac{\langle x|y \rangle}{\langle x|x \rangle} x + y - \frac{\langle x|y \rangle}{\langle x|x \rangle} x \middle| \frac{\langle x|y \rangle}{\langle x|x \rangle} x + y - \frac{\langle x|y \rangle}{\langle x|x \rangle} x \right\rangle$$

Concrete example: using target rewriting (1/2)

Instead of trying with rewriting, let's step back:

Initial goal

$$\forall s\ \langle.|.\rangle\ u\ v.\ \mathit{isInnerSpace}\ (s, \langle.|.\rangle) \wedge u \in s \wedge v \in s$$
$$\Rightarrow \mathit{norm}(\langle x|y\rangle)^2 \leq \mathit{realOfComplex}\ ($$
$$\langle x|x\rangle. \left\langle \frac{\langle x|y\rangle}{\langle x|x\rangle}x + y - \frac{\langle x|y\rangle}{\langle x|x\rangle}x \middle| \frac{\langle x|y\rangle}{\langle x|x\rangle}x + y - \frac{\langle x|y\rangle}{\langle x|x\rangle}x \right\rangle$$

Question: *why do we want to obtain this goal?*

Answer: because we actually want to use the following theorem afterwards:

Theorem (≈ Commutativity of sum and inner product under orthogonality)

$$\forall s\ \mathit{inprod}\ u\ v.\ \mathit{isInnerSpace}\ (s, \langle.|.\rangle) \wedge u \in s \wedge v \in s \wedge u \perp v$$
$$\Rightarrow \langle u + v|u + v\rangle = \langle u|u\rangle + \langle v|v\rangle$$

Concrete example: using target rewriting (2/2)

So that's what target rewriting does:

Initial goal

$$\forall s\ \langle.|.\rangle\ x\ y.\ \mathit{isInnerSpace}\ (s, \langle.|.\rangle) \wedge x \in s \wedge y \in s$$
$$\Rightarrow \mathit{norm}(\langle x|y\rangle)^2 \leq \mathit{realOfComplex}\ (\langle x|x\rangle.\langle y|y\rangle)$$

```
# TARGET_REWRITE_TAC [DECOMPOSITION_THM] ORTHOGONAL_PROD_SUM;;
```

Obtained goal

$$\forall s\ \langle.|.\rangle\ u\ v.\ \mathit{isInnerSpace}\ (s, \langle.|.\rangle) \wedge u \in s \wedge v \in s$$
$$\Rightarrow \mathit{norm}(\langle x|y\rangle)^2 \leq \mathit{realOfComplex}\ ($$
$$\langle x|x\rangle.\left\langle \frac{\langle x|y\rangle}{\langle x|x\rangle}x + y - \frac{\langle x|y\rangle}{\langle x|x\rangle}x \middle| \frac{\langle x|y\rangle}{\langle x|x\rangle}x + y - \frac{\langle x|y\rangle}{\langle x|x\rangle}x \right\rangle)$$

$\rightarrow$ yields the expected goal

Outline

Summary

- Less need for manual input
- Better feedback
- Time saved

Implementation:

- Now integrated in HOL Light
- Implementation in HOL4 also available (but still buggy for now)

Related Work

Implicational rewriting:

- dependent rewriting (HOL4, Homeier) and many similar works in HOL Light, Isabelle, Coq
- fact of reasoning deeply: *deep inference*
- if formulas seen as clauses: *superposition calculus*

Target rewriting:

- similar to *smart matching* in Matita, but different usage/philosophy (and much less efficient)
- conceptually can be seen as an instance of *deduction modulo*
- *AC-rewriting* is a particular case

Thanks!

Discussion

Do people really want this kind of automation?

- I hear many times that people prefer to state explicitly some subgoals, in particular for readability
- I am more in favour of the machine finding out what I want
- And then retrieve the information by replaying the scripts, or using a tool like Proviola[1]
- I think it is better for proof engineering to make the scripts less fragile
- But I also hear exactly the same argument precisely to go towards more explicit information in the scripts...

In any case, one can still use the tactic when meaningful and not use it otherwise...

[1] Tanking, Geuvers, McKinna, Wiedijk