Importing SMT and Connection proofs as expansion trees: examples

Giselle Reis
INRIA-Saclay, France
giselle.reis@inria.fr

This report contains some examples of proofs from the automated theorem provers leanCoP and veriT and shows how they can be imported in GAPT. All files are available in the examples directory of the software.

The versions of the softwares used were:

- GAPT: master branch as of 30/07/2015[1]
- LeanCoP 2.2
- VeriT 201410

We show here how these proofs are visualized in prooftool, but by typing `help` in GAPT's command line, one can see a list of available functions for other purposes.

1 LeanCoP

The following file represents the problem of determining whether there exists two irrational numbers x and y such that x to the power of y is rational.

```
fof(a, axiom, i(sr2)).
fof(b, axiom, ~i(two)).
fof(c, axiom, times(sr2,sr2) = two).
fof(d, axiom, ![X,Y,Z] : exp(exp(X, Y), Z) = exp(X, times(Y,Z))).
fof(e, axiom, ![X] : exp(X, two) = times(X,X)).
fof(f, conjecture, ?[X,Y] : (~i(exp(X,Y)) & i(X) & i(Y))).
```

LeanCoP's `leantptp` proof (with extra line breaks to fit the width) of this problem is:

```
fof(f, conjecture, ? [_13459, _13462] : (~ i(exp(_13459, _13462)) & i(_13459) & i(_13462)),
  file('samples/irrationals.p', f)).
fof(a, axiom, i(sr2), file('samples/irrationals.p', a)).
fof(b, axiom, ~ i(two), file('samples/irrationals.p', b)).
fof(c, axiom, times(sr2, sr2) = two, file('samples/irrationals.p', c)).
fof(d, axiom, ! [_13784, _13787, _13790] :
  exp(exp(_13784, _13787), _13790) = exp(_13784, times(_13787, _13790)),
    file('samples/irrationals.p', d)).
fof(e, axiom, ! [_13973] : exp(_13973, two) = times(_13973, _13973),
  file('samples/irrationals.p', e)).

cnf(1, plain, [-(i(exp(_7304, _7358))), i(_7304), i(_7358)], clausify(f)).
cnf(2, plain, [-(i(sr2))], clausify(a)).
cnf(3, plain, [i(two)], clausify(b)).
```

[1] If you are using the system after this date, all functionality described here should work. If this is not the case, please file a bug report.

Submitted to:
PxTP 2015

```
cnf(4, plain, [-(times(sr2, sr2) = two)], clausify(c)).
cnf(5, plain, [-(exp(exp(_6527, _6583), _6638) = exp(_6527, times(_6583, _6638)))], clausify(d)).
cnf(6, plain, [-(exp(_6984, two) = times(_6984, _6984))], clausify(e)).
cnf(7, plain, [-(exp(_4828, _4961) = exp(_4895, _5026)), _4828 = _4895, _4961 = _5026],
  theory(equality)).
cnf(8, plain, [-(i(_4533)), _4484 = _4533, i(_4484)], theory(equality)).
cnf(9, plain, [-(_3586 = _3586)], theory(equality)).

cnf('1',plain,[i(two)],start(3)).
cnf('1.1',plain,[-(i(two)), times(sr2, sr2) = two, i(times(sr2, sr2))],
  extension(8,bind([[_4533, _4484], [two, times(sr2, sr2)]]))).
cnf('1.1.1',plain,[-(times(sr2, sr2) = two)],extension(4)).
cnf('1.1.2',plain,[-(i(times(sr2, sr2))), exp(sr2, two) = times(sr2, sr2), i(exp(sr2, two))],
  extension(8,bind([[_4533, _4484], [times(sr2, sr2), exp(sr2, two)]]))).
cnf('1.1.2.1',plain,[-(exp(sr2, two) = times(sr2, sr2))],extension(6,bind([[_6984], [sr2]]))).
cnf('1.1.2.2',plain,[-(i(exp(sr2, two))),
  exp(sr2, times(sr2, sr2)) = exp(sr2, two), i(exp(sr2, times(sr2, sr2)))],
  extension(8,bind([[_4533, _4484], [exp(sr2, two), exp(sr2, times(sr2, sr2))]]))).
cnf('1.1.2.2.1',plain,[-(exp(sr2, times(sr2, sr2)) = exp(sr2, two)),
  sr2 = sr2, times(sr2, sr2) = two],
  extension(7,bind([[_4828, _4895, _4961, _5026], [sr2, sr2, times(sr2, sr2), two]]))).
cnf('1.1.2.2.1.1',plain,[-(sr2 = sr2)],extension(9,bind([[_3586], [sr2]]))).
cnf('1.1.2.2.1.2',plain,[-(times(sr2, sr2) = two)],extension(4)).
cnf('1.1.2.2.2',plain,[-(i(exp(sr2, times(sr2, sr2)))),
  exp(exp(sr2, sr2), sr2) = exp(sr2, times(sr2, sr2)), i(exp(exp(sr2, sr2), sr2))],
  extension(8,bind([[_4533, _4484], [exp(sr2, times(sr2, sr2)), exp(exp(sr2, sr2), sr2)]]))).
cnf('1.1.2.2.2.1',plain,[-(exp(exp(sr2, sr2), sr2) = exp(sr2, times(sr2, sr2)))],
  extension(5,bind([[_6527, _6583, _6638], [sr2, sr2, sr2]]))).
cnf('1.1.2.2.2.2',plain,[-(i(exp(exp(sr2, sr2), sr2))), i(exp(sr2, sr2)), i(sr2)],
  extension(1,bind([[_7304, _7358], [exp(sr2, sr2), sr2]]))).
cnf('1.1.2.2.2.2.1',plain,[-(i(exp(sr2, sr2))), i(sr2), i(sr2)],
  extension(1,bind([[_7304, _7358], [sr2, sr2]]))).
cnf('1.1.2.2.2.2.1.1',plain,[-(i(sr2))],extension(2)).
cnf('1.1.2.2.2.2.1.2',plain,[-(i(sr2))],extension(2)).
cnf('1.1.2.2.2.2.2',plain,[-(i(sr2))],extension(2)).
```

One can load it in GAPT using the command:

```
gapt> val es = loadLeanCoPProof("examples/import/irrationals.leancop.s")
es: Option[at.logic.gapt.proofs.expansionTrees.ExpansionSequent] = ...
```

Running prooftool on this object (`prooftool(es.get)`) will open a window with the visualization of the expansion proof, as shown in Figure 1. Note that the succedent is already expanded (clicking on a quantified formula will expand it to the instances used) and we can see the two pairs used: $(\sqrt{2},\sqrt{2})$ and $(\sqrt{2}^{\sqrt{2}},\sqrt{2})$.

2 VeriT

The following is a simple proof which needs the equality axiom of congruence on predicates:

```
(set-logic QF_UF)
(set-info :smt-lib-version 2.0)
(declare-sort U 0)
(declare-fun f (U U) U)
```

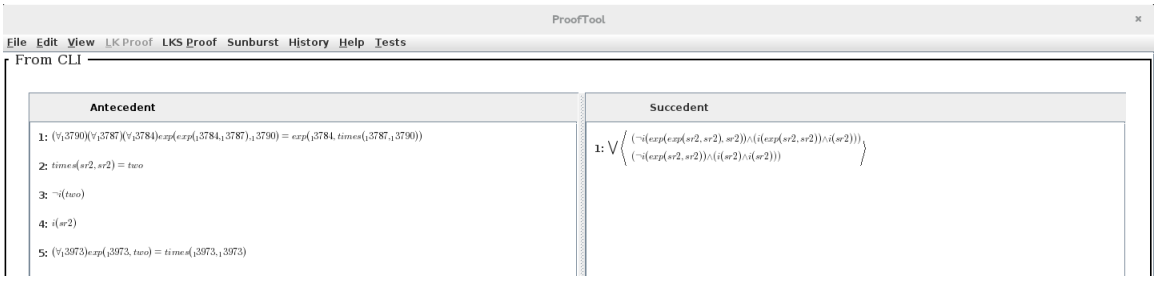

Figure 1: Visualization of the expansion tree for the proof of irrational numbers.

```
(declare-fun a () U)
(declare-fun b () U)
(declare-fun p (U) Bool)
(assert (p a))
(assert (and (= (f a b) (f (f a b) b))
         (= (p (f (f a b) b)) (p a))))
(assert (not (p (f a b))))
(check-sat)
(exit)
```

Running veriT on this problem with the option `--proof-version=1` generates the proof object (with extra line breaks):

```
veriT 201410 - the SMT-solver veriT (UFRN/LORIA).
success
success
success
success
success
success
success
success
unsat
success
(set .c1 (input :conclusion ((p a))))
(set .c2 (input :conclusion ((and (= (f a b) (f (f a b) b)) (= (p (f (f a b) b)) (p a))))))
(set .c3 (input :conclusion ((not (p (f a b)))))))
(set .c4 (and :clauses (.c2) :conclusion ((= (f a b) (f (f a b) b)))))
(set .c5 (and :clauses (.c2) :conclusion ((= (p (f (f a b) b)) (p a)))))
(set .c6 (equiv1 :clauses (.c5) :conclusion ((not (p (f (f a b) b))) (p a))))
(set .c7 (equiv2 :clauses (.c5) :conclusion ((p (f (f a b) b)) (not (p a)))))
(set .c8 (resolution :clauses (.c7 .c1) :conclusion ((p (f (f a b) b)))))
(set .c9 (eq_congruent_pred :conclusion ((not (= (f a b) (f (f a b) b)))
  (not (p (f (f a b) b))) (p (f a b)))))
(set .c10 (resolution :clauses (.c9 .c4 .c8 .c3) :conclusion ()))
```

Analogous to the leanCoP case, we can load the proof in GAPT and open the corresponding expansion proof in prooftool:

```
val p = loadVeriTProof("examples/import/predcong.verit.s")
p: Option[at.logic.gapt.proofs.expansionTrees.ExpansionSequent] = ...

gapt> prooftool(p.get)
```

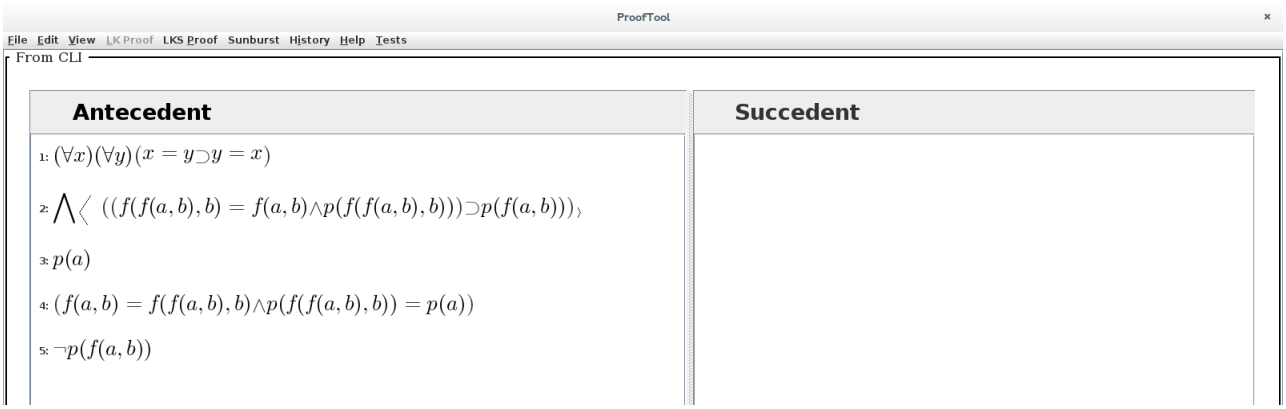

Figure 2: Visualization of the expansion tree for a simple veriT proof.

The result is in Figure 2. In this case, the instance of the predicate congruence axiom used in expanded.

Acknowledgments The author would like to thank Pascal Fontaine and Jens Otten for clarifications about the tools used and fruitful discussions; Pascal Fontaine and Geoff Sutcliffe for providing the dataset of proofs; Sonia Marin for comments on an early draft; and the reviewers for very useful remarks and for taking the time to try the system.