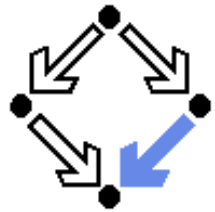
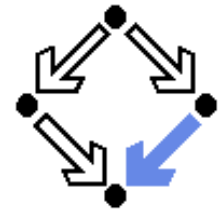
The RISC ProofNavigator
Second Status Report

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.uni-linz.ac.at



Motivation

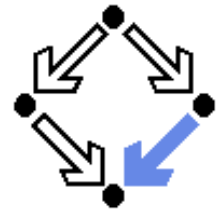


Component of an envisioned integrated program reasoning environment.

- Based on 2005 evaluation of various proving assistants.
  - PVS, Isabelle/HOL, Coq.
- Key goals:
  - Aggressive proof state simplification.
  - User-friendly proof state presentation.
  - Intuitive proof tree navigation.
  - Helpful proof automation (proving on the small scale).
  - Open to human guidance (proving on the large scale).
- Principle of "least surprise"
  - If in doubt, do the least surprising thing.
  - E.g., start potentially costly simplifications only at explicit request.
- Open source implementation
  - Java-based application and class library.
  - Stanford CVCL (Cooperating Validity Checker Lite)

http://www.risc.uni-linz.ac.at/research/formal/software/ProofNavigator

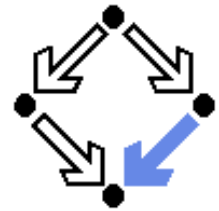
Conclusions in November 2005



- Basic framework operational and stable.
  - About 150 classes and 20000 lines of code.
  - But only toy examples have been tested so far.
- Next: better CVCL interaction.
  - Error detection and recovery.
  - Re-translation of canonized formulas to human-friendly form.
  - Decomposing formulas for validity checking of individual parts.
- Goal: run my test cases successfully and analyze the results.
  - Certainly more work will arise from this...
- Finally: documentation and paper.
  - Release of system planned for first half of 2006.
- Perhaps: extension by other automated decision procedures.
  - Interfaces for Simplify and ICS may be interesting.

Hopefully I can show some more examples soon.

Status in March 2006

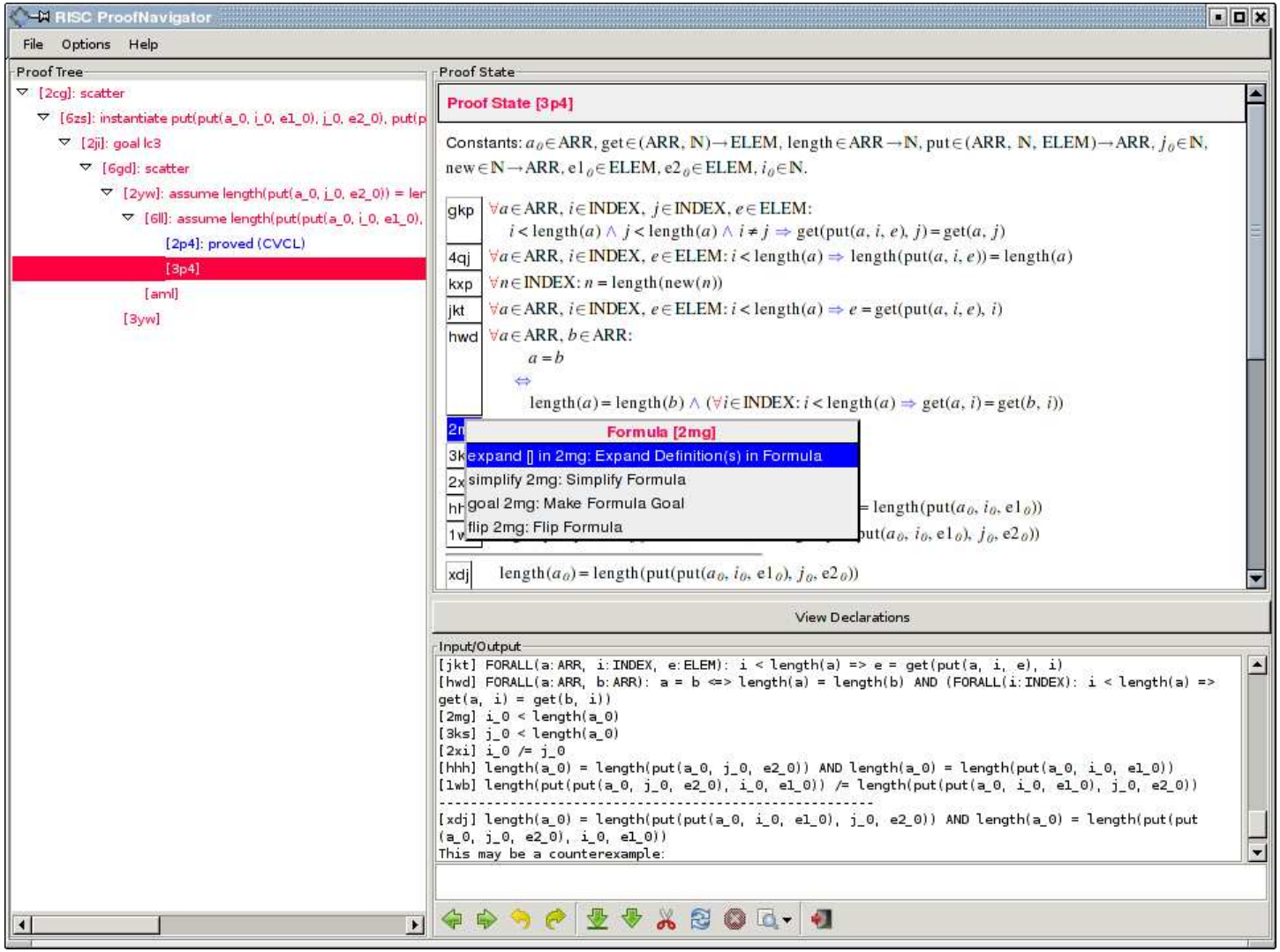


ProofNavigator version 0.52 installed in the RISC environment.

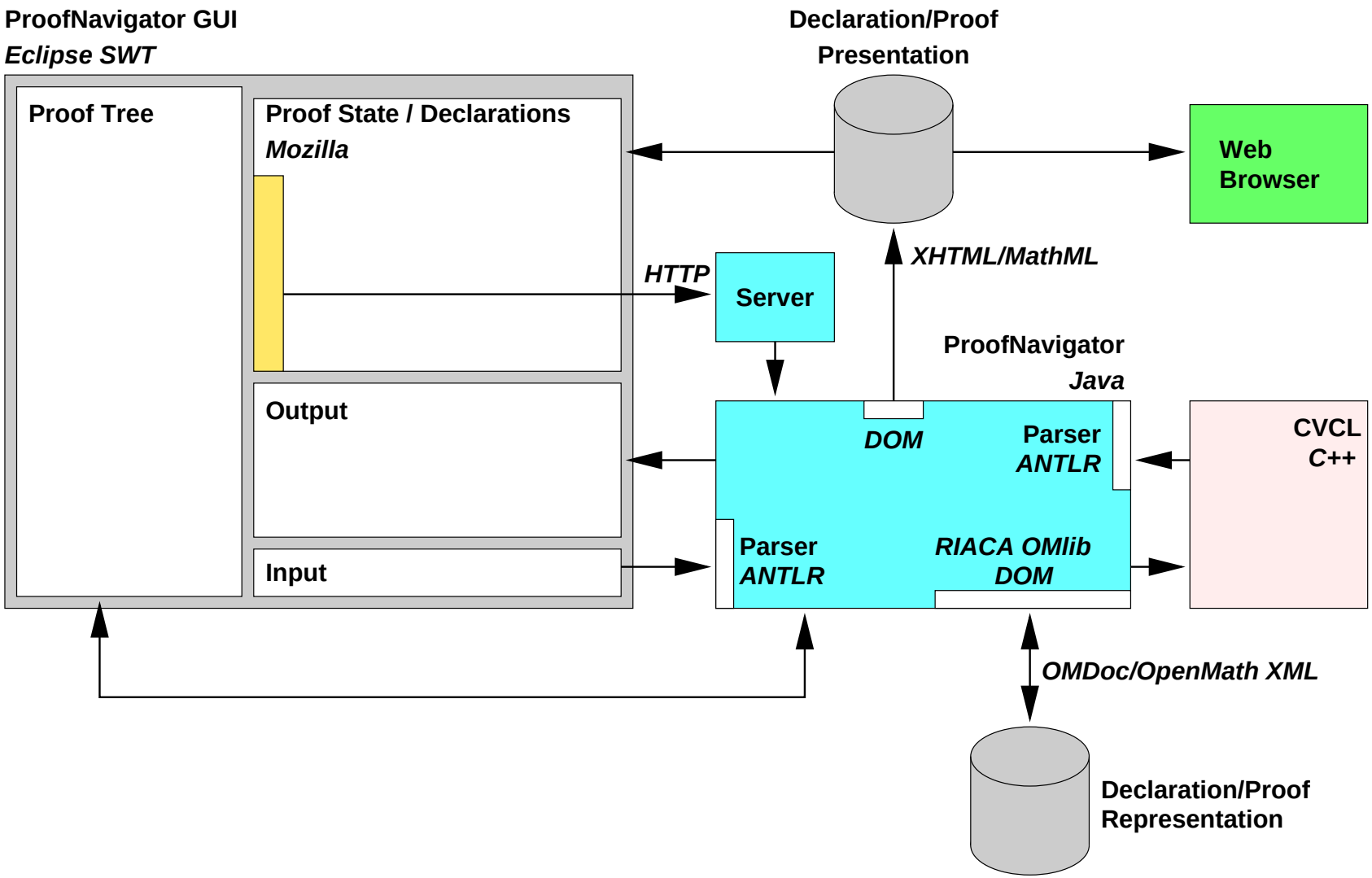
- New point-and-click interface.
  - Proof navigation.
  - Proof state manipulation.
- Improvement of CVCL interaction.
  - Manual abortion of validity check.
  - Translation of simplification results to human-friendly form.
- Numerous internal modifications.
  - Fixing bugs and conceptual errors, replacing hacks by real solutions.
  - Fine-tuning to minimize user surprise.
- Successfully proved most test cases.
  - Axiomatic arrays with proof of "commutes" property.
  - Constructive arrays with proofs of "commutes" and extensionality.
  - Verification of linear search.
  - Verification of binary search.

Status almost as planned.

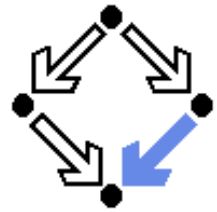
ProofNavigator GUI



ProofNavigator Architecture



Proof Examples

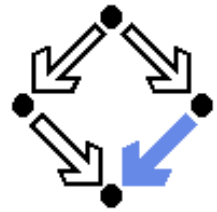


Examples contained in 0.52 release.

- Induction proofs: `sum` and `sum0`.
- Quantifier proofs: `quant` and `quant0`.
- Axiomatization of arrays: `arrays1`.
- Constructive definition of arrays: `arrays2`.
- Linear search: `linsearch`.
- Binary search: `binarysearch`.

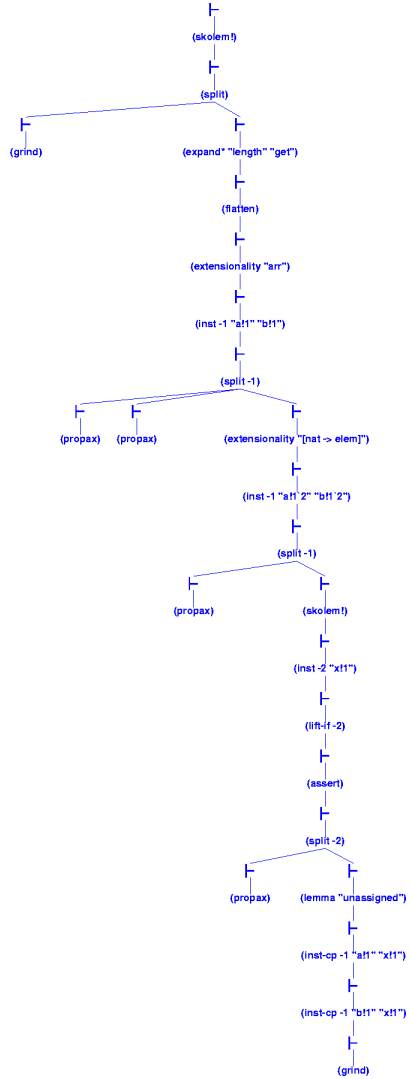
Now for some online demonstrations.

Constructive Definition of Arrays

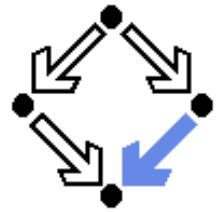


ProofNavigator versus PVS.

```
[adt]: expand get, length, content
  [cw2]: scatter
    [qey]: proved (CVCL)
    [rey]: assume b_0.1 = a_0.1
      [zpt]: proved (CVCL)
      [1pt]: instantiate a_0.1, b_0.1 in 1fm
        [y51]: goal 2sq
          [ku2]: scatter
            [iub]: instantiate a_0, i_0 in 3p3
              [1ef]: instantiate b_0, i_0 in 3p3
                [y4r]: instantiate i_0 in ruq
                  [kbc]: proved (CVCL)
```



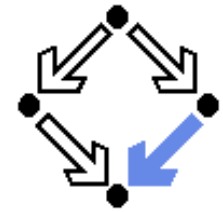
Linear Search: Conditions A and B1



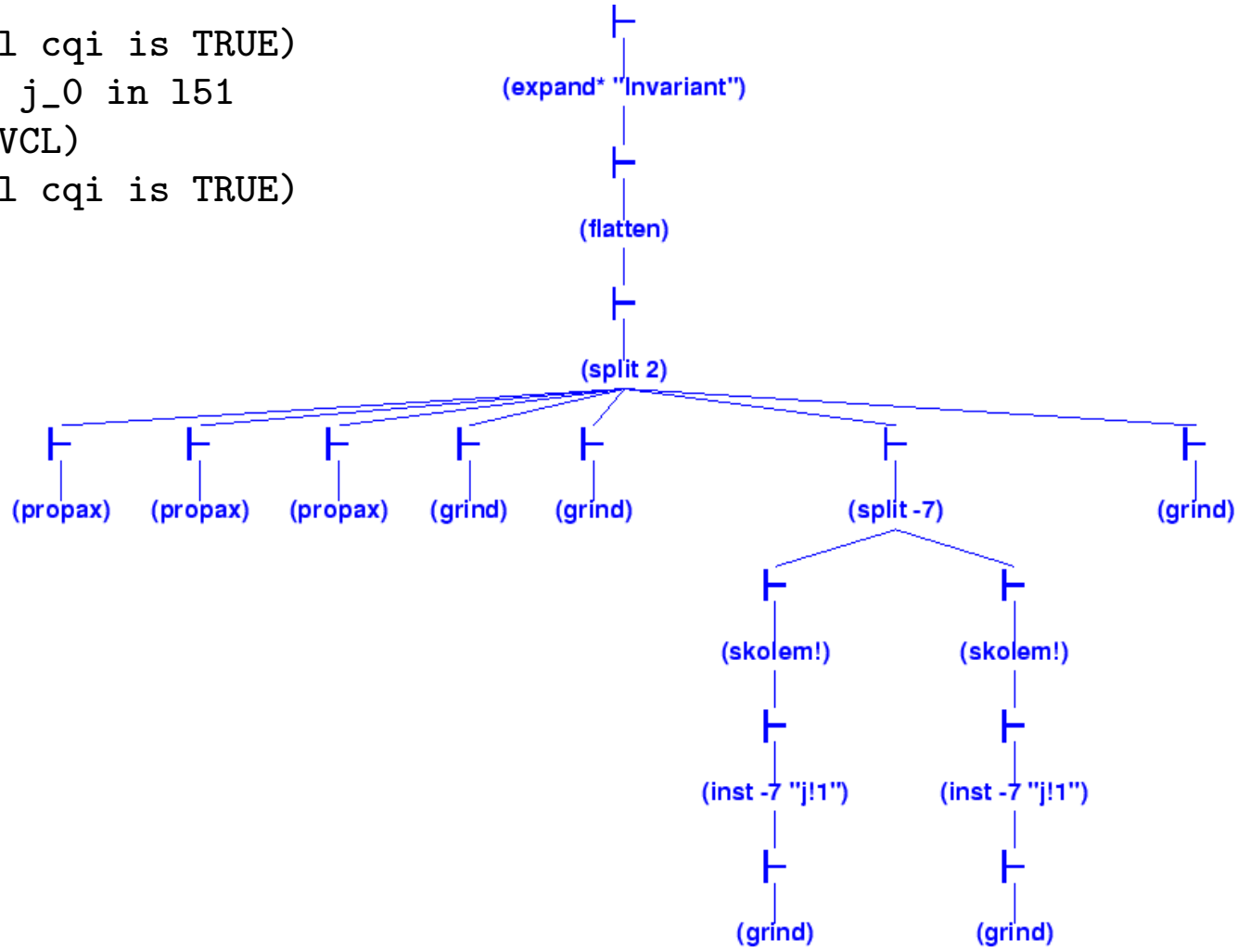
```
[bca]: expand Input, Invariant
  [fuo]: scatter
    [bxg]: proved (CVCL)
```

```
[p1b]: expand Invariant
  [lf6]: proved (CVCL)
```

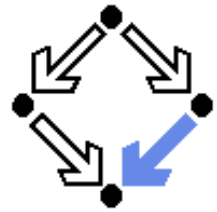
Linear Search: Condition B2



```
[q1b]: expand Invariant
  [slx]: scatter
    [a1y]: proved (Goal cqi is TRUE)
    [b1y]: instantiate j_0 in l51
      [joz]: proved (CVCL)
    [c1y]: proved (Goal cqi is TRUE)
```

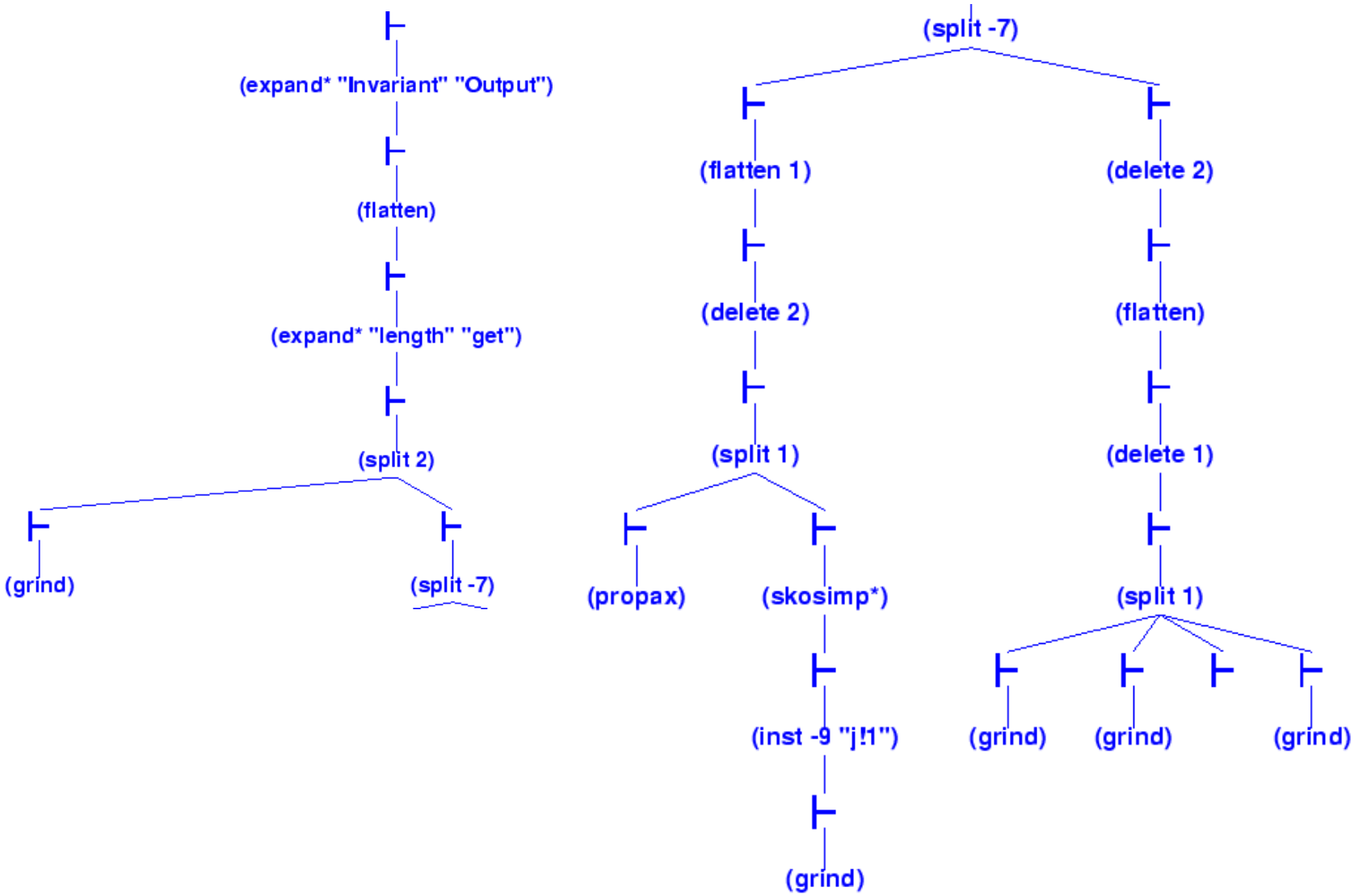


Linear Search: Condition C

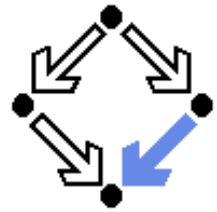


```
[dca]: expand Invariant, Output
  [tvy]: scatter
    [dcu]: instantiate j_0 in l51
      [t4c]: proved (CVCL)
    [ecu]: goal ueq
      [kel]: scatter
        [ulf]: proved (CVCL)
        [vlf]: instantiate j_0 in l51
          [rzb]: proved (CVCL)
    [fcu]: instantiate j_0 in l51
      [blt]: proved (CVCL)
    [gcu]: proved (CVCL)
```

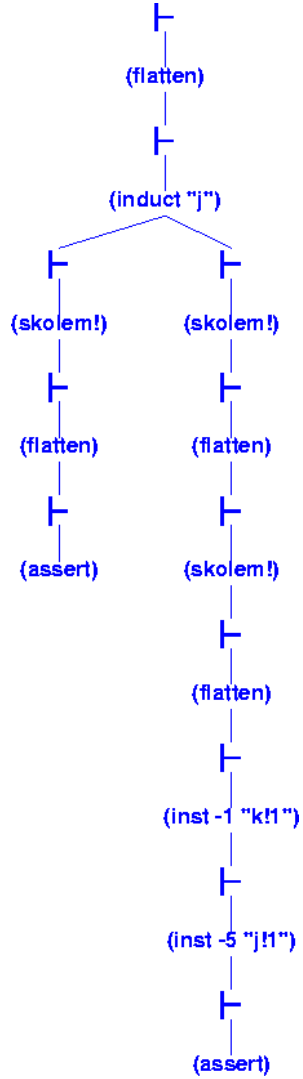
Linear Search: Condition C (Contd)



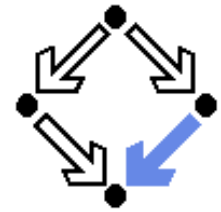
Binary Search: Lemma L



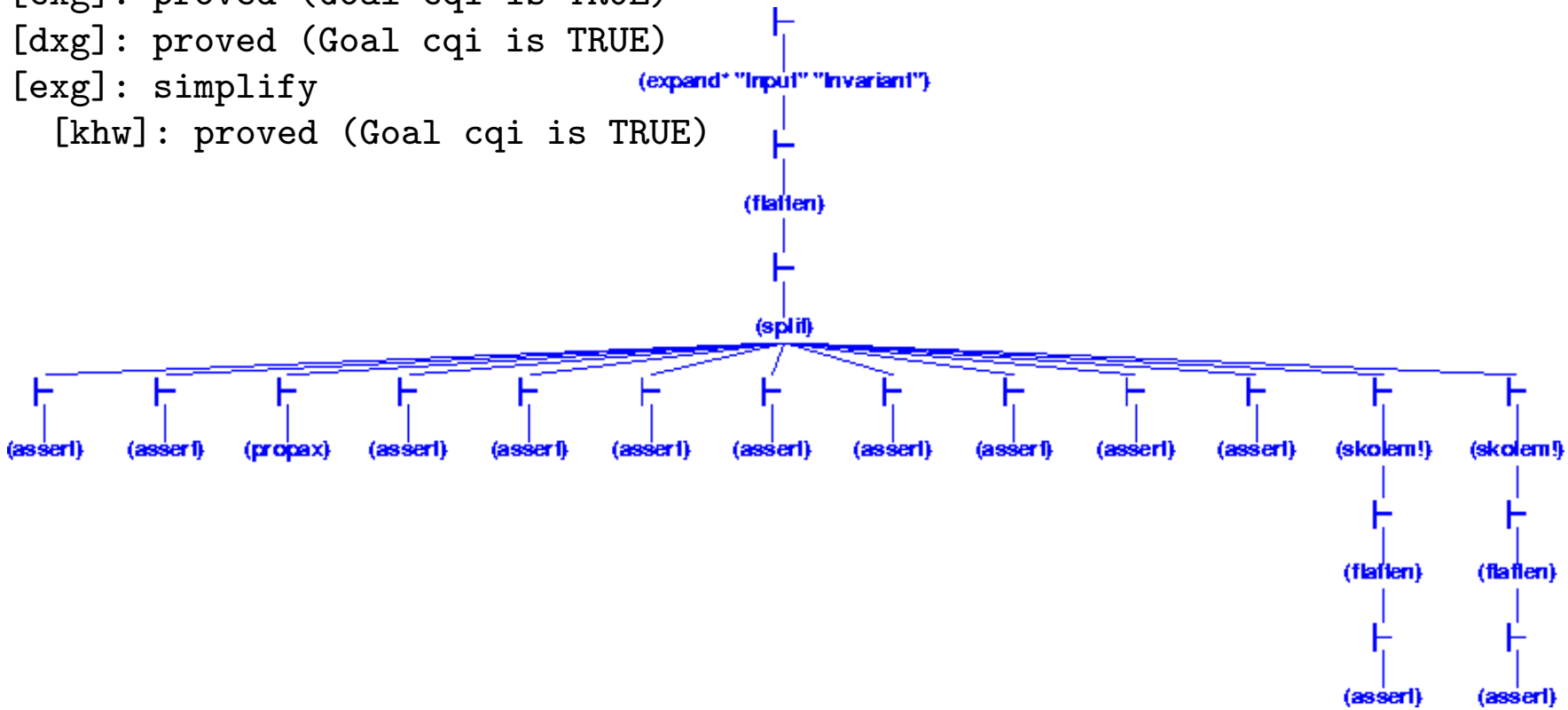
```
[mca]: flatten
  [c3f]: induction j z42
    [3da]: scatter
      [s3k]: proved (CVCL)
    [4da]: scatter
      [jdt]: instantiate k_0 in cgs
        [4kl]: instantiate j_0 in iuh
          [ji1]: proved (CVCL)
```



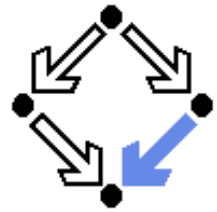
Binary Search: Condition A



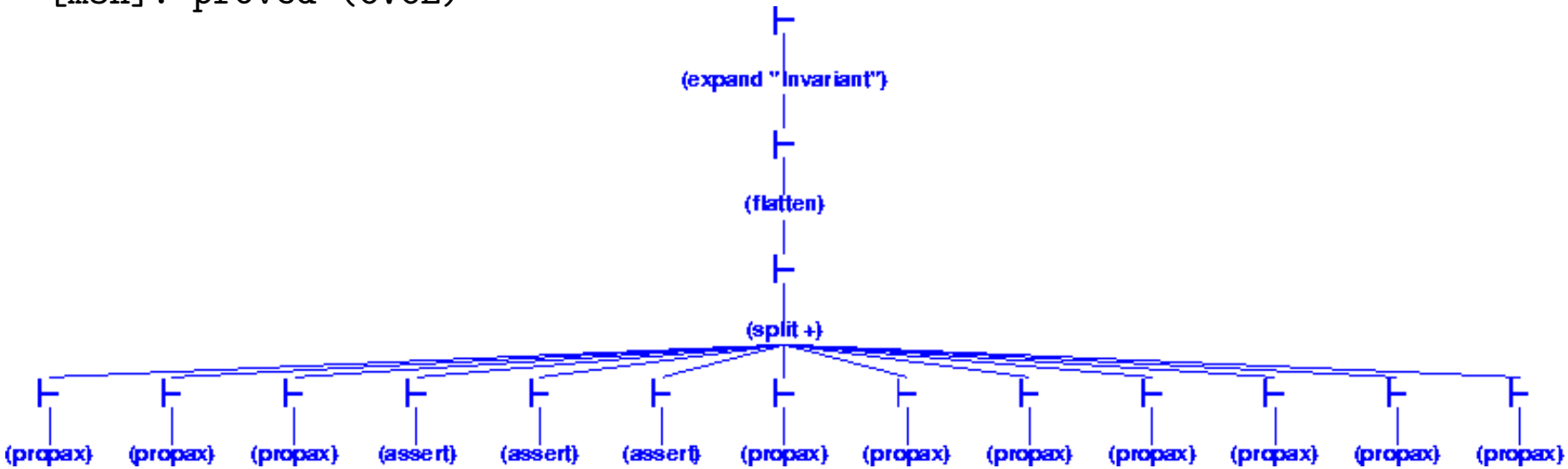
```
[bca]: expand Input, Invariant in ne5
  [fuo]: scatter
    [bxg]: simplify
      [ft4]: proved (Goal cqi is TRUE)
    [cxg]: proved (Goal cqi is TRUE)
    [dxg]: proved (Goal cqi is TRUE)
    [exg]: simplify
      [khw]: proved (Goal cqi is TRUE)
```



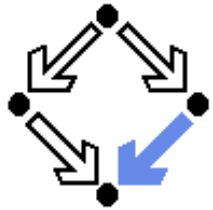
Binary Search: Condition B1



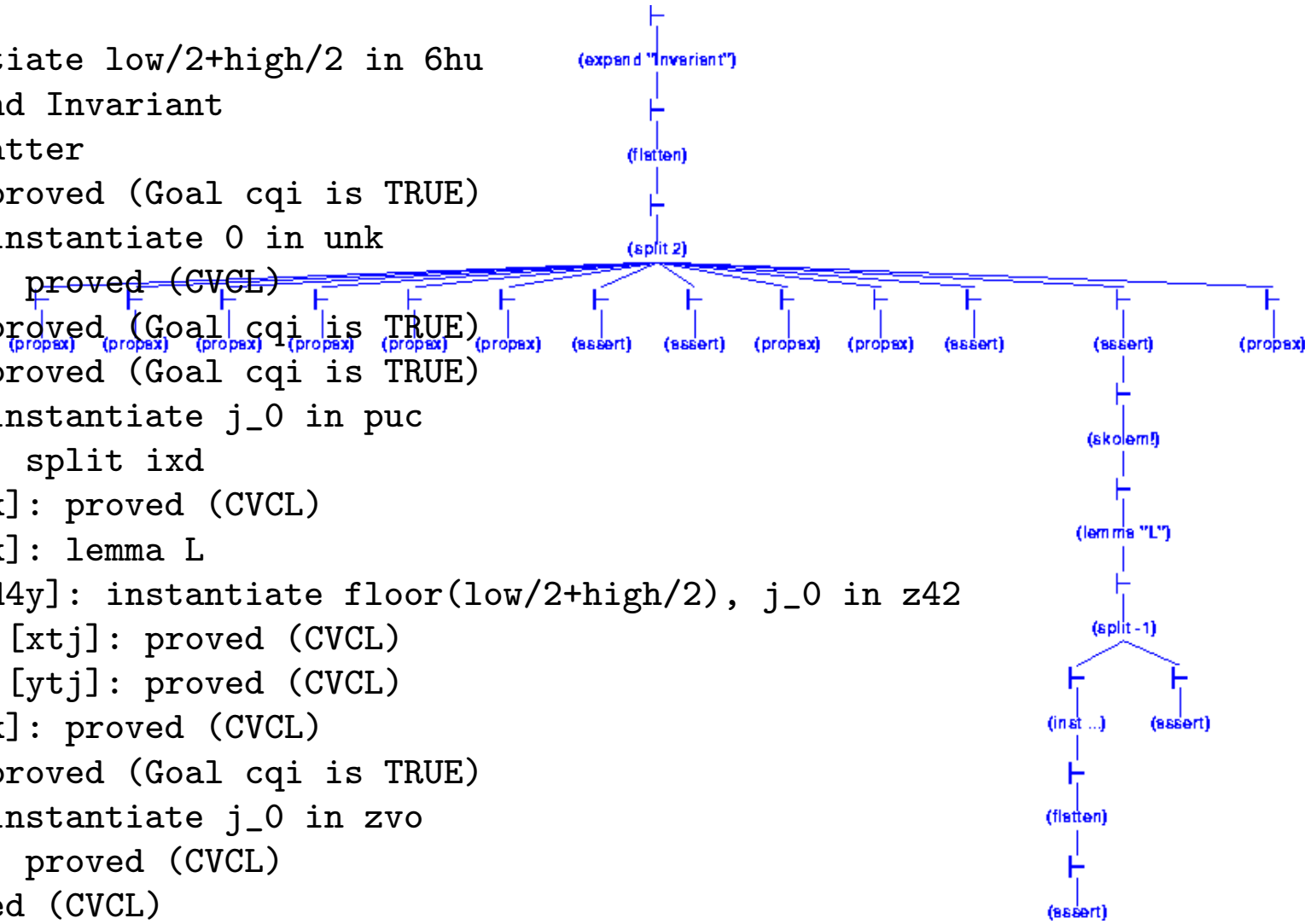
```
[p1b]: expand Invariant
  [lf6]: decompose
    [p6i]: instantiate low/2+high/2 in 6hu
      [l3h]: scatter
        [p3f]: proved (Goal cqi is TRUE)
        [q3f]: proved (Goal cqi is TRUE)
        [r3f]: proved (Goal cqi is TRUE)
        [s3f]: instantiate 0 in unk
          [asu]: proved (CVCL)
        [t3f]: proved (Goal cqi is TRUE)
      [m3h]: proved (CVCL)
```



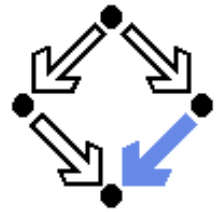
Binary Search: Conditions B2/B3



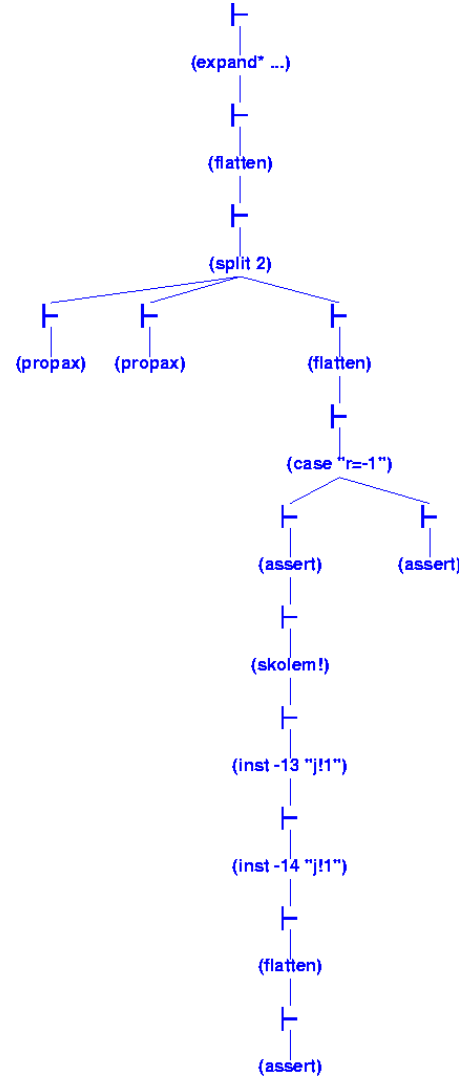
```
[q1b]: instantiate low/2+high/2 in 6hu
  [s1x]: expand Invariant
    [a1y]: scatter
      [cch]: proved (Goal cqi is TRUE)
      [dch]: instantiate 0 in unk
        [xmk]: proved (CVCL)
      [ech]: proved (Goal cqi is TRUE)
      [fch]: proved (Goal cqi is TRUE)
      [gch]: instantiate j_0 in puc
        [mkm]: split ixd
          [wgk]: proved (CVCL)
          [xgk]: lemma L
            [d4y]: instantiate floor(low/2+high/2), j_0 in z42
              [xtj]: proved (CVCL)
              [ytj]: proved (CVCL)
          [ygk]: proved (CVCL)
      [hch]: proved (Goal cqi is TRUE)
      [ich]: instantiate j_0 in zvo
        [1iy]: proved (CVCL)
  [t1x]: proved (CVCL)
```



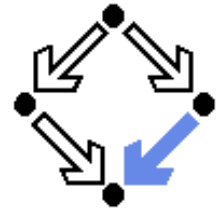
Binary Search: Condition C



```
[dca]: expand Invariant, Output
  [tvy]: scatter
    [dcu]: instantiate j_0 in zvo
      [t4c]: instantiate j_0 in puc
        [dcu]: proved (CVCL)
    [ecu]: goal ueq
      [kel]: scatter
        [ulf]: proved (Goal cqi is TRUE)
        [vlf]: instantiate j_0 in zvo
          [rzb]: instantiate j_0 in puc
            [vor]: proved (CVCL)
```

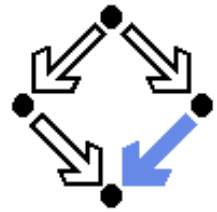


Axiomatic Definition of Arrays

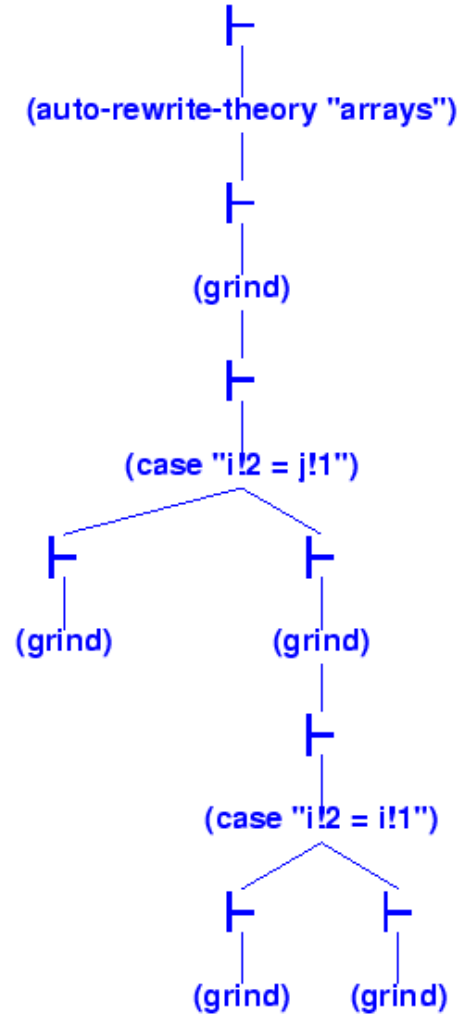


```
[2cg]: scatter
  [6zs]: assume length(put(a_0, i_0, e1_0)) = length(a_0) AND length(put(a_0, j_0, e2_0)) = length(a_0)
    [2ji]: assume length(put(put(a_0, i_0, e1_0), j_0, e2_0)) = length(a_0) AND
                  length(put(put(a_0, j_0, e2_0), i_0, e1_0)) = length(a_0)
      [6gd]: decompose
        [2yw]: instantiate put(put(a_0, j_0, e2_0), i_0, e1_0), put(put(a_0, i_0, e1_0), j_0, e2_0) in hwd
          [6ll]: goal ppq
            [2p4]: scatter
              [6ip]: case i_0 = i_1
                [2oi]: instantiate put(a_0, j_0, e2_0), i_1, e1_0 in jkt
                  [64s]: split jw2
                    [2vd]: proved (CVCL)
                    [3vd]: instantiate put(a_0, i_1, e1_0), j_0, i_1, e2_0 in gkp
                      [gin]: split 514
                        [mbq]: proved (CVCL)
                        [nbq]: proved (CVCL)
                        [obq]: instantiate a_0, i_1, e1_0 in jkt
                          [er1]: proved (CVCL)
                [3oi]: case j_0 = i_1
                  [gvb]: instantiate put(a_0, i_0, e1_0), j_0, e2_0 in jkt
                    [m5b]: split blu
                      [wtx]: proved (CVCL)
                      [xtx]: instantiate put(a_0, j_0, e2_0), i_0, j_0, e1_0 in gkp
                        [dc3]: split nr2
                          [xu6]: proved (CVCL)
                          [yu6]: proved (CVCL)
                          [zu6]: instantiate a_0, j_0, e2_0 in jkt
                            [rzj]: proved (CVCL)
```

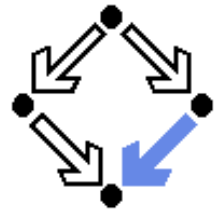
Axiomatic Definition of Arrays (Contd)



```
                [hvb]: instantiate put(a_0, j_0, e2_0), i_0, i_1, e1_0 i
                  [ti2]: split lyr
                    [hgp]: proved (CVCL)
                    [igp]: proved (CVCL)
                    [jgp]: instantiate put(a_0, i_0, e1_0), j_0, i_1, e2
                      [beq]: split 36c
                        [j3l]: proved (CVCL)
                        [k3l]: proved (CVCL)
                        [l3l]: instantiate a_0, i_0, i_1, e1_0 in gkp
                          [p1h]: split srs
                            [lx5]: proved (CVCL)
                            [mx5]: proved (CVCL)
                            [nx5]: instantiate a_0, j_0, i_1, e2_0 in gk
                              [4lv]: proved (CVCL)
  [ahd]: split
    [crr]: instantiate put(a_0, i_0, e1_0), j_0, e2_0 in 4qj
      [qch]: proved (CVCL)
    [drr]: instantiate put(a_0, j_0, e2_0), i_0, e1_0 in 4qj
      [xnb]: proved (CVCL)
[3ji]: split
  [gmn]: instantiate a_0, i_0, e1_0 in 4qj
    [mfr]: proved (CVCL)
  [hmn]: instantiate a_0, j_0, e2_0 in 4qj
    [t4l]: proved (CVCL)
```

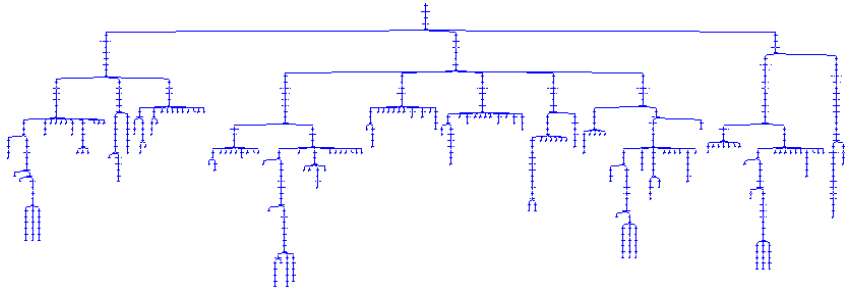


Next Steps



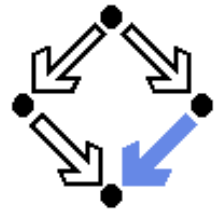
Some more refinements.

- Automatic instantiation before proof state simplification.
  - Instantiate universal assumptions and existential goals.
  - Instantiation values are those ground terms that occur in proof state.
  - Will cut proof trees that only depend on trivial instantiations.
- Larger evaluation example.
  - Client/server system.
- User and reference manual.



Release in the second half of the semester.

Acknowledgements



The ProofNavigator is based on various pieces of open source software.

- CVC Lite 2.0
- RIACA OpenMath Library 2.0
- General Purpose Hash Function Algorithms Library
- ANTLR 2.7.6b2
- Eclipse Standard Widget Toolkit 3.1.2
- Mozilla Browser 1.7.8 GTK2
- GIMP Toolkit GTK+ 2.X
- Java Development Kit 5.0
- Tango Icon Library

It is important to contribute to the open source pool.