Proving Calculational Proofs Correct

Andrew T. Walter, Ankit Kumar, Panagiotis Manolios

Calculational Proofs in ACL2s, doi: 10.48550/arXiv.2307.12224

ACL2s Systems Programming, ACL2 2022, doi: 10.4204/EPTCS.359.12

History

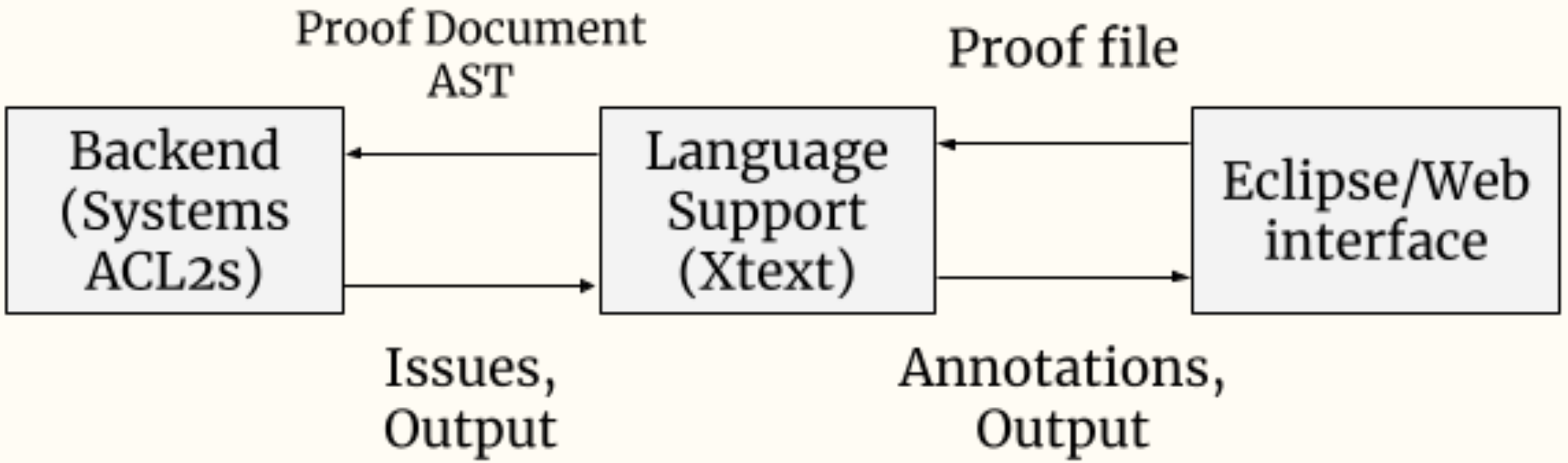
- CS2800: Logic and Computation
- Challenge: teach proof vs. wishful thinking
- Common mistakes -> proof format -> checks

Who should care

- Teaching using ACL2?
- Like the proofs you see?
- Want to generate proof-builder instructions?

Demo

Architecture



Backend

- Phase 1: Local checks
- Phase 2: `proof-builder`

proof-builder

```
ACL2S !>(verify (=> (natp x) (intp x)))
->: (casesplit (< x 0))

Creating one new goal:  (acl2::main . 1).
->: s

The proof of the current goal, acl2::main, has been completed.  However,
the following subgoals remain to be proved:
  (acl2::main . 1).
Now proving (acl2::main . 1).
->: th
*** Top-level hypotheses:
1. (<= 0 x)

The current subterm is:
(implies (natp x) (integerp x))
->: s
*!*!*!*!*!*!* All goals have been proved! *!*!*!*!*!*!*
```

proof-builder pain points

```
ACL2S !>(verify (=> (natp x) (intp x)))
->: pro
->: th
*** Top-level hypotheses:
1. (natp x)

The current subterm is:
(integerp x)
->: (retain 1)
*** NO CHANGE *** -- All hypotheses are to be retained.

Macroexpansion of instruction (acl2-pc::retain 1) failed!
->:
```

Keeping track of hypotheses

```
ACL2S !>(verify (=> (natp x) (intp x)))
->: pro
->: (claim (and y z) :hints none)
;; ...
->: th
*** Top-level hypotheses:
1. (natp x)
2. y
3. z

The current subterm is:
(integerp x)
->:
```

Keeping track of goals

```
(verify (implies (and (tlp x) (tlp y) (endp y))
                 (== (ap x y) x)))
->: induct
;; ... ;; first we must discharge the induction case
->: pro
;; ...
->: th
*** Top-level hypotheses:
1. (and (tlp x) (tlp y))
2. (not (endp x))
3. (implies (and (tlp (cdr x)) (tlp y) (endp y))
            (equal (ap (cdr x) y) (cdr x)))
4. (tlp x)
5. (tlp y)
6. (endp y)

The current subterm is:
(equal (ap x y) x)
```

Future Work

- Improved UIs
- Expert mode
- More proof strategies
- AI proof tutor?

Thanks!

Questions?

Reach out: walter.a@northeastern.edu

Backup Slides

Definec

```
1  (definec ap (a :tl b :tl) :tl
2    (if (endp a) b
3      (cons (car a) (ap (cdr a) b))))
4
5  ;; Expands into...
6  (defun ap (a b)
7    (declare (xargs :guard (and (true-listp a) (true-listp b)) ...))
8    (mbe :logic
9         (if (and (true-listp a) (true-listp b))
10            (if (endp a) b
11              (cons (car a) (ap (cdr a) b)))
12          (acl2s-true-list-undefined 'ap (list a b)))
13        :exec ...))
```

Contract Completion

```
(definec ap (a :tl b :tl) :tl
  (if (endp a) b
    (cons (car a) (ap (cdr a) b))))
;; recall that the above will result in a defun with a guard of
;; (and (true-listp a) (true-listp b))

;; The guard obligation for the statement
(implies (and (tlp x) (== (ap y y) x) (tlp y))
         (== (ap x y) (ap z y)))
;; is (and (tlp y) (tlp z)), since (tlp x) is already known to be true
;; before it is used in a call of ap, but (tlp y) and (tlp z) are not.
;; (order matters!)

;; The below is contract-completed.
(implies (and (tlp x) (tlp y) (tlp z) (== (ap y y) x))
         (== (ap x y) (ap z y)))
```

Proving a step

```
;; Below are instructions for the step:
(rv (ap x y))
== { C1, def ap }
(rv y)
;; The step is turned into a claim with sub-instructions
(:claim-simple (== (rv (ap x y)) (rv y)) :hints
  (("Goal" :instructions
    (:pro-or-skip
     ;; We already know that the guard obligations for this statement hold,
     ;; since we proved this in Phase 1. Should be easy for ACL2 to prove.
     (:claim (and (true-listp x) (true-listp y)) :do-not-flatten t)
     ;; Only keep the hypotheses corresponding to used context items,
     ;; type-hypothesis like context items & the guard obligation claim.
     (:retain-or-skip 5 1 2 3)
     ;; Enable whatever rules the user's hints allow them to use, plus cont
     (:in-theory (union-theories (theory 'contract-theory) '(ap)))
     ;; Attempt to automatically prove, only using simplification.
     (:finish :bash)))))
```

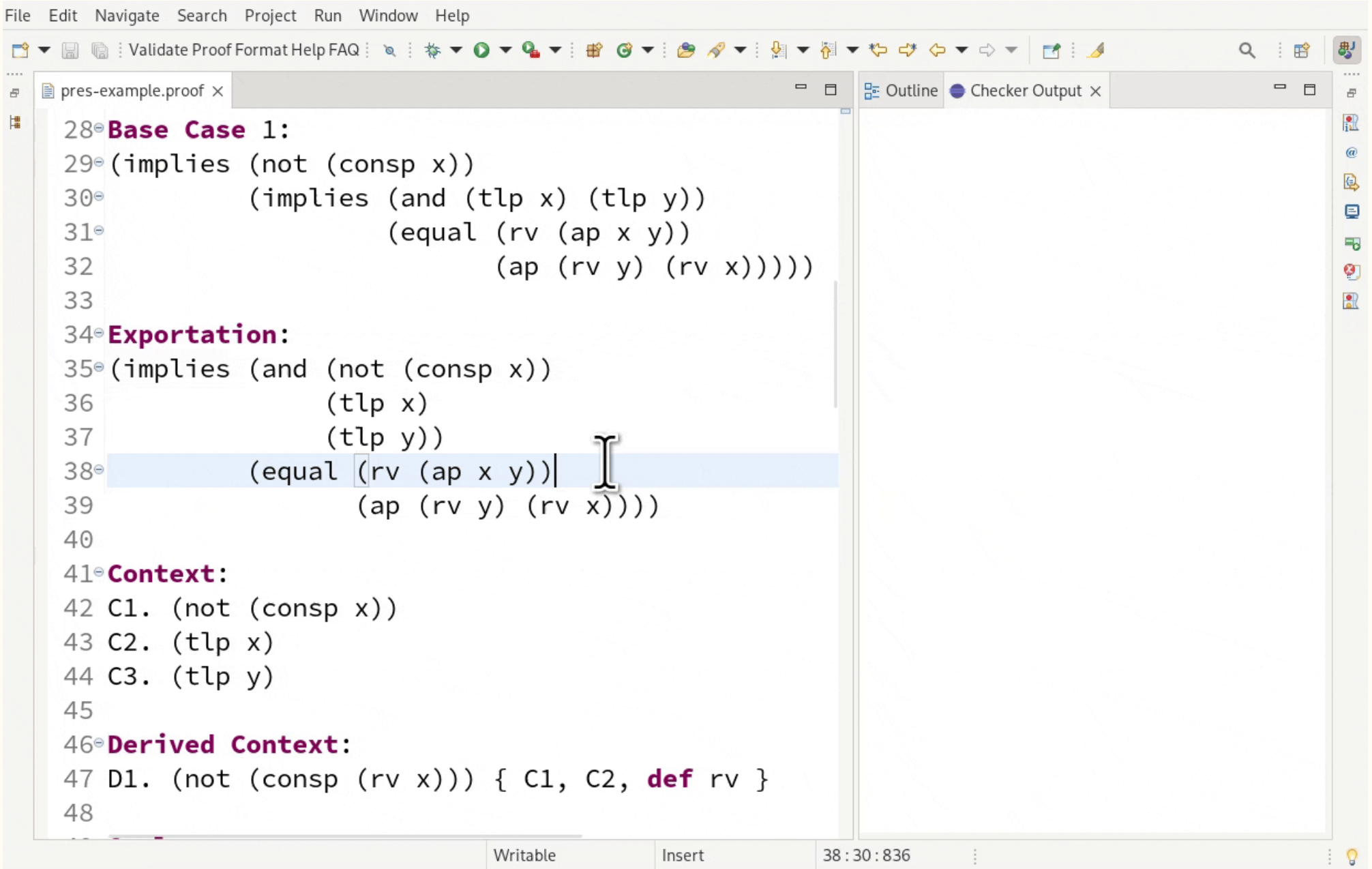
non-inductive proof

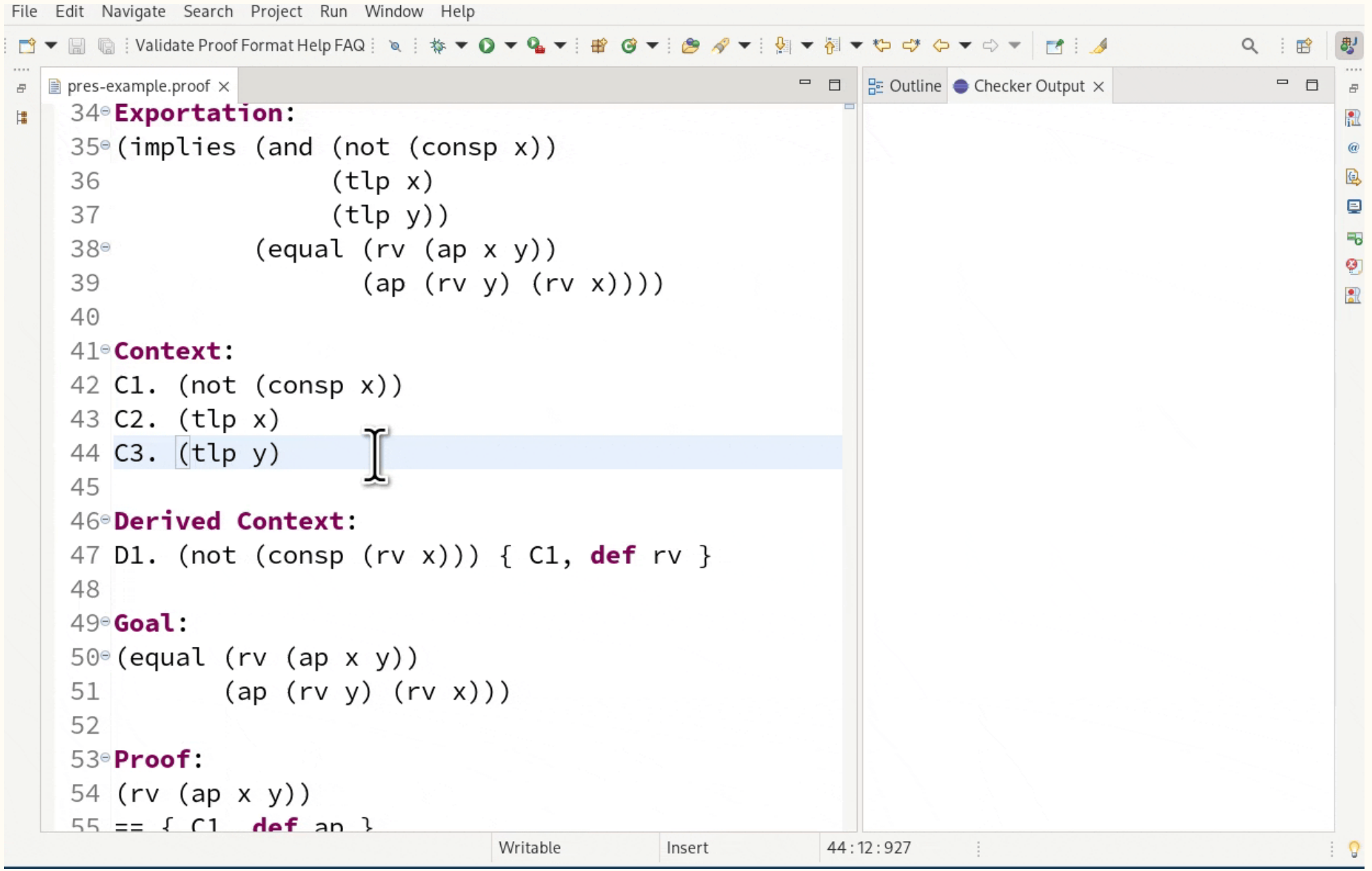
```
(thm
 (implies (and (not (consp x)) (tlp x) (tlp y))
          (equal (rv (ap x y)) (ap (rv y) (rv x))))
 :hints (("Goal" :instructions
   (:pro-or-skip
    ;; Add a claim for each derived context item (proving = t)
    ;; ...
    ;; Add a claim for each step
    ;; ...
    ;; Turn the proof builder goal back into an implication
    :demote
    ;; Prove that the derived context items + steps imply the proof stmt
    (:finish (:repeat-until-done (:split-in-theory min-executable-theory))))))
```

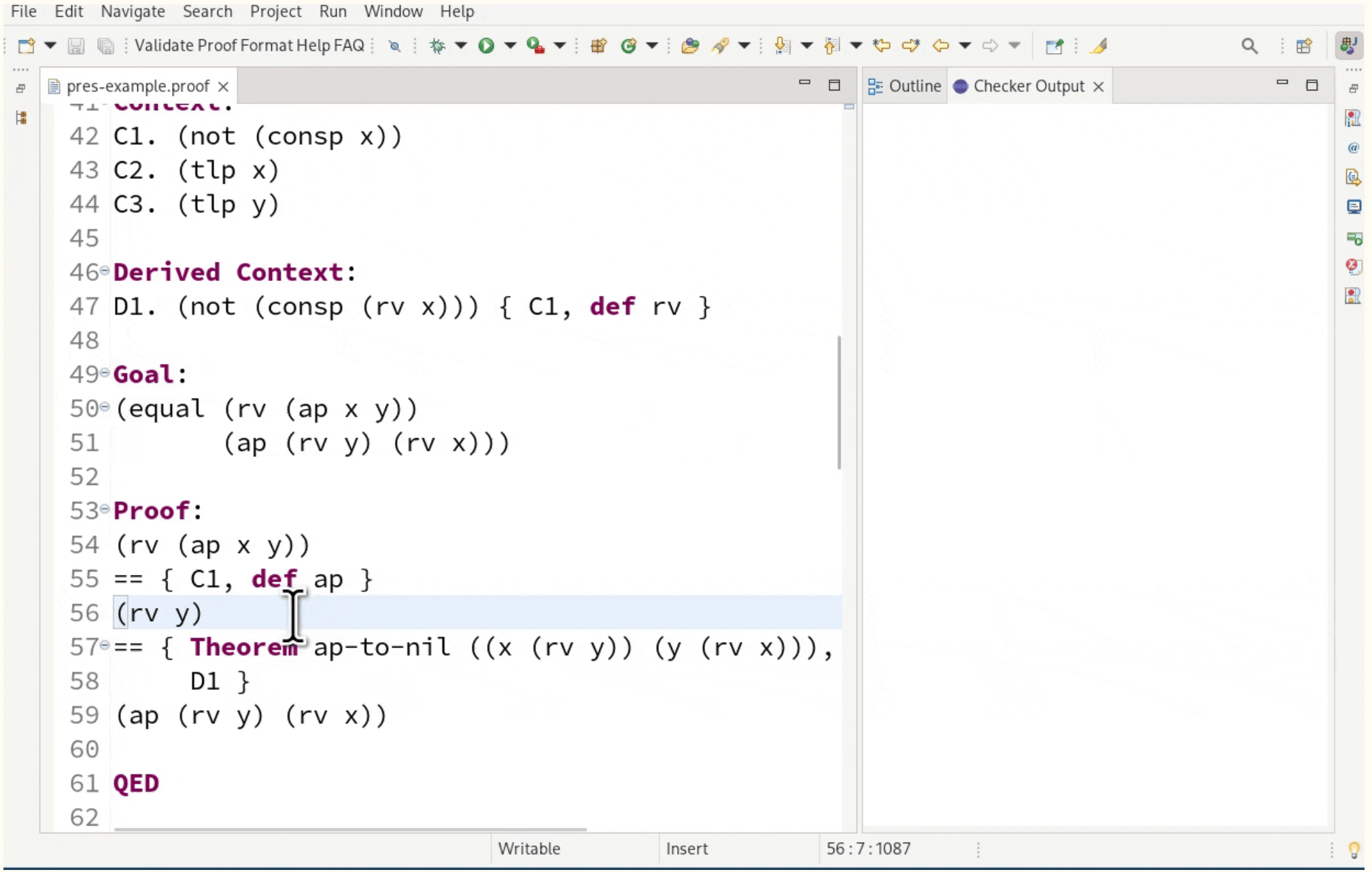
inductive proof

```
(encapsulate ()
  ;; Local defthm for each subproof
  (local (defthm |ap-rv-base case1| ...))
  (local (defthm |ap-rv-induction case1| ...))
  ;; prove the top-level theorem
  (thm (implies (and (tlp x) (tlp y))
                (equal (rv (ap x y)) (ap (rv y) (rv x))))
       :hints (("Goal" :instructions
        (:pro-or-skip
         (:induct (tlp x))
         ;; change to first induction proof obligation
         (:cg-or-skip (acl2::main . 1))
         (:finish
           ;; Demote and then prove equal to the exported statement
           :demote
           (:= (implies (and (not (consp x)) (tlp x) (tlp y))
                        (equal (rv (ap x y)) (ap (rv y) (rv x)))))
           ;; then use the local defthm above
           :by |ap-rv-base case 1|)
```

Error Messages


File Edit Navigate Search Project Run Window Help
Validate Proof Format Help FAQ
pres-example.proof
Outline
Checker Output
28 Base Case 1:
29 (implies (not (consp x))
30 (implies (and (tlp x) (tlp y))
31 (equal (rv (ap x y))
32 (ap (rv y) (rv x)))))
33
34 Exportation:
35 (implies (and (not (consp x))
36 (tlp x)
37 (tlp y))
38 (equal (rv (ap x y))
39 (ap (rv y) (rv x))))
40
41 Context:
42 C1. (not (consp x))
43 C2. (tlp x)
44 C3. (tlp y)
45
46 Derived Context:
47 D1. (not (consp (rv x))) { C1, C2, def rv }
48
Writable
Insert
38 : 30 : 836


File Edit Navigate Search Project Run Window Help
Validate Proof Format Help FAQ
pres-example.proof
Outline
Checker Output
34 Exportation:
35 (implies (and (not (consp x))
36 (tlp x)
37 (tlp y))
38 (equal (rv (ap x y))
39 (ap (rv y) (rv x))))
40
41 Context:
42 C1. (not (consp x))
43 C2. (tlp x)
44 C3. (tlp y)
45
46 Derived Context:
47 D1. (not (consp (rv x))) { C1, def rv }
48
49 Goal:
50 (equal (rv (ap x y))
51 (ap (rv y) (rv x)))
52
53 Proof:
54 (rv (ap x y))
55 == { C1, def ap }
Writable
Insert
44 : 12 : 927


File Edit Navigate Search Project Run Window Help
Validate Proof Format Help FAQ
pres-example.proof
Outline
Checker Output
42 C1. (not (consp x))
43 C2. (tlp x)
44 C3. (tlp y)
45
46 Derived Context:
47 D1. (not (consp (rv x))) { C1, def rv }
48
49 Goal:
50 (equal (rv (ap x y))
51 (ap (rv y) (rv x)))
52
53 Proof:
54 (rv (ap x y))
55 == { C1, def ap }
56 (rv y)
57 == { Theorem ap-to-nil ((x (rv y)) (y (rv x))),
58 D1 }
59 (ap (rv y) (rv x))
60
61 QED
62
Writable
Insert
56 : 7 : 1087