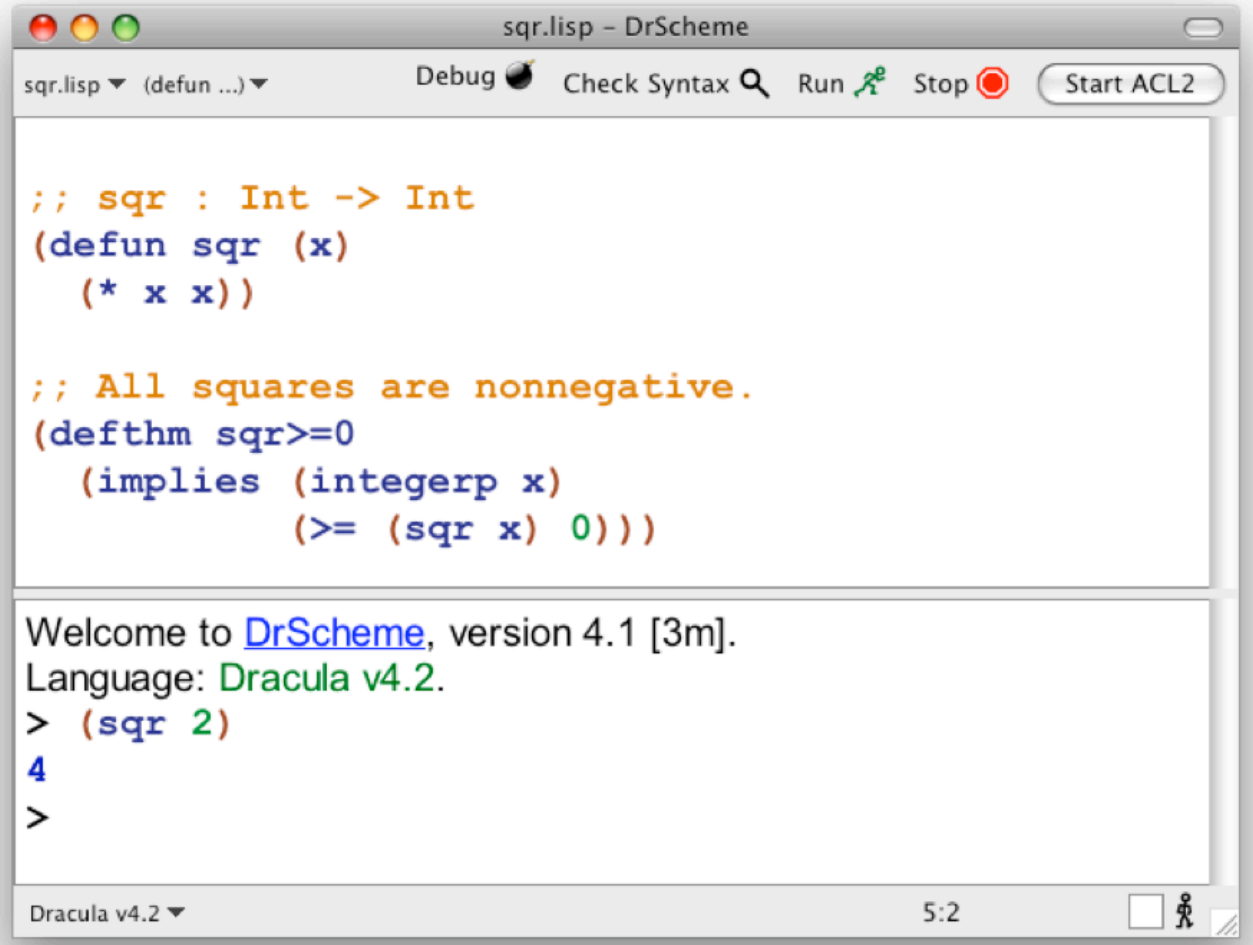
DoubleCheck Your Theorems

Carl Eastlund
cce@ccs.neu.edu

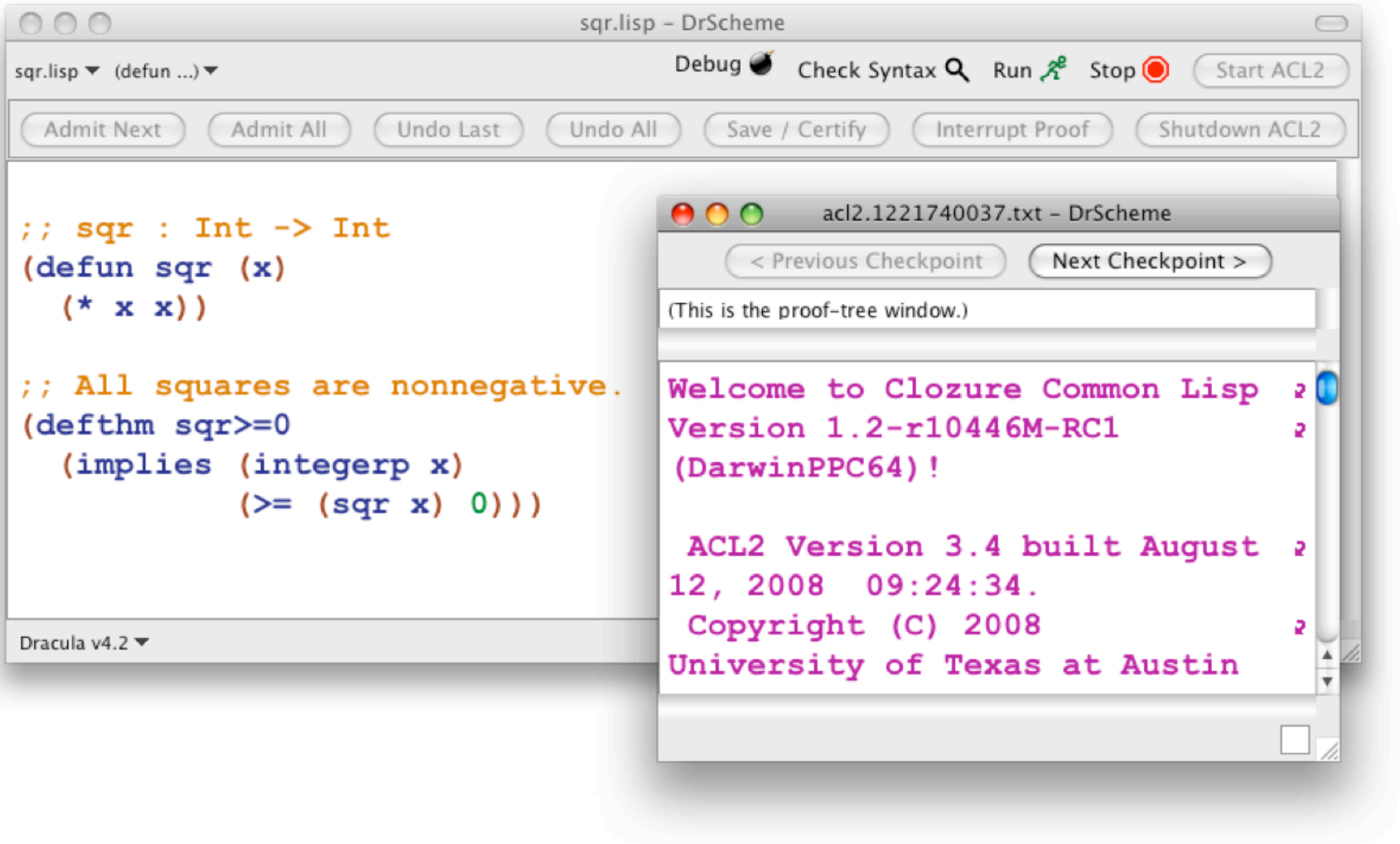
Northeastern University
Boston, Massachusetts

A Tale Of Two Students

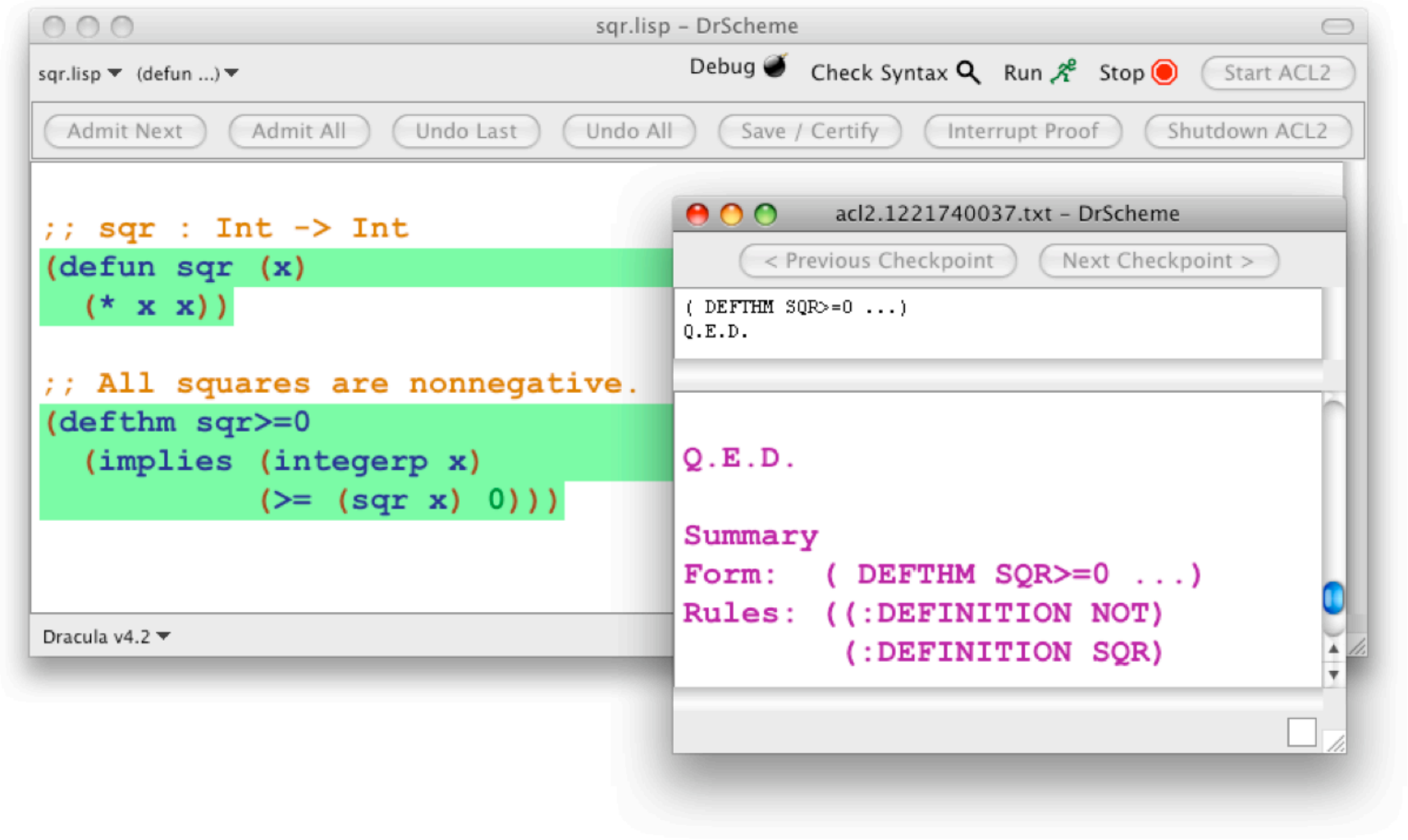
Happy Student



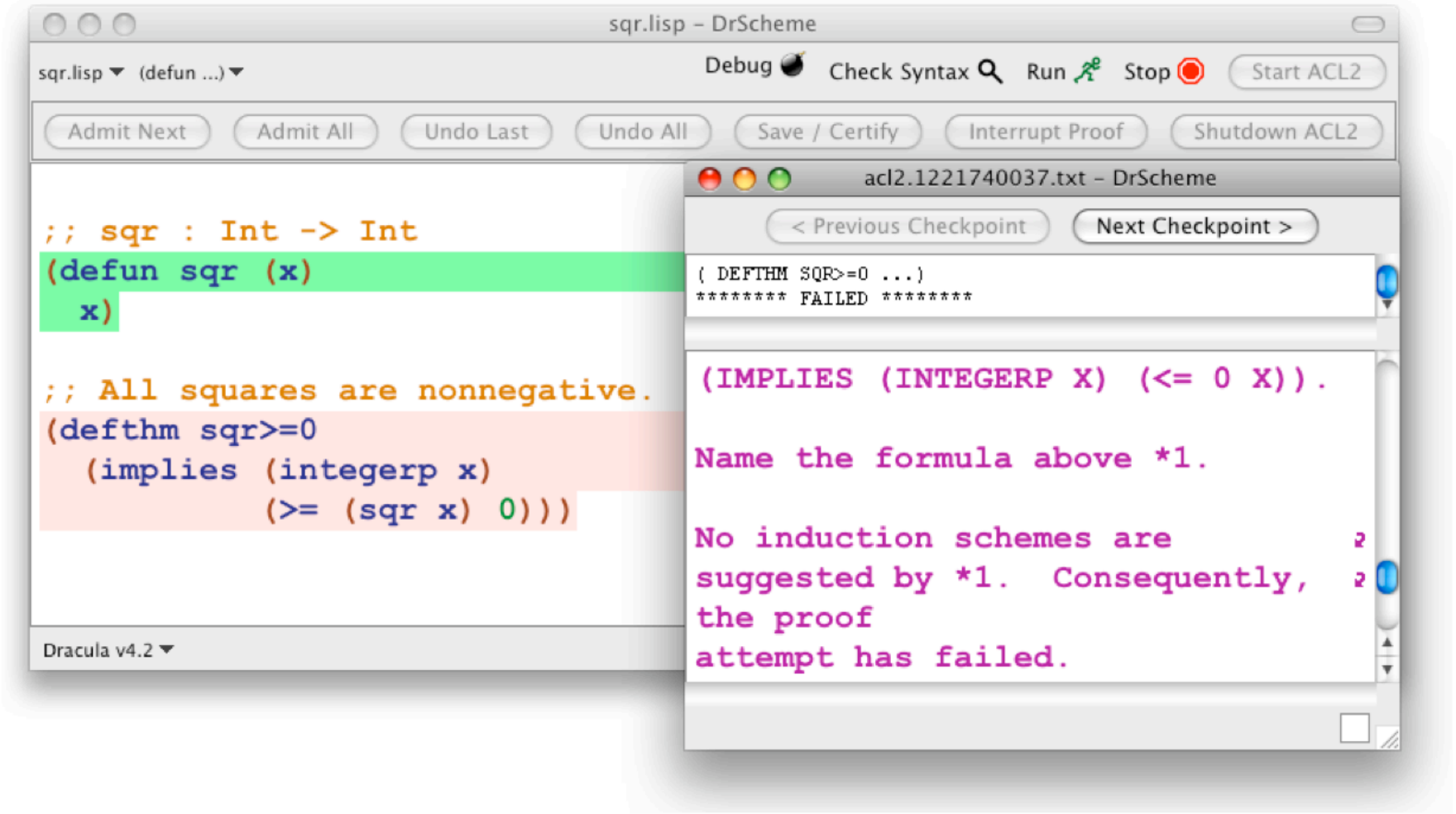
Happy Student



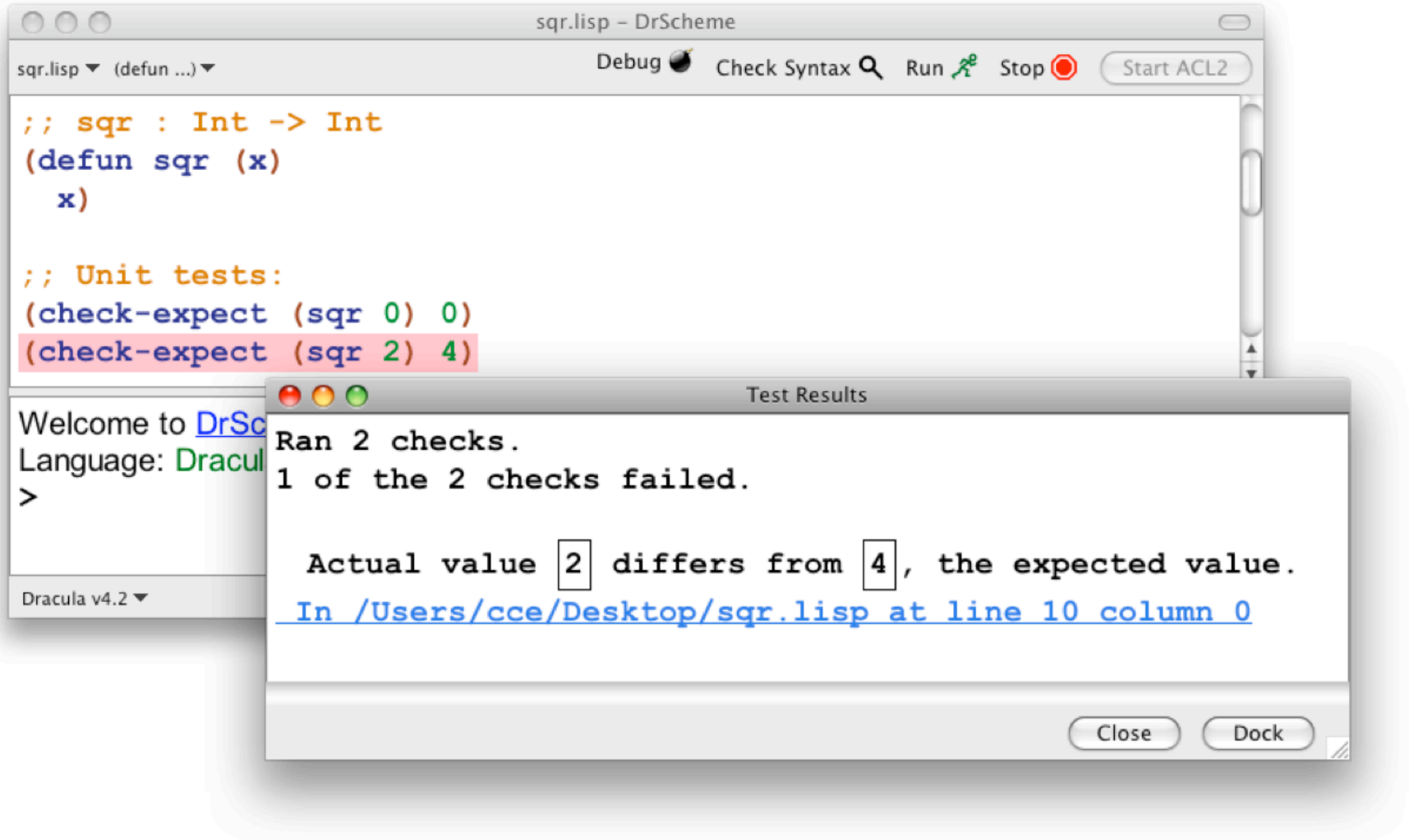
Happy Student



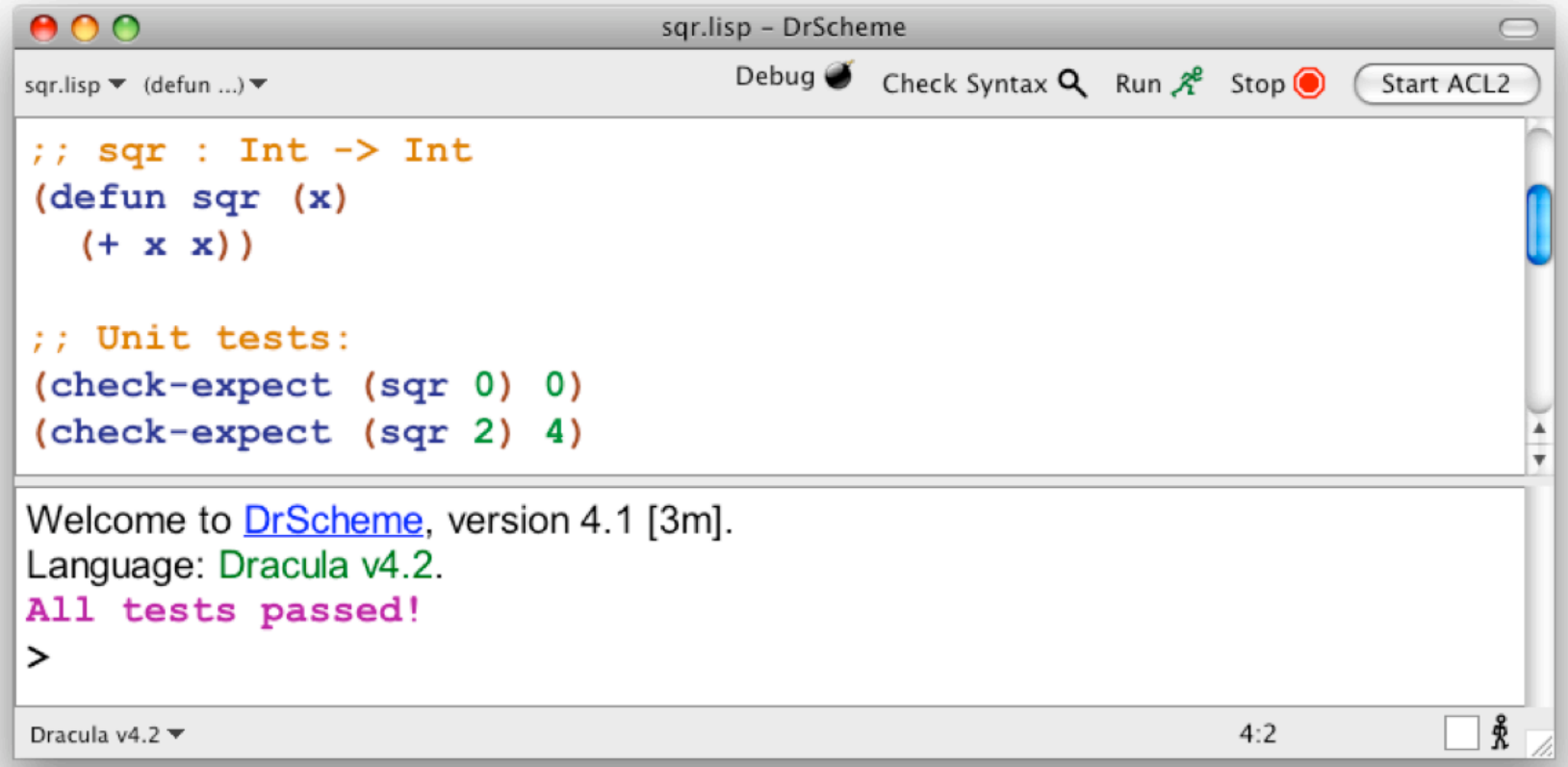
Sad Student


sqr.lisp – DrScheme
sqr.lisp
(defun ...)
Debug
Check Syntax
Run
Stop
Start ACL2
Admit Next
Admit All
Undo Last
Undo All
Save / Certify
Interrupt Proof
Shutdown ACL2
;; sqr : Int -> Int
(defun sqr (x)
x)
;; All squares are nonnegative.
(defthm sqr>=0
(implies (integerp x)
(>= (sqr x) 0)))
Dracula v4.2
acl2.1221740037.txt – DrScheme
< Previous Checkpoint
Next Checkpoint >
(DEFTHM SQR>=0 ...)
******** FAILED ********
(IMPLIES (INTEGERP X) (<= 0 X)).
Name the formula above *1.
No induction schemes are
suggested by *1. Consequently,
the proof
attempt has failed.

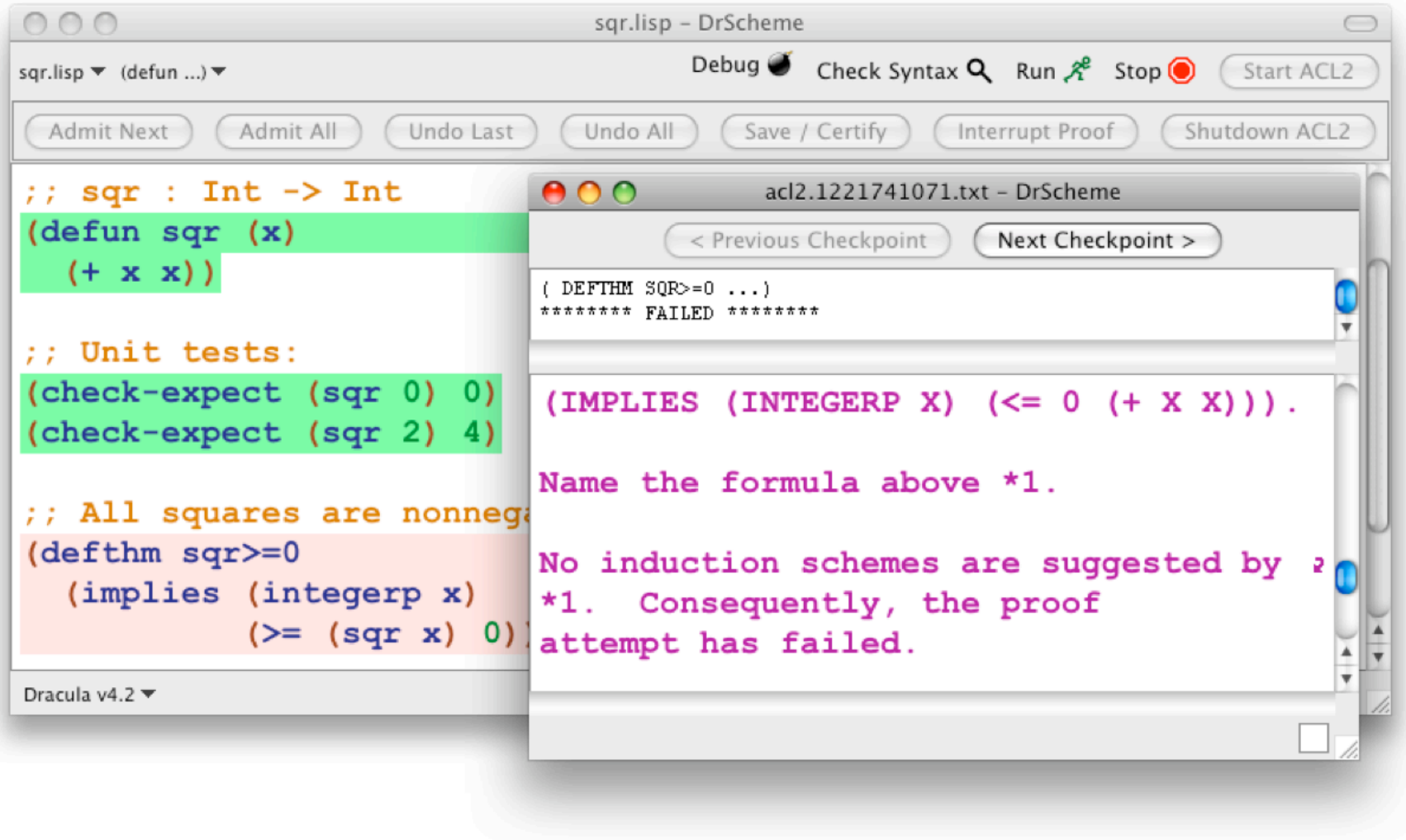
Sad Student


sqr.lisp – DrScheme
sqr.lisp
(defun ...)
Debug
Check Syntax
Run
Stop
Start ACL2
;; sqr : Int -> Int
(defun sqr (x)
x)
;; Unit tests:
(check-expect (sqr 0) 0)
(check-expect (sqr 2) 4)
Welcome to DrSc
Language: Dracul
>
Dracula v4.2
Test Results
Ran 2 checks.
1 of the 2 checks failed.
Actual value 2 differs from 4, the expected value.
In /Users/cce/Desktop/sqr.lisp at line 10 column 0
Close
Dock

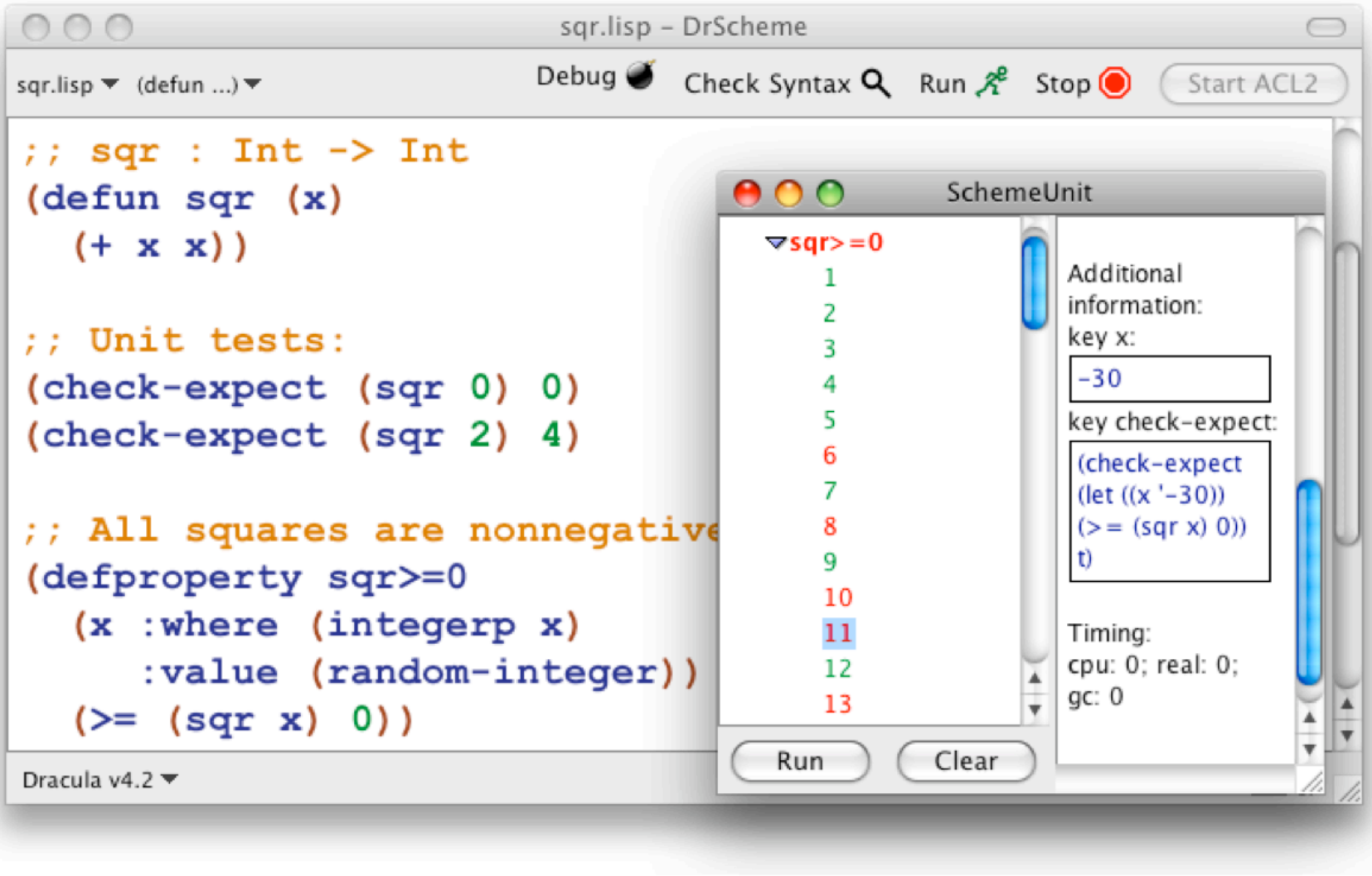
Sad Student


sqr.lisp – DrScheme
sqr.lisp
(defun ...)
Debug
Check Syntax
Run
Stop
Start ACL2
;; sqr : Int -> Int
(defun sqr (x)
(+ x x))
;; Unit tests:
(check-expect (sqr 0) 0)
(check-expect (sqr 2) 4)
Welcome to DrScheme, version 4.1 [3m].
Language: Dracula v4.2.
All tests passed!
>
Dracula v4.2
4:2

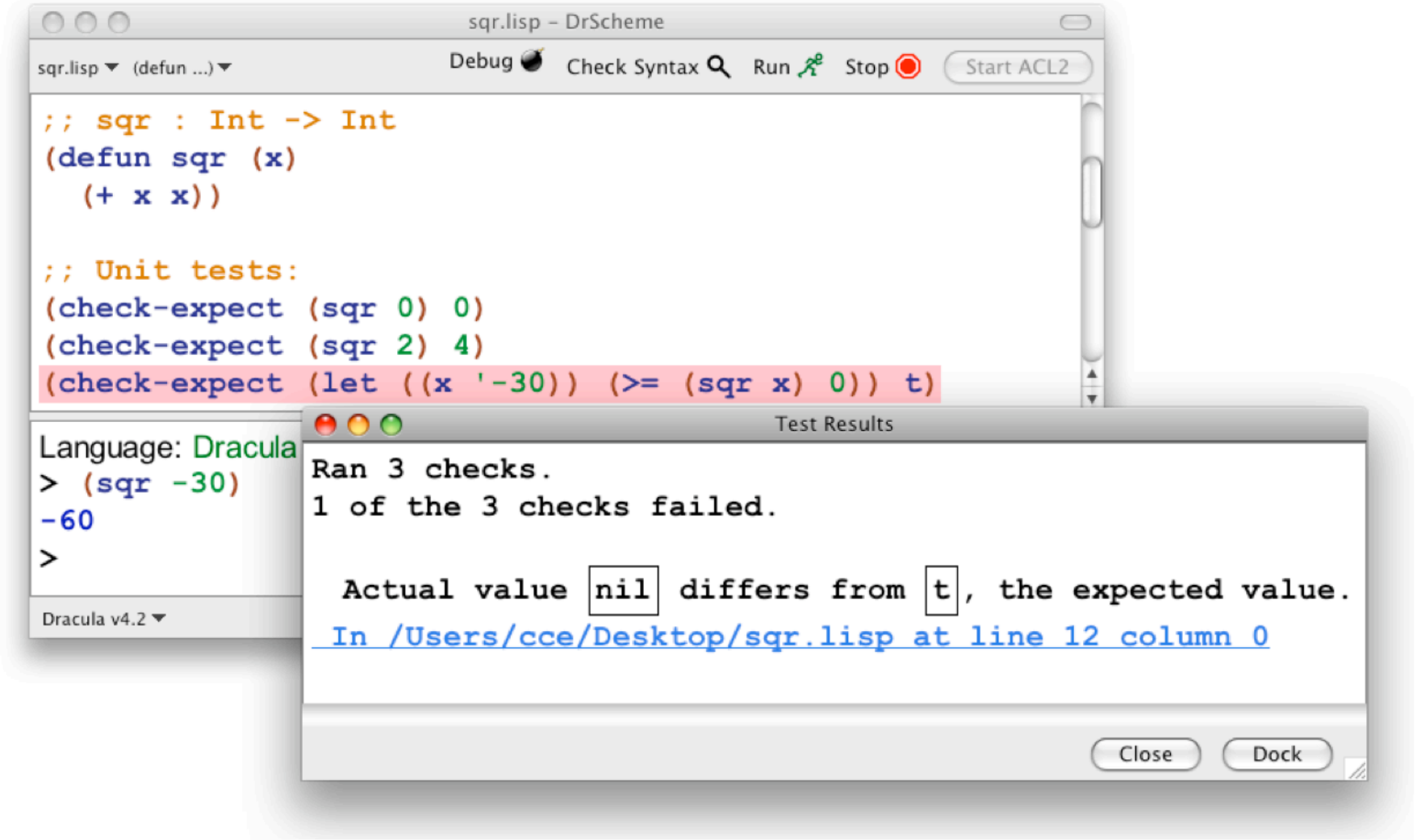
Mad Student


sqr.lisp – DrScheme
sqr.lisp
(defun ...)
Debug
Check Syntax
Run
Stop
Start ACL2
Admit Next
Admit All
Undo Last
Undo All
Save / Certify
Interrupt Proof
Shutdown ACL2
;; sqr : Int -> Int
(defun sqr (x)
(+ x x))
;; Unit tests:
(check-expect (sqr 0) 0)
(check-expect (sqr 2) 4)
;; All squares are nonnega
(defthm sqr>=0
(implies (integerp x)
(>= (sqr x) 0)
Dracula v4.2
acl2.1221741071.txt – DrScheme
< Previous Checkpoint
Next Checkpoint >
(DEFTHM SQR>=0 ...)
******** FAILED ********
(IMPLIES (INTEGERP X) (<= 0 (+ X X))).
Name the formula above *1.
No induction schemes are suggested by
*1. Consequently, the proof
attempt has failed.

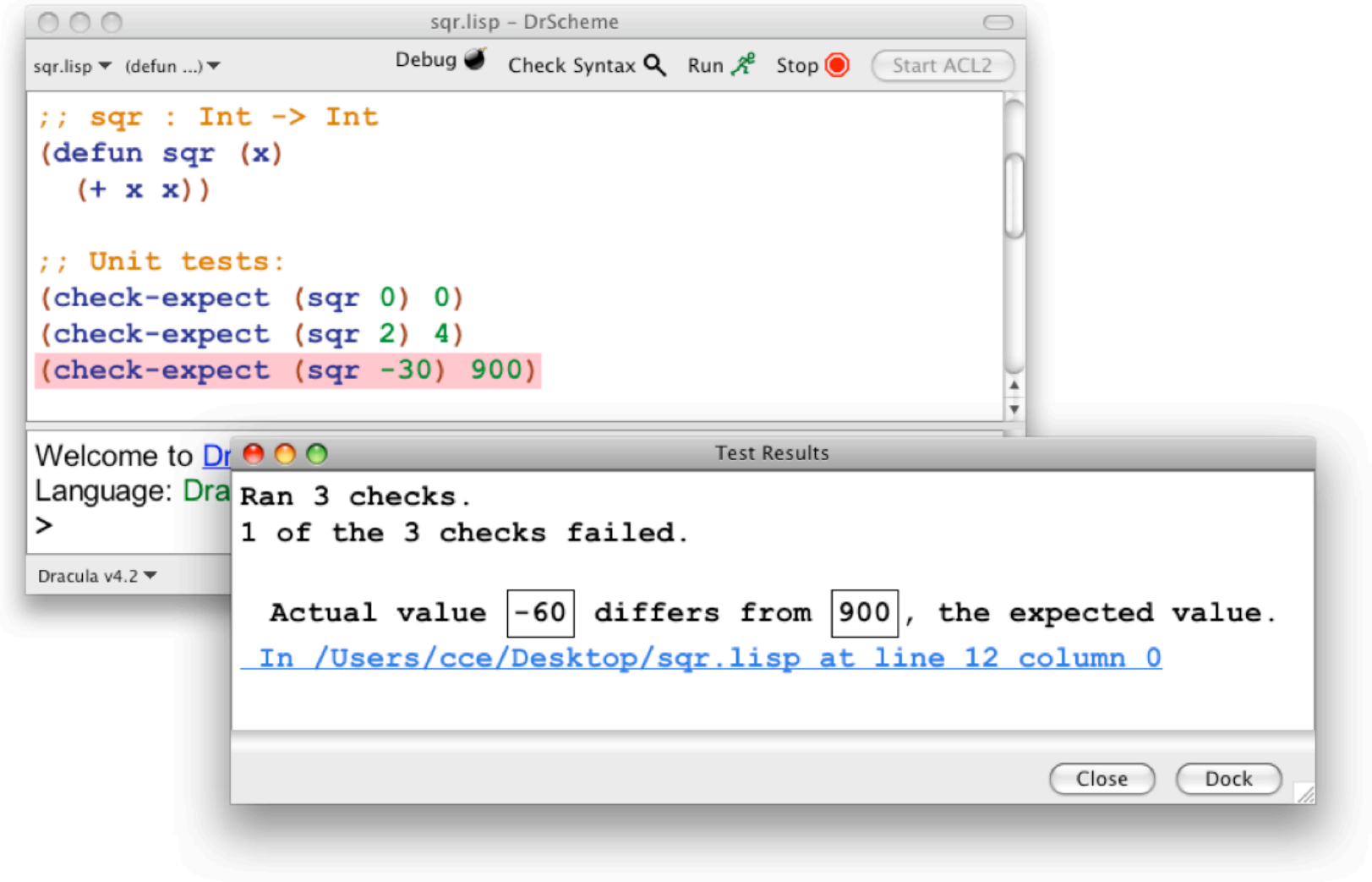
Mad Student


sqr.lisp – DrScheme
sqr.lisp
(defun ...)
Debug
Check Syntax
Run
Stop
Start ACL2
;; sqr : Int -> Int
(defun sqr (x)
(+ x x))
;; Unit tests:
(check-expect (sqr 0) 0)
(check-expect (sqr 2) 4)
;; All squares are nonnegative
(defproperty sqr>=0
(x :where (integerp x)
:value (random-integer))
(>= (sqr x) 0))
Dracula v4.2
SchemeUnit
sqr>=0
1
2
3
4
5
6
7
8
9
10
11
12
13
Additional information:
key x:
-30
key check-expect:
(check-expect
(let ((x '-30))
(>= (sqr x) 0))
t)
Timing:
cpu: 0; real: 0;
gc: 0
Run
Clear

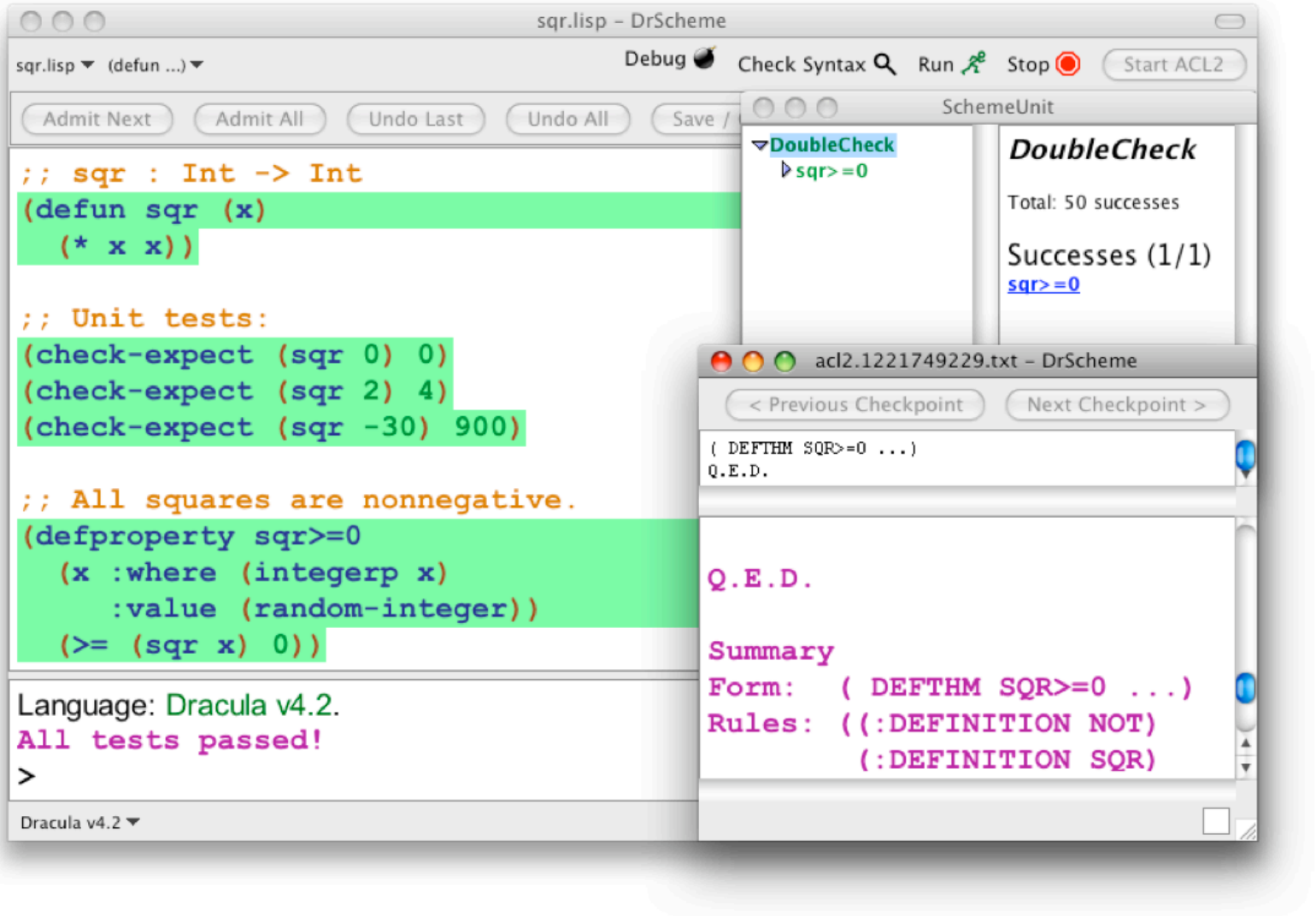
Mad Student


sqr.lisp – DrScheme
sqr.lisp
(defun ...)
Debug
Check Syntax
Run
Stop
Start ACL2
;; sqr : Int -> Int
(defun sqr (x)
(+ x x))
;; Unit tests:
(check-expect (sqr 0) 0)
(check-expect (sqr 2) 4)
(check-expect (let ((x '-30)) (>= (sqr x) 0)) t)
Language: Dracula
> (sqr -30)
-60
>
Dracula v4.2
Test Results
Ran 3 checks.
1 of the 3 checks failed.
Actual value nil differs from t, the expected value.
In /Users/cce/Desktop/sqr.lisp at line 12 column 0
Close
Dock

Mad Student


sqr.lisp – DrScheme
sqr.lisp
(defun ...)
Debug
Check Syntax
Run
Stop
Start ACL2
;; sqr : Int -> Int
(defun sqr (x)
(+ x x))
;; Unit tests:
(check-expect (sqr 0) 0)
(check-expect (sqr 2) 4)
(check-expect (sqr -30) 900)
Welcome to Dr
Language: Dra
>
Dracula v4.2
Test Results
Ran 3 checks.
1 of the 3 checks failed.
Actual value -60 differs from 900, the expected value.
In /Users/cce/Desktop/sqr.lisp at line 12 column 0
Close
Dock

Another Happy Student



Check and DoubleCheck

Check

```
(defthm       sqr>=0

  (implies   (integerp x)
    (>= (sqr x) 0)))
```

DoubleCheck

```
(include-book "doublecheck" :dir :teachpacks)


(defproperty sqr>=0
  (x)
  (implies  (integerp x)
    (>= (sqr x) 0)))


(generate-properties)
```

DoubleCheck

```
(include-book "doublecheck" :dir :teachpacks)


(defproperty sqr>=0
  (x :value (random-integer)
     :where (integerp x))
    (>= (sqr x) 0))


(generate-properties)
```

DoubleCheck

```
(include-book "doublecheck" :dir :teachpacks)


(defproperty sqr>=0 :repeat 1000 :limit 5000
  (x :value (random-integer)
     :where (integerp x))
    (>= (sqr x) 0))


(generate-properties)
```

DoubleCheck

```
(defthm       sqr>=0

  (implies   (integerp x)
    (>= (sqr x) 0)))
```

Random Generators

```
(random-boolean)
(random-char)
(random-string)
(random-symbol)

(random-atom)
(random-sexp)

(random-element-of lst)
```

```
(random-natural)
(random-integer)
(random-rational)
(random-number)

(random-data-size)
(random-between lo hi)
```

Random Generators

```
(random-list-of expr [:size size])
(random-sexp-of expr [:size size])

(random-case expr [:weight weight] ...)

(defrandom name (arg ...) body)
```

Random Generators

```
(random-list-of expr [:size size])
(random-sexp-of expr [:size size])

(random-case expr [:weight weight] ...)

(defrandom name (arg ...) body)


; random-multiset : [Listof X] -> [Listof X]
(defrandom random-multiset (elements)
  (random-case
   nil :weight 1/4
   (cons (random-element-of elements)
         (random-multiset elements))))
```

RealityCheck

- Random testing
  - based on SchemeUnit
  - `defproperty` constructs test suite & all cases
  - `generate-properties` runs all suites
  - random values pulled from lazy stream
- Theorem proving
  - macro-expands to `defthm`
  - generators are vacuous, program-mode

Other Approaches

- Claessen and Hughes. QuickCheck: a lightweight tool for random testing of Haskell programs. ICFP 2000.
- Runciman et al. SmallCheck and Lazy SmallCheck: automatic exhaustive testing for small values. Haskell 2008.
- Berghofer and Nipkow. Random testing in Isabelle/HOL. SEFM 2004.
- Spiridinov and Khurshid. Pythia: automatic generation of counterexamples for ACL2 using Alloy. ACL2 2007.
- Sumners. Checking ACL2 theorems via SAT checking. ACL2 2002.

Thank You.