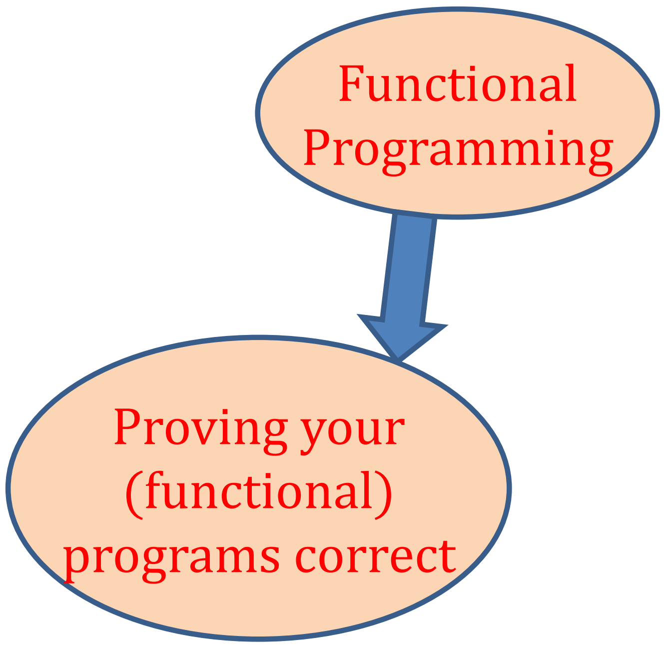
Software Verification

(preview of COS 510 “Programming Languages”)

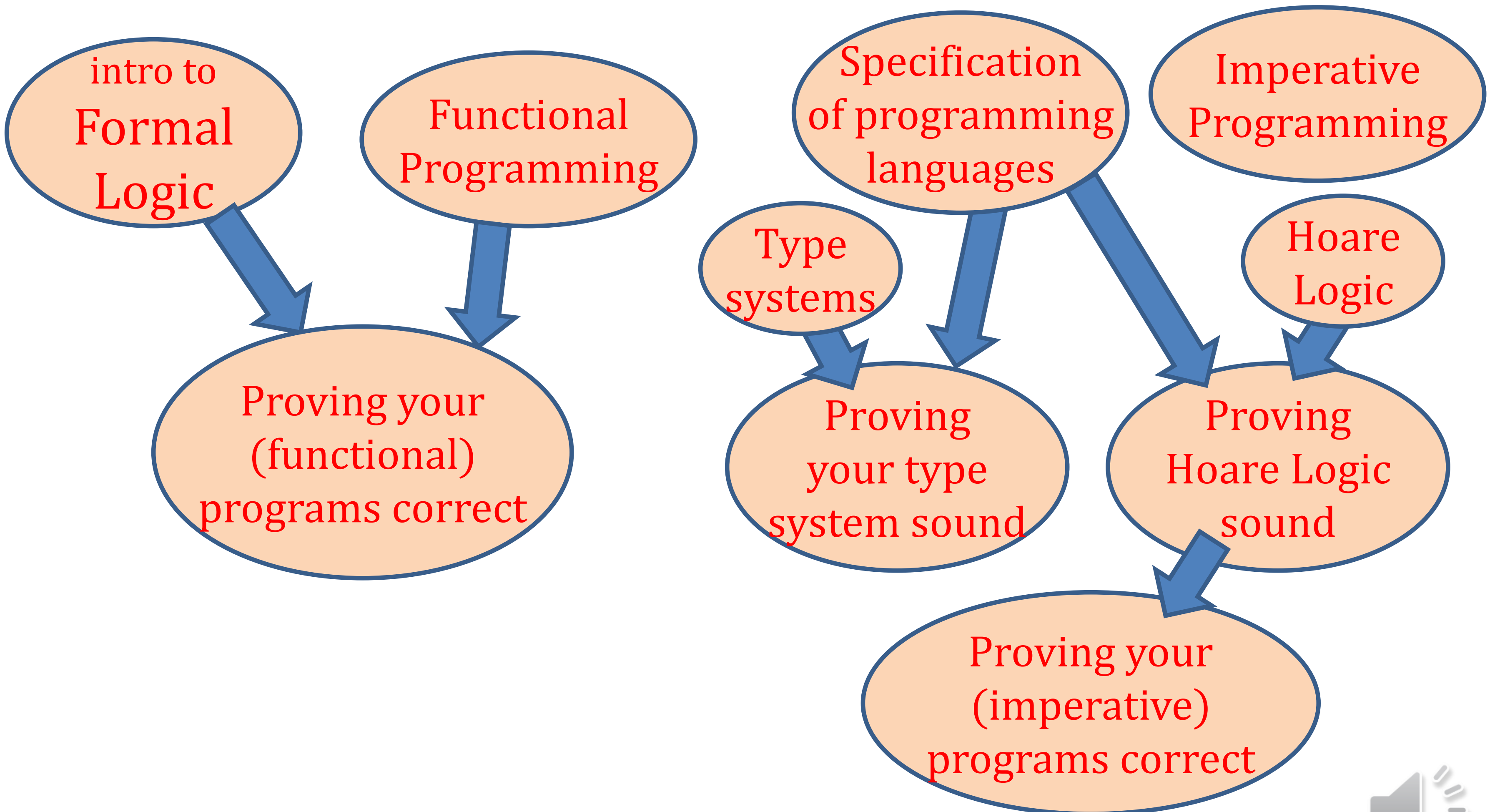
Andrew W. Appel

Princeton University

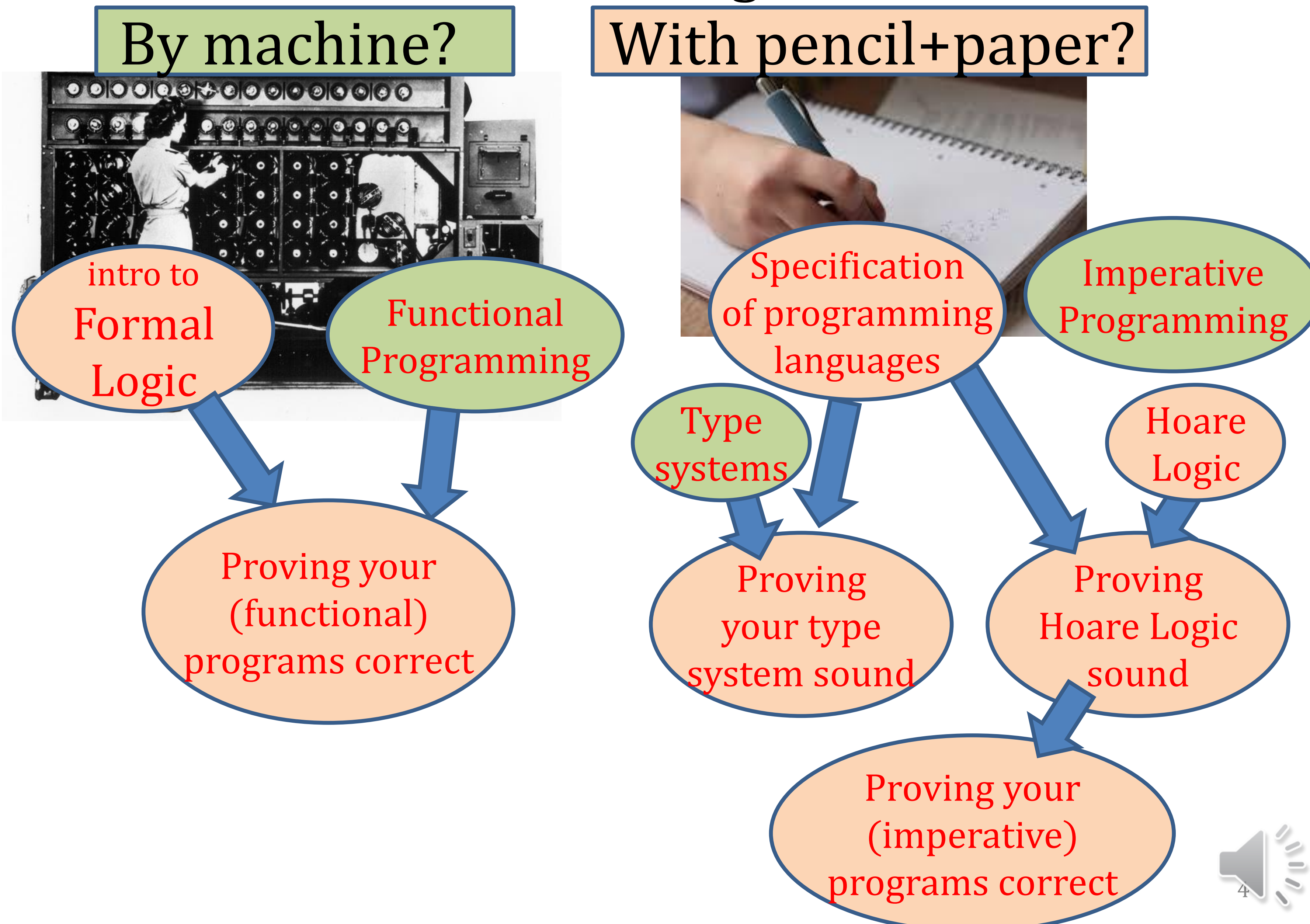
Formal reasoning about programs



Formal reasoning
about programs and programming languages

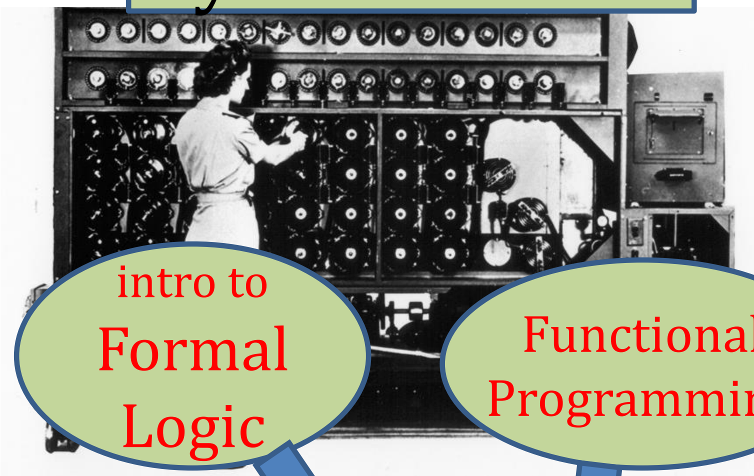


Which of these things do we do



We can do all of these

By machine!

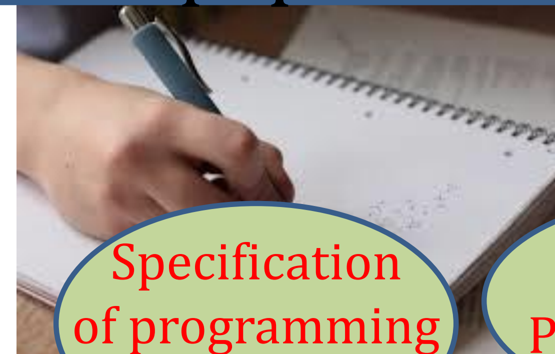


intro to Formal Logic

Functional Programming

Proving your (functional) programs correct

pencil+paper? Really?



Specification of programming languages

Imperative Programming

Type systems

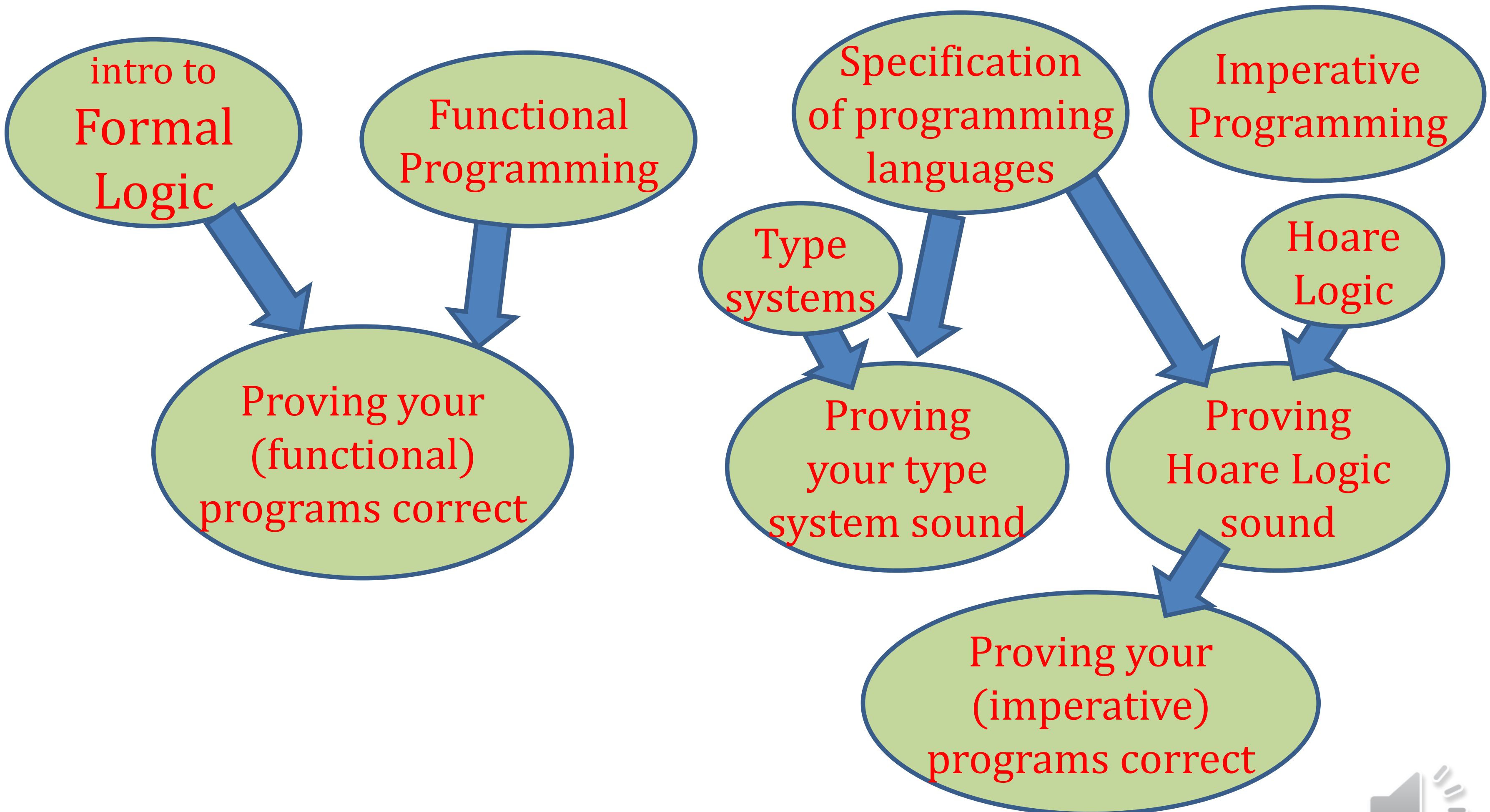
Hoare Logic

Proving your type system sound

Proving Hoare Logic sound

Proving your (imperative) programs correct

COS 510: Machine-checked, formal reasoning about programs and programming languages



EXAMPLE: LENGTH, APP

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Require Import List.

Fixpoint length {A} (xs: list A) : nat :=
 match xs with
 | nil => 0
 | x::xs' => 1 + length xs'
 end.

Eval compute in length (1::2::3::4::nil).


Fixpoint app {A} (xs ys: list A) : list A :=
 match xs with
 | nil => ys
 | x::xs' => x :: app xs' ys
 end.

Eval compute in app (1::2::3::nil) (7::8::nil).

Eval compute in length (app (1::2::3::nil) (7::8::nil)).
```

Messages Errors Jobs

Ready Line: 7 Char: 6 0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Require Import List.

Fixpoint length {A} (xs: list A) : nat :=
 match xs with
 | nil => 0
 | x::xs' => 1 + length xs'
 end.

Eval compute in length (1::2::3::4::nil).


Fixpoint app {A} (xs ys: list A) : list A :=
 match xs with
 | nil => ys
 | x::xs' => x :: app xs' ys
 end.

Eval compute in app (1::2::3::nil) (7::8::nil).

Eval compute in length (app (1::2::3::nil) (7::8::nil)).
```

Messages ↗ Errors ↗ Jobs ↗

```
= 4
: nat
```

Ready Line: 9 Char: 42 0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Require Import List.

Fixpoint length {A} (xs: list A) : nat :=
 match xs with
 | nil => 0
 | x::xs' => 1 + length xs'
 end.

Eval compute in length (1::2::3::4::nil).


Fixpoint app {A} (xs ys: list A) : list A :=
 match xs with
 | nil => ys
 | x::xs' => x :: app xs' ys
 end.

Eval compute in app (1::2::3::nil) (7::8::nil).

Eval compute in length (app (1::2::3::nil) (7::8::nil)).
```

Messages Errors Jobs

Ready Line: 17 Char: 1 0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Require Import List.

Fixpoint length {A} (xs: list A) : nat :=
 match xs with
 | nil => 0
 | x::xs' => 1 + length xs'
 end.

Eval compute in length (1::2::3::4::nil).


Fixpoint app {A} (xs ys: list A) : list A :=
 match xs with
 | nil => ys
 | x::xs' => x :: app xs' ys
 end.

Eval compute in app (1::2::3::nil) (7::8::nil).

Eval compute in length (app (1::2::3::nil) (7::8::nil)).
```

Messages Errors Jobs

```
= 1 :: 2 :: 3 :: 7 :: 8 :: nil
: list nat
```

Ready Line: 18 Char: 48 0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

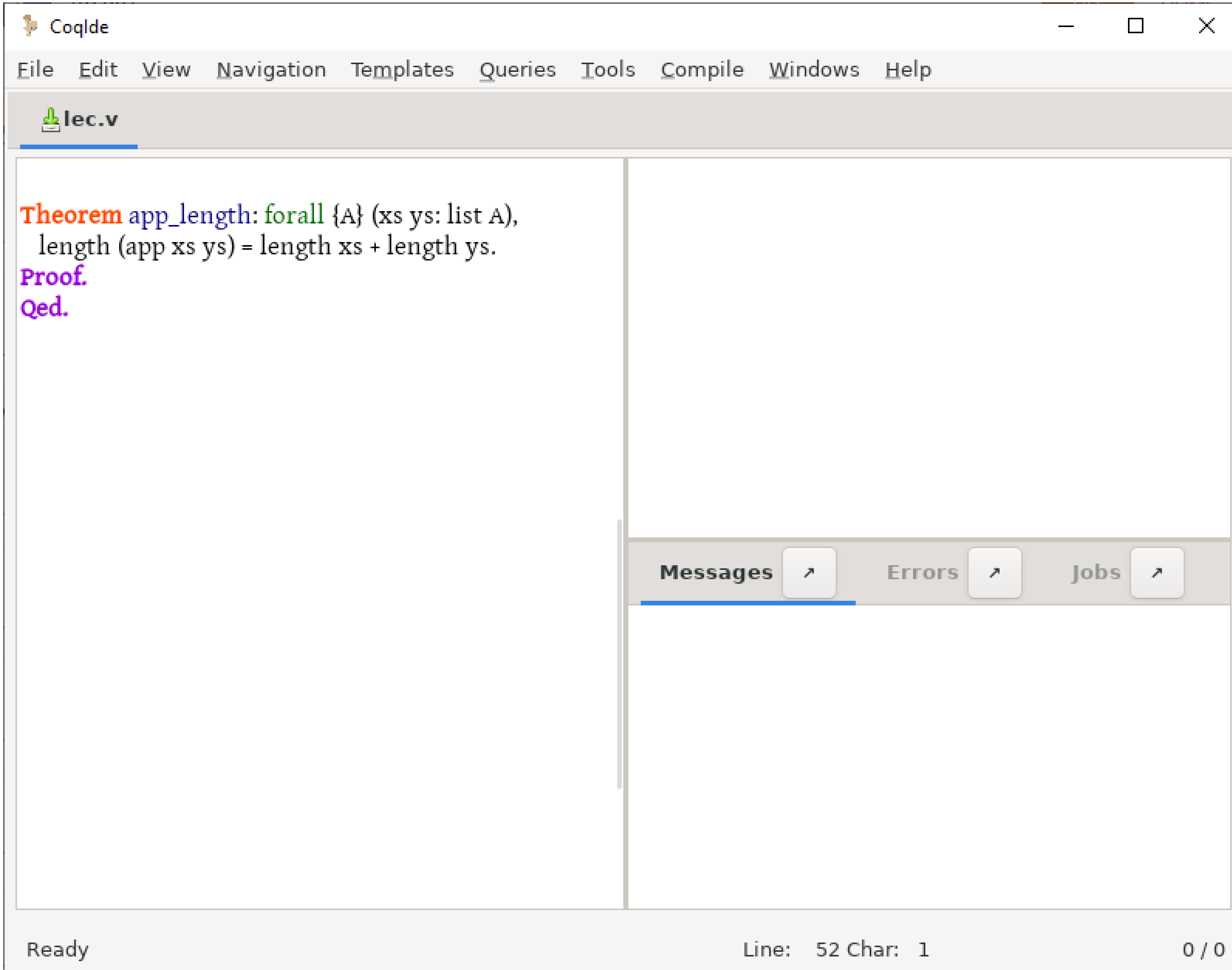
lec.v

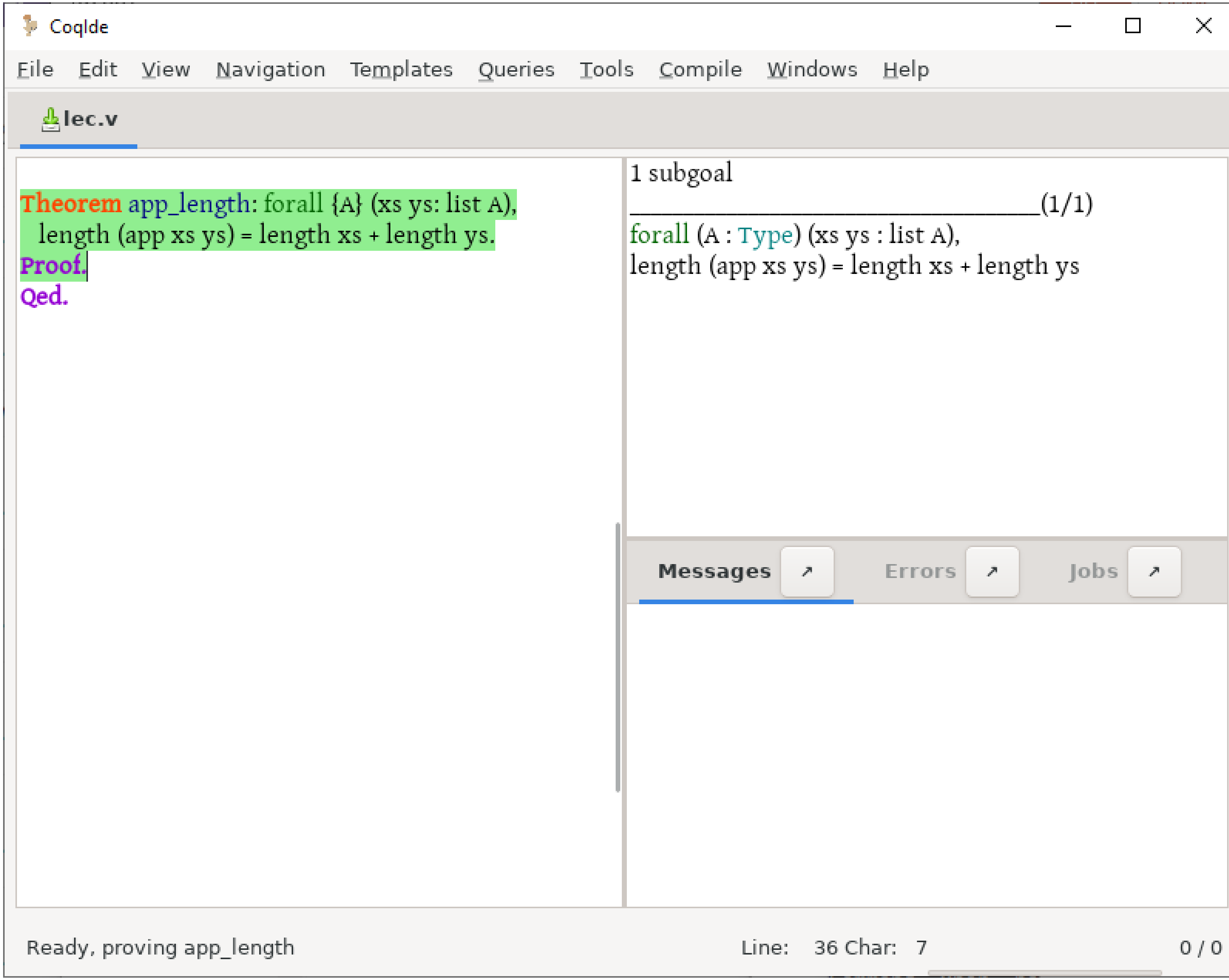
```
Require Import List.

Fixpoint length {A} (xs: list A) : nat :=
 match xs with
 | nil => 0
 | x::xs' => 1 + length xs'
 end.

Eval compute in length (1::2::3::4::nil).


Fixpoint app {A} (xs ys: list A) : list A :=
 match xs with
 | nil => ys
 | x::xs' => x :: app xs' ys
 end.

Eval compute in app (1::2::3::nil) (7::8::nil).

Eval compute in length (app (1::2::3::nil) (7::8::nil)).
```

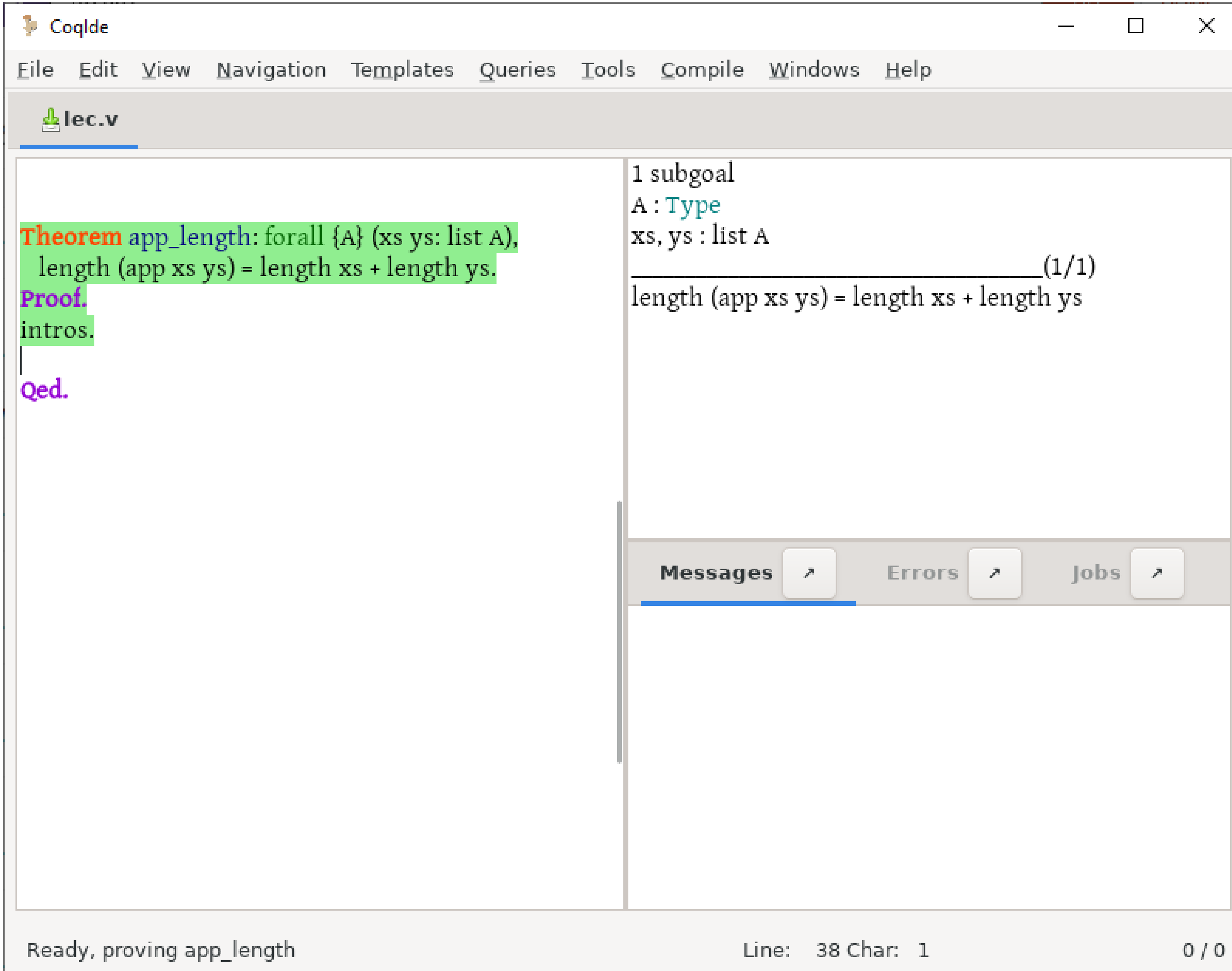
Messages | Errors | Jobs

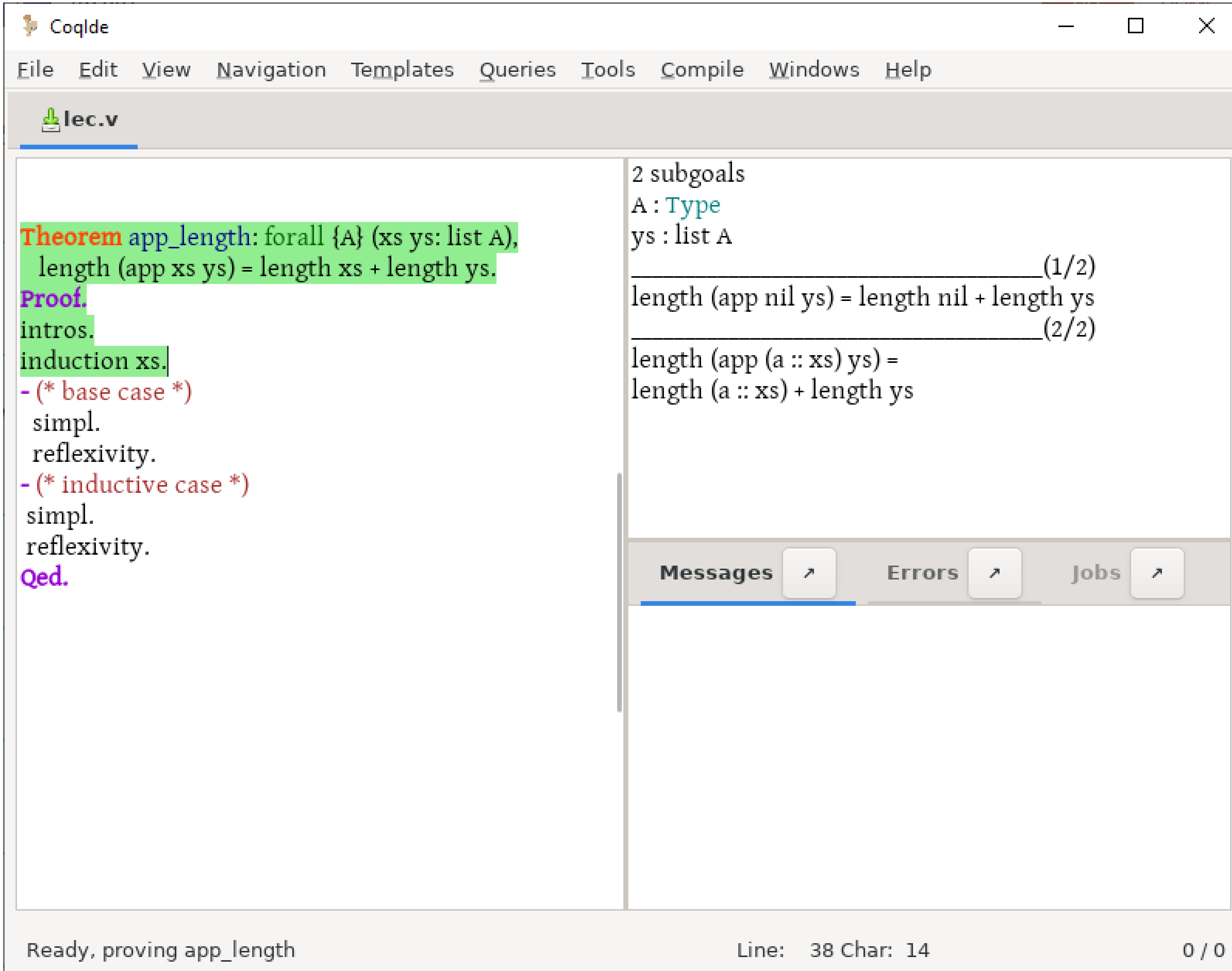
```
= 5
: nat
```

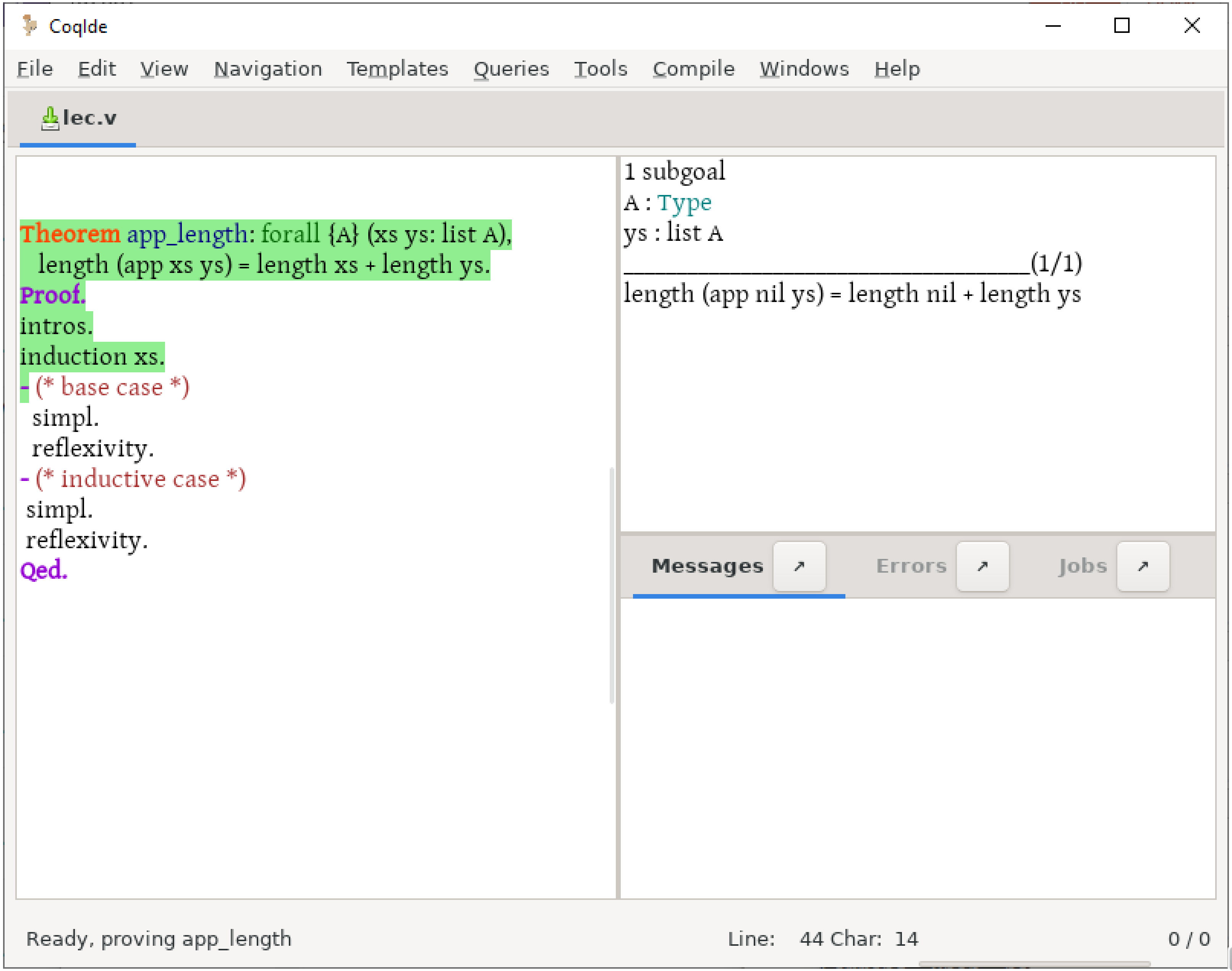
Ready | Line: 13 Char: 15 | 0 / 0

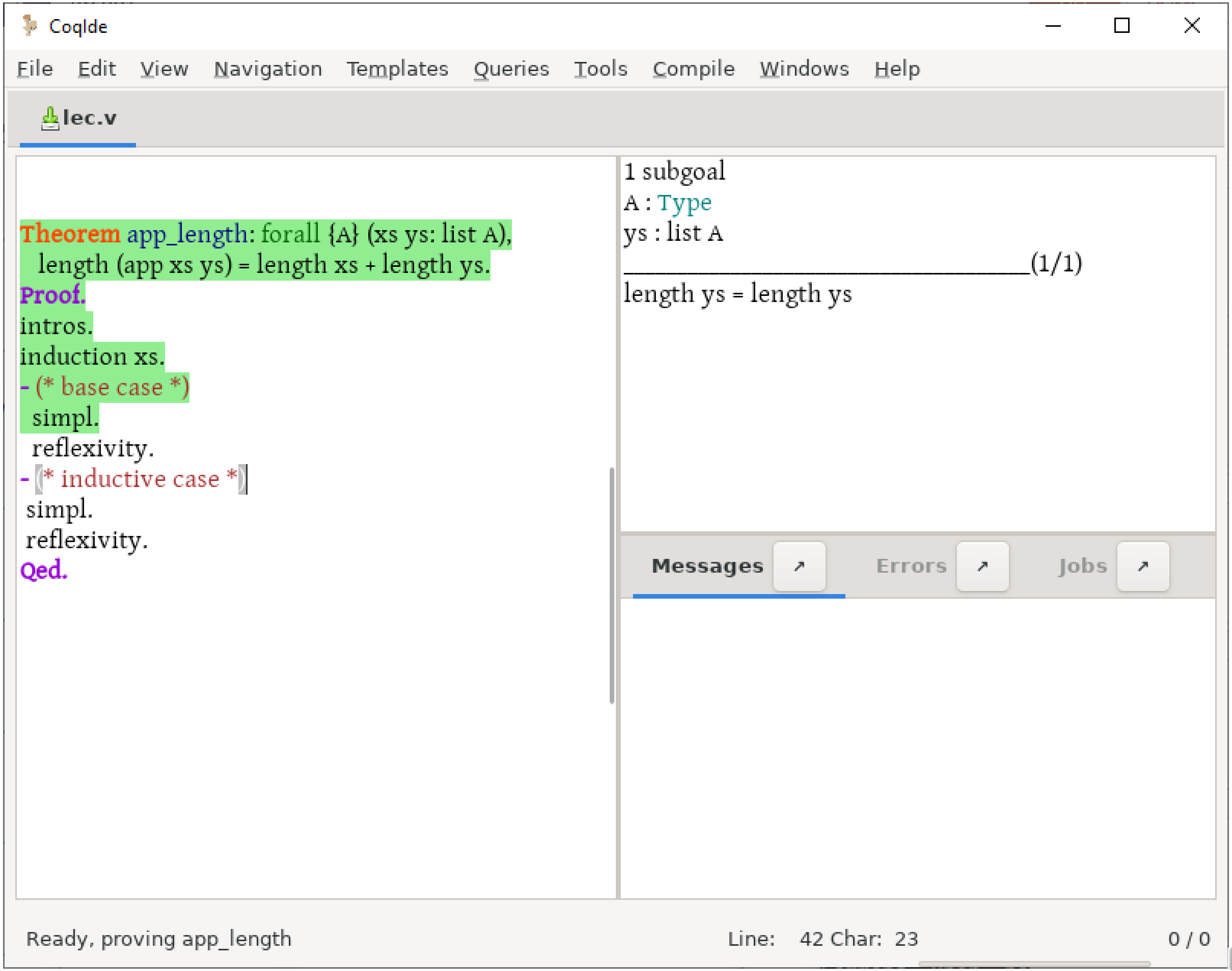

CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
length (app xs ys) = length xs + length ys.
Proof.
Qed.
Messages
Errors
Jobs
Ready
Line: 52 Char: 1
0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
length (app xs ys) = length xs + length ys.
Proof.
Qed.
1 subgoal
(1/1)
forall (A : Type) (xs ys : list A),
length (app xs ys) = length xs + length ys
Messages
Errors
Jobs
Ready, proving app_length
Line: 36 Char: 7
0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
length (app xs ys) = length xs + length ys.
Proof.
intros.
Qed.
1 subgoal
A : Type
xs, ys : list A
(1/1)
length (app xs ys) = length xs + length ys
Messages
Errors
Jobs
Ready, proving app_length
Line: 38 Char: 1
0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
simpl.
reflexivity.
- (* inductive case *)
simpl.
reflexivity.
Qed.
2 subgoals
A : Type
ys : list A
(1/2)
length (app nil ys) = length nil + length ys
(2/2)
length (app (a :: xs) ys) =
length (a :: xs) + length ys
Messages
Errors
Jobs
Ready, proving app_length
Line: 38 Char: 14
0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
simpl.
reflexivity.
- (* inductive case *)
simpl.
reflexivity.
Qed.
1 subgoal
A : Type
ys : list A
(1/1)
length (app nil ys) = length nil + length ys
Messages
Errors
Jobs
Ready, proving app_length
Line: 44 Char: 14
0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
simpl.
reflexivity.
- (* inductive case *)
simpl.
reflexivity.
Qed.
1 subgoal
A : Type
ys : list A
(1/1)
length ys = length ys
Messages
Errors
Jobs
Ready, proving app_length
Line: 42 Char: 23
0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Theorem app_length: forall {A} (xs ys: list A),
  length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
  simpl.
  reflexivity.
- (* inductive case *)
 simpl.
 reflexivity.
Qed.
```

```
1 subgoal
A : Type
ys : list A
______________________________________(1/1)
length (app nil ys) = length nil + length ys
```

Messages ↗ Errors ↗ Jobs ↗

Ready, proving app_length Line: 44 Char: 14 0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Theorem app_length: forall {A} (xs ys: list A),
  length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
  simpl.
  reflexivity.
- (* inductive case *)
 simpl.
 reflexivity.
Qed.
```

```
1 subgoal
A : Type
ys : list A
______________________________________(1/1)
length ys = length ys
```

Messages Errors Jobs

Ready, proving app_length   Line: 42 Char: 23   0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Theorem app_length: forall {A} (xs ys: list A),
  length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
  simpl.
  reflexivity.
- (* inductive case *)
 simpl.
 reflexivity.
Qed.
```

This subproof is complete, but there are some unfocused goals:

______________________________________(1/1)

length (app (a :: xs) ys) =
length (a :: xs) + length ys

Messages Errors Jobs

Ready, proving app_length Line: 41 Char: 15 0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Theorem app_length: forall {A} (xs ys: list A),
  length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
  simpl.
  reflexivity.
- (* inductive case *)
 simpl.
 reflexivity.
Qed.
```

```
1 subgoal
A : Type
a : A
xs, ys : list A
IHxs : length (app xs ys) =
      length xs + length ys
______________________________________(1/1)
length (app (a :: xs) ys) =
length (a :: xs) + length ys
```

Messages ↗ Errors ↗ Jobs ↗

Ready, proving app_length Line: 42 Char: 23 0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

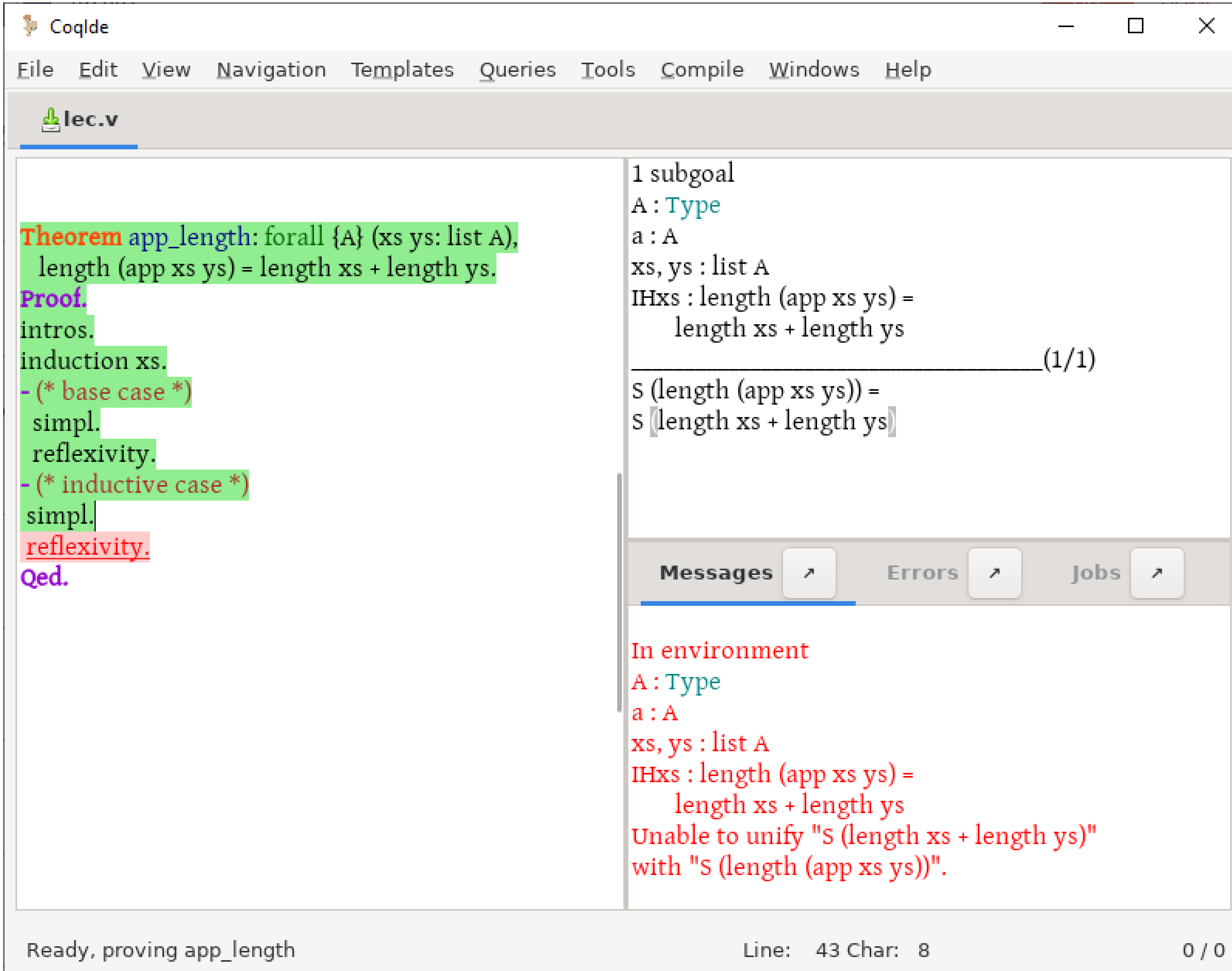
lec.v

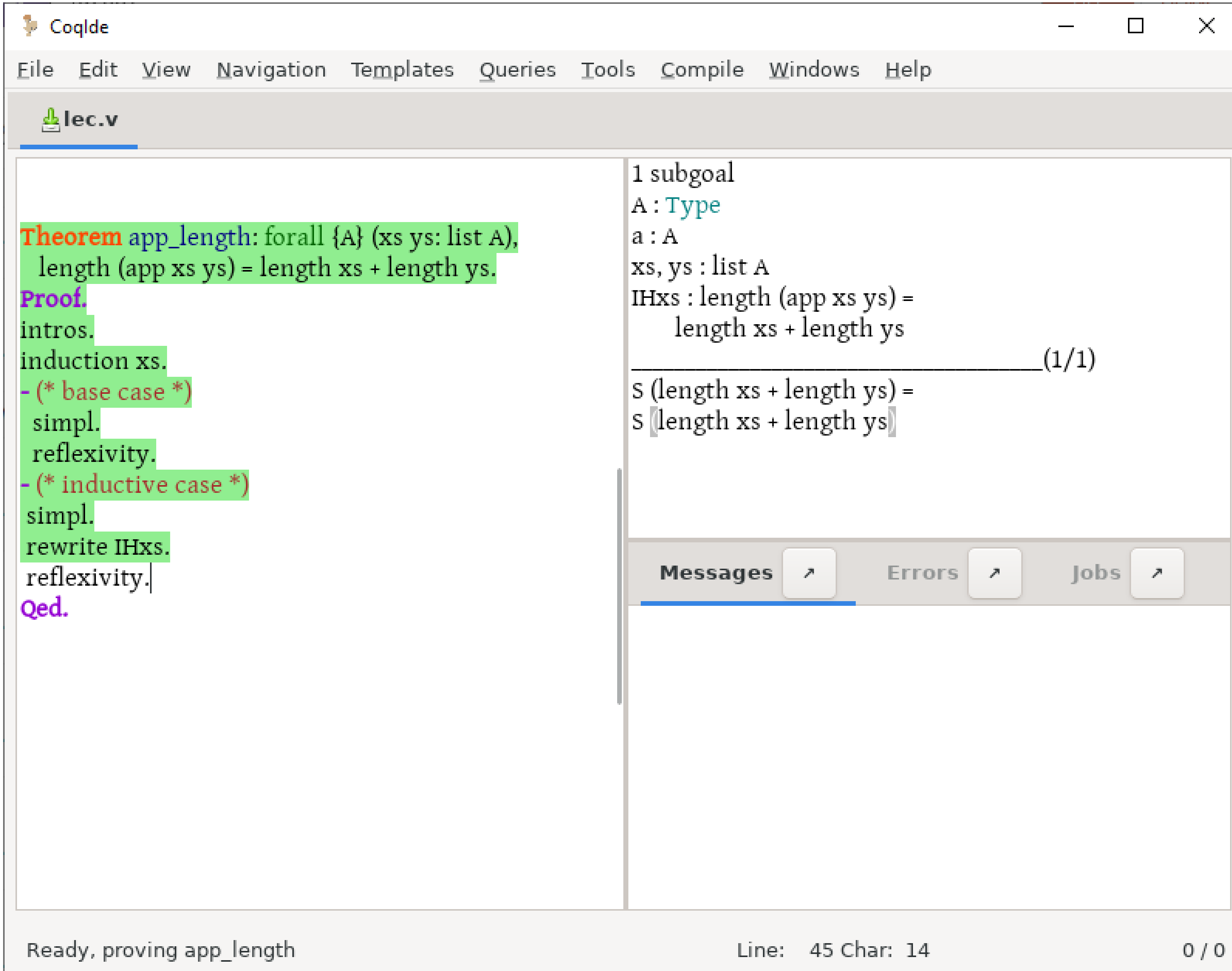
```
Theorem app_length: forall {A} (xs ys: list A),
  length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
  simpl.
  reflexivity.
- (* inductive case *)
 simpl.
 reflexivity.
Qed.
```

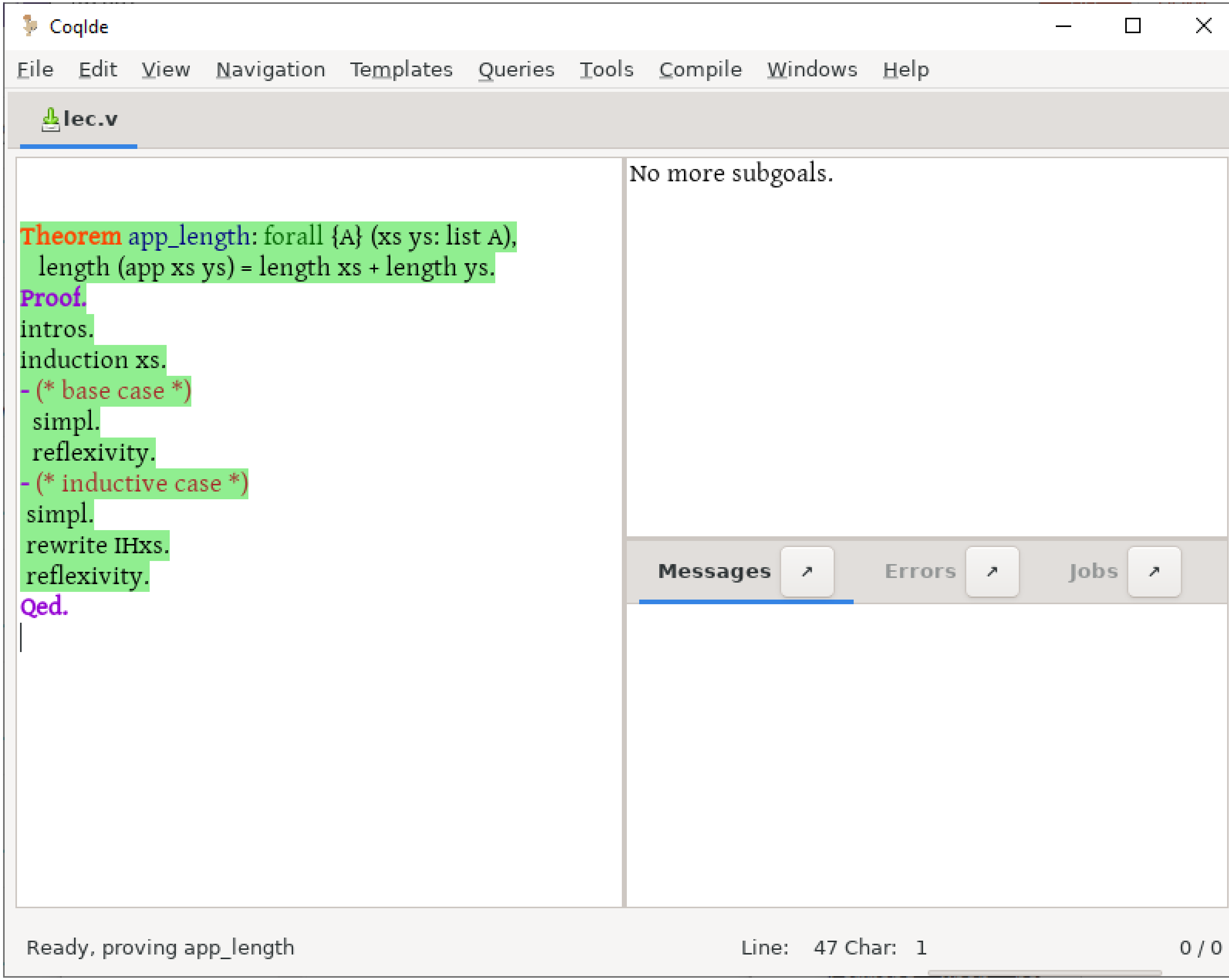
```
1 subgoal
A : Type
a : A
xs, ys : list A
IHxs : length (app xs ys) =
       length xs + length ys
______________________________________(1/1)
S (length (app xs ys)) =
S (length xs + length ys)
```

Messages Errors Jobs

Ready, proving app_length Line: 43 Char: 8 0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
simpl.
reflexivity.
- (* inductive case *)
simpl.
reflexivity.
Qed.
1 subgoal
A : Type
a : A
xs, ys : list A
IHxs : length (app xs ys) =
length xs + length ys
(1/1)
S (length (app xs ys)) =
S (length xs + length ys)
Messages
Errors
Jobs
In environment
A : Type
a : A
xs, ys : list A
IHxs : length (app xs ys) =
length xs + length ys
Unable to unify "S (length xs + length ys)"
with "S (length (app xs ys))".
Ready, proving app_length
Line: 43 Char: 8
0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
simpl.
reflexivity.
- (* inductive case *)
simpl.
rewrite IHxs.
reflexivity.
Qed.
1 subgoal
A : Type
a : A
xs, ys : list A
IHxs : length (app xs ys) =
length xs + length ys
(1/1)
S (length xs + length ys) =
S (length xs + length ys)
Messages
Errors
Jobs
Ready, proving app_length
Line: 45 Char: 14
0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_length: forall {A} (xs ys: list A),
  length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
  simpl.
  reflexivity.
- (* inductive case *)
 simpl.
 rewrite IHxs.
 reflexivity.
Qed.
No more subgoals.
Messages
Errors
Jobs
Ready, proving app_length
Line: 47 Char: 1
0 / 0

CoqIde

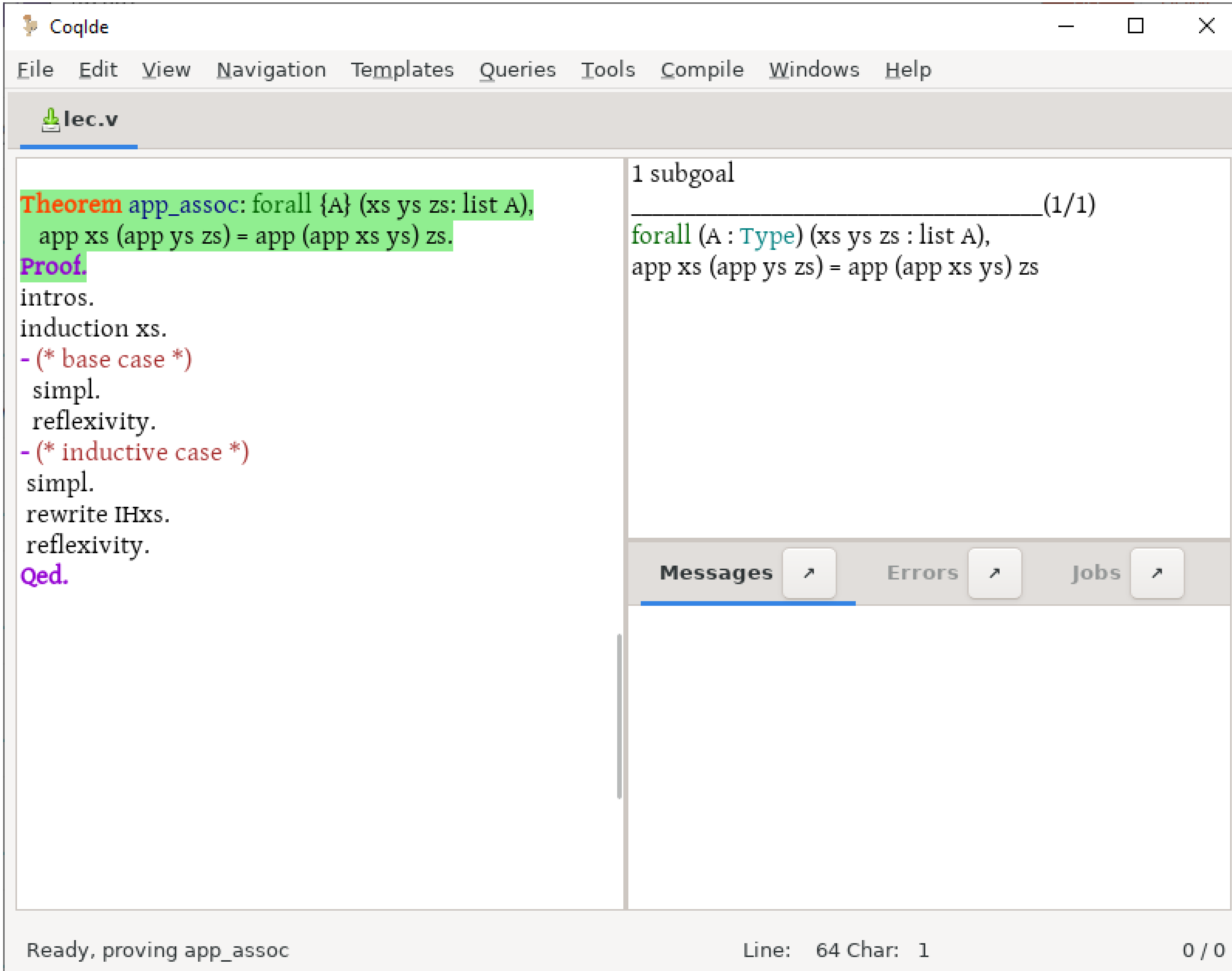
File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Theorem app_length: forall {A} (xs ys: list A),
  length (app xs ys) = length xs + length ys.
Proof.
intros.
induction xs.
- (* base case *)
  simpl.
  reflexivity.
- (* inductive case *)
 simpl.
 rewrite IHxs.
 reflexivity.
Qed.
```

Messages ↗ Errors ↗ Jobs ↗

Ready Line: 41 Char: 15 0 / 0


CoqIde
File Edit View Navigation Templates Queries Tools Compile Windows Help
lec.v
Theorem app_assoc: forall {A} (xs ys zs: list A),
app xs (app ys zs) = app (app xs ys) zs.
Proof.
intros.
induction xs.
- (* base case *)
simpl.
reflexivity.
- (* inductive case *)
simpl.
rewrite IHxs.
reflexivity.
Qed.
1 subgoal
(1/1)
forall (A : Type) (xs ys zs : list A),
app xs (app ys zs) = app (app xs ys) zs
Messages
Errors
Jobs
Ready, proving app_assoc
Line: 64 Char: 1
0 / 0

CoqIde

File Edit View Navigation Templates Queries Tools Compile Windows Help

lec.v

```
Theorem app_assoc: forall {A} (xs ys zs: list A),
  app xs (app ys zs) = app (app xs ys) zs.
Proof.
intros.
induction xs.
- (* base case *)
  simpl.
  reflexivity.
- (* inductive case *)
 simpl.
 rewrite IHxs.
 reflexivity.
Qed.
```

Messages Errors Jobs

Ready Line: 82 Char: 1 0 / 0

Applications of Formal Methods

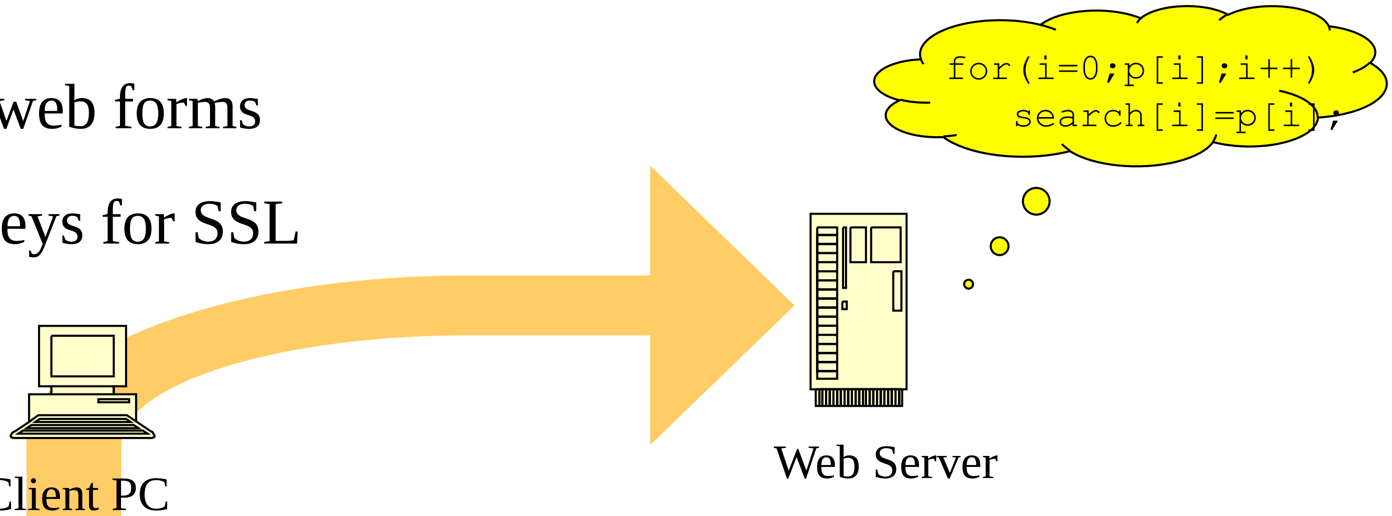
Attacking a web server

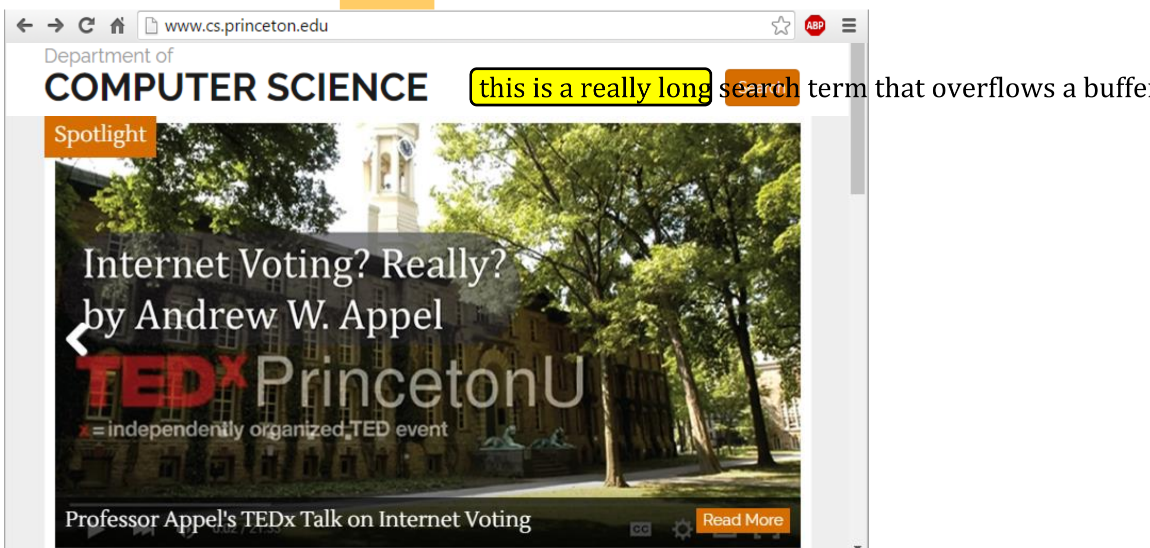
URLs

Input in web forms

Crypto keys for SSL

etc.





Attacking a web browser

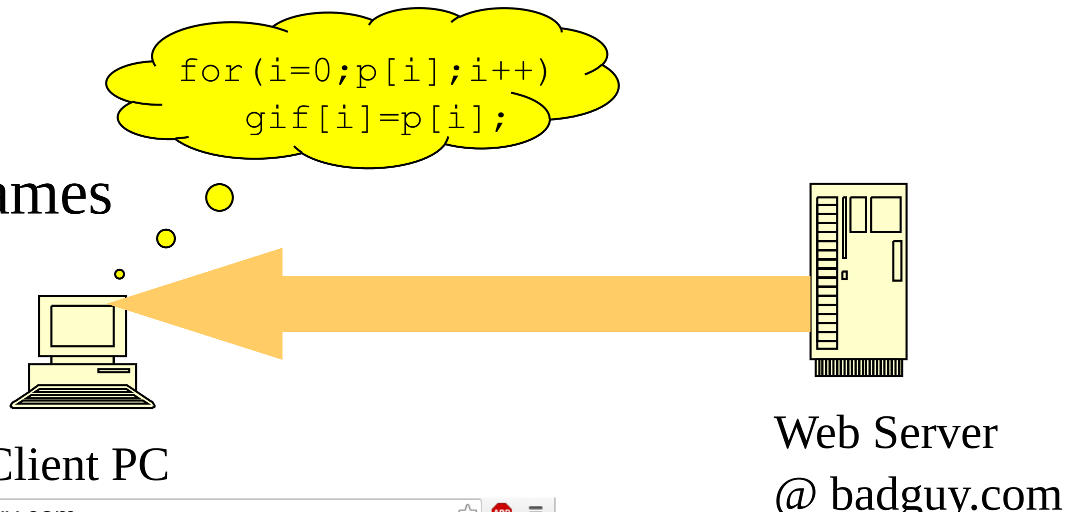
HTML keywords

Images

Image names

URLs

etc.



Attacking everything in sight

E-mail client

PDF viewer

Web browser

Operating-system kernel

TCP/IP stack

Any application that ever sees input directly from the outside

Solution: implement the outward-facing parts of software <u>without any bugs</u>!

E-mail client

PDF viewer

Web browser

Operating-system kernel

TCP/IP stack

Any application that ever sees input directly from the outside

In recent years, great progress in . . .

- Proved-correct optimizing C compiler (France)
- Proved-correct ML compiler (Sweden, Princeton)
- Proved-correct O.S. kernels (Australia, New Haven)
- Proved-correct crypto (Princeton NJ, Cambridge MA)
- Proved-correct distributed systems (Seattle, Israel)
- Proved-correct web server (Philadelphia)
- Proved-correct malloc/free library (Princeton, Hoboken)

Automated verification in industry

Amazon

Microsoft

Intel

Facebook

Google

Galois, HRL, Rockwell, Bedrock, …

Recent Princeton JIW / Sr. Thesis

- Katherine Ye ’16 verified crypto security
- Naphat Sanguansin ’16 verified crypto impl’n
- Brian McSwiggen ’18 verified B-trees
- Katja Vassilev ’19 verified dead-var elimination
- John Li ’19 verified uncurrying
- Jake Waksbaum ’20 verified Burrows-Wheeler
- Anvay Grover ’20 verified CPS-conversion

ACM Conference on Computer and Communications Security 2017

Verified Correctness and Security of mbedTLS HMAC-DRBG

Katherine Q. Ye '16
Princeton U., Carnegie Mellon U.

Matthew Green
Johns Hopkins University

Naphat Sanguansin'16
Princeton University

Lennart Beringer
Princeton University

Adam Petcher
Oracle

Andrew W. Appel '81
Princeton University

ABSTRACT

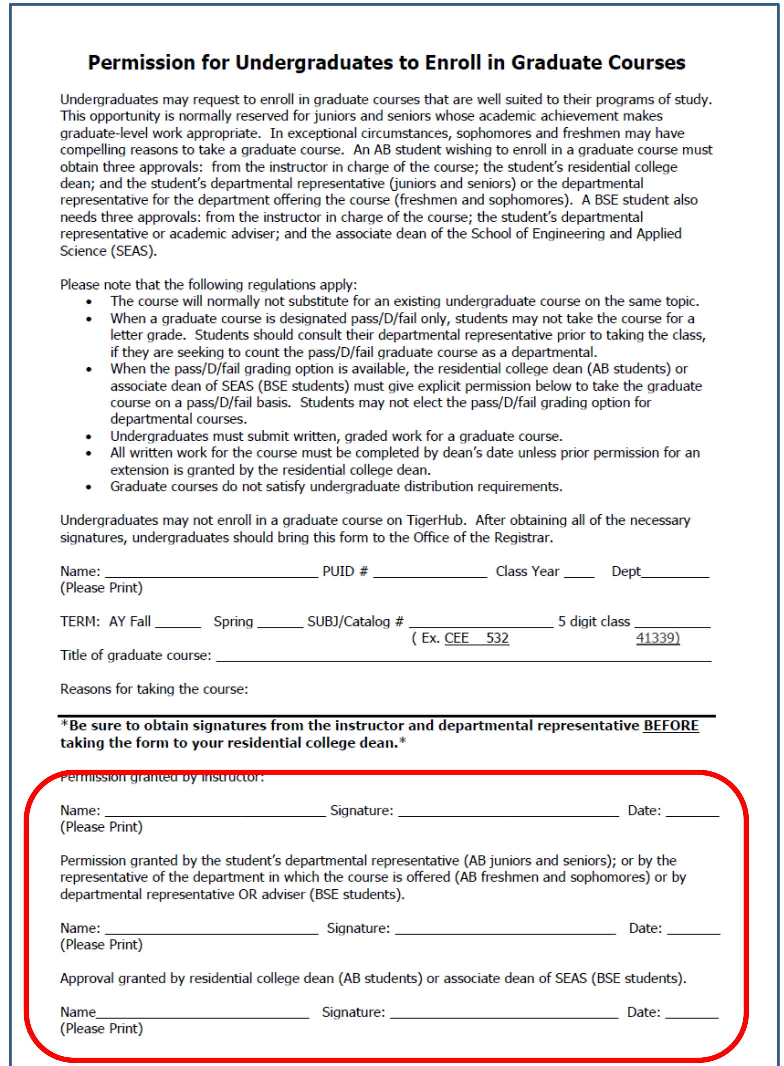
We have formalized the functional specification of HMAC-DRBG (NIST 800-90A), and we have proved its cryptographic security—that its output is pseudorandom—using a hybrid game-based proof. We have also proved that the mbedTLS implementation (C program) correctly implements this functional specification. That proof composes with an existing C compiler correctness proof to guarantee, end-to-end, that the machine language program gives strong pseudorandomness. All proofs (hybrid games, C program verification, compiler, and their composition) are machine-checked in the Coq proof assistant. Our proofs are modular: the hybrid game proof holds on any implementation of HMAC-DRBG that satisfies our functional specification. Therefore, our functional specification can serve as a high-assurance reference.

Prerequisites for COS 510 if you're an undergrad

1. COS 326
   Functional Programming

2. Enjoy the proofs in COS 326

3. Get the form signed
   by Colleen Kenny-McGinley, room 210 (one-stop shopping, all three signatures):



Permission for Undergraduates to Enroll in Graduate Courses

Undergraduates may request to enroll in graduate courses that are well suited to their programs of study. This opportunity is normally reserved for juniors and seniors whose academic achievement makes graduate-level work appropriate. In exceptional circumstances, sophomores and freshmen may have compelling reasons to take a graduate course. An AB student wishing to enroll in a graduate course must obtain three approvals: from the instructor in charge of the course; the student's residential college dean; and the student's departmental representative (juniors and seniors) or the departmental representative for the department offering the course (freshmen and sophomores). A BSE student also needs three approvals: from the instructor in charge of the course; the student's departmental representative or academic adviser; and the associate dean of the School of Engineering and Applied Science (SEAS).

Please note that the following regulations apply:

- The course will normally not substitute for an existing undergraduate course on the same topic.
- When a graduate course is designated pass/D/fail only, students may not take the course for a letter grade. Students should consult their departmental representative prior to taking the class, if they are seeking to count the pass/D/fail graduate course as a departmental.
- When the pass/D/fail grading option is available, the residential college dean (AB students) or associate dean of SEAS (BSE students) must give explicit permission below to take the graduate course on a pass/D/fail basis. Students may not elect the pass/D/fail grading option for departmental courses.
- Undergraduates must submit written, graded work for a graduate course.
- All written work for the course must be completed by dean's date unless prior permission for an extension is granted by the residential college dean.
- Graduate courses do not satisfy undergraduate distribution requirements.

Undergraduates may not enroll in a graduate course on TigerHub. After obtaining all of the necessary signatures, undergraduates should bring this form to the Office of the Registrar.

Name: ________________________ PUID # ____________ Class Year ____ Dept________
(Please Print)

TERM: AY Fall _____ Spring _____ SUBJ/Catalog # ________________ 5 digit class _________
(Ex. CEE 532 41339)

Title of graduate course: ____

Reasons for taking the course:

Be sure to obtain signatures from the instructor and departmental representative BEFORE taking the form to your residential college dean.

Permission granted by instructor:

Name: ________________________ Signature: ________________________ Date: ______
(Please Print)

Permission granted by the student's departmental representative (AB juniors and seniors); or by the representative of the department in which the course is offered (AB freshmen and sophomores) or by departmental representative OR adviser (BSE students).

Name: ________________________ Signature: ________________________ Date: ______
(Please Print)

Approval granted by residential college dean (AB students) or associate dean of SEAS (BSE students).

Name________________________ Signature: ________________________ Date: ______
(Please Print)