

# Isabelle and Proof General: Preview

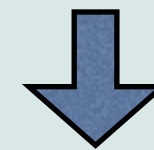
---

Lawrence C. Paulson  
University of Cambridge



# Getting Started

- Install Isabelle, following instructions on the [download page](#).
- Install Proof General.
- Proof General requires the editor XEmacs to be installed.
- If you have not used XEmacs before, practice on plain text files before attempting proofs!
- Launch Isabelle from the command line.
- Here, Isabelle has been installed at `/usr/local` and is used to open one of the standard theories.



Launching Isabelle at the  
UNIX command line



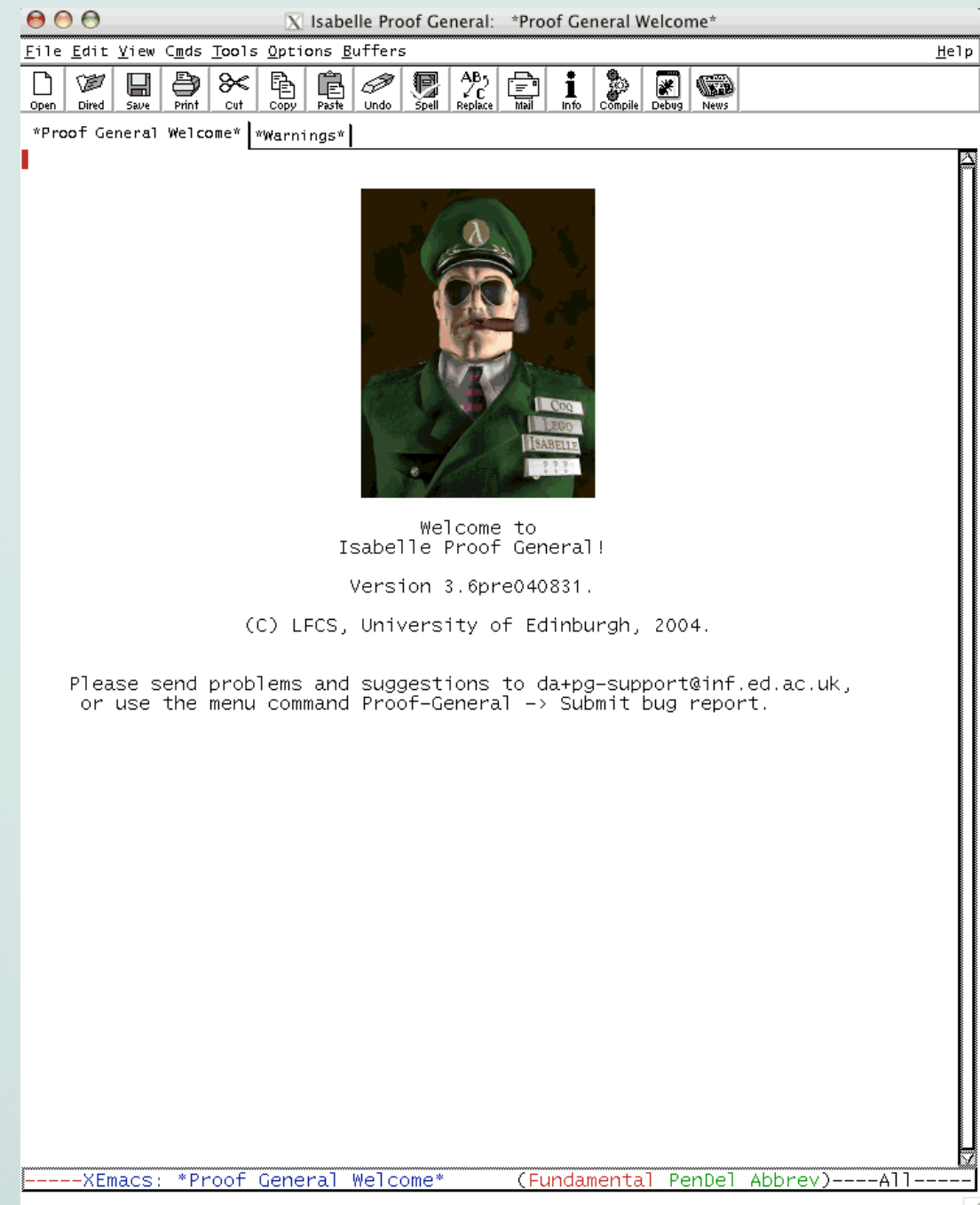
```
xterm
/usr/local/Isabelle: ./bin/Isabelle src/HOL/NumberTheory/Fib.thy&
[6] 5837
/usr/local/Isabelle: █
```





# Proof General

- Proof General launches within XEmacs.
- If you don't see this splash screen, Proof General is not correctly installed.





# The Theory File

- The theory opens in Proof General.
- Theory files visited from XEmacs also open in Proof General.
- Syntax colouring distinguishes constants, types, keywords, etc.
- The toolbar gives quick access to basic proof operations.
- This theory defines the Fibonacci function and proves theorems about it.

```
Fib.thy
(* ID:          $Id: Fib.thy,v 1.11 2005/01/14 11:00:27 nipkow Exp $
   Author:      Lawrence C Paulson, Cambridge University Computer Laboratory
   Copyright    1997 University of Cambridge
*)

header {* The Fibonacci function *}

theory Fib = Primes:

text {*
  Fibonacci numbers: proofs of laws taken from:
  R. L. Graham, D. E. Knuth, O. Patashnik. Concrete Mathematics.
  (Addison-Wesley, 1989)
*}

\bigskip
*}

consts fib :: "nat => nat"
recdef fib "measure (\lambda x. x)"
  zero:  "fib 0 = 0"
  one:   "fib (Suc 0) = Suc 0"
  Suc_Suc: "fib (Suc (Suc x)) = fib x + fib (Suc x)"
[]
text {*
  \medskip The difficulty in these proofs is to ensure that the
  induction hypotheses are applied before the definition of @{term
  fib}. Towards this end, the @{term fib} equations are not declared
  to the Simplifier and are applied very selectively at first.
*}

text{*We disable @{term fib.Suc_Suc} for simplification ...*}
declare fib.Suc_Suc [simp del]

text{*...then prove a version that has a more restrictive pattern.*}
lemma fib_Suc3: "fib (Suc (Suc (Suc n))) = fib (Suc n) + fib (Suc (Suc n))"
  by (rule fib.Suc_Suc)

text {* \medskip Concrete Mathematics, page 280 *}

lemma fib_add: "fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n"
  apply (induct n rule: fib.induct)
  prefer 3
  txt {* simplify the LHS just enough to apply the induction hypotheses *}
  apply (simp add: fib_Suc3)
  apply (simp_all add: fib.Suc_Suc add_mult_distrib2)
  done

lemma fib_Suc_neq_0: "fib (Suc n) \neq 0"
  apply (induct n rule: fib.induct)
  apply (simp_all add: fib.Suc_Suc)
  done

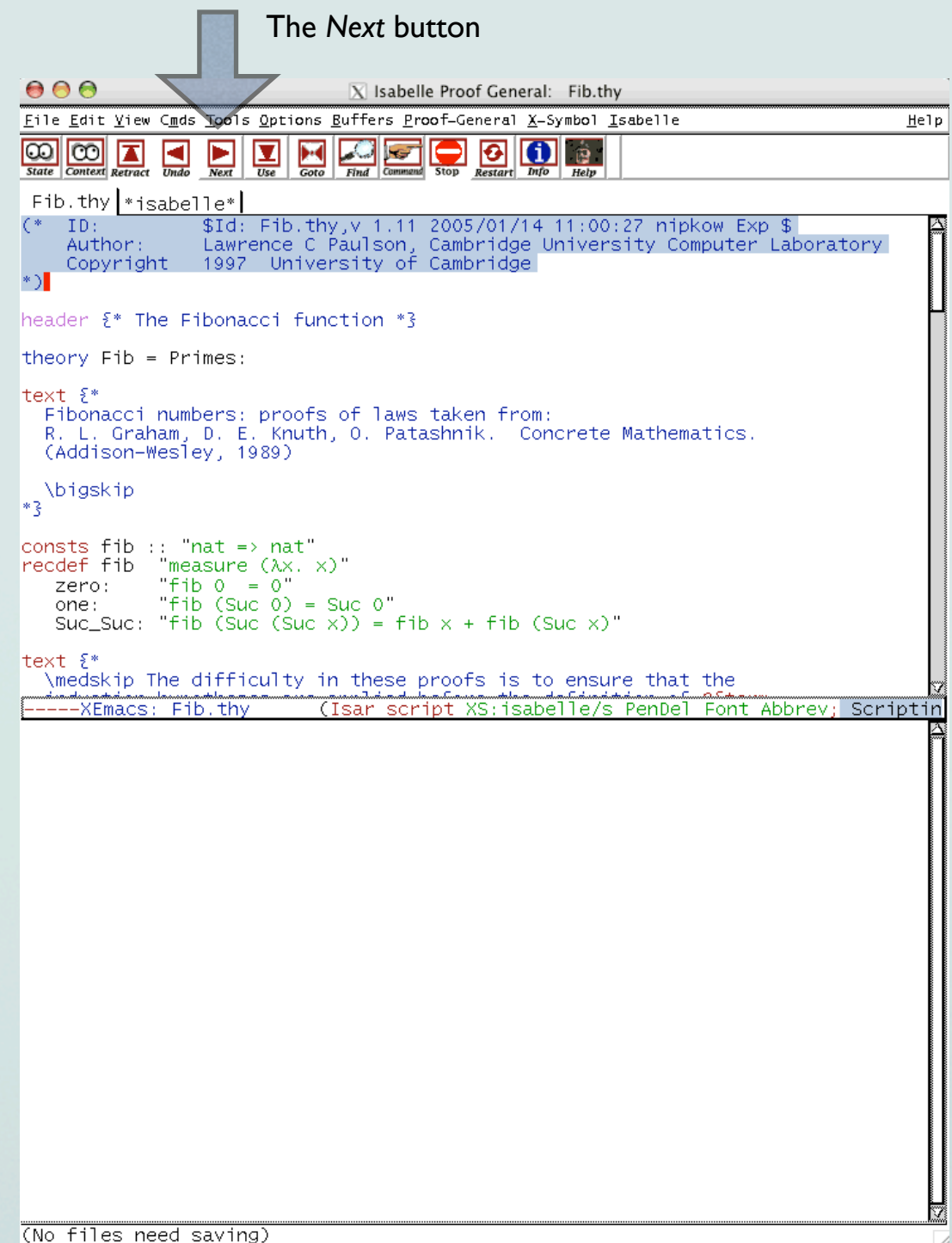
-----XEmacs: Fib.thy      (Isar script XS:isabelle/s PenDel Font Abbrev;)-----Top-
```





# Basic Navigation

- A theory file contains definitions, proofs, LaTeX markup, and general commands.
- Clicking on *Next* starts Isabelle and processes the first item: a comment.
- Repeated clicks on *Next* step through the definitions.
- Proof General highlights material that has been processed in blue.





## The *Undo* button

The image is a screenshot of the Isabelle proof editor. At the top, the title bar reads "Isabelle Proof General: Fib.thy". Below the title bar is a menu bar with "File", "Edit", "View", "Tools", "Options", "Buffers", "Proof-General", "X-Symbol", and "Isabelle". Underneath the menu bar is a toolbar with icons for "State", "Context", "Retract", "Undo", "Next", "Use", "Goto", "Find", "Command", "Stop", "Restart", "Info", and "Help". The main text area contains the following Isabelle code:

```
Fib.thy |*isabelle*|
(* ID: $Id: Fib.thy,v 1.11 2005/01/14 11:00:27 nipkow Exp $
   Author: Lawrence C Paulson, Cambridge University Computer Laboratory
   Copyright 1997 University of Cambridge
*)

header {* The Fibonacci function *}

theory Fib = Primes:

text {*
  Fibonacci numbers: proofs of laws taken from:
  R. L. Graham, D. E. Knuth, O. Patashnik. Concrete Mathematics.
  (Addison-Wesley, 1989)

  \bigskip
*}

consts fib :: "nat => nat"
recdef fib "measure (\lambda x. x)"
  zero: "fib 0 = 0"
  one: "fib (Suc 0) = Suc 0"
  Suc_Suc: "fib (Suc (Suc x)) = fib x + fib (Suc x)"

text {*
  \medskip The difficulty in these proofs is to ensure that the
  -----XEmacs: Fib.thy (Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin
Simple arithmetic decision procedure failed.
Now trying full Presburger arithmetic...

[Isabelle] ### Search depth = 1
```

- 6





# Running a Proof

- We are about to replay a small proof relating the Fibonacci function, addition and multiplication.
- Processing the `lemma` command displays one subgoal in the proof window.
- The commands `lemma`, `theorem` and `corollary` are essentially equivalent.

The screenshot shows the Isabelle Proof General interface. The top window, titled "Fib.thy", contains the following Isabelle code:

```
Fib.thy |*isabelle*|
text{*We disable @{text fib.Suc_Suc} for simplification ...*}
declare fib.Suc_Suc [simp del]

text{*...then prove a version that has a more restrictive pattern.*}
lemma fib_Suc3: "fib (Suc (Suc (Suc n))) = fib (Suc n) + fib (Suc (Suc n))"
  by (rule fib.Suc_Suc)

text {* \medskip Concrete Mathematics, page 280 *}

lemma fib_add: "fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n"
  apply (induct n rule: fib.induct)
  prefer 3
  txt {* simplify the LHS just enough to apply the induction hypotheses *}
  apply (simp add: fib_Suc3)
  apply (simp_all add: fib.Suc_Suc add_mult_distrib2)
  done

lemma fib_Suc_neq_0: "fib (Suc n)  $\neq$  0"
  apply (induct n rule: fib.induct)
  apply (simp_all add: fib.Suc_Suc)
  done

lemma fib_Suc_gr_0: "0 < fib (Suc n)"
  by (insert fib_Suc_neq_0 [of n], simp)

-----XEmacs: Fib.thy (Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin
proof (prove): step 0

fixed variables: n, k

goal (lemma (fib_add), 1 subgoal):
  1. fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n
```

The bottom window shows the proof state for the `fib_add` lemma, indicating the current goal and the fixed variables `n` and `k`.



# Performing Induction

- The first command performs induction on  $n$  using the rule `fib.induct`.
- Isabelle produced this rule while processing the recursive definition of the Fibonacci function.
- The proof window now displays three subgoals.

```
Fib.thy |*isabelle*|
text{*We disable @{text fib.Suc_Suc} for simplification ...*}
declare fib.Suc_Suc [simp del]

text{*...then prove a version that has a more restrictive pattern.*}
lemma fib_Suc3: "fib (Suc (Suc (Suc n))) = fib (Suc n) + fib (Suc (Suc n))"
  by (rule fib.Suc_Suc)

text {* \medskip Concrete Mathematics, page 280 *}

lemma fib_add: "fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n"
  apply (induct n rule: fib.induct)
  prefer 3
  txt {* simplify the LHS just enough to apply the induction hypotheses *}
  apply (simp add: fib_Suc3)
  apply (simp_all add: fib.Suc_Suc add_mult_distrib2)
  done

lemma fib_Suc_neq_0: "fib (Suc n)  $\neq$  0"
  apply (induct n rule: fib.induct)
  apply (simp_all add: fib.Suc_Suc)
  done

lemma fib_Suc_gr_0: "0 < fib (Suc n)"
  by (insert fib_Suc_neq_0 [of n], simp)

-----XEmacs: Fib.thy (Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin
proof (prove): step 1

fixed variables: n, k

goal (lemma (fib_add), 3 subgoals):
1. fib (Suc (0 + k)) = fib (Suc k) * fib (Suc 0) + fib k * fib 0
2. fib (Suc (Suc 0 + k)) =
   fib (Suc k) * fib (Suc (Suc 0)) + fib k * fib (Suc 0)
3.  $\forall x$ . [[fib (Suc (Suc x + k)) =
         fib (Suc k) * fib (Suc (Suc x)) + fib k * fib (Suc x);
         fib (Suc (x + k)) = fib (Suc k) * fib (Suc x) + fib k * fib x]]
    $\Rightarrow$  fib (Suc (Suc (Suc x) + k)) =
        fib (Suc k) * fib (Suc (Suc (Suc x))) + fib k * fib (Suc (Suc x))
```





# A Rewriting Step

- The third subgoal is selected: prefer 3.
- Then, it is simplified with the help of a rewrite rule called fib\_Suc3.
- This subgoal is still rather complicated!

```
Fib.thy |*isabelle*|
text{*We disable @{text fib.Suc_Suc} for simplification ...*}
declare fib.Suc_Suc [simp del]

text{*...then prove a version that has a more restrictive pattern.*}
lemma fib_Suc3: "fib (Suc (Suc (Suc n))) = fib (Suc n) + fib (Suc (Suc n))"
  by (rule fib.Suc_Suc)

text {* \medskip Concrete Mathematics, page 280 *}

lemma fib_add: "fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n"
  apply (induct n rule: fib.induct)
  prefer 3
  txt {* simplify the LHS just enough to apply the induction hypotheses *}
  apply (simp add: fib_Suc3)
  apply (simp_all add: fib.Suc_Suc add_mult_distrib2)
  done

lemma fib_Suc_neq_0: "fib (Suc n) ≠ 0"
  apply (induct n rule: fib.induct)
  apply (simp_all add: fib.Suc_Suc)
  done

lemma fib_Suc_gr_0: "0 < fib (Suc n)"
  by (insert fib_Suc_neq_0 [of n], simp)

-----XEmacs: Fib.thy (Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin
proof (prove): step 4

fixed variables: n, k

goal (lemma (fib_add), 3 subgoals):
1.  $\forall x. [\text{fib} (\text{Suc} (\text{Suc} (x + k))) = \text{fib} (\text{Suc} k) * \text{fib} (\text{Suc} (\text{Suc} x)) + \text{fib} k * \text{fib} (\text{Suc} x); \text{fib} (\text{Suc} (x + k)) = \text{fib} (\text{Suc} k) * \text{fib} (\text{Suc} x) + \text{fib} k * \text{fib} x] \Rightarrow \text{fib} (\text{Suc} k) * \text{fib} (\text{Suc} x) + \text{fib} k * \text{fib} x + (\text{fib} (\text{Suc} k) * \text{fib} (\text{Suc} (\text{Suc} x)) + \text{fib} k * \text{fib} (\text{Suc} x)) = \text{fib} (\text{Suc} k) * (\text{fib} (\text{Suc} x) + \text{fib} (\text{Suc} (\text{Suc} x))) + \text{fib} k * \text{fib} (\text{Suc} (\text{Suc} x))$ 
2.  $\text{fib} (\text{Suc} (0 + k)) = \text{fib} (\text{Suc} k) * \text{fib} (\text{Suc} 0) + \text{fib} k * \text{fib} 0$ 
3.  $\text{fib} (\text{Suc} (\text{Suc} 0 + k)) = \text{fib} (\text{Suc} k) * \text{fib} (\text{Suc} (\text{Suc} 0)) + \text{fib} k * \text{fib} (\text{Suc} 0)$ 
```



# Finishing the Proof

- Next, all three subgoals are simplified, with the help of the rewrite rules shown.
- The simplifier automatically includes hundreds of other rewrite rules, as well as various decision procedures.
- This time, no subgoals remain.

```
Fib.thy |*isabelle*|
text{*We disable @{text fib.Suc_Suc} for simplification ...*}
declare fib.Suc_Suc [simp del]

text{*...then prove a version that has a more restrictive pattern.*}
lemma fib_Suc3: "fib (Suc (Suc (Suc n))) = fib (Suc n) + fib (Suc (Suc n))"
  by (rule fib.Suc_Suc)

text {* \medskip Concrete Mathematics, page 280 *}

lemma fib_add: "fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n"
  apply (induct n rule: fib.induct)
  prefer 3
  txt {* simplify the LHS just enough to apply the induction hypotheses *}
  apply (simp add: fib_Suc3)
  apply (simp_all add: fib.Suc_Suc add_mult_distrib2)
  done

lemma fib_Suc_neq_0: "fib (Suc n)  $\neq$  0"
  apply (induct n rule: fib.induct)
  apply (simp_all add: fib.Suc_Suc)
  done

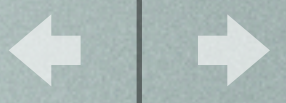
lemma fib_Suc_gr_0: "0 < fib (Suc n)"
  by (insert fib_Suc_neq_0 [of n], simp)

-----XEmacs: Fib.thy (Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin
proof (prove): step 5

fixed variables: n, k

goal (lemma (fib_add)):
No subgoals!
```





# Storing the Theorem

- The done command causes Isabelle to accept the proof, storing the theorem.
- If you were proving this theorem for the first time, you would try various commands right in the editor buffer. You would use *Undo* frequently!
- Once you have succeeded, the file will hold the final version of your proof.
- Using *Undo* on a done command moves the cursor above its proof. Isabelle “forgets” the theorem.

```
Fib.thy |*isabelle*|
text{*We disable @{text fib.Suc_Suc} for simplification ...*}
declare fib.Suc_Suc [simp del]

text{*...then prove a version that has a more restrictive pattern.*}
lemma fib_Suc3: "fib (Suc (Suc (Suc n))) = fib (Suc n) + fib (Suc (Suc n))"
  by (rule fib.Suc_Suc)

text {* \medskip Concrete Mathematics, page 280 *}

lemma fib_add: "fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n"
  apply (induct n rule: fib.induct)
  prefer 3
  txt {* simplify the LHS just enough to apply the induction hypotheses *}
  apply (simp add: fib_Suc3)
  apply (simp_all add: fib.Suc_Suc add_mult_distrib2)
  done

lemma fib_Suc_neq_0: "fib (Suc n)  $\neq$  0"
  apply (induct n rule: fib.induct)
  apply (simp_all add: fib.Suc_Suc)
  done

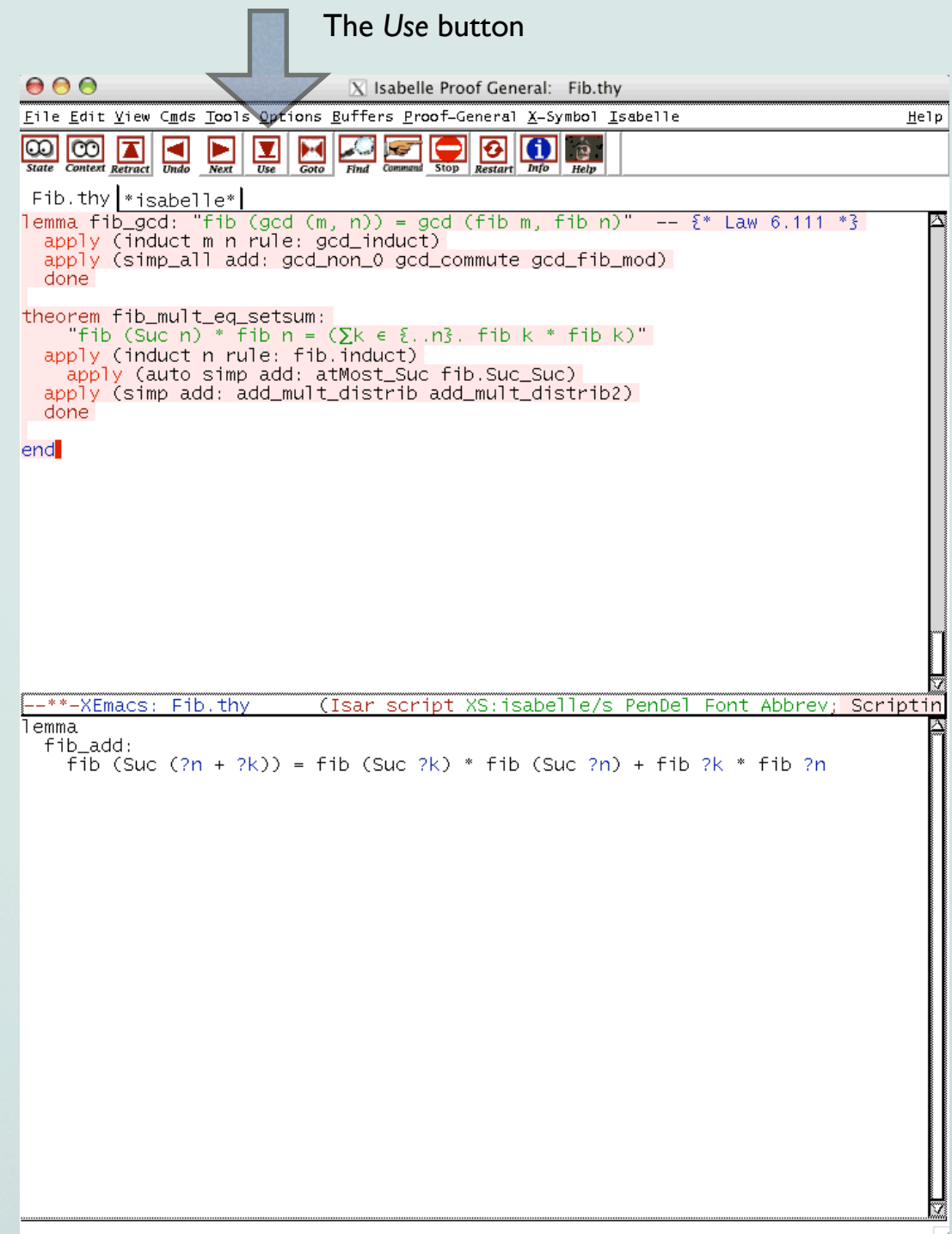
lemma fib_Suc_gr_0: "0 < fib (Suc n)"
  by (insert fib_Suc_neq_0 [of n], simp)

-----XEmacs: Fib.thy      (Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin
lemma
  fib_add:
    fib (Suc (?n + ?k)) = fib (Suc ?k) * fib (Suc ?n) + fib ?k * fib ?n
```



# Processing a Theory

- To run a theory right to the end, click on the *Use* button.
- Now the rest of the theory appears in pink until Isabelle can process it.







# Stop!

- Proof taking too long? Simplifier's looping? Clicked the wrong button? Just click on *Stop*.
- If things behave weirdly after this, perhaps Proof General has got out of sync with Isabelle.
- To get back into sync, use *Goto* to go back to the start of the current proof.
- You can use *Revert* to go back to the top of the theory file.

The Revert button

The Stop button

```
Fib.thy |*isabelle*|
lemma fib_gcd: "fib (gcd (m, n)) = gcd (fib m, fib n)" -- {* Law 6.111 *}
  apply (induct m n rule: gcd_induct)
  apply (simp_all add: gcd_non_0 gcd_commute gcd_fib_mod)
  done

theorem fib_mult_eq_setsum:
  "fib (Suc n) * fib n = (Σ k ∈ {..n}. fib k * fib k)"
  apply (induct n rule: fib_induct)
  apply (auto simp add: atMost_Suc fib_Suc_Suc)
  apply (simp add: add_mult_distrib add_mult_distrib2)
  done

end

---XEmacs: Fib.thy (Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin
*** Interrupt.
*** At command "apply".

Interrupt: script management may be in an inconsistent state
        (but it's probably okay)
```

Use C-c C-. to jump to end of processed region



# Where Am I?

- If a proof fails—or is interrupted—in a long theory file, how do we locate the critical spot?
- You could simply scroll through the file until you find the end of the blue region.
- To jump right there, use the menu item Proof General > Goto Locked End. The key combination CTRL/C- . does the same thing.
- The proof was interrupted during a call to presburger, an arithmetic decision procedure.





# The Proof State

- Clicking on the *State* button reveals the proof state at the given point.
- Here, there was one subgoal left when the proof was interrupted.

The *State* button

```

Isabelle Proof General: Fib.thy
File Edit View Cnds Tools Options Buffers Proof-General X-Symbol Isabelle Help

State Context Retract Undo Next Use Goto Find Command Stop Restart Info Help

Fib.thy |*isabelle*|
\medskip Concrete Mathematics, page 278: Cassini's identity. The proof is
much easier using integers, not natural numbers!
*3
lemma fib_Cassini_int:
  "int (fib (Suc (Suc n)) * fib n) =
   (if n mod 2 = 0 then int (fib (Suc n) * fib (Suc n)) - 1
    else int (fib (Suc n) * fib (Suc n)) + 1)"
  apply (induct n rule: fib.induct)
  apply (simp add: fib.Suc_Suc)
  apply (simp add: fib.Suc_Suc mod_Suc)
  apply (simp add: fib.Suc_Suc add_mult_distrib add_mult_distrib2
    mod_Suc zmult_int [symmetric])
  apply presburger
  done

text{*We now obtain a version for the natural numbers via the coercion
function @{term int}.*3}
theorem fib_Cassini:
  "fib (Suc (Suc n)) * fib n =
   (if n mod 2 = 0 then fib (Suc n) * fib (Suc n) - 1
    else fib (Suc n) * fib (Suc n) + 1)"
  apply (rule int_int_eq [THEN iffD1])
  apply (simp add: fib_Cassini_int)
  apply (subst zdiff_int [symmetric])
  done
--*-XEmacs: Fib.thy (Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin
proof (prove): step 4
fixed variables: n
goal (lemma (fib_Cassini_int), 1 subgoal):
1.  $\forall x. [2 * (\text{int} (\text{fib} (\text{Suc } x)) * \text{int} (\text{fib} (\text{Suc } x))) +$ 
 $\text{int} (\text{fib } x) * \text{int} (\text{fib} (\text{Suc } x)) =$ 
 $(\text{if } (\text{if } \text{Suc } 0 = x \bmod 2 \text{ then } 0 \text{ else } \text{Suc } (x \bmod 2)) = 0$ 
 $\text{then } \text{int} (\text{fib} (\text{Suc} (\text{Suc } x)) * \text{fib} (\text{Suc} (\text{Suc } x))) - 1$ 
 $\text{else } \text{int} (\text{fib} (\text{Suc} (\text{Suc } x)) * \text{fib} (\text{Suc} (\text{Suc } x))) + 1);$ 
 $\text{int} (\text{fib } x) * \text{int} (\text{fib } x) + \text{int} (\text{fib} (\text{Suc } x)) * \text{int} (\text{fib } x) =$ 
 $(\text{if } x \bmod 2 = 0 \text{ then } \text{int} (\text{fib} (\text{Suc } x) * \text{fib} (\text{Suc } x)) - 1$ 
 $\text{else } \text{int} (\text{fib} (\text{Suc } x) * \text{fib} (\text{Suc } x)) + 1)]$ 
 $\Rightarrow \text{Suc } 0 \neq x \bmod 2 \rightarrow x \bmod 2 = 0$ 

```

Use C-c C-o to rotate output buffers; C-c C-w to clear response & trace.





# Finding Theorems

- Isabelle provides thousands of lemmas. How do you find the ones you need? One way is to click the *Find* button.
- Then, type some constants—or entire terms—into the XEmacs minibuffer.
- Type the term " $x+y*z = u/v$ ", with the quotation marks. Isabelle finds all theorems involving the infix operators  $+$ ,  $*$ ,  $/$  and  $=$ .
- A term stands for the set of constants it contains: it is not used as a pattern.

The *Find* button

The screenshot shows the Isabelle XEmacs editor window titled "Isabelle Proof General: Fib.thy". The toolbar at the top includes buttons for State, Context, Retract, Undo, Next, Use, Goto, Find, Command, Stop, Restart, Info, and Help. The Find button is highlighted with a blue arrow pointing to it from the text "The Find button".

The main text area shows the content of `Fib.thy`, which includes a lemma `fib_Cassini_int` and a theorem `fib_Cassini`. The lemma is defined as:

```
lemma fib_Cassini_int:
  "int (fib (Suc (Suc n)) * fib n) =
   (if n mod 2 = 0 then int (fib (Suc n) * fib (Suc n)) - 1
    else int (fib (Suc n) * fib (Suc n)) + 1)"
  apply (induct n rule: fib.induct)
  apply (simp add: fib.Suc_Suc)
  apply (simp add: fib.Suc_Suc mod_Suc)
  apply (simp add: fib.Suc_Suc add_mult_distrib2 mod_Suc zmult_int [symmetric])
  apply presburger
  done
```

The theorem `fib_Cassini` is defined as:

```
theorem fib_Cassini:
  "fib (Suc (Suc n)) * fib n =
   (if n mod 2 = 0 then fib (Suc n) * fib (Suc n) - 1
    else fib (Suc n) * fib (Suc n) + 1)"
  apply (rule int_int_eq [THEN iffD1])
  apply (simp add: fib_Cassini_int)
  apply (subst zdiff_int [symmetric])
  done
```

The minibuffer at the bottom shows the search results for the term `fib Suc 0`:

```
Find theorems containing: fib Suc 0
```

Type some constants





# Theorems Found

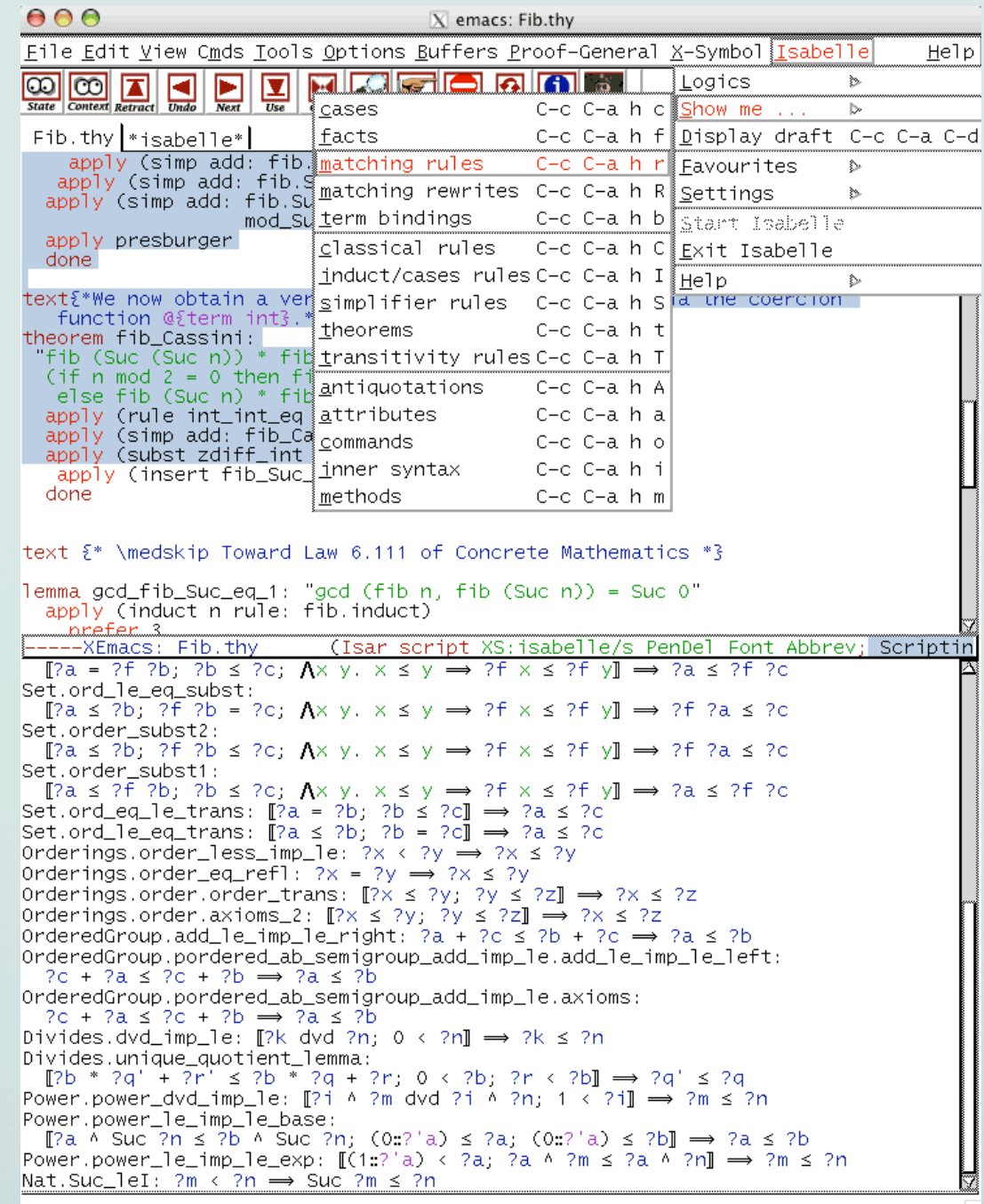
- The response buffer lists the theorems containing *all* of the listed constants.
- If you are lucky, there will be just a few rather than hundreds!
- The more constants you type, the fewer theorems will be displayed.
- Variables mentioned in the current goal are viewed as constants for this purpose.





# The Isabelle Menu

- The Isabelle menu gives access to Isabelle commands and information.
- Isabelle > Show me... provides other ways of finding theorems: matching rules and matching rewrites.
- In the example, the current subgoal has the form  $x \leq y$ , and matching rules displays all known theorems that can prove a conclusion of that form.

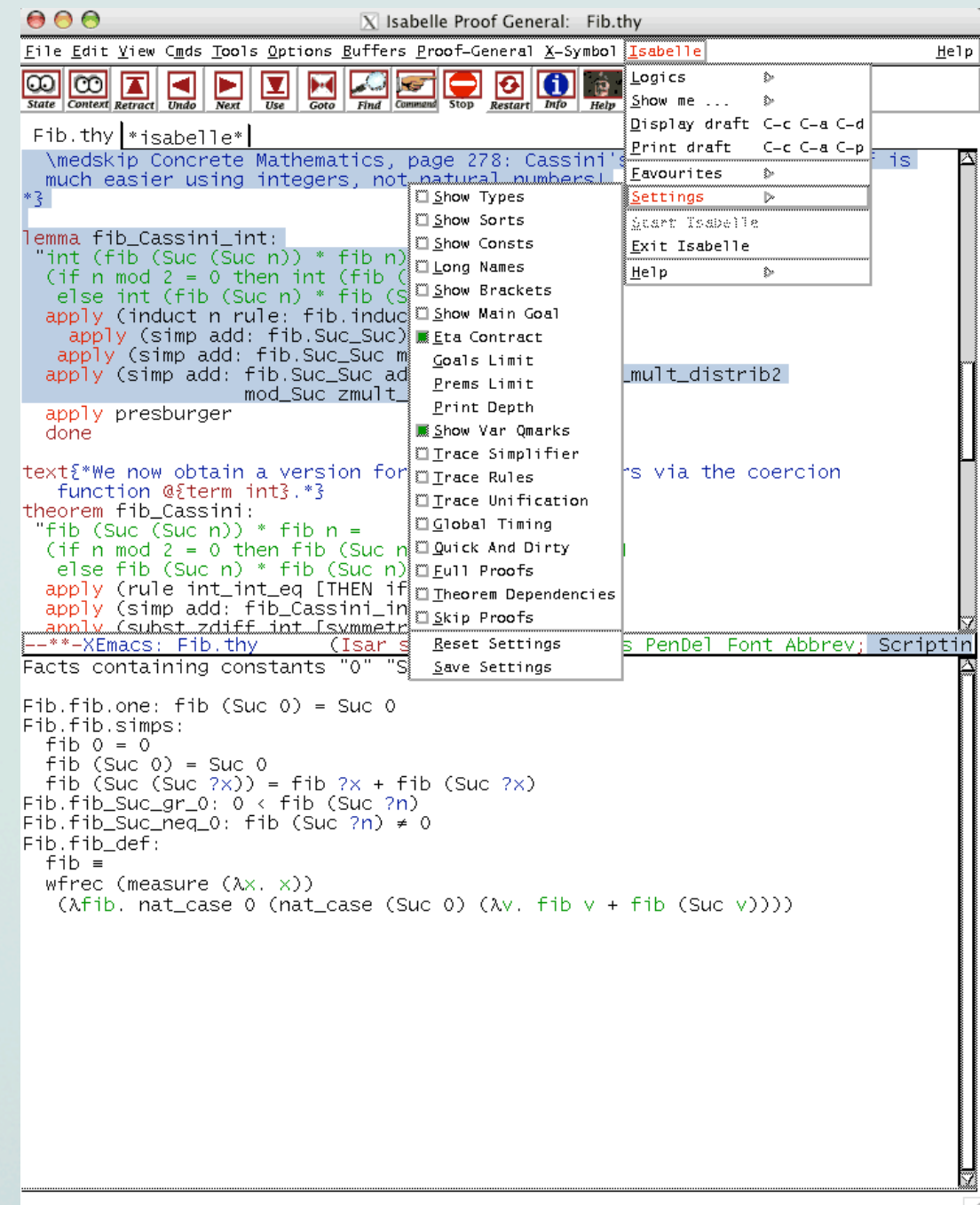






# Settings

- The menu Isabelle > Settings can request the display of types, execution times, and various traces.
- There are printing options to suit special situations, such as enormous subgoals.
- Use Show Types and Show Sorts to cause more type information to be displayed.
- The various Show options make the output more verbose, but more explicit, and are helpful for diagnosing problems.

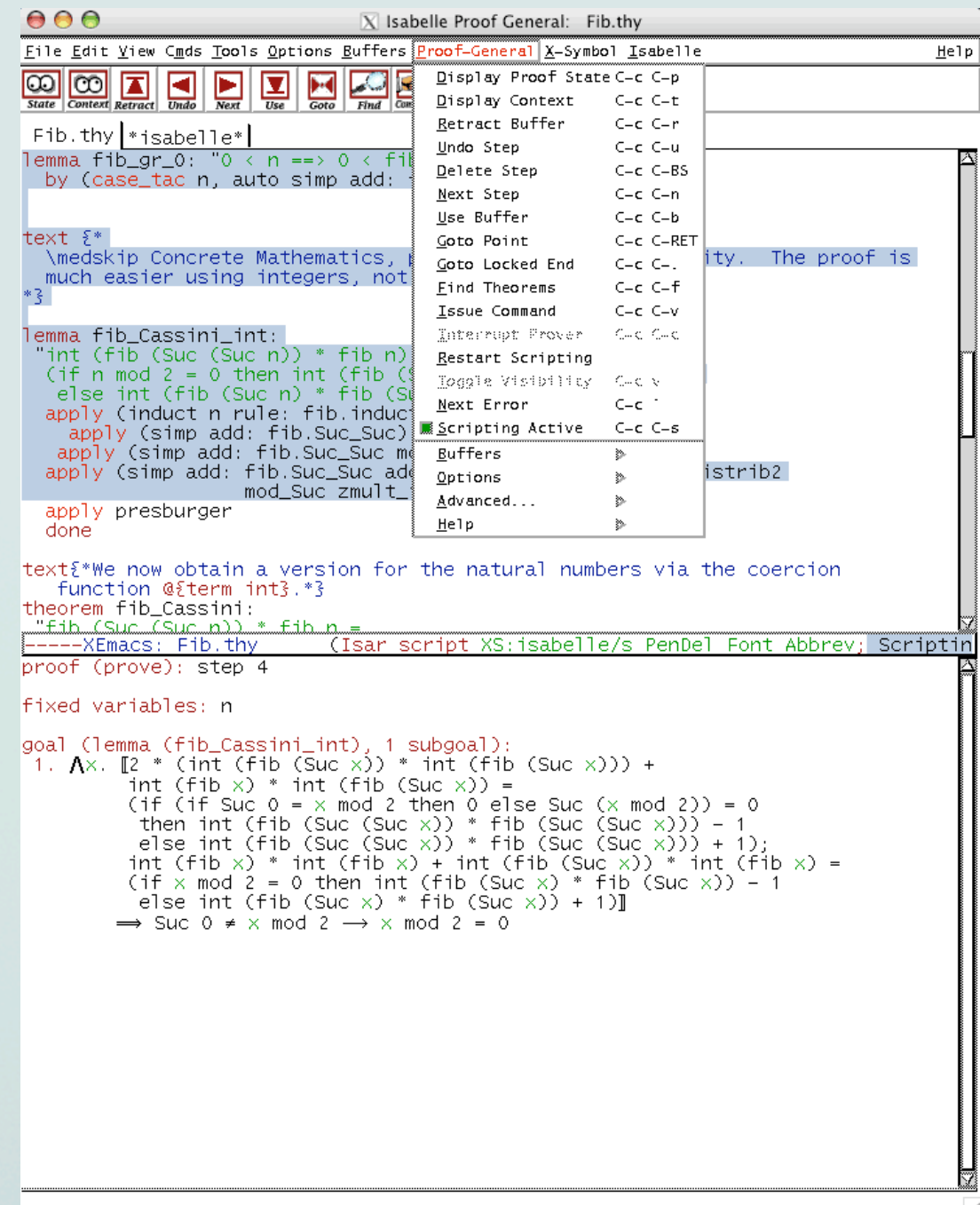






# The PG Menu

- The Proof General menu gives access to many commands.
- The main commands are available from the toolbar. A notable exception is Goto Locked End.
- Choose Proof General > Buffers > Trace to see tracing output.

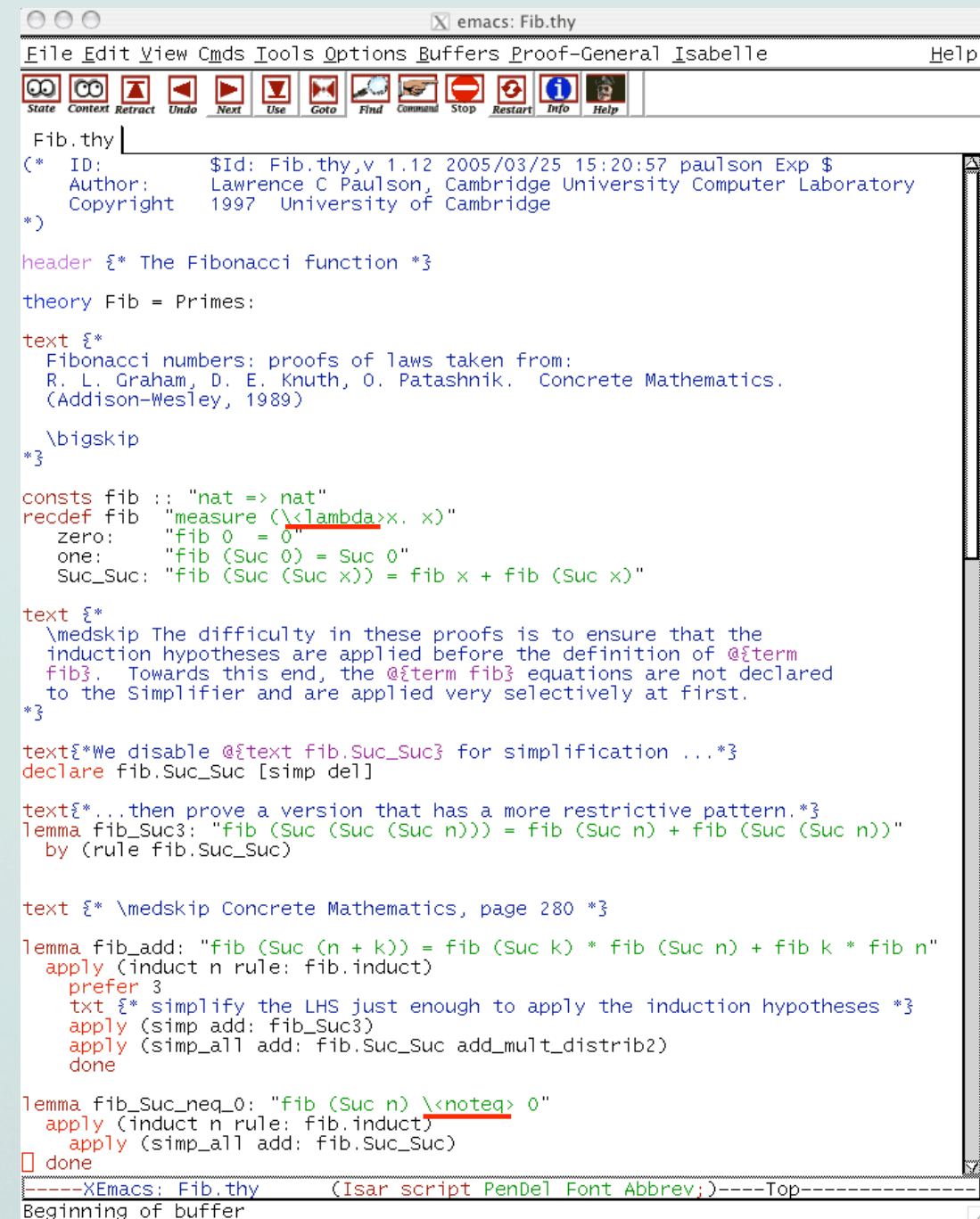






# Mathematical Symbols

- Proof General uses the X-Symbol package to display mathematical symbols such as  $\lambda \leq \neq \in \notin \cup$  and  $\cap$ .
- The package is included with Proof General, but may need to be switched on.
- If X-Symbol mode is off, Proof General will display ASCII escape sequences, as shown on the right.



```
emac: Fib.thy
File Edit View Cms Tools Options Buffers Proof-General Isabelle Help
State Context Retract Undo Next Use Goto Find Command Stop Restart Info Help

Fib.thy
(* ID: $Id: Fib.thy,v 1.12 2005/03/25 15:20:57 paulson Exp $
   Author: Lawrence C Paulson, Cambridge University Computer Laboratory
   Copyright 1997 University of Cambridge
*)

header {* The Fibonacci function *}

theory Fib = Primes:

text {*
  Fibonacci numbers: proofs of laws taken from:
  R. L. Graham, D. E. Knuth, O. Patashnik. Concrete Mathematics.
  (Addison-Wesley, 1989)
  \bigskip
  *}

consts fib :: "nat => nat"
recdef fib "measure (\<lambda>x. x)"
  zero: "fib 0 = 0"
  one: "fib (Suc 0) = Suc 0"
  Suc_Suc: "fib (Suc (Suc x)) = fib x + fib (Suc x)"

text {*
  \medskip The difficulty in these proofs is to ensure that the
  induction hypotheses are applied before the definition of @term
  fib3. Towards this end, the @term fib3 equations are not declared
  to the Simplifier and are applied very selectively at first.
  *}

text{*We disable @text fib.Suc_Suc3 for simplification ...*}
declare fib.Suc_Suc [simp del]

text{*...then prove a version that has a more restrictive pattern.*}
lemma fib_Suc3: "fib (Suc (Suc (Suc n))) = fib (Suc n) + fib (Suc (Suc n))"
  by (rule fib.Suc_Suc)

text {* \medskip Concrete Mathematics, page 280 *}

lemma fib_add: "fib (Suc (n + k)) = fib (Suc n) * fib (Suc k) + fib n * fib (Suc k)"
  apply (induct n rule: fib.induct)
  prefer 3
  txt {* simplify the LHS just enough to apply the induction hypotheses *}
  apply (simp add: fib_Suc3)
  apply (simp_all add: fib.Suc_Suc add_mult_distrib2)
  done

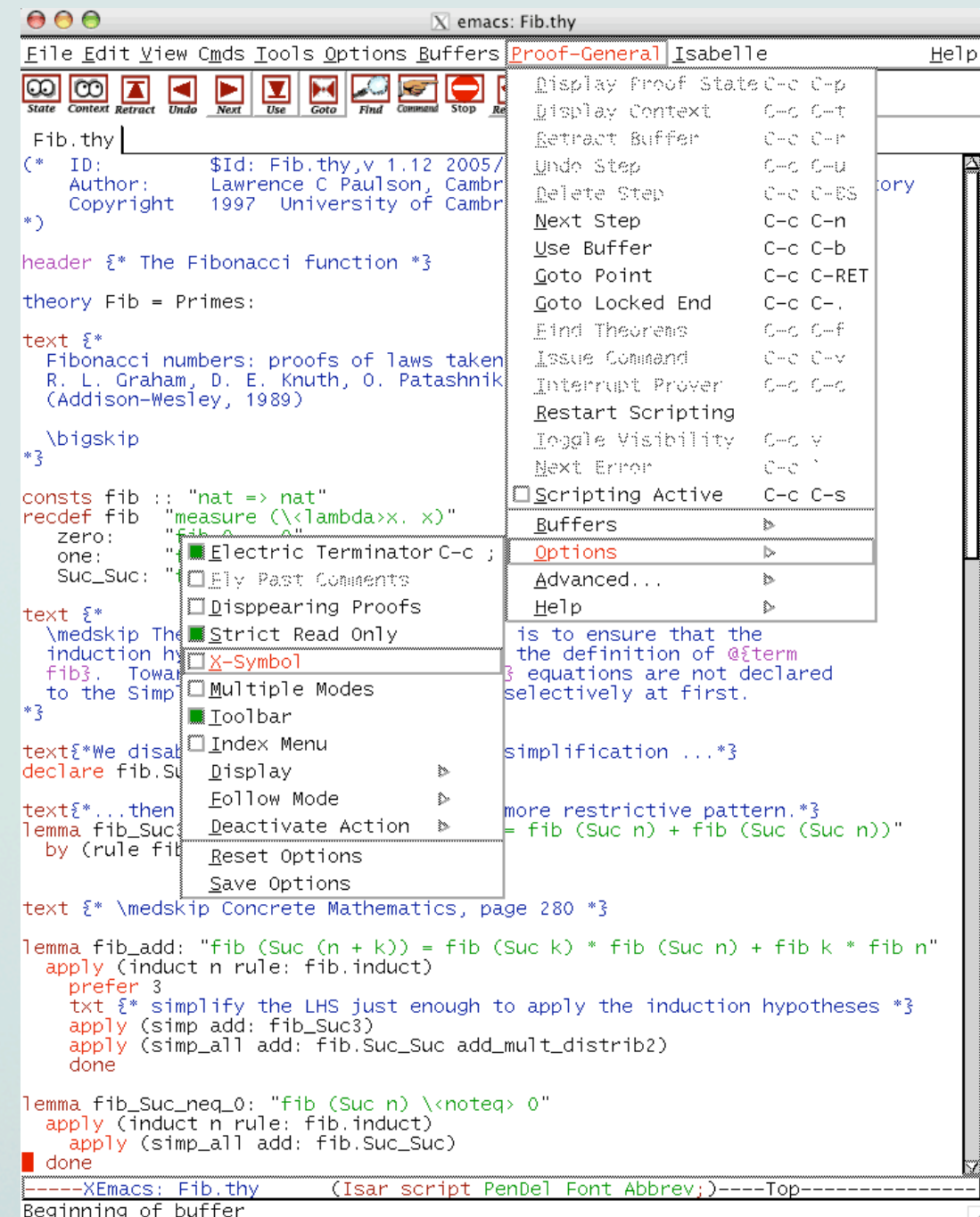
lemma fib_Suc_neq_0: "fib (Suc n) \<noteq> 0"
  apply (induct n rule: fib.induct)
  apply (simp_all add: fib.Suc_Suc)
  done
-----XEmacs: Fib.thy (Isar script PenDel Font Abbrev;)-----Top-----
Beginning of buffer
```





# Enabling Symbols

- To enable X-Symbol mode, select the menu item Proof General > Options > X-Symbol.
- Then, make this setting permanent using Proof General > Options > Save Options.
- Take the time to explore the many other options and settings on offer.







# Go Forth and Prove!

- Try out this theory yourself: you will find it in `src/HOL/NumberTheory/Fib.thy`.
- For more information on Isabelle, read the [documentation](#).
- For more information on Proof General, see its [user manual](#).
- Have fun!

The screenshot shows the Isabelle Proof General interface with the file `Fib.thy` open. The window title is "Isabelle Proof General: Fib.thy". The menu bar includes File, Edit, View, Cmps, Tools, Options, Buffers, Proof-General, X-Symbol, Isabelle, and Help. The toolbar contains icons for State, Context, Retract, Undo, Next, Use, Goto, Find, Command, Stop, Restart, Info, and Help. The main text area shows the following code:

```
Fib.thy |*isabelle*|
apply (induct m n rule: gcd_induct)
apply (simp_all add: gcd_non_0 gcd_commute gcd_fib_mod)
done

theorem fib_mult_eq_setsum:
  "fib (Suc n) * fib n = (Σ k ∈ {..n}. fib k * fib k)"
  apply (induct n rule: fib_induct)
  apply (auto simp add: atMost_Suc fib_Suc_Suc)
  apply (simp add: add_mult_distrib add_mult_distrib2)
  done
end
```

The status bar at the bottom indicates the file is `Fib.thy` and the script is `(Isar script XS:isabelle/s PenDel Font Abbrev; Scriptin`.