

Interactive Formal Verification

6: Sets

Tjark Weber
(Slides: Lawrence C Paulson)
Computer Laboratory
University of Cambridge

Set Notation in Isabelle

Set Notation in Isabelle

- Set notation is crucial to mathematical discourse.

Set Notation in Isabelle

- Set notation is crucial to mathematical discourse.
- Set-theoretic abstractions naturally express many complex constructions.

Set Notation in Isabelle

- Set notation is crucial to mathematical discourse.
- Set-theoretic abstractions naturally express many complex constructions.
- A set in higher-order logic is a *boolean-valued map*.

Set Notation in Isabelle

- Set notation is crucial to mathematical discourse.
- Set-theoretic abstractions naturally express many complex constructions.
- A set in higher-order logic is a *boolean-valued map*.
- Its elements must all have the *same type*

Set Theory Primitives

Set Theory Primitives

- The type `α set`, which abbreviates `$\alpha \Rightarrow \text{bool}$`

Set Theory Primitives

- The type α `set`, which abbreviates $\alpha \Rightarrow \text{bool}$
- The membership relation: \in

Set Theory Primitives

- The type α `set`, which abbreviates $\alpha \Rightarrow \text{bool}$
- The membership relation: \in
- The subset relation: \subseteq
 - Reflexive, anti-symmetric, transitive

Set Theory Primitives

- The type α `set`, which abbreviates $\alpha \Rightarrow \text{bool}$
- The membership relation: \in
- The subset relation: \subseteq
 - Reflexive, anti-symmetric, transitive
- The empty set: $\{\}$

Set Theory Primitives

- The type α `set`, which abbreviates $\alpha \Rightarrow \text{bool}$
- The membership relation: \in
- The subset relation: \subseteq
 - Reflexive, anti-symmetric, transitive
- The empty set: $\{\}$
- The universal set: $UNIV$

Basic Set Theory Operations

Basic Set Theory Operations

$$e \in \{x. P(x)\} \iff P(e)$$

Basic Set Theory Operations

$$e \in \{x. P(x)\} \iff P(e)$$

$$e \in \{x \in A. P(x)\} \iff e \in A \wedge P(e)$$

Basic Set Theory Operations

$$e \in \{x. P(x)\} \iff P(e)$$

$$e \in \{x \in A. P(x)\} \iff e \in A \wedge P(e)$$

$$e \in -A \iff e \notin A$$

Basic Set Theory Operations

$$e \in \{x. P(x)\} \iff P(e)$$

$$e \in \{x \in A. P(x)\} \iff e \in A \wedge P(e)$$

$$e \in -A \iff e \notin A$$

$$e \in A \cup B \iff e \in A \vee e \in B$$

Basic Set Theory Operations

$$e \in \{x. P(x)\} \iff P(e)$$

$$e \in \{x \in A. P(x)\} \iff e \in A \wedge P(e)$$

$$e \in -A \iff e \notin A$$

$$e \in A \cup B \iff e \in A \vee e \in B$$

$$e \in A \cap B \iff e \in A \wedge e \in B$$

Basic Set Theory Operations

$$e \in \{x. P(x)\} \iff P(e)$$

$$e \in \{x \in A. P(x)\} \iff e \in A \wedge P(e)$$

$$e \in -A \iff e \notin A$$

$$e \in A \cup B \iff e \in A \vee e \in B$$

$$e \in A \cap B \iff e \in A \wedge e \in B$$

$$e \in \text{Pow}(A) \iff e \subseteq A$$

Big Union and Intersection

Big Union and Intersection

$$e \in \left(\bigcup x. B(x) \right) \iff \exists x. e \in B(x)$$

Big Union and Intersection

$$e \in \left(\bigcup x. B(x) \right) \iff \exists x. e \in B(x)$$

$$e \in \left(\bigcup_{x \in A} B(x) \right) \iff \exists x \in A. e \in B(x)$$

Big Union and Intersection

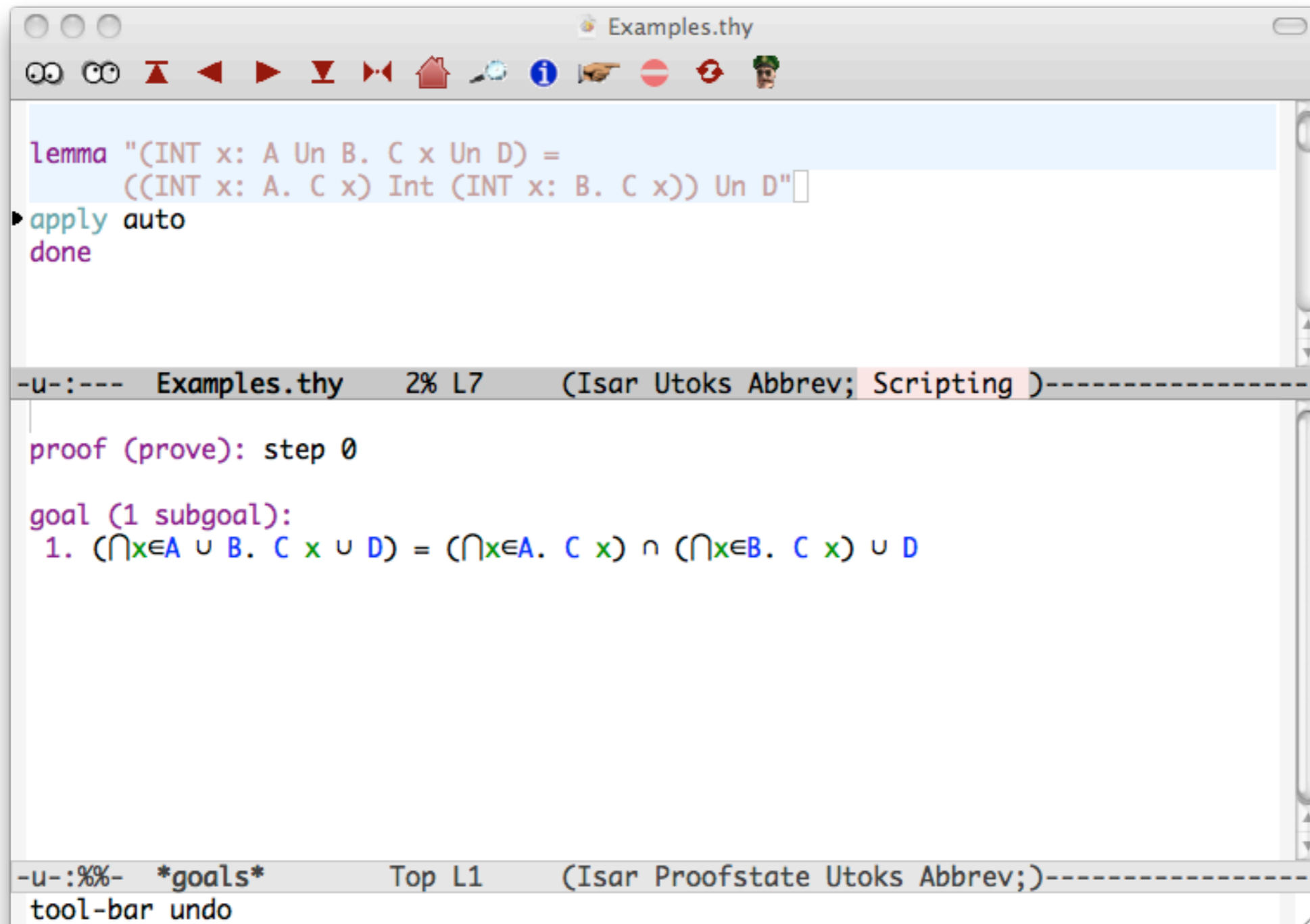
$$e \in \left(\bigcup x. B(x) \right) \iff \exists x. e \in B(x)$$
$$e \in \left(\bigcup_{x \in A} B(x) \right) \iff \exists x \in A. e \in B(x)$$
$$e \in \bigcup A \iff \exists x \in A. e \in x$$

Big Union and Intersection

$$e \in \left(\bigcup x. B(x) \right) \iff \exists x. e \in B(x)$$
$$e \in \left(\bigcup_{x \in A} B(x) \right) \iff \exists x \in A. e \in B(x)$$
$$e \in \bigcup A \iff \exists x \in A. e \in x$$

And the analogous forms of intersections...

A Simple Set Theory Proof



The screenshot shows a window titled "Examples.thy" with a toolbar at the top. The main text area contains the following code:

```
lemma "(INT x: A Un B. C x Un D) =  
      ((INT x: A. C x) Int (INT x: B. C x)) Un D"  
apply auto  
done
```

Below the code is a status bar with the text: `-u-:--- Examples.thy 2% L7 (Isar Utoks Abbrev; Scripting)`

The proof state is shown in a separate window below, with the text:

```
proof (prove): step 0  
goal (1 subgoal):  
1.  $(\bigcap_{x \in A \cup B} C x \cup D) = (\bigcap_{x \in A} C x) \cap (\bigcap_{x \in B} C x) \cup D$ 
```

At the bottom, another status bar shows: `-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)` and a `tool-bar undo` button.

A Simple Set Theory Proof

The screenshot shows a proof assistant window titled "Examples.thy". The main text area contains the following code:

```
lemma "(INT x: A Un B. C x Un D) =  
      ((INT x: A. C x) Int (INT x: B. C x)) Un D"  
apply auto  
done
```

Below the code is a status bar with the text: "-u-:--- Examples.thy 2% L7 (Isar Utop)".

The goal section shows:

```
proof (prove): step 0  
goal (1 subgoal):  
1.  $(\bigcap_{x \in A \cup B}. C x \cup D) = (\bigcap_{x \in A}. C x) \cap (\bigcap_{x \in B}. C x) \cup D$ 
```

At the bottom, another status bar reads: "-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
tool-bar undo".

A blue callout box with white text says: "plain ASCII syntax is an alternative to special symbols". Two red arrows point from this box to the lemma statement and the goal statement in the code above.

A Simple Set Theory Proof

The screenshot shows a window titled "Examples.thy" with a toolbar containing navigation icons. The main text area contains the following code:

```
lemma "(INT x: A Un B. C x Un D) =  
      ((INT x: A. C x) Int (INT x: B. C x)) Un D"  
apply auto  
done
```

Below the code is a status bar with the text: `-u-:--- Examples.thy 2% L9 (Isar Utoks Abbrev; Scripting)`. The bottom section of the window shows the proof state:

```
proof (prove): step 1  
goal:  
No subgoals!
```

At the very bottom, another status bar reads: `-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)` and a `tool-bar next` button is visible.

Functions

Functions

$$e \in (f' A) \iff \exists x \in A. e = f(x)$$

Functions

$$e \in (f' A) \iff \exists x \in A. e = f(x)$$

$$e \in (f^{-1} A) \iff f(e) \in A$$

Functions

$$e \in (f' A) \iff \exists x \in A. e = f(x)$$

$$e \in (f^{-1} A) \iff f(e) \in A$$

$$f(x:=y) = (\lambda z. \text{if } z = x \text{ then } y \text{ else } f(z))$$

Functions

$$e \in (f' A) \iff \exists x \in A. e = f(x)$$

$$e \in (f^{-1} A) \iff f(e) \in A$$

$$f(x:=y) = (\lambda z. \text{if } z = x \text{ then } y \text{ else } f(z))$$

- Also **inj**, **surj**, **bij**, **inv**, etc. (injective,...)

Functions

$$e \in (f' A) \iff \exists x \in A. e = f(x)$$

$$e \in (f^{-1} A) \iff f(e) \in A$$

$$f(x:=y) = (\lambda z. \text{if } z = x \text{ then } y \text{ else } f(z))$$

- Also *inj*, *surj*, *bij*, *inv*, etc. (injective,...)
- Don't *re-invent* image and inverse image!!

Finite Set Notation

Finite Set Notation

$$\{a_1, \dots, a_n\} = \text{insert}(a_1, \dots, \text{insert}(a_n, \{\}))$$

Finite Set Notation

$$\{a_1, \dots, a_n\} = \text{insert}(a_1, \dots, \text{insert}(a_n, \{\}))$$

$$e \in \text{insert}(a, B) \iff e = a \vee e \in B$$

Finite Sets

A finite set is defined *inductively*
in terms of `{}` and `insert`

Finite Sets

A finite set is defined *inductively*
in terms of `{}` and `insert`

$$\text{finite}(A \cup B) = (\text{finite } A \wedge \text{finite } B)$$

Finite Sets

A finite set is defined *inductively*
in terms of `{}` and `insert`

$$\text{finite}(A \cup B) = (\text{finite } A \wedge \text{finite } B)$$

$$\text{finite } A \implies \text{card}(\text{Pow } A) = 2^{\text{card } A}$$

Intervals, Sums and Products

Intervals, Sums and Products

$$\{ \cdot \cdot < u \} == \{ x \cdot \quad x < u \}$$

$$\{ \cdot \cdot u \} == \{ x \cdot \quad x \leq u \}$$

$$\{ 1 < \cdot \cdot \} == \{ x \cdot \quad 1 < x \}$$

$$\{ 1 \cdot \cdot \} == \{ x \cdot \quad 1 \leq x \}$$

$$\{ 1 < \cdot \cdot < u \} == \{ 1 < \cdot \cdot \} \cap \{ \cdot \cdot < u \}$$

$$\{ 1 \cdot \cdot < u \} == \{ 1 \cdot \cdot \} \cap \{ \cdot \cdot < u \}$$

Intervals, Sums and Products

$$\{..<u\} == \{x. \ x < u\}$$

$$\{..u\} == \{x. \ x \leq u\}$$

$$\{1<..\} == \{x. \ 1 < x\}$$

$$\{1..\} == \{x. \ 1 \leq x\}$$

$$\{1<..<u\} == \{1<..\} \cap \{..<u\}$$

$$\{1..\<u\} == \{1..\} \cap \{..<u\}$$

setsum f A **and** setprod f A

Intervals, Sums and Products

$$\{..<u\} == \{x. x < u\}$$

$$\{..u\} == \{x. x \leq u\}$$

$$\{1<..\} == \{x. 1 < x\}$$

$$\{1..\} == \{x. 1 \leq x\}$$

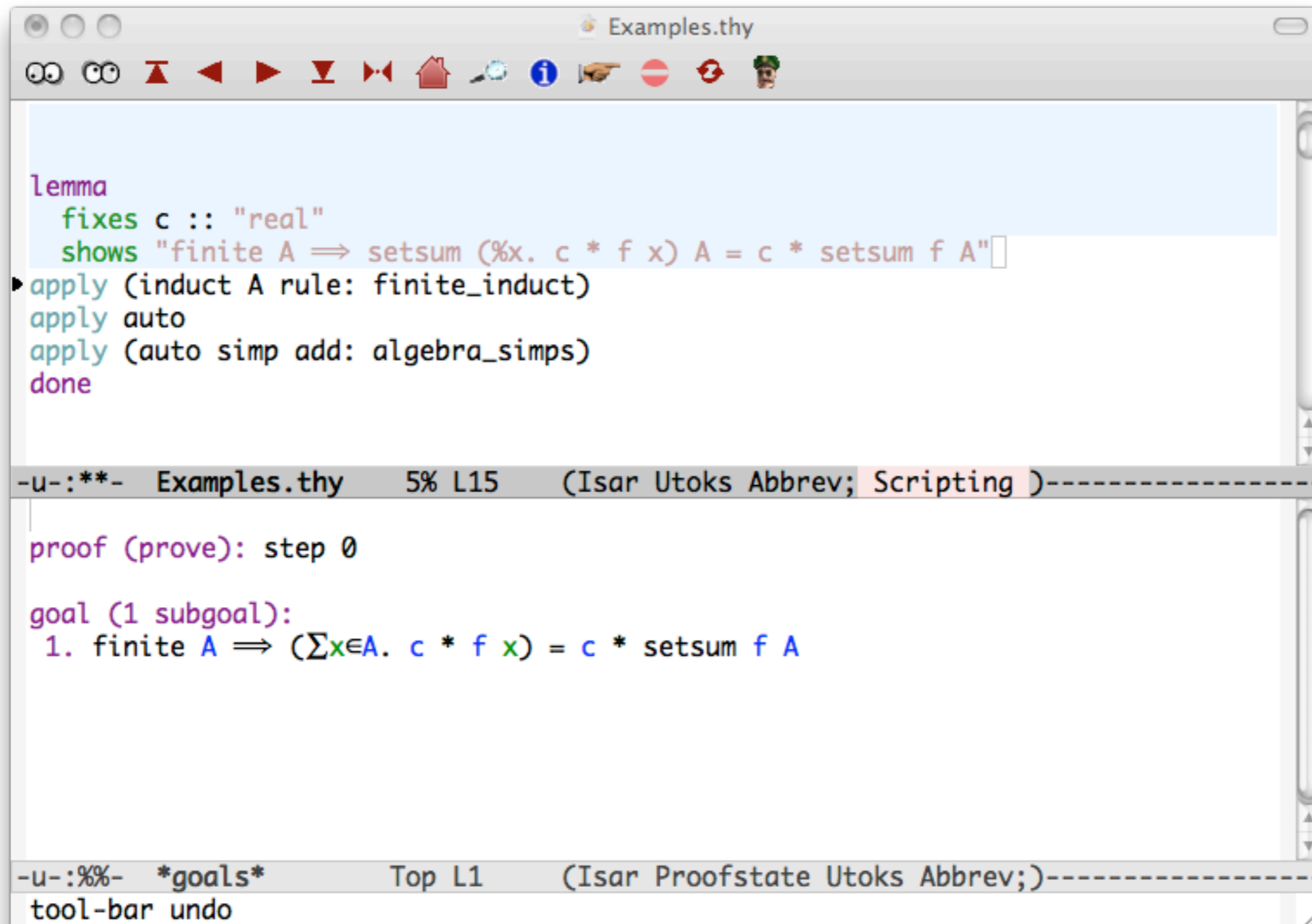
$$\{1<..<u\} == \{1<..\} \cap \{..<u\}$$

$$\{1..\<u\} == \{1..\} \cap \{..<u\}$$

setsum f A **and** setprod f A

$\sum_{i \in I}. f$ **and** $\prod_{i \in I}. f$

A Harder Proof Involving Sets



```
Examples.thy
- - - - -
lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
- apply (induct A rule: finite_induct)
- apply auto
- apply (auto simp add: algebra_simps)
- done

- u-:***- Examples.thy 5% L15 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 0
goal (1 subgoal):
  1. finite A  $\implies$  ( $\sum_{x \in A}. c * f x$ ) = c * setsum f A

- u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
tool-bar undo
```

A Harder Proof Involving Sets

The screenshot shows a window titled "Examples.thy" with a toolbar at the top. The main text area contains the following code:

```
lemma
  fixes c :: "real"
  shows "finite A  $\Rightarrow$  setsum (%x. c * f x) A = c * setsum f A"
  apply (induct A rule: finite_induct)
  apply auto
  apply (auto simp add: algebra_simps)
  done
```

An orange arrow points from a blue callout box containing the text "a way to specify the types of variables" to the line `fixes c :: "real"`.

Below the code, a status bar shows: `-u-:**- Examples.thy 5% L15 (Isar Utoks Abbrev; Scripting)`

The proof state is shown in a separate window below, with the following content:

```
proof (prove): step 0

goal (1 subgoal):
  1. finite A  $\Rightarrow$  ( $\sum_{x \in A} c * f x$ ) = c * setsum f A
```

At the bottom, another status bar shows: `-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)` and a `tool-bar undo` button.

A Harder Proof Involving Sets

The screenshot shows a theorem prover interface with a toolbar at the top. The main window displays the following code:

```
lemma
  fixes c :: "real"
  shows "finite A  $\Rightarrow$  setsum (%x. c * f x) A = c * setsum f A"
  apply (induct A rule: finite_induct)
  apply auto
  apply (auto simp add: algebra_simps)
  done
```

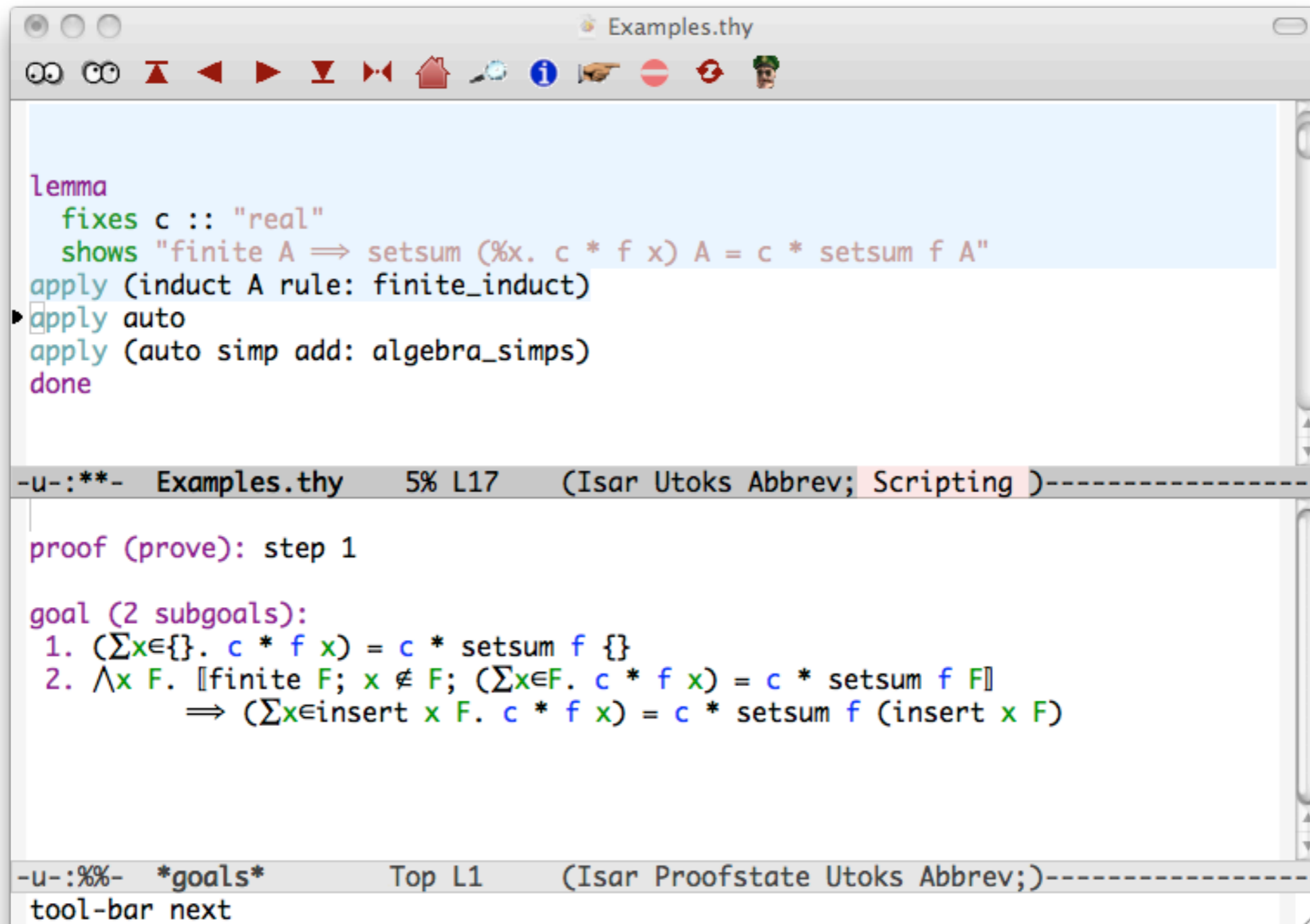
Two callout boxes with red arrows point to specific parts of the code:

- A box labeled "a way to specify the types of variables" points to the line `fixes c :: "real"`.
- A box labeled "induction on the finite set, A" points to the line `apply (induct A rule: finite_induct)`.

The bottom of the window shows the proof state:

```
-u-:***- Examples.thy 5% L15 (Isar Utoks Abbrev; Scripting )-----
|
| proof (prove): step 0
|
| goal (1 subgoal):
| 1. finite A  $\Rightarrow$  ( $\sum_{x \in A} c * f x$ ) = c * setsum f A
|
|-----
-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
|
| tool-bar undo
```

Outcome of the Induction



```
Examples.thy
--u-:***- Examples.thy 5% L17 (Isar Utoks Abbrev; Scripting )-----
lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
apply (induct A rule: finite_induct)
 $\square$  apply auto
apply (auto simp add: algebra_simps)
done

--u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
proof (prove): step 1

goal (2 subgoals):
1.  $(\sum_{x \in \{ \}}. c * f x) = c * \text{setsum } f \{ \}$ 
2.  $\wedge x F. [\text{finite } F; x \notin F; (\sum_{x \in F}. c * f x) = c * \text{setsum } f F]$ 
 $\implies (\sum_{x \in \text{insert } x F}. c * f x) = c * \text{setsum } f (\text{insert } x F)$ 

tool-bar next
```

Outcome of the Induction

```
Examples.thy

lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
apply (induct A rule: finite_induct)
 $\square$  apply auto
apply (auto simp add: algebra_simps)
done

-u-:***- Examples.thy 5% L17 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 1
goal (2 subgoals):
1.  $(\sum_{x \in \{ \}}. c * f x) = c * \text{setsum } f \{ \}$ 
2.  $\wedge x F. [\text{finite } F; x \notin F; (\sum_{x \in F}. c * f x) = c * \text{setsum } f F]$ 
 $\implies (\sum_{x \in \text{insert } x F}. c * f x) = c * \text{setsum } f (\text{insert } x F)$ 

-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
tool-bar next
```

base case: A is empty



Outcome of the Induction

```
Examples.thy

lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
  apply (induct A rule: finite_induct)
  apply auto
  apply (auto simp add: algebra_simps)
  done

-u-:***- Examples.thy 5% L17 (Isar Utoks Abbrev; Scripting )-----

proof (prove): step 1

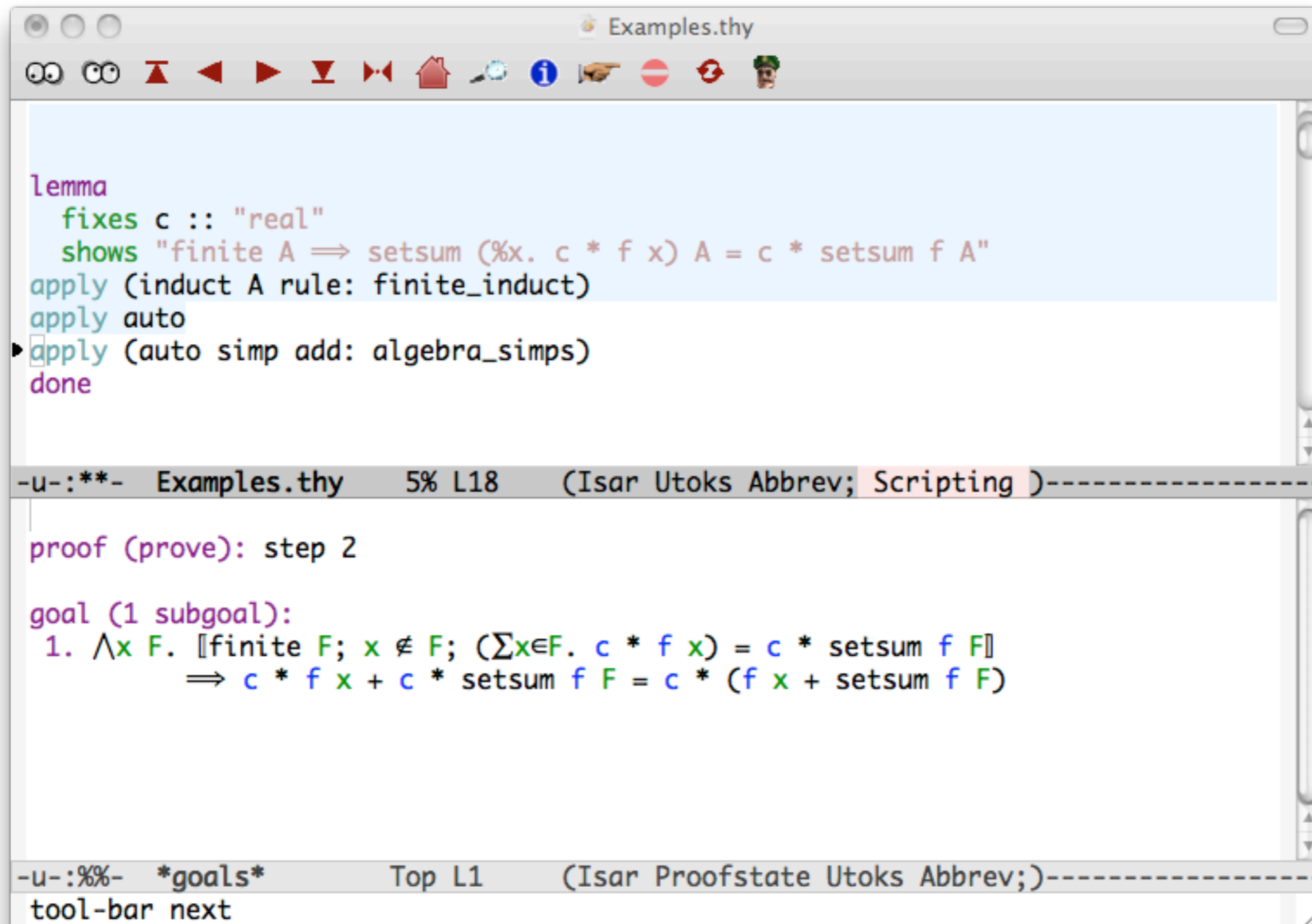
goal (2 subgoals):
1.  $(\sum_{x \in \{ \}}$  c * f x) = c * setsum f { }
2.  $\wedge x F. [$ finite F; x  $\notin$  F;  $(\sum_{x \in F. c * f x) = c * setsum f F]$ 
 $\implies (\sum_{x \in \text{insert } x F. c * f x) = c * setsum f (\text{insert } x F)$ 

-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
tool-bar next
```

base case: A is empty

inductive step: A = insert x F

Almost There!



```
Examples.thy
--u-:***- Examples.thy 5% L18 (Isar Utoks Abbrev; Scripting )-----
lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
  apply (induct A rule: finite_induct)
  apply auto
  apply (auto simp add: algebra_simps)
  done

proof (prove): step 2

goal (1 subgoal):
  1.  $\forall x \in F. [\text{finite } F; x \notin F; (\sum_{x \in F}. c * f x) = c * \text{setsum } f F]$ 
      $\implies c * f x + c * \text{setsum } f F = c * (f x + \text{setsum } f F)$ 

--u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
tool-bar next
```

Almost There!

```
Examples.thy

lemma
  fixes c :: "real"
  shows "finite A  $\Rightarrow$  setsum (%x. c * f x) A = c * setsum f A"
apply (induct A rule: finite_induct)
apply auto
 $\square$  apply (auto simp add: algebra_simps)
done

-u-:***- Examples.thy 5% L18 (Isar Utoks Abbrev; Scripting )-----

proof (prove): step 2

goal (1 subgoal):
1.  $\wedge x \in F. [\text{finite } F; x \notin F; (\sum_{x \in F} c * f x) = c * \text{setsum } f F]$ 
 $\Rightarrow c * f x + c * \text{setsum } f F = c * (f x + \text{setsum } f F)$ 

need to apply a distributive law

-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
tool-bar next
```

Almost There!

```
Examples.thy

lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
apply (induct A rule: finite_induct)
apply auto
▶ apply (auto simp add: algebra_simps)
done

-u-:***- Examples.thy 5% L18 (Isar Utoks Abbrev; Scripting )-----

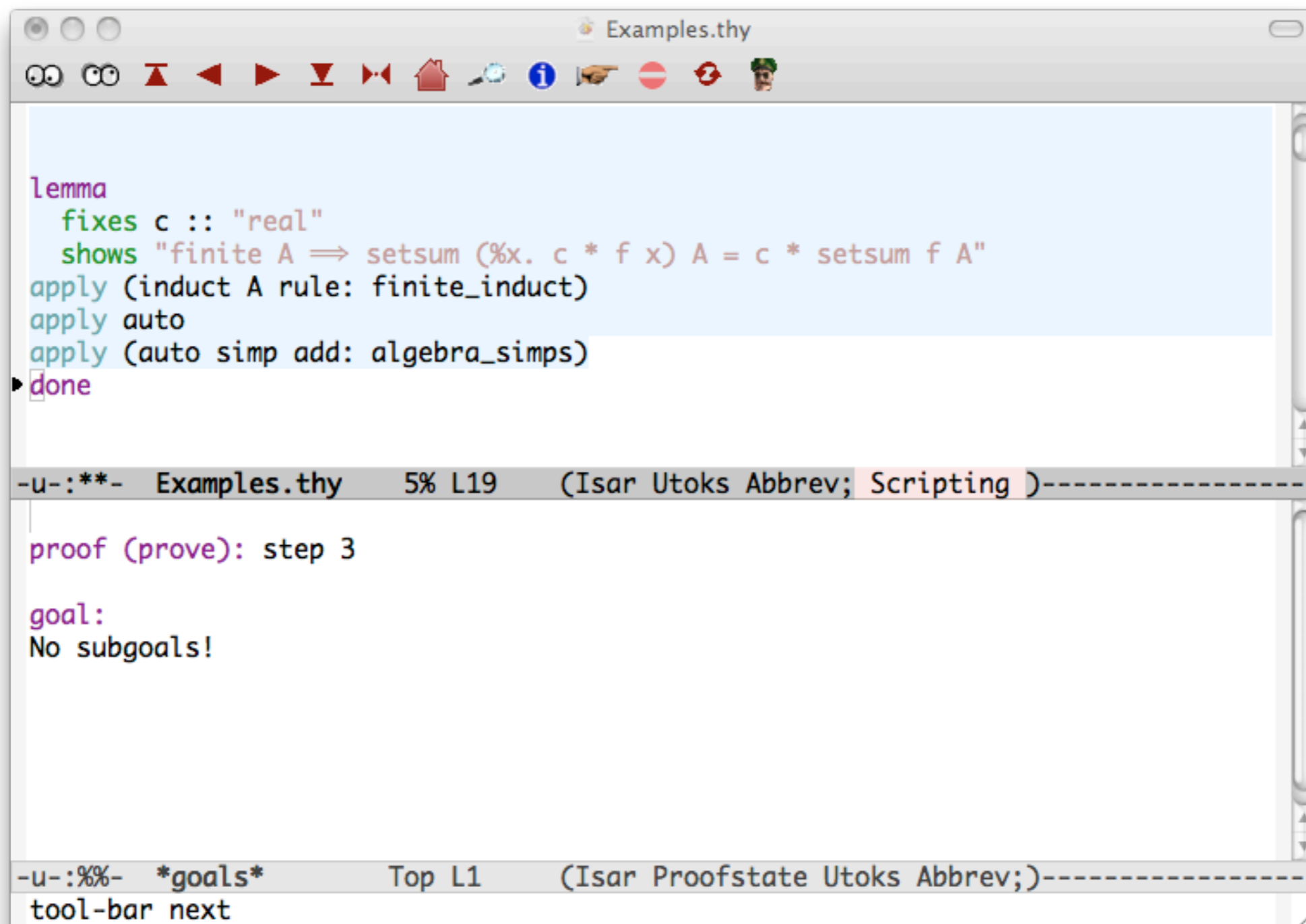
proof (prove): step 2

goal (1 subgoal):
1.  $\forall x \in F. [\text{finite } F; x \notin F; (\sum_{x \in F}. c * f x) = c * \text{setsum } f F]$ 
 $\implies c * f x + c * \text{setsum } f F = c * (f x + \text{setsum } f F)$ 

need to apply a distributive law

-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
tool-bar next
```

Finished!



The screenshot shows a window titled "Examples.thy" with a toolbar at the top. The main text area contains the following code:

```
lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
apply (induct A rule: finite_induct)
apply auto
apply (auto simp add: algebra_simps)
done
```

Below the code, a status bar indicates the current position: "-u-:***- Examples.thy 5% L19 (Isar Utoks Abbrev; Scripting)-----".

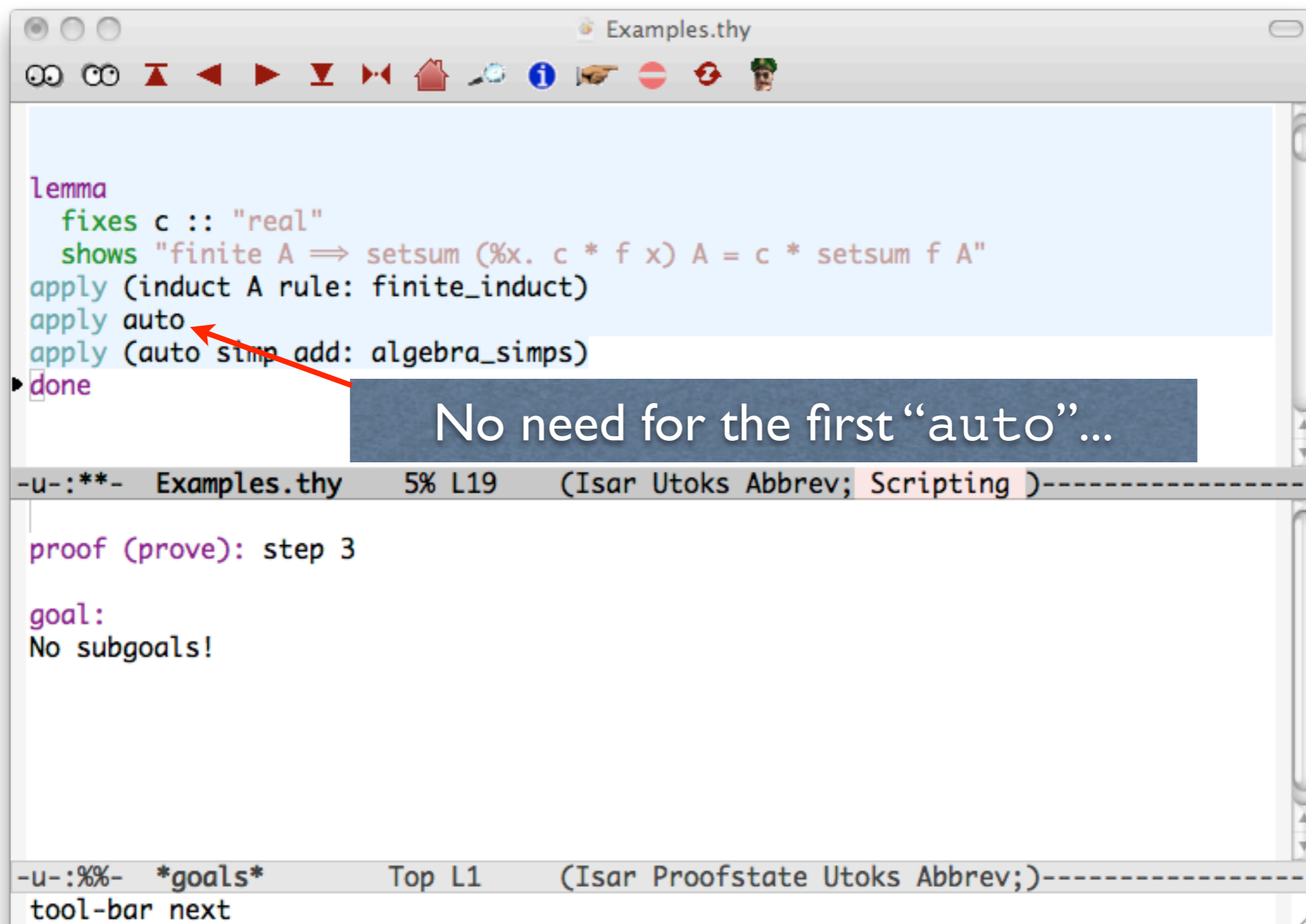
The next section of the interface shows the result of the proof:

```
proof (prove): step 3

goal:
No subgoals!
```

At the bottom, another status bar reads: "-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----" followed by "tool-bar next".

Finished!



The screenshot shows a window titled "Examples.thy" with a toolbar at the top. The main text area contains the following code:

```
lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
  apply (induct A rule: finite_induct)
  apply auto
  apply (auto simp add: algebra_simps)
  done
```

A red arrow points from the text "No need for the first 'auto'..." to the first `apply auto` line. Below the code, a status bar shows: `-u-:**- Examples.thy 5% L19 (Isar Utoks Abbrev; Scripting)`. The bottom section of the window shows the proof state:

```
proof (prove): step 3
goal:
No subgoals!
```

The bottom status bar shows: `-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)` and `tool-bar next`.

Proving Theorems about Sets

Proving Theorems about Sets

- It is not practical to learn all the built-in lemmas.

Proving Theorems about Sets

- It is not practical to learn all the built-in lemmas.
- Instead, try an automatic proof method:
 - `auto`
 - `force`
 - `blast`

Proving Theorems about Sets

- It is not practical to learn all the built-in lemmas.
- Instead, try an automatic proof method:
 - `auto`
 - `force`
 - `blast`
- Each uses the built-in library, comprising hundreds of facts, with powerful heuristics.

Finding Theorems about Sets

The screenshot shows a window titled "Examples.thy" with a toolbar containing navigation icons. The main text area contains the following Isabelle code:

```
lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
apply (induct A rule: finite_induct)
apply auto
apply (auto simp add: algebra_simps)
done
```

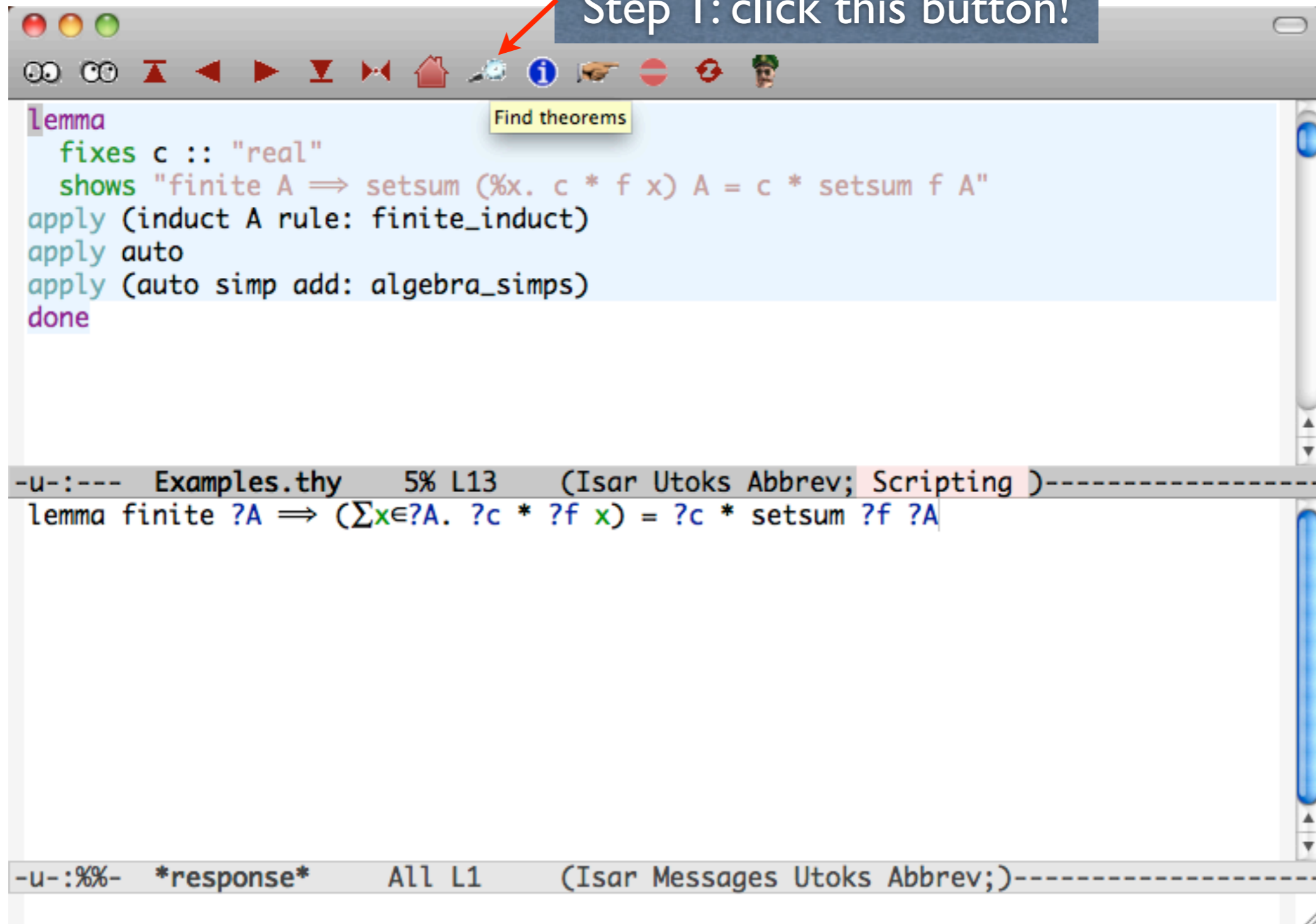
A yellow tooltip labeled "Find theorems" is positioned over the code. Below the code area, a status bar shows the file name "Examples.thy", line number "5% L13", and the text "(Isar Utoks Abbrev; Scripting)". Below this, the formal representation of the lemma is shown:

```
-u-:--- Examples.thy 5% L13 (Isar Utoks Abbrev; Scripting )-----
lemma finite ?A  $\implies$  ( $\sum_{x \in ?A} ?c * ?f x$ ) = ?c * setsum ?f ?A
```

At the bottom, another status bar shows "-u-:%%- *response* All L1 (Isar Messages Utoks Abbrev;)"

Finding Theorems about Sets

Step 1: click this button!



The screenshot shows a software interface for a theorem prover. At the top, there is a toolbar with various navigation icons. A red arrow points from a dark blue callout box containing the text "Step 1: click this button!" to a magnifying glass icon in the toolbar. Below the toolbar, a yellow tooltip with the text "Find theorems" is visible. The main area contains a code editor with the following text:

```
Lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
apply (induct A rule: finite_induct)
apply auto
apply (auto simp add: algebra_simps)
done
```

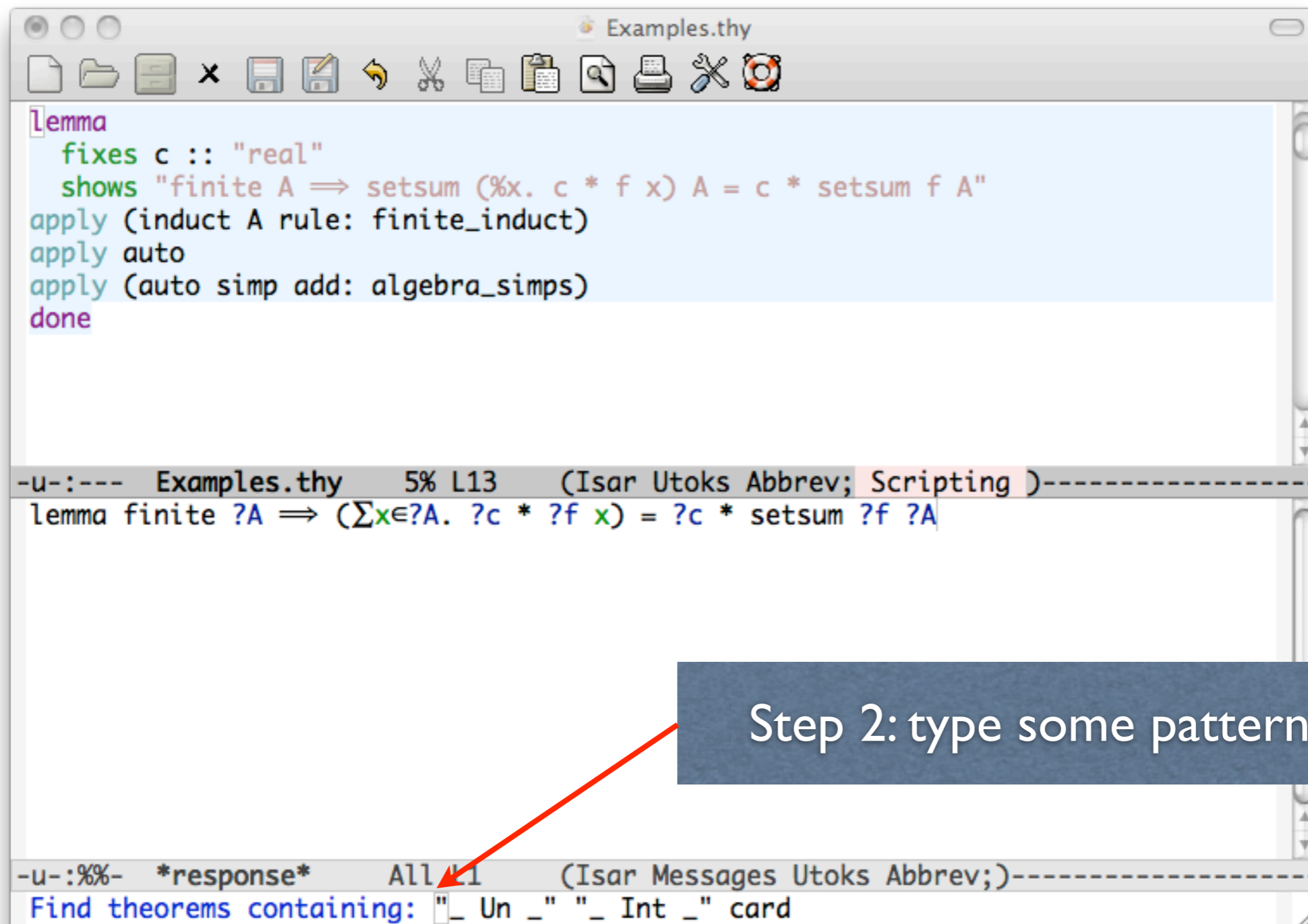
Below the code editor, there are two horizontal panels. The top panel shows the file path "Examples.thy" and line 5, with the text "(Isar Utoks Abbrev; Scripting)". The bottom panel shows the text "*response*" and line 1, with the text "(Isar Messages Utoks Abbrev;)".

-u-:--- Examples.thy 5% L13 (Isar Utoks Abbrev; Scripting)-----

lemma finite ?A \implies ($\sum_{x \in ?A} ?c * ?f x$) = ?c * setsum ?f ?A

-u-:%%- *response* All L1 (Isar Messages Utoks Abbrev;)-----

Finding Theorems about Sets



The screenshot shows a window titled "Examples.thy" with a toolbar at the top. The main text area contains the following code:

```
lemma
  fixes c :: "real"
  shows "finite A  $\implies$  setsum (%x. c * f x) A = c * setsum f A"
apply (induct A rule: finite_induct)
apply auto
apply (auto simp add: algebra_simps)
done
```

Below the code is a status bar with the text: "-u-:--- Examples.thy 5% L13 (Isar Utoks Abbrev; Scripting)-----".

Below the status bar is a search bar with the text: "lemma finite ?A \implies ($\sum_{x \in ?A} ?c * ?f x$) = ?c * setsum ?f ?A".

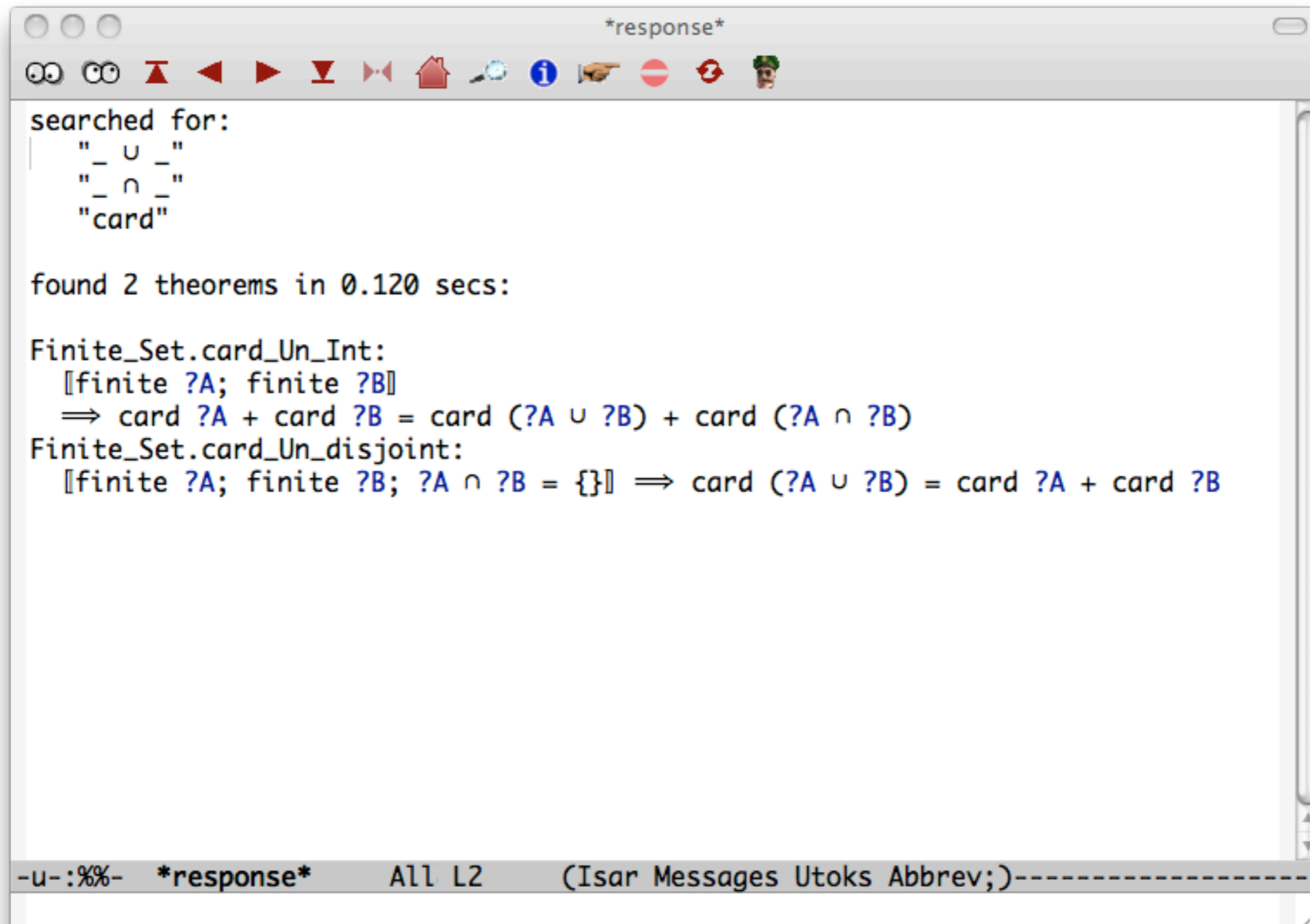
At the bottom of the window is another status bar with the text: "-u-:%%- *response* All 11 (Isar Messages Utoks Abbrev;)-----".

Below the bottom status bar is a search bar with the text: "Find theorems containing: "_ Un _" "_ Int _" card".

A red arrow points from a blue box containing the text "Step 2: type some patterns" to the search bar at the bottom of the window.

Step 2: type some patterns

What Theorems Were Found?



```
*response*
searched for:
  "_ u _"
  "_ n _"
  "card"

found 2 theorems in 0.120 secs:

Finite_Set.card_Un_Int:
  [[finite ?A; finite ?B]
   => card ?A + card ?B = card (?A ∪ ?B) + card (?A ∩ ?B)]
Finite_Set.card_Un_disjoint:
  [[finite ?A; finite ?B; ?A ∩ ?B = {}]] => card (?A ∪ ?B) = card ?A + card ?B

-u-:%%- *response* All L2 (Isar Messages Utoks Abbrev;)
```