

Interactive Formal Verification

7: Inductive Definitions

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

Defining a Set Inductively

Defining a Set Inductively

- The set of even numbers is the least set such that
 - 0 is even.
 - If n is even, then $n+2$ is even.

Defining a Set Inductively

- The set of even numbers is the least set such that
 - 0 is even.
 - If n is even, then $n+2$ is even.
- These can be viewed as *introduction rules*.

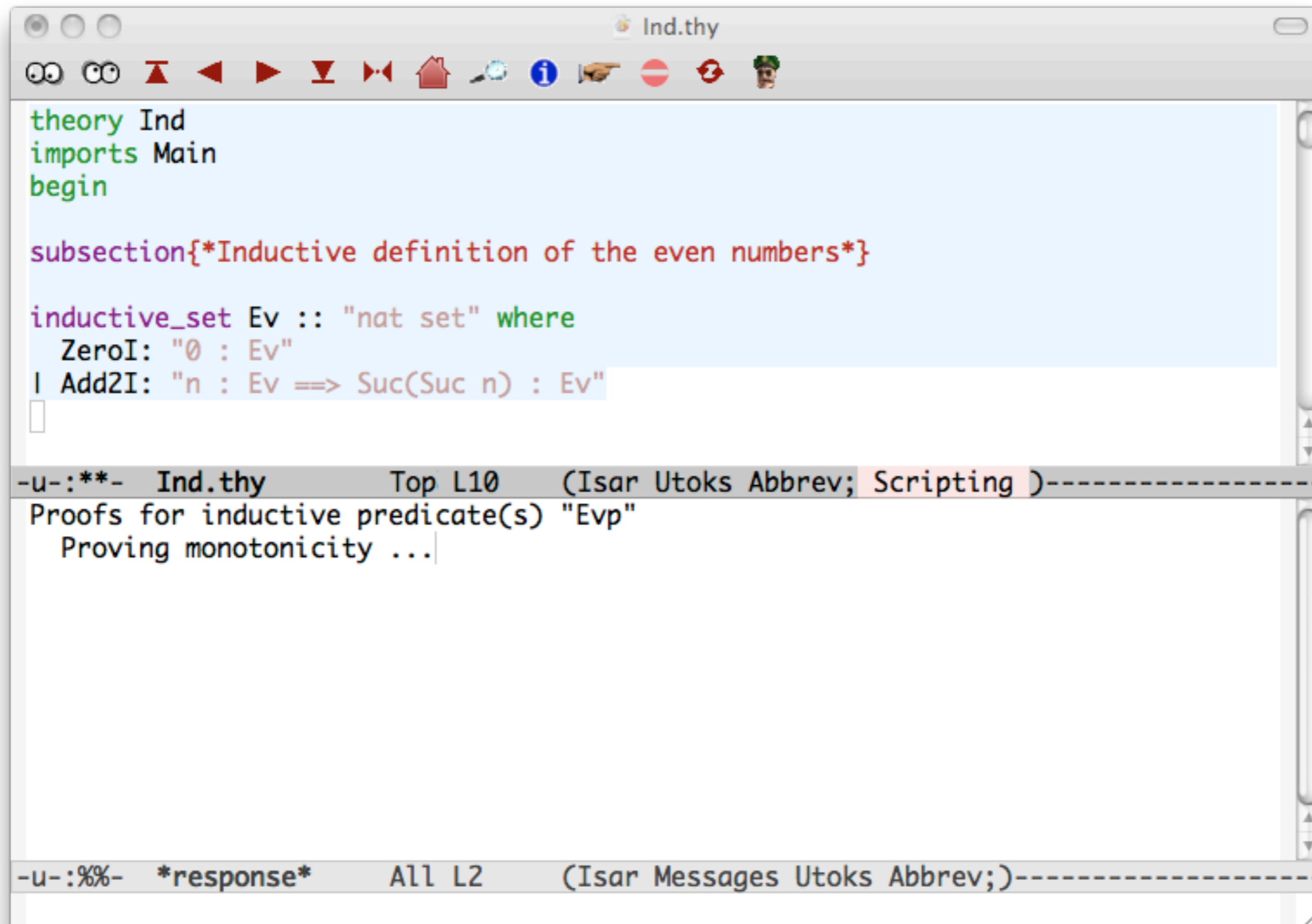
Defining a Set Inductively

- The set of even numbers is the least set such that
 - 0 is even.
 - If n is even, then $n+2$ is even.
- These can be viewed as *introduction rules*.
- We get an *induction principle* to express that no other numbers are even.

Defining a Set Inductively

- The set of even numbers is the least set such that
 - 0 is even.
 - If n is even, then $n+2$ is even.
- These can be viewed as *introduction rules*.
- We get an *induction principle* to express that no other numbers are even.
- Induction is used throughout mathematics, and to express the semantics of programming languages.

Inductive Definitions in Isabelle



```
theory Ind
imports Main
begin

subsection{*Inductive definition of the even numbers*}

inductive_set Ev :: "nat set" where
  ZeroI: "0 : Ev"
| Add2I: "n : Ev ==> Suc(Suc n) : Ev"

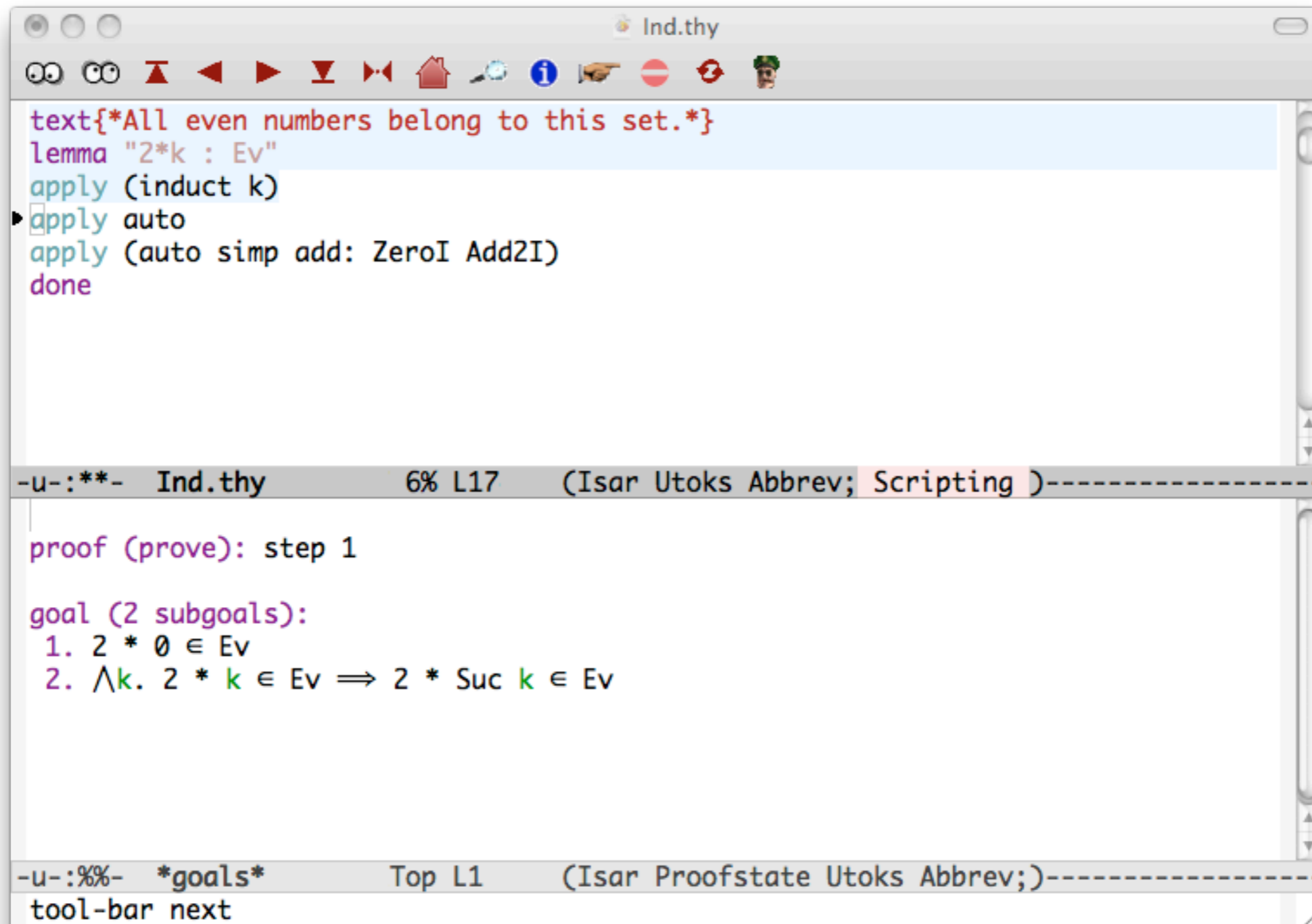
```

-u-:**- Ind.thy Top L10 (Isar Utoks Abbrev; Scripting)-----

Proofs for inductive predicate(s) "Ev"
Proving monotonicity ...|

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

Even Numbers Belong to Ev



The screenshot shows a proof assistant window titled "Ind.thy". The main editor contains the following code:

```
text{*All even numbers belong to this set.*}  
lemma "2*k : Ev"  
  apply (induct k)  
  apply auto  
  apply (auto simp add: ZeroI Add2I)  
  done
```

The status bar at the bottom of the editor shows: `-u-:***- Ind.thy 6% L17 (Isar Utoks Abbrev; Scripting)`. Below the editor, a goal state is displayed:

```
proof (prove): step 1  
goal (2 subgoals):  
1.  $2 * 0 \in \text{Ev}$   
2.  $\wedge k. 2 * k \in \text{Ev} \implies 2 * \text{Suc } k \in \text{Ev}$ 
```

The bottom status bar shows: `-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)`. At the very bottom, the text "tool-bar next" is visible.

Even Numbers Belong to Ev

The screenshot shows a proof assistant window titled "Ind.thy". The top toolbar contains various navigation icons. The main text area contains the following code:

```
text{*All even numbers belong to this set.*}  
lemma "2*k : Ev"  
  apply (induct k)  
  apply auto  
  apply (auto simp add: ZeroI Add2I)  
  done
```

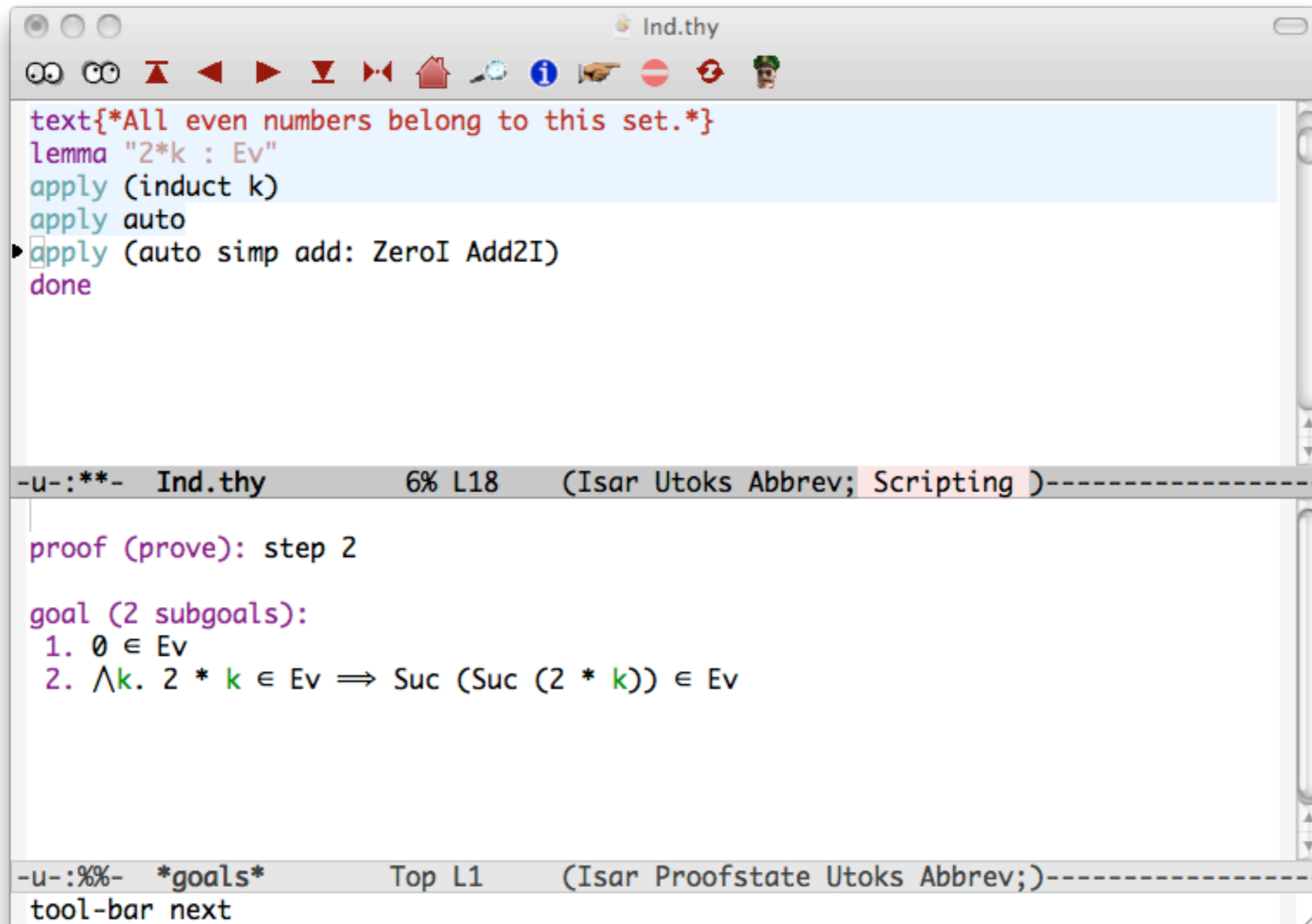
Below the code, the proof state is shown. The first line is "proof (prove): step 1". The second line is "goal (2 subgoals):". The goals are:

1. $2 * 0 \in \text{Ev}$
2. $\wedge k. 2 * k \in \text{Ev} \implies 2 * \text{Suc } k \in \text{Ev}$

Two red arrows point from a blue text box to the code and the goals. The text box contains the text "ordinary induction yields two subgoals".

At the bottom of the window, the status bar shows "-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----" and "tool-bar next".

Proving Set Membership



```
Ind.thy
text{*All even numbers belong to this set.*}
lemma "2*k : Ev"
apply (induct k)
apply auto
apply (auto simp add: ZeroI Add2I)
done

-u-:**- Ind.thy          6% L18  (Isar Utoks Abbrev; Scripting )-----
|
proof (prove): step 2

goal (2 subgoals):
1. 0 ∈ Ev
2.  $\wedge k. 2 * k \in \text{Ev} \implies \text{Suc} (\text{Suc} (2 * k)) \in \text{Ev}$ 

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

Proving Set Membership

The screenshot shows a theorem prover interface with a toolbar at the top and a main text area. The text area contains the following code:

```
text{*All even numbers belong to this set.*}  
lemma "2*k : Ev"  
  apply (induct k)  
  apply auto  
  apply (auto simp add: ZeroI Add2I)  
done
```

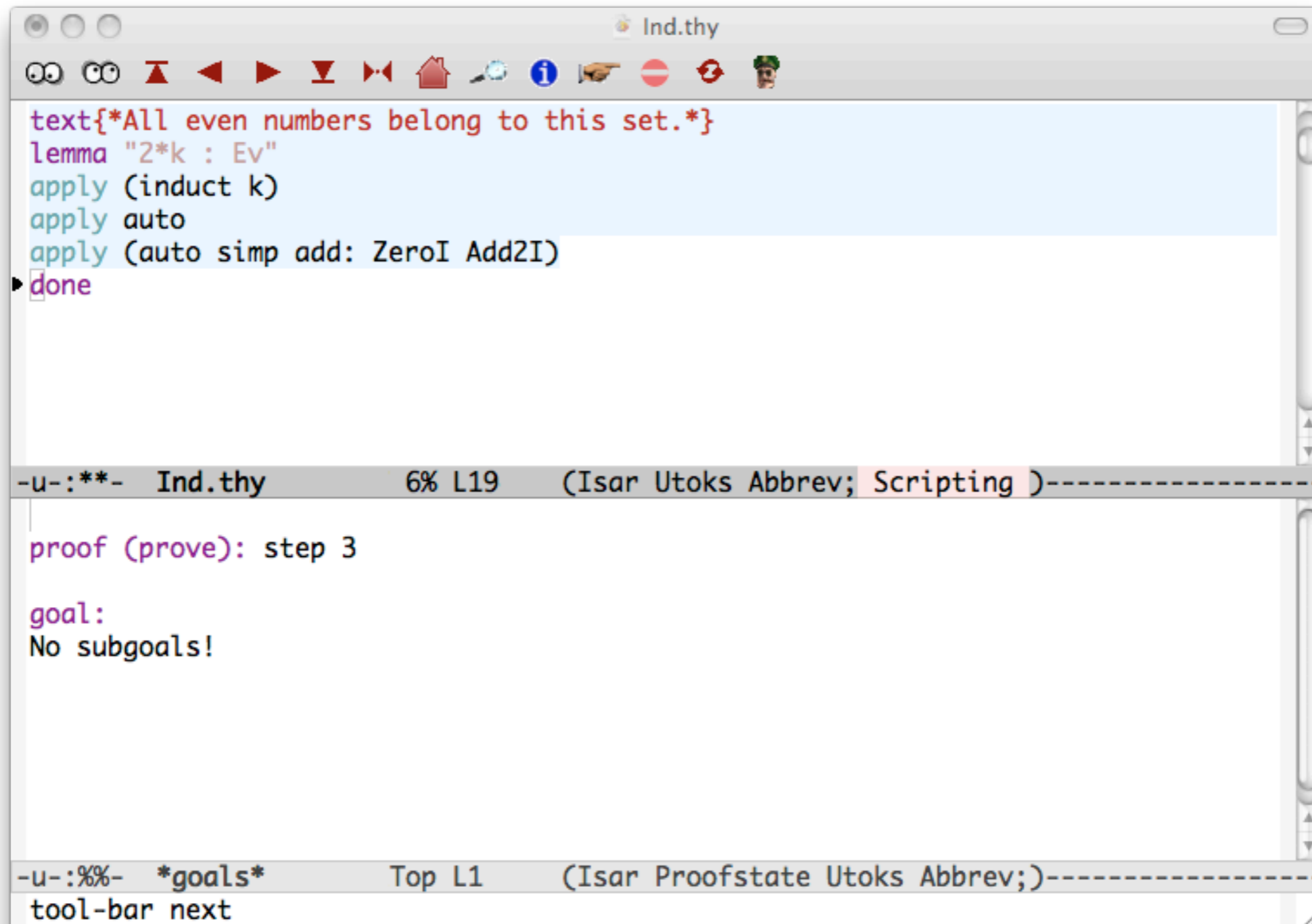
Below the code, a status bar shows the current file as "Ind.thy" at line 6, column 18. The main text area now displays the result of the proof script:

```
proof (prove): step 2  
goal (2 subgoals):  
1.  $0 \in \text{Ev}$   
2.  $\wedge k. 2 * k \in \text{Ev} \implies \text{Suc} (\text{Suc} (2 * k)) \in \text{Ev}$ 
```

At the bottom, another status bar shows the current state as "Top L1" and "(Isar Proofstate Utoks Abbrev;)" with a dashed line. The text "tool-bar next" is visible at the very bottom.

Two red arrows point from a dark blue callout box to the code and the goals. The callout box contains the text: "after simplification, the subgoals resemble the introduction rules".

Finishing the Proof



The screenshot shows a window titled "Ind.thy" with a toolbar and a text editor. The text editor contains the following code:

```
text{*All even numbers belong to this set.*}  
lemma "2*k : Ev"  
  apply (induct k)  
  apply auto  
  apply (auto simp add: ZeroI Add2I)  
done
```

Below the editor is a status bar with the text: "-u-:**- Ind.thy 6% L19 (Isar Utoks Abbrev; Scripting)-----".

Below the status bar is a text area showing the proof state:

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

At the bottom of the window is another status bar with the text: "-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----".

At the very bottom of the window is the text: "tool-bar next".

Finishing the Proof

```
text{*All even numbers belong to this set.*}
lemma "2*k : Ev"
apply (induct k)
apply auto
apply (auto simp add: ZeroI Add2I)
done
```

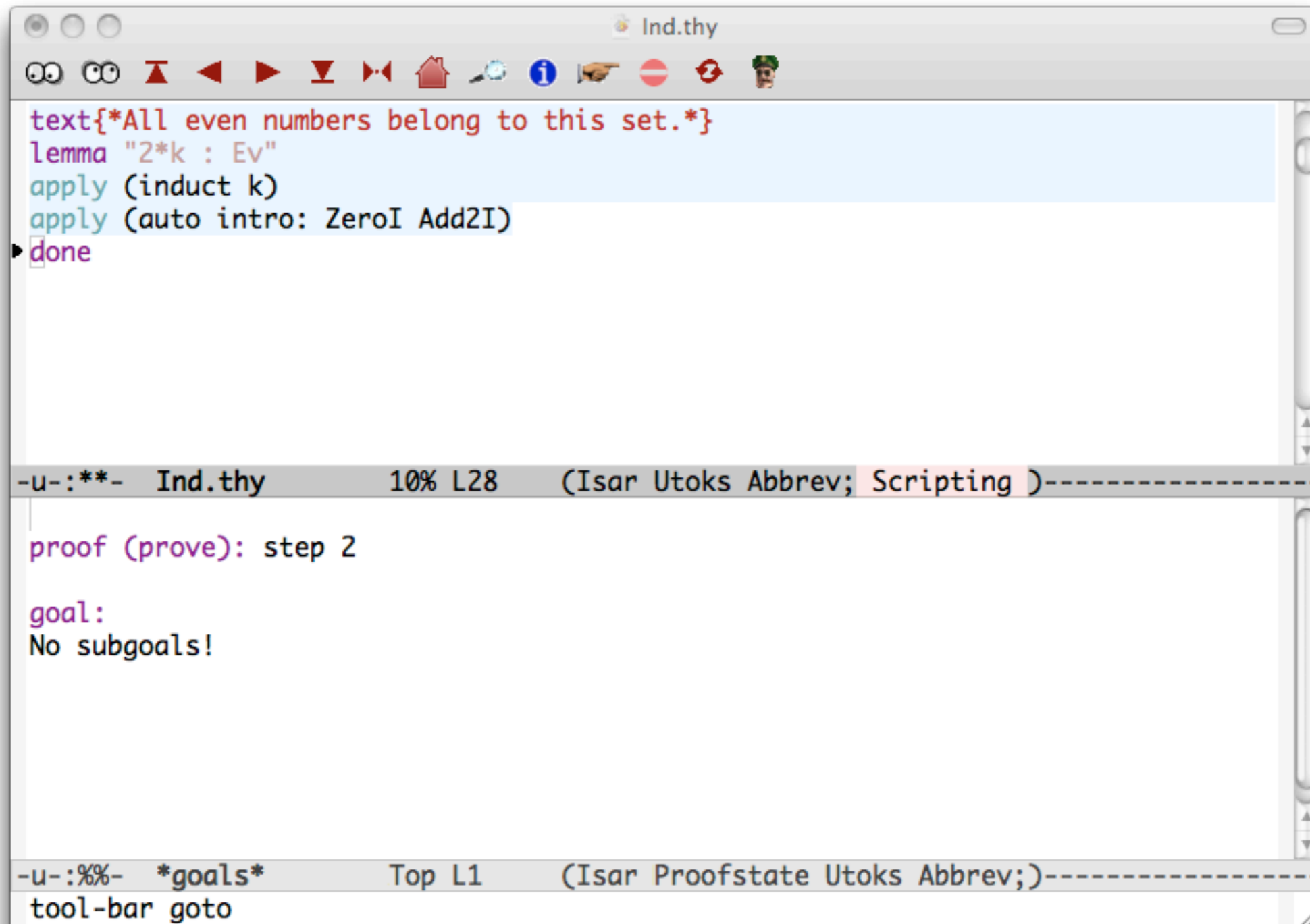
-u-:***- Ind.thy 6%

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

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

We have used these as
conditional rewrite rules.

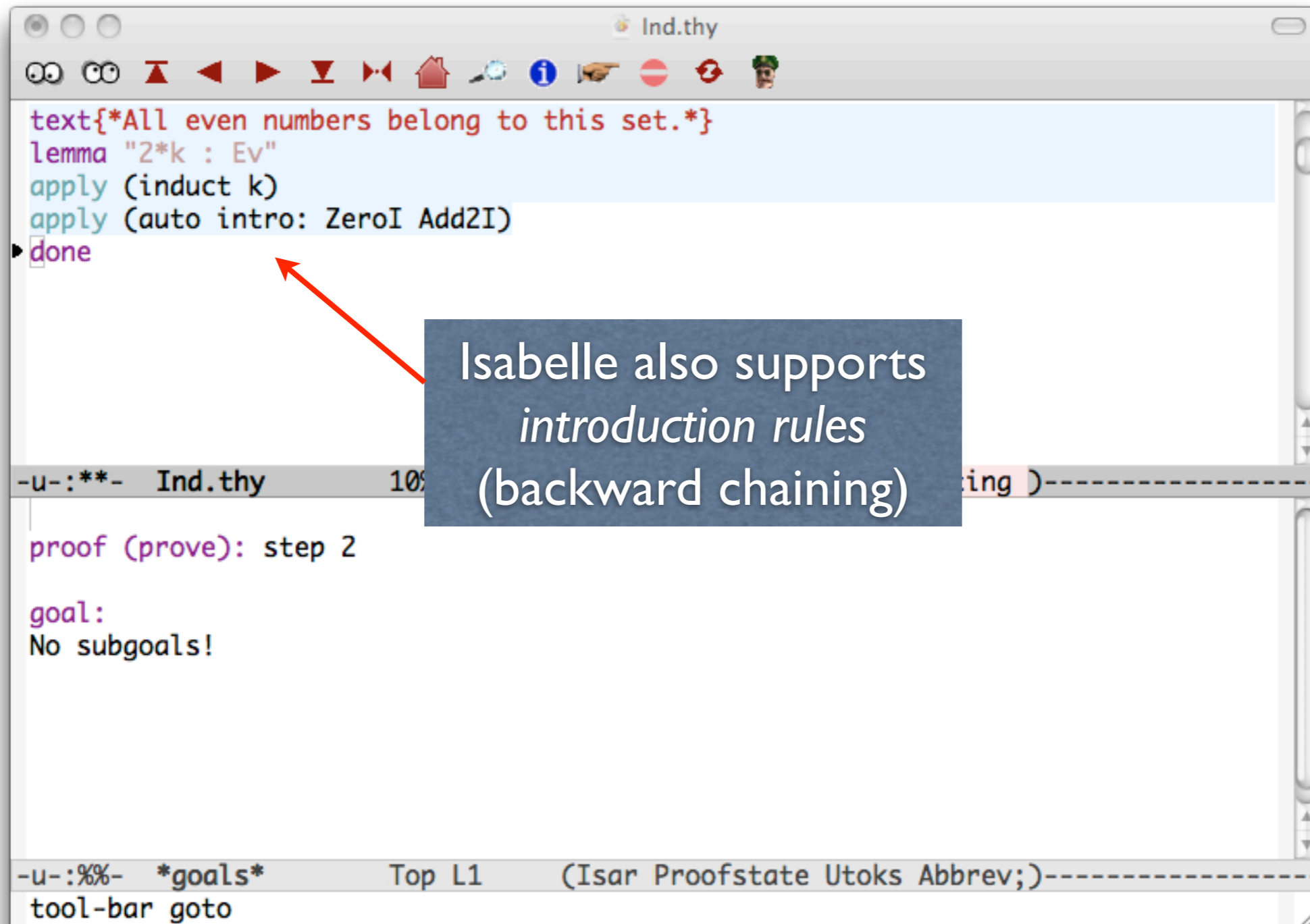
Finishing the Proof



```
Ind.thy
text{*All even numbers belong to this set.*}
lemma "2*k : Ev"
  apply (induct k)
  apply (auto intro: ZeroI Add2I)
  done

-u-:**- Ind.thy          10% L28  (Isar Utoks Abbrev; Scripting )-----
|
| proof (prove): step 2
|
| goal:
| No subgoals!
|
-u-:%%- *goals*          Top L1  (Isar Proofstate Utoks Abbrev;)-----
tool-bar goto
```

Finishing the Proof



```
Ind.thy
text{*All even numbers belong to this set.*}
lemma "2*k : Ev"
  apply (induct k)
  apply (auto intro: ZeroI Add2I)
  done

proof (prove): step 2
goal:
No subgoals!
```

The screenshot shows the Isabelle proof editor interface. The main window displays a proof script for a lemma. The script is as follows:

```
text{*All even numbers belong to this set.*}
lemma "2*k : Ev"
  apply (induct k)
  apply (auto intro: ZeroI Add2I)
  done
```

The word "done" is highlighted in purple, indicating the proof is complete. A red arrow points from a text box to the "done" keyword. The text box contains the text: "Isabelle also supports *introduction rules* (backward chaining)".

Below the main window, the status bar shows the current proof state: "proof (prove): step 2", "goal:", and "No subgoals!". At the bottom, the tool-bar shows "goto" and other navigation options.

Isabelle also supports *introduction rules* (backward chaining)

Rule Induction

Rule Induction

- Proving something about *every* element of the set.

Rule Induction

- Proving something about *every* element of the set.
- It expresses that the inductive set is *minimal*.

Rule Induction

- Proving something about *every* element of the set.
- It expresses that the inductive set is *minimal*.
- It is sometimes called “induction on derivations”

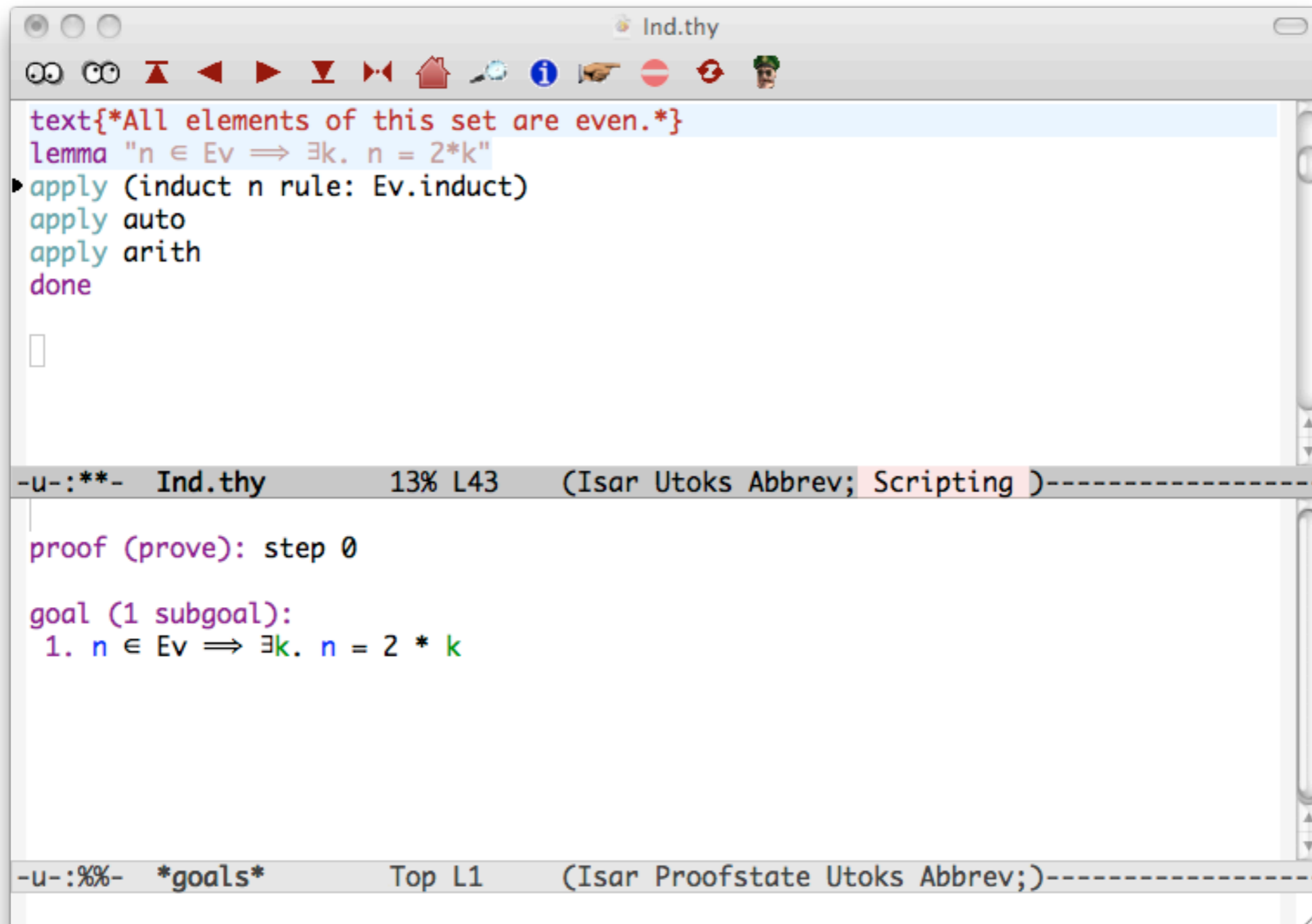
Rule Induction

- Proving something about *every* element of the set.
- It expresses that the inductive set is *minimal*.
- It is sometimes called “induction on derivations”
- There is a *base case* for every non-recursive introduction rule

Rule Induction

- Proving something about *every* element of the set.
- It expresses that the inductive set is *minimal*.
- It is sometimes called “induction on derivations”
- There is a *base case* for every non-recursive introduction rule
- ...and an *inductive step* for the other rules.

Ev Has only Even Numbers



The screenshot shows a window titled "Ind.thy" with a toolbar and a text editor. The text editor contains the following code:

```
text{*All elements of this set are even.*}
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"
▸ apply (induct n rule: Ev.induct)
  apply auto
  apply arith
  done

□
```

Below the editor is a status bar with the text: `-u-:**- Ind.thy 13% L43 (Isar Utoks Abbrev; Scripting)-----`

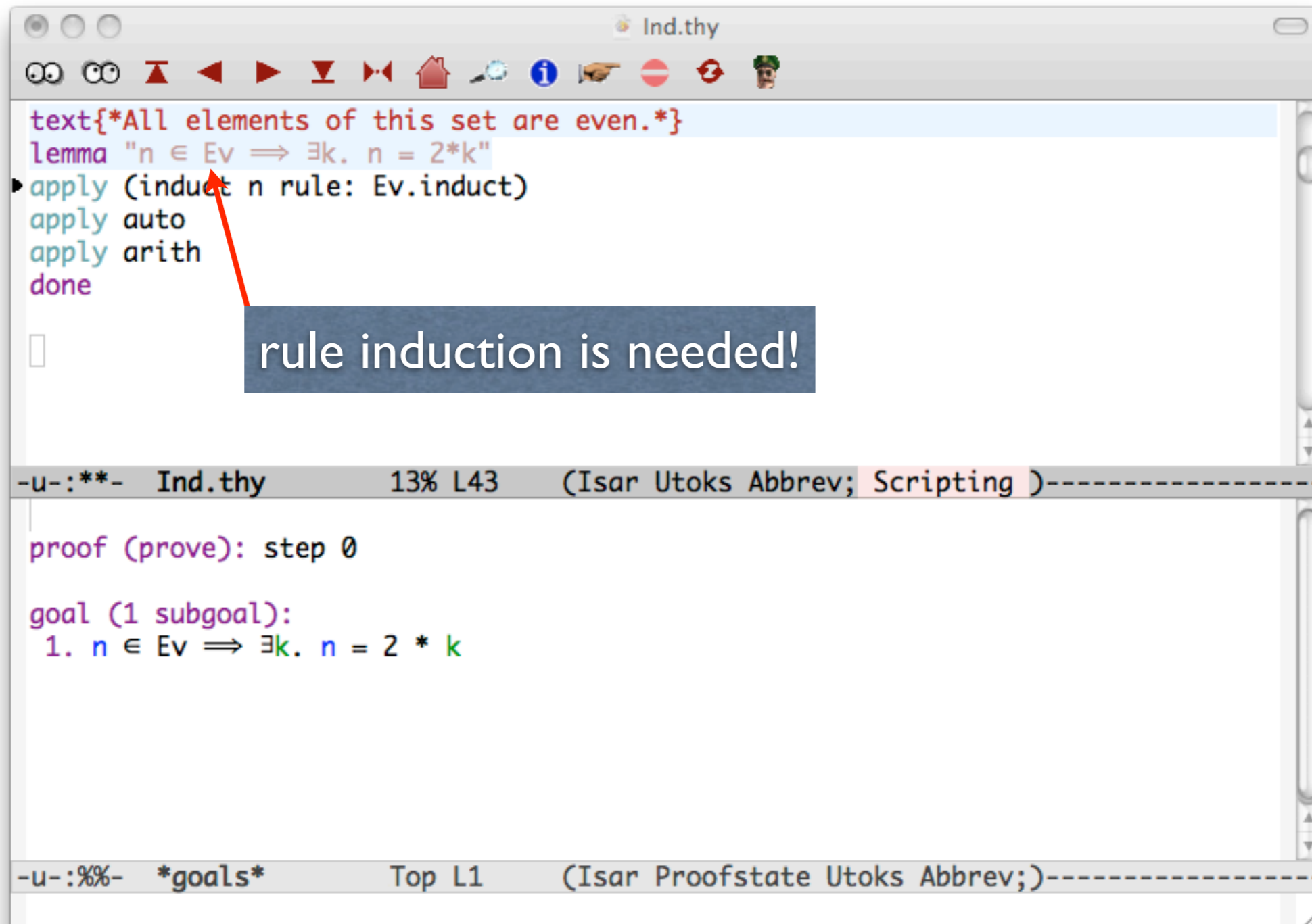
The next section shows the proof state:

```
proof (prove): step 0

goal (1 subgoal):
  1. n ∈ Ev ⇒ ∃k. n = 2 * k
```

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

Ev Has only Even Numbers



The screenshot shows a theorem prover interface with a window titled "Ind.thy". The main editor contains the following text:

```
text{*All elements of this set are even.*}  
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"  
• apply (induct n rule: Ev.induct)  
  apply auto  
  apply arith  
  done
```

A red arrow points from a dark blue callout box containing the text "rule induction is needed!" to the `induct` keyword in the proof script. Below the editor, a status bar shows the current position: "-u-:**- Ind.thy 13% L43 (Isar Utoks Abbrev; Scripting)". The bottom panel shows the current goal state:

```
proof (prove): step 0  
goal (1 subgoal):  
1. n ∈ Ev ⇒ ∃k. n = 2 * k
```

The bottom status bar shows: "-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)"

Ev Has only Even Numbers

```
Ind.thy
text{*All elements of this set are even.*}
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"
  apply (induct n rule: Ev.induct)
  apply auto
  apply arith
  done

□

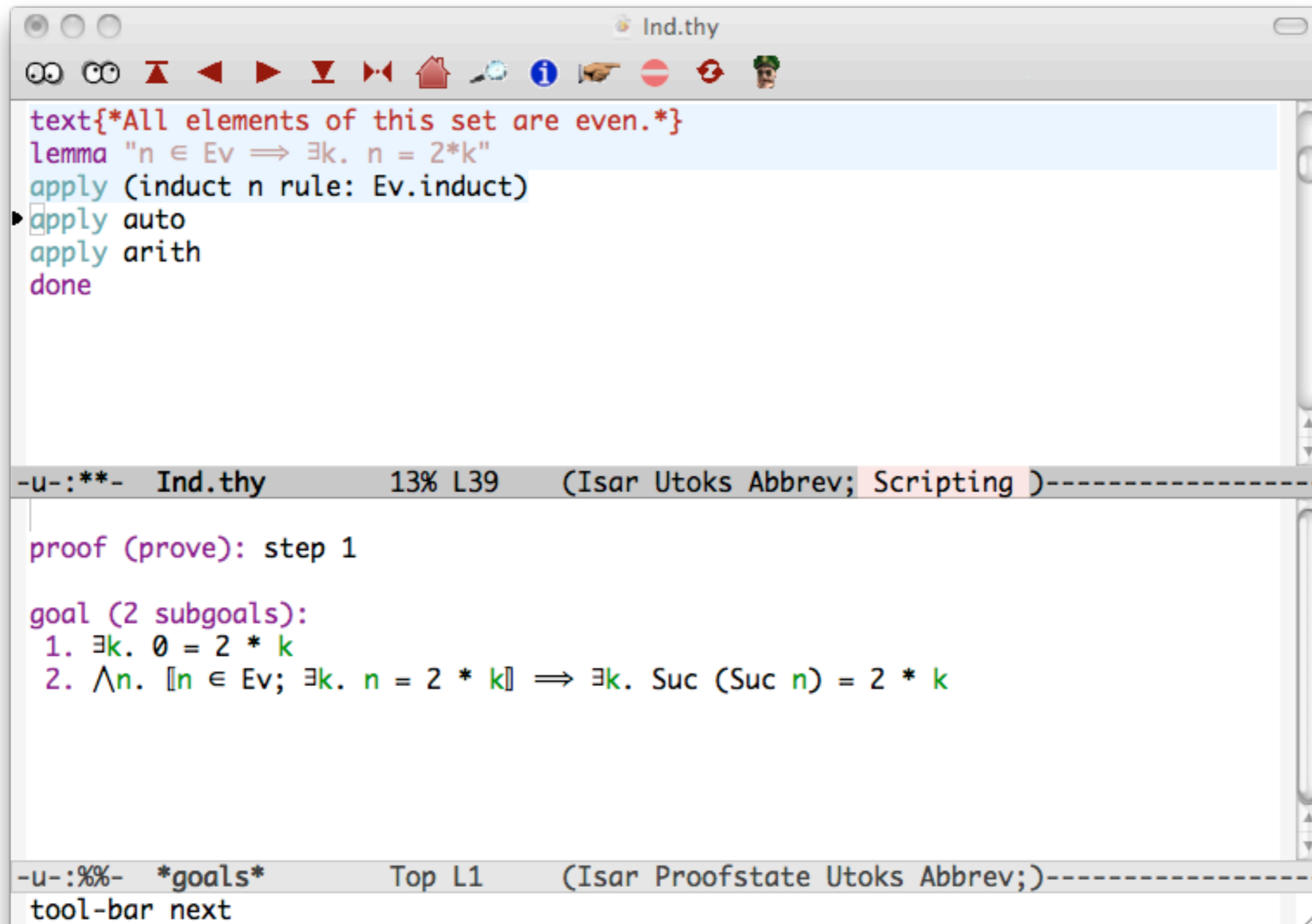
-u-:**- Ind.thy 13% L43 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 0
goal (1 subgoal):
1. n ∈ Ev ⇒ ∃k. n = 2 * k

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

Annotations:

- rule induction is needed!
- naming the induction rule

An Example of Rule Induction



```
Ind.thy
text{*All elements of this set are even.*}
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"
apply (induct n rule: Ev.induct)
apply auto
apply arith
done

-u-:**- Ind.thy 13% L39 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 1
goal (2 subgoals):
1. ∃k. 0 = 2 * k
2. ∧n. [n ∈ Ev; ∃k. n = 2 * k] ⇒ ∃k. Suc (Suc n) = 2 * k

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

An Example of Rule Induction

The screenshot shows a theorem prover interface with a window titled "Ind.thy". The main text area contains the following code:

```
text{*All elements of this set are even.*}  
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"  
apply (induct n rule: Ev.induct)  
apply auto  
apply arith  
done
```

The status bar below the text area shows: "-u-:**- Ind.thy 13% L39 (Isar Utoks Abbrev; Scripting)".

The next section of the interface shows a proof step:

```
proof (prove): step 1  
goal (2 subgoals):  
1. ∃k. 0 = 2 * k  
2. ∧n. [n ∈ Ev; ∃k. n = 2 * k] ⇒ ∃k. Suc (Suc n) = 2 * k
```

A red arrow points from the text "base case: n replaced by 0" to the first subgoal. The status bar below this section shows: "-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)".

At the bottom of the interface, there is a "tool-bar next" label.

An Example of Rule Induction

```
Ind.thy
text{*All elements of this set are even.*}
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"
apply (induct n rule: Ev.induct)
apply auto
apply arith
done

-u-:**- Ind.thy 13% L39 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 1 base case: n replaced by 0
goal (2 subgoals):
1. ∃k. 0 = 2 * k
2. ∧n. [n ∈ Ev; ∃k. n = 2 * k] ⇒ ∃k. Suc (Suc n) = 2 * k
induction step: n replaced by Suc (Suc n)
-u-:%%- fstate Utoks Abbrev;)-----
tool-bar next
```

Nearly There!

The screenshot shows a theorem prover interface with a window titled "Ind.thy". The main editor contains the following text:

```
text{*All elements of this set are even.*}  
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"  
apply (induct n rule: Ev.induct)  
apply auto  
▸ apply arith  
done
```

Below the editor is a status bar with the text: "-u-:**- Ind.thy 13% L40 (Isar Utoks Abbrev; Scripting)-----".

The bottom panel shows the current goal state:

```
proof (prove): step 2  
goal (1 subgoal):  
1.  $\wedge k. 2 * k \in \text{Ev} \Rightarrow \exists ka. \text{Suc} (\text{Suc} (2 * k)) = 2 * ka$ 
```

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

Nearly There!

```
Ind.thy
text{*All elements of this set are even.*}
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"
apply (induct n rule: Ev.induct)
apply auto
▸ apply arith
done

-u-:***- Ind.thy 13% L40 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 2
goal (1 subgoal):
1.  $\wedge k. 2 * k \in \text{Ev} \Rightarrow \exists ka. \text{Suc} (\text{Suc} (2 * k)) = 2 * ka$ 
Too difficult for auto
-u-:%%- *goals* Top L1 (Isar Proofstate Utoks Abbrev;)-----
tool-bar next
```

Nearly There!

```
Ind.thy
text{*All elements of this set are even.*}
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"
apply (induct n rule: Ev.induct)
apply auto
apply arith
done

-u-:***- Ind.thy 13% L40 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 2
goal (1 subgoal):
1.  $\wedge k. 2 * k \in \text{Ev} \Rightarrow \exists ka. \text{Suc} (\text{Suc} (2 * k)) = 2 * ka$ 
    ↑
    Too difficult for auto

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

The arith Proof Method

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

```
text{*All elements of this set are even.*}  
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"  
apply (induct n rule: Ev.induct)  
apply auto  
apply arith  
done
```

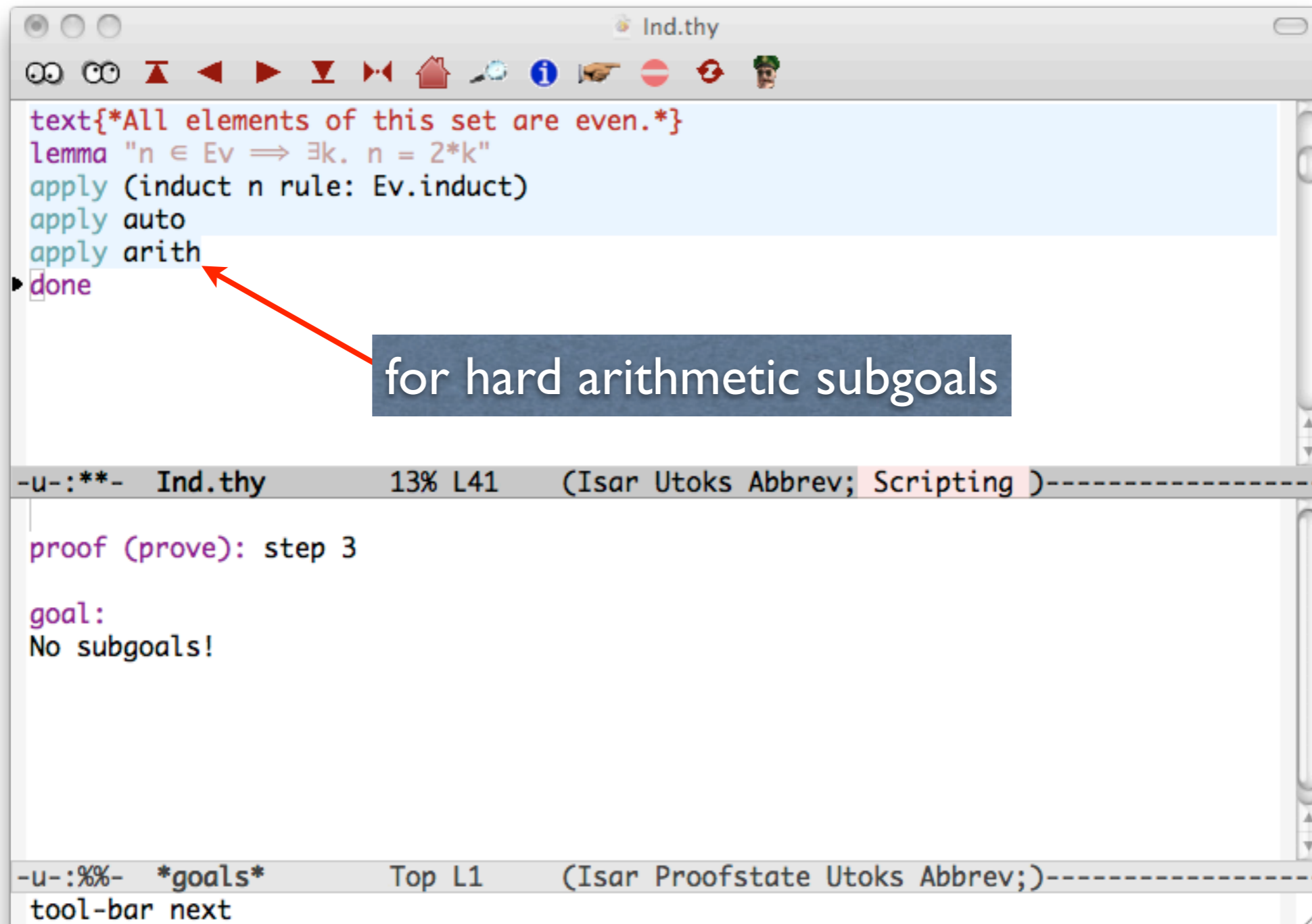
Below the code is a status bar with the text: "-u-:**- Ind.thy 13% L41 (Isar Utoks Abbrev; Scripting)-----".

The bottom section of the window shows the current proof state:

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

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

The arith Proof Method



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

```
text{*All elements of this set are even.*}  
lemma "n ∈ Ev ⇒ ∃k. n = 2*k"  
apply (induct n rule: Ev.induct)  
apply auto  
apply arith  
done
```

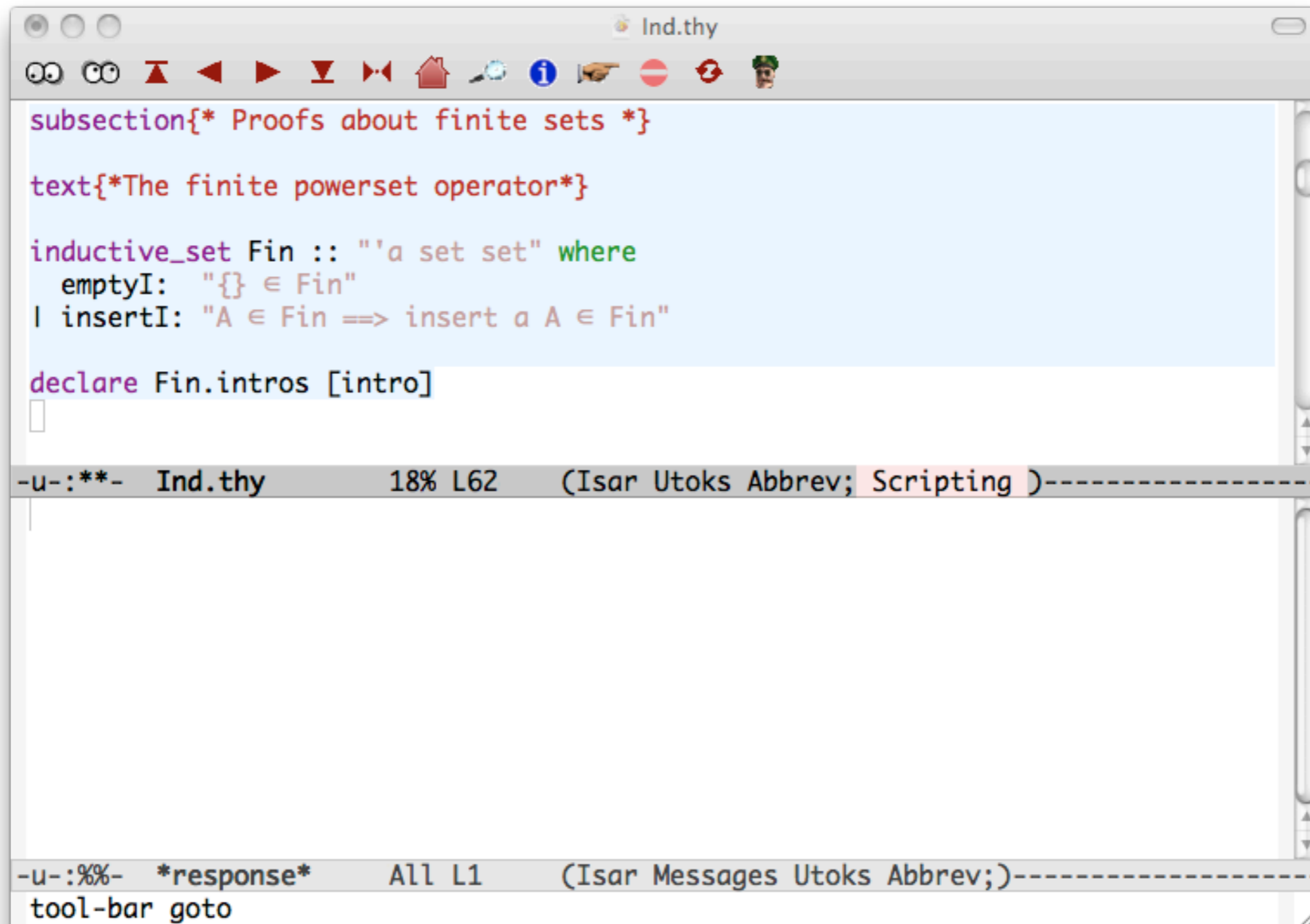
A red arrow points from a dark blue callout box containing the text "for hard arithmetic subgoals" to the `arith` command in the script. Below the code, a status bar shows the current position: "-u-:**- Ind.thy 13% L41 (Isar Utoks Abbrev; Scripting)".

The bottom panel shows the execution state:

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

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

Defining Finiteness



```
subsection{* Proofs about finite sets *}

text{*The finite powerset operator*}

inductive_set Fin :: "'a set set" where
  emptyI: "{} ∈ Fin"
| insertI: "A ∈ Fin ==> insert a A ∈ Fin"

declare Fin.intros [intro]

```

-u-:***- Ind.thy 18% L62 (Isar Utoks Abbrev; Scripting)-----

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

tool-bar goto

Defining Finiteness

```
subsection{* Proofs about finite sets *}

text{*The finite powerset operator*}

inductive_set Fin :: "'a set set" where
  emptyI: "{} ∈ Fin"
| insertI: "A ∈ Fin ==> insert a A ∈ Fin"

declare Fin.intros [intro]

```

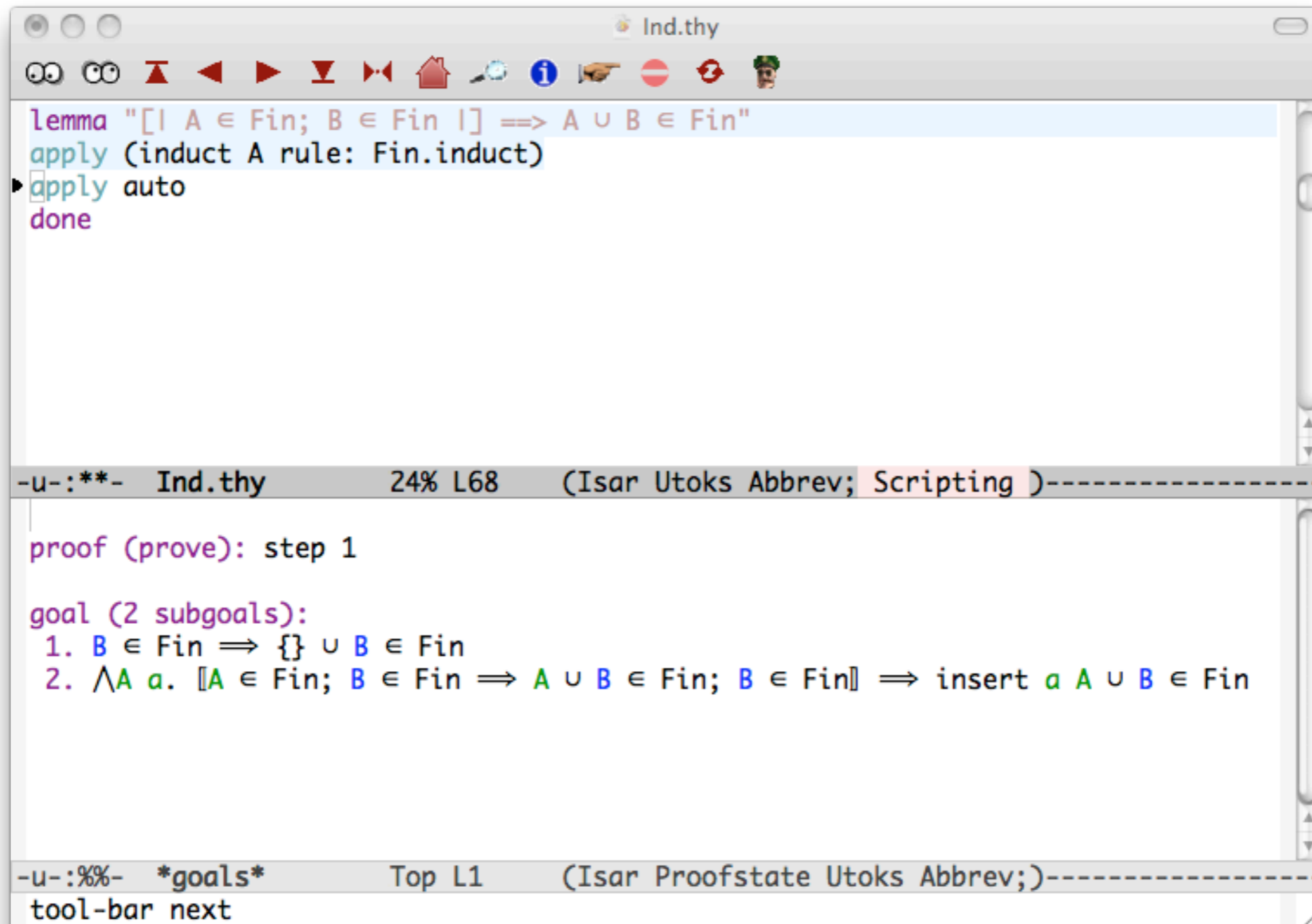
make the rules available to auto,blast

-u-:***- Ind.thy 18% L62

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

tool-bar goto

The Union of Two Finite Sets



```
Ind.thy
lemmas "[| A ∈ Fin; B ∈ Fin |] ==> A ∪ B ∈ Fin"
apply (induct A rule: Fin.induct)
apply auto
done

-u-:***- Ind.thy 24% L68 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 1
goal (2 subgoals):
1. B ∈ Fin ==> {} ∪ B ∈ Fin
2. ∧A a. [A ∈ Fin; B ∈ Fin ==> A ∪ B ∈ Fin; B ∈ Fin] ==> insert a A ∪ B ∈ Fin

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

The Union of Two Finite Sets

The screenshot shows a theorem prover interface with a toolbar at the top. The main text area contains the following code:

```
lemma "[| A ∈ Fin; B ∈ Fin |] ==> A ∪ B ∈ Fin"  
  apply (induct A rule: Fin.induct)  
  apply auto  
done
```

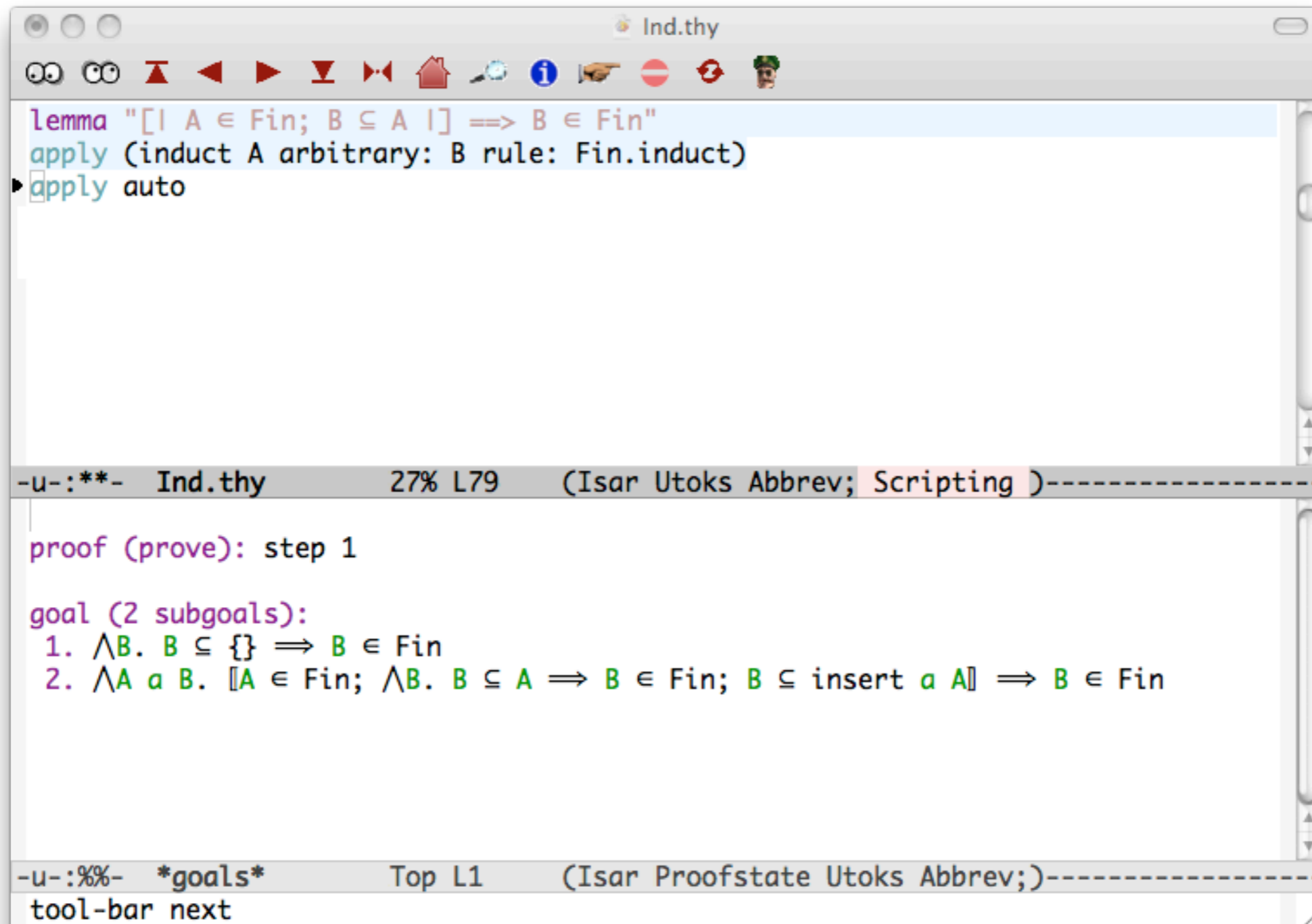
A red arrow points from a dark blue box containing the text "perform induction on A" to the `apply (induct A rule: Fin.induct)` line. Below the code, a status bar shows the file name "Ind.thy", progress "24% L68", and a list of loaded files including "(Isar Utoks Abbrev; Scripting)".

The bottom pane shows the current proof state:

```
proof (prove): step 1  
goal (2 subgoals):  
1. B ∈ Fin ==> {} ∪ B ∈ Fin  
2. ∧A a. [A ∈ Fin; B ∈ Fin ==> A ∪ B ∈ Fin; B ∈ Fin] ==> insert a A ∪ B ∈ Fin
```

The bottom status bar shows the file name "*goals*", progress "Top L1", and a list of loaded files including "(Isar Proofstate Utoks Abbrev;)", along with the text "tool-bar next".

A Subset of a Finite Set



```
Ind.thy
lemmas "[| A ∈ Fin; B ⊆ A |] ==> B ∈ Fin"
apply (induct A arbitrary: B rule: Fin.induct)
apply auto

-u-:***- Ind.thy 27% L79 (Isar Utoks Abbrev; Scripting )-----
proof (prove): step 1
goal (2 subgoals):
1.  $\wedge B. B \subseteq \{\}$   $\Rightarrow B \in \text{Fin}$ 
2.  $\wedge A a B. [A \in \text{Fin}; \wedge B. B \subseteq A \Rightarrow B \in \text{Fin}; B \subseteq \text{insert } a A] \Rightarrow B \in \text{Fin}$ 

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

A Subset of a Finite Set

The screenshot shows a theorem prover interface with a toolbar at the top. The main text area contains the following code:

```
lemma "[| A ∈ Fin; B ⊆ A |] ==> B ∈ Fin"  
apply (induct A arbitrary: B rule: Fin.induct)  
apply auto
```

A red arrow points from a blue callout box to the variable `B` in the lemma statement. The callout box contains the text: "to prove that every subset of A is finite".

The status bar at the bottom of the window shows: `-u-:***- Ind.thy 27% L79 (Isar Utoks Abbrev; Scripting)`

The proof state area shows:

```
proof (prove): step 1  
goal (2 subgoals):  
1.  $\wedge B. B \subseteq \{\}$   $\Rightarrow B \in \text{Fin}$   
2.  $\wedge A a B. [A \in \text{Fin}; \wedge B. B \subseteq A \Rightarrow B \in \text{Fin}; B \subseteq \text{insert } a A] \Rightarrow B \in \text{Fin}$ 
```

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

A Subset of a Finite Set

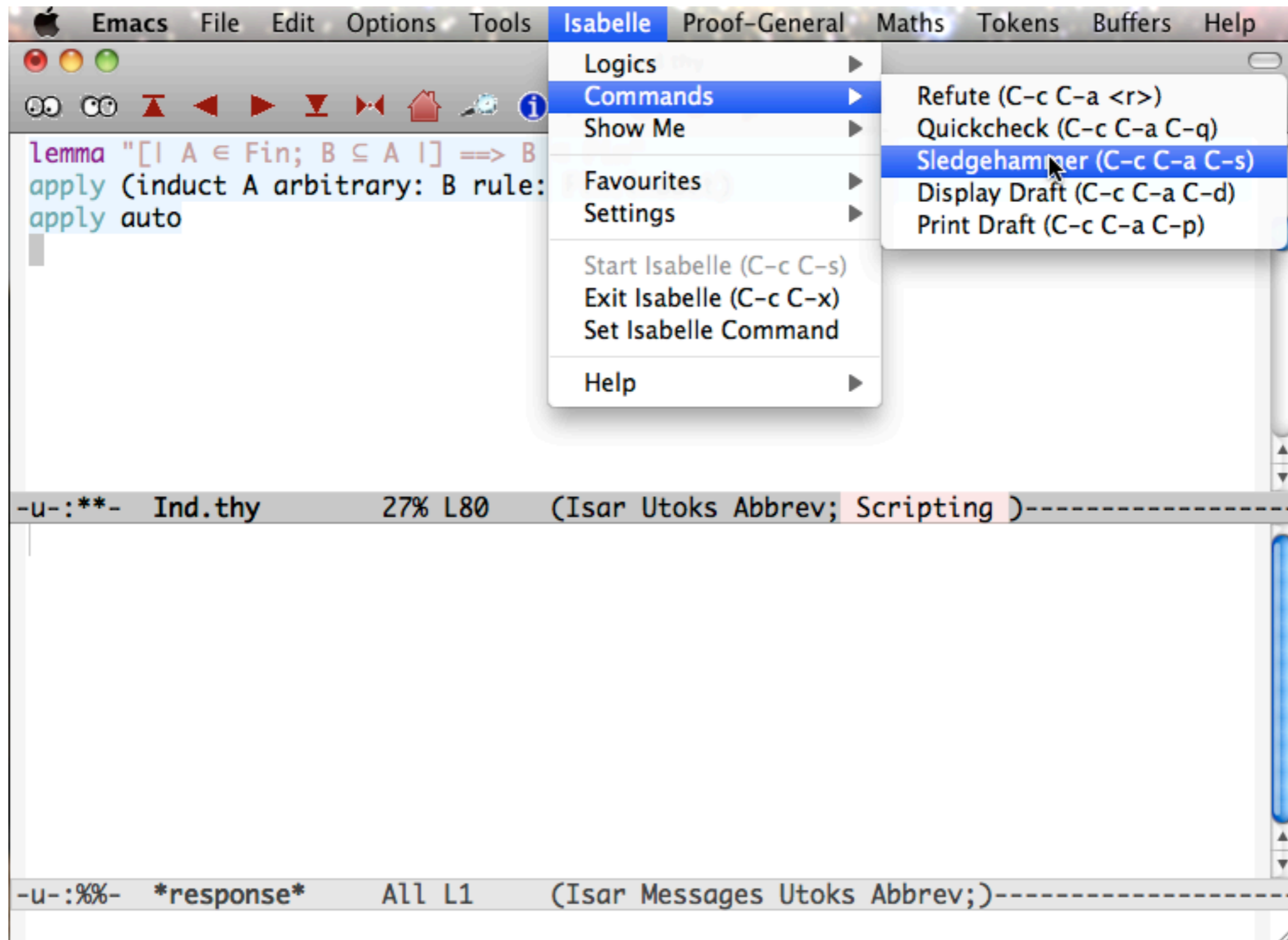
```
Ind.thy
lemma "[| A ∈ Fin; B ⊆ A |] ==> B ∈ Fin"
  apply (induct A arbitrary: B rule: Fin.induct)
  apply auto

proof (prove): step 1
goal (2 subgoals):
1.  $\wedge B. B \subseteq \{\}$   $\Rightarrow B \in \text{Fin}$ 
2.  $\wedge A a B. [A \in \text{Fin}; \wedge B. B \subseteq A \Rightarrow B \in \text{Fin}; B \subseteq \text{insert } a A] \Rightarrow B \in \text{Fin}$ 
tool-bar next
```

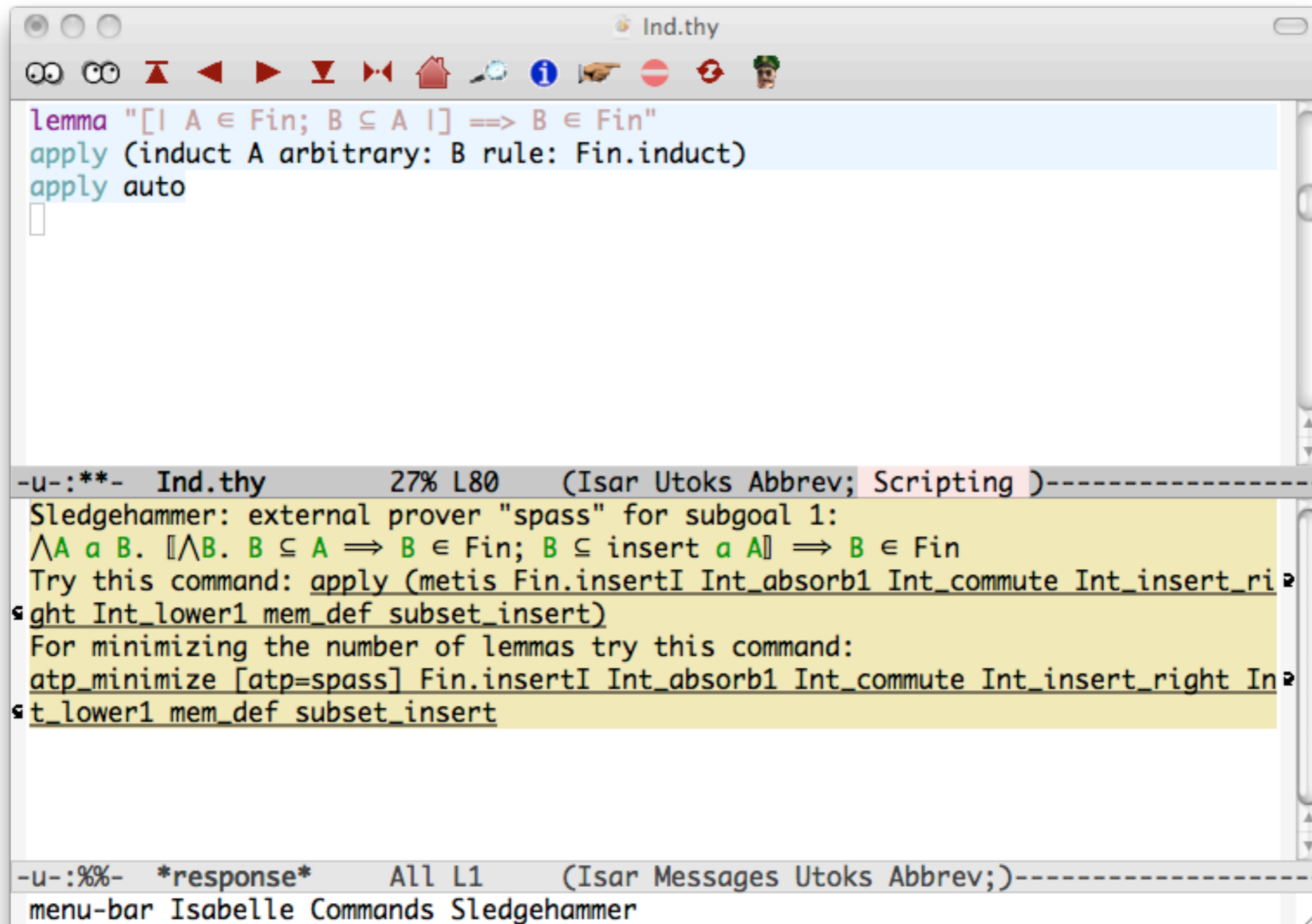
to prove that every subset of A is finite

as seen in the induction hypothesis

Time to Try Sledgehammer!



Success!



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

```
lemma "[| A ∈ Fin; B ⊆ A |] ==> B ∈ Fin"  
apply (induct A arbitrary: B rule: Fin.induct)  
apply auto  
□
```

Below the code is a status bar showing the current position: "-u-:**- Ind.thy 27% L80 (Isar Utoks Abbrev; Scripting)".

The bottom pane shows the Sledgehammer prover's response:

```
Sledgehammer: external prover "spass" for subgoal 1:  
∧A a B. [∧B. B ⊆ A ==> B ∈ Fin; B ⊆ insert a A] ==> B ∈ Fin  
Try this command: apply (metis Fin.insertI Int_absorb1 Int_commute Int_insert_right  
Int_lower1 mem_def subset_insert)  
For minimizing the number of lemmas try this command:  
atp_minimize [atp=spass] Fin.insertI Int_absorb1 Int_commute Int_insert_right Int_lower1 mem_def subset_insert
```

The bottom status bar shows: "-u-:%%- *response* All L1 (Isar Messages Utoks Abbrev;)" and the menu bar includes "Isabelle Commands Sledgehammer".

Success!

```
Ind.thy
lemma "[| A ∈ Fin; B ⊆ A |] ==> B ∈ Fin"
  apply (induct A arbitrary: B rule: Fin.induct)
  apply auto
  []

-u-:***- Ind.thy 27% L80 (Isar Utoks Abbrev)
Sledgehammer: external prover "spass" for subgoal 1

$$\bigwedge A a B. [\bigwedge B. B \subseteq A \implies B \in \text{Fin}; B \subseteq \text{insert } a A] \implies B \in \text{Fin}$$

Try this command: apply (metis Fin.insertI Int_absorb1 Int_commute Int_insert_right Int_lower1 mem_def subset_insert)
For minimizing the number of lemmas try this command:
atp_minimize [atp=spass] Fin.insertI Int_absorb1 Int_commute Int_insert_right Int_lower1 mem_def subset_insert

-u-:%%- *response* All L1 (Isar Messages Utoks Abbrev;)-----
menu-bar Isabelle Commands Sledgehammer
```

Success!

```
lemma "[| A ∈ Fin; B ⊆ A |] ==> B ∈ Fin"
apply (induct A arbitrary: B rule: Fin.induct)
apply auto
□
```

-u-:***- Ind.thy 27% L80 (Isar Utoks Abbrev)

Sledgehammer: external prover "spass" for subgoal 1

$\wedge A a B. [\wedge B. B \subseteq A \implies B \in \text{Fin}; B \subseteq \text{insert } a A] \implies B \in \text{Fin}$

Try this command: `apply (metis Fin.insertI Int_absorb1 Int_commute Int_insert_right Int_lower1 mem_def subset_insert)`

For minimizing the number of lemmas try this command:
`atp_minimize [atp=spass] Fin.insertI Int_absorb1 Int_commute Int_insert_right Int_lower1 mem_def subset_insert`

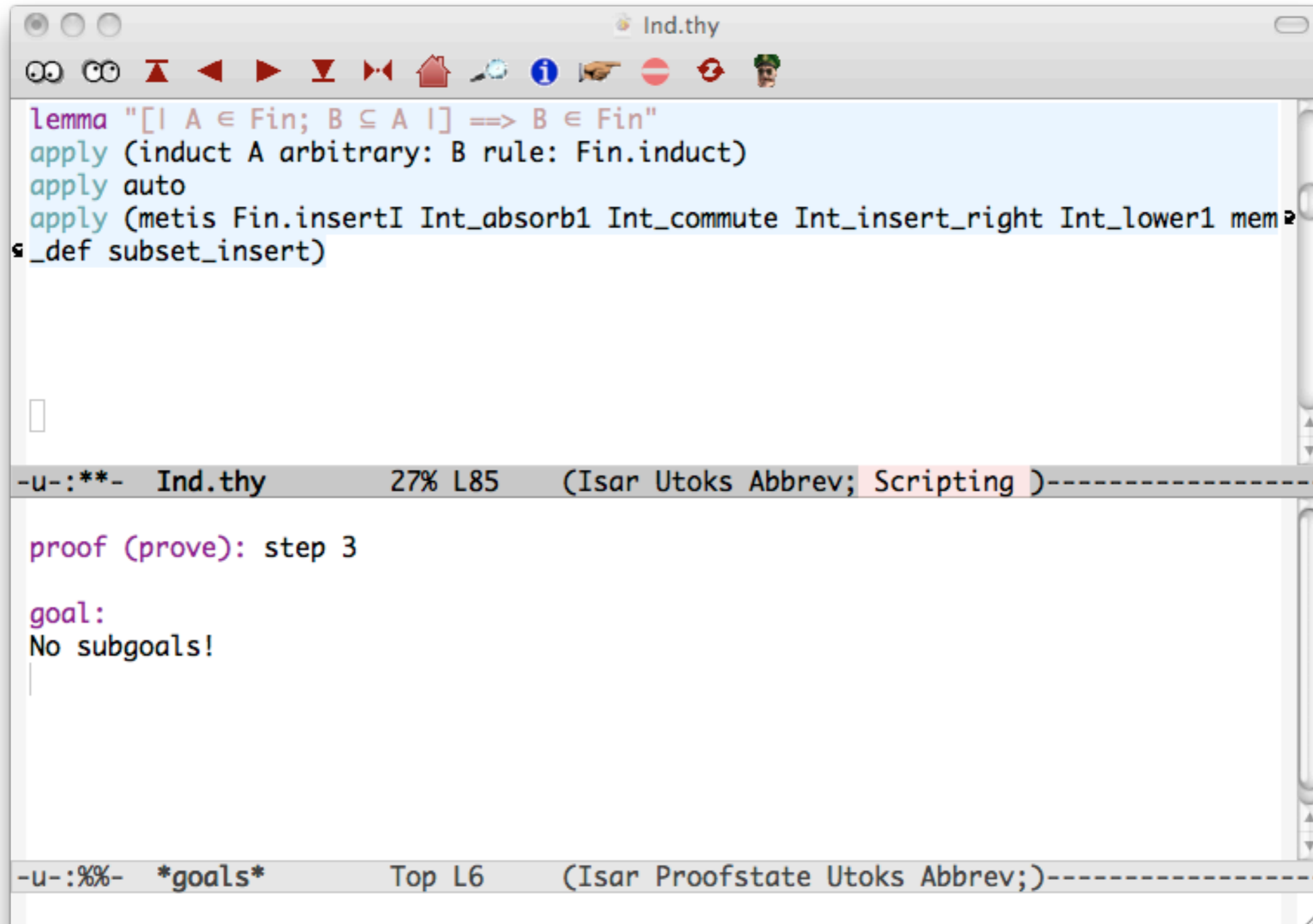
-u-:%%- *response* All L1 (Isar Messages Utoks Abbrev;)

menu-bar Isabelle Commands Sledgehammer

this command should prove the goal

this one may return a more compact command

The Completed Proof



The screenshot shows a proof assistant window titled "Ind.thy". The main editor contains the following code:

```
lemma "[| A ∈ Fin; B ⊆ A |] ==> B ∈ Fin"  
  apply (induct A arbitrary: B rule: Fin.induct)  
  apply auto  
  apply (metis Fin.insertI Int_absorb1 Int_commute Int_insert_right Int_lower1 mem  
_def subset_insert)
```

Below the code is a status bar with the text: "-u-:***- Ind.thy 27% L85 (Isar Utoks Abbrev; Scripting)-----".

The bottom panel shows the proof state:

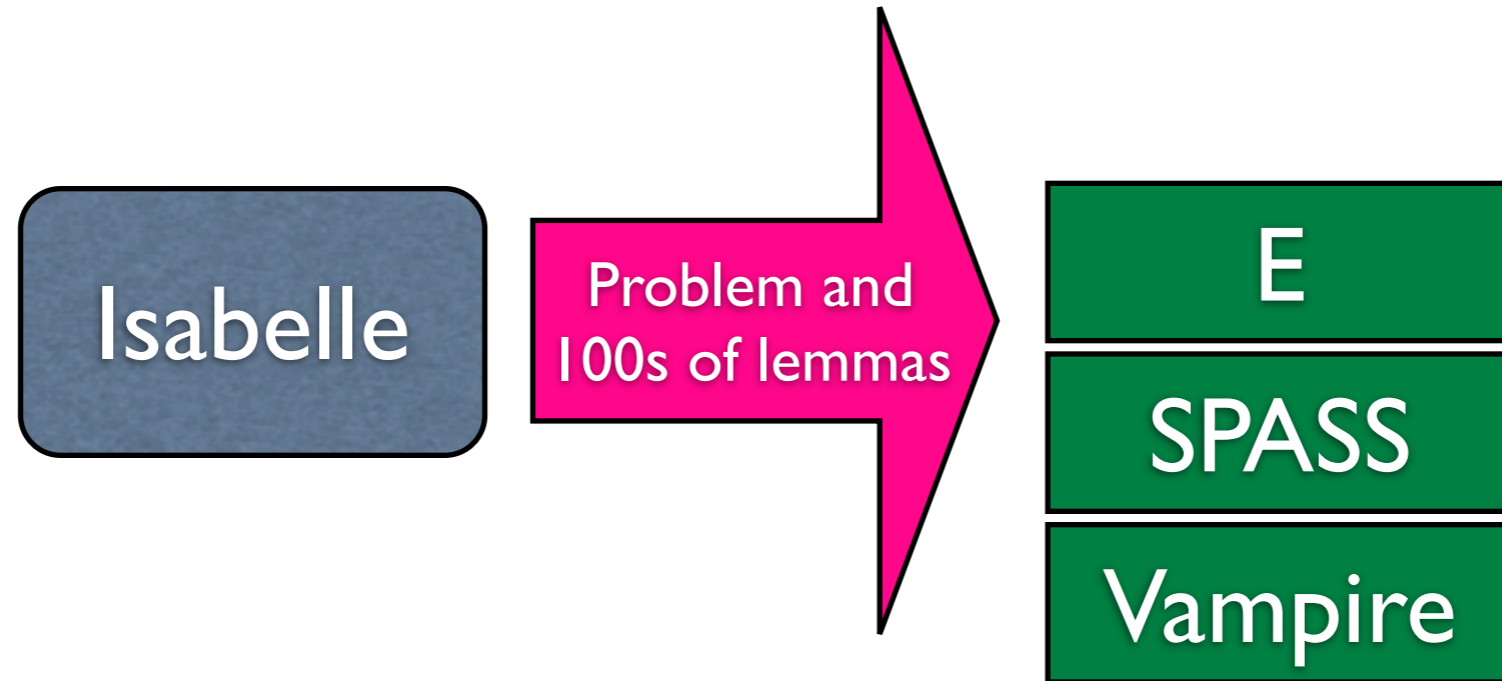
```
proof (prove): step 3  
goal:  
No subgoals!  
|
```

Below the proof state is another status bar with the text: "-u-:%%- *goals* Top L6 (Isar Proofstate Utoks Abbrev;)-----".

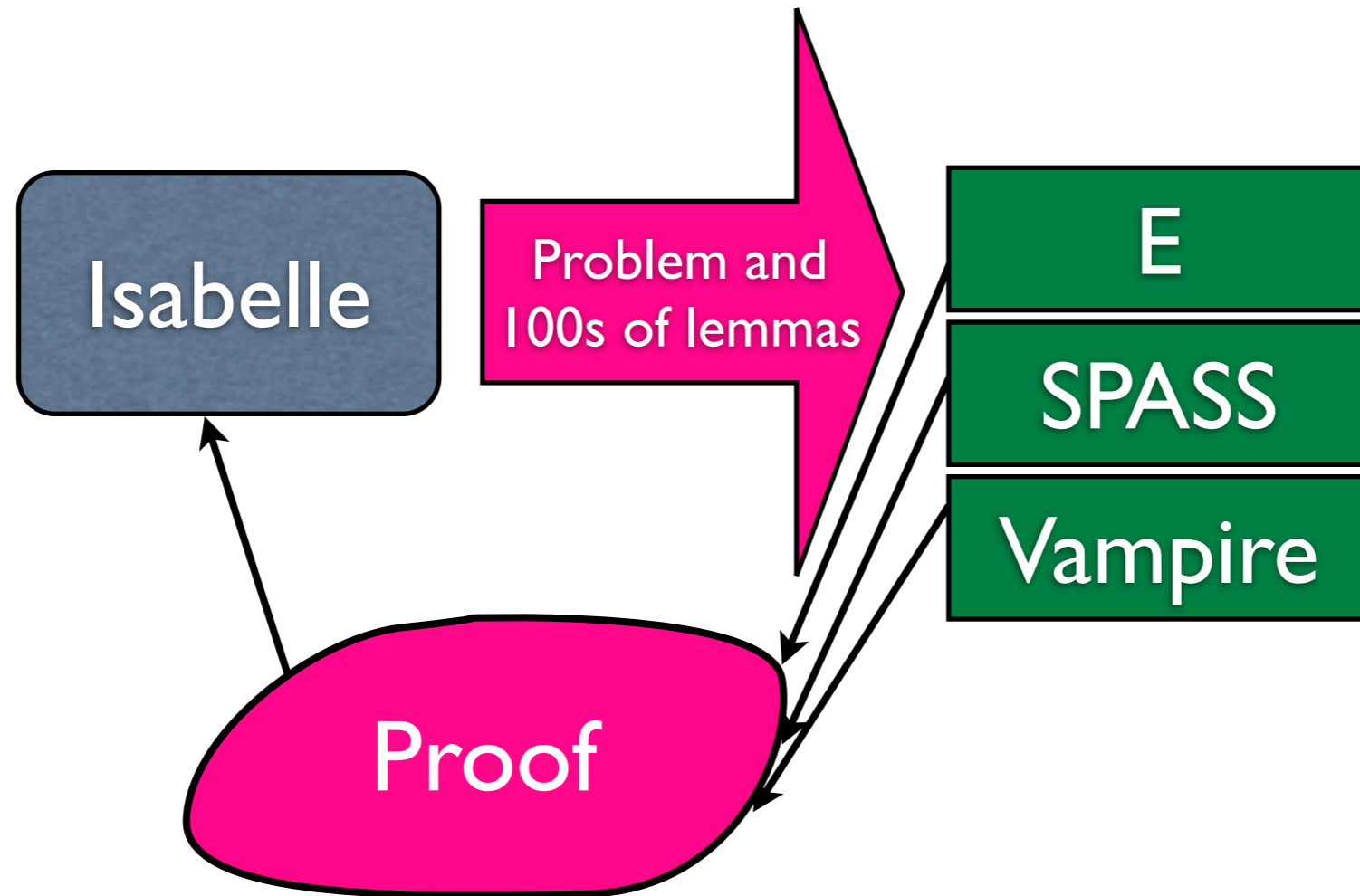
How Sledgehammer Works

Isabelle

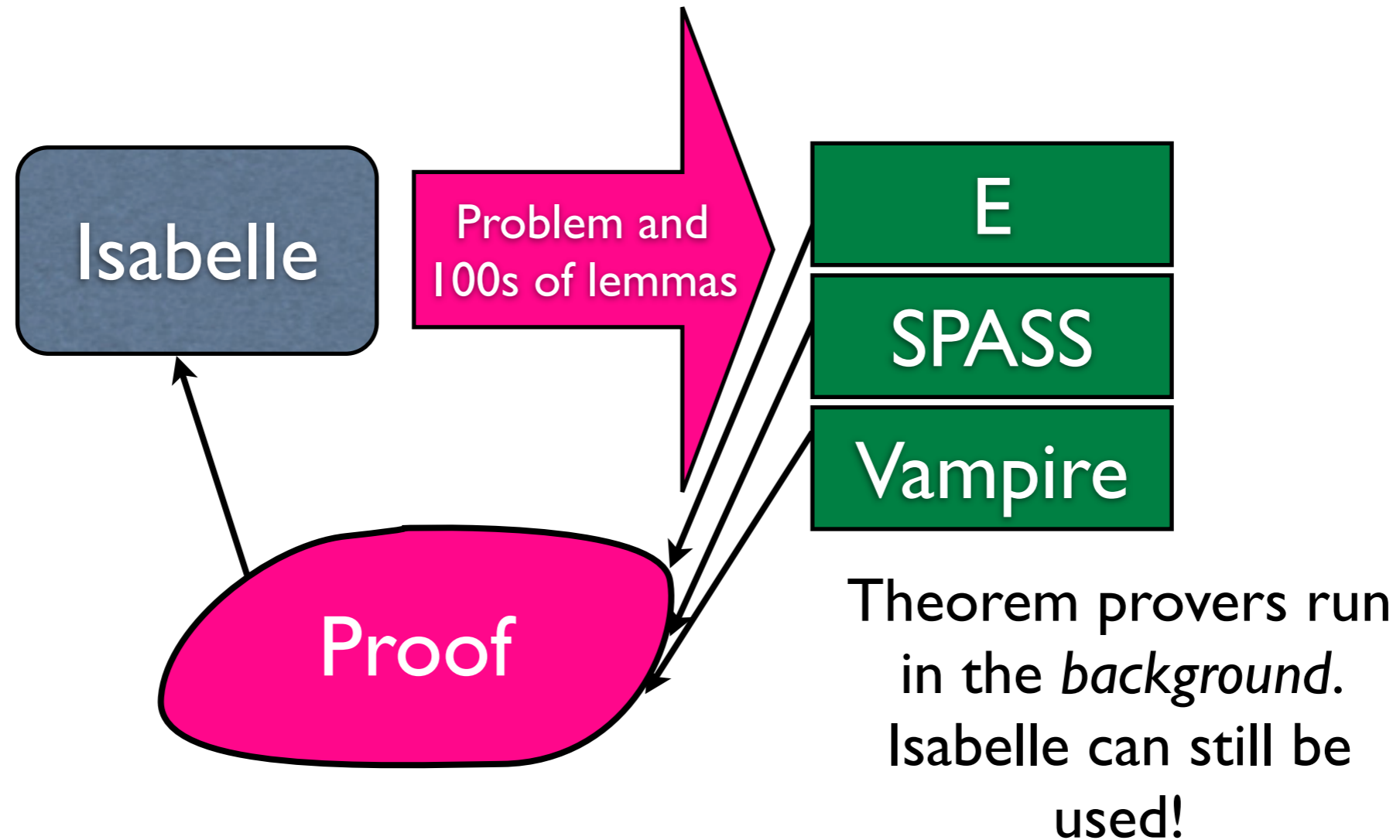
How Sledgehammer Works



How Sledgehammer Works



How Sledgehammer Works



Notes on Sledgehammer

Notes on Sledgehammer

- It is always available, though it usually fails...

Notes on Sledgehammer

- It is always available, though it usually fails...
- It does not prove the goal, but returns a call to `metis`. This command *usually* works...

Notes on Sledgehammer

- It is always available, though it usually fails...
- It does not prove the goal, but returns a call to `metis`. This command *usually* works...
- The minimise option removes redundant theorems, increasing the likelihood of success.

Notes on Sledgehammer

- It is always available, though it usually fails...
- It does not prove the goal, but returns a call to `metis`. This command *usually* works...
- The `minimise` option removes redundant theorems, increasing the likelihood of success.
- Calling `metis` directly is difficult unless you know exactly which lemmas are needed.