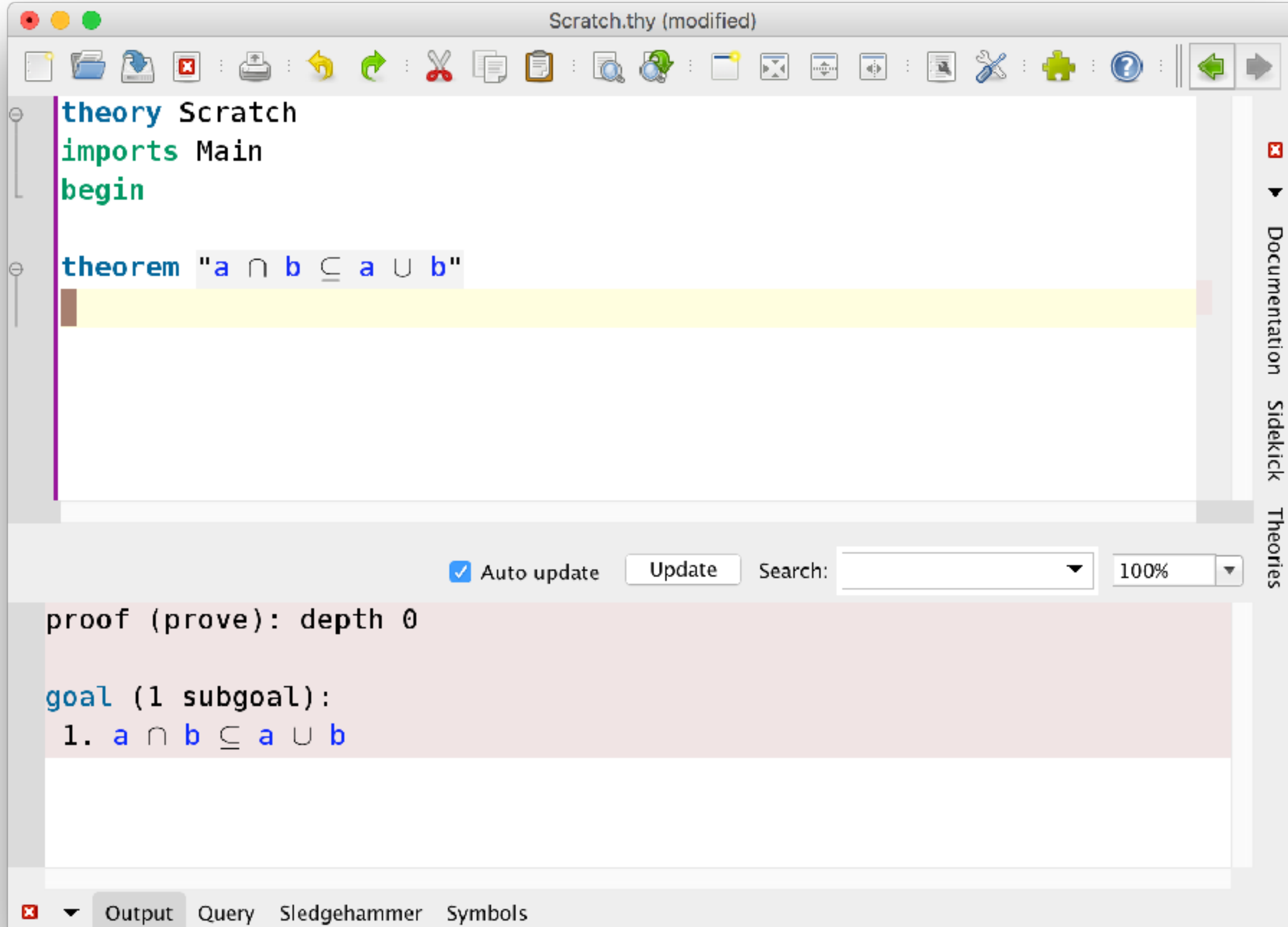
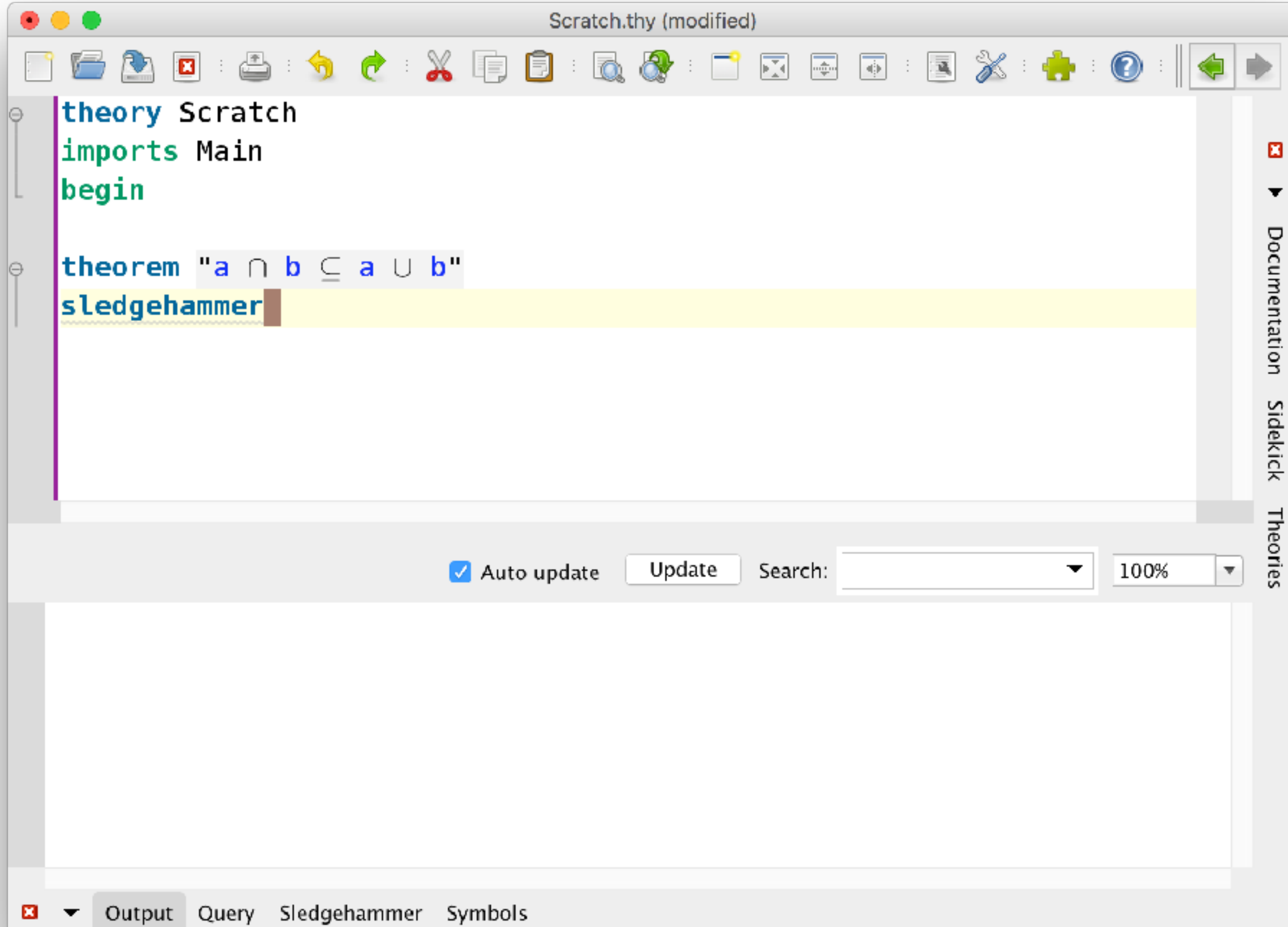
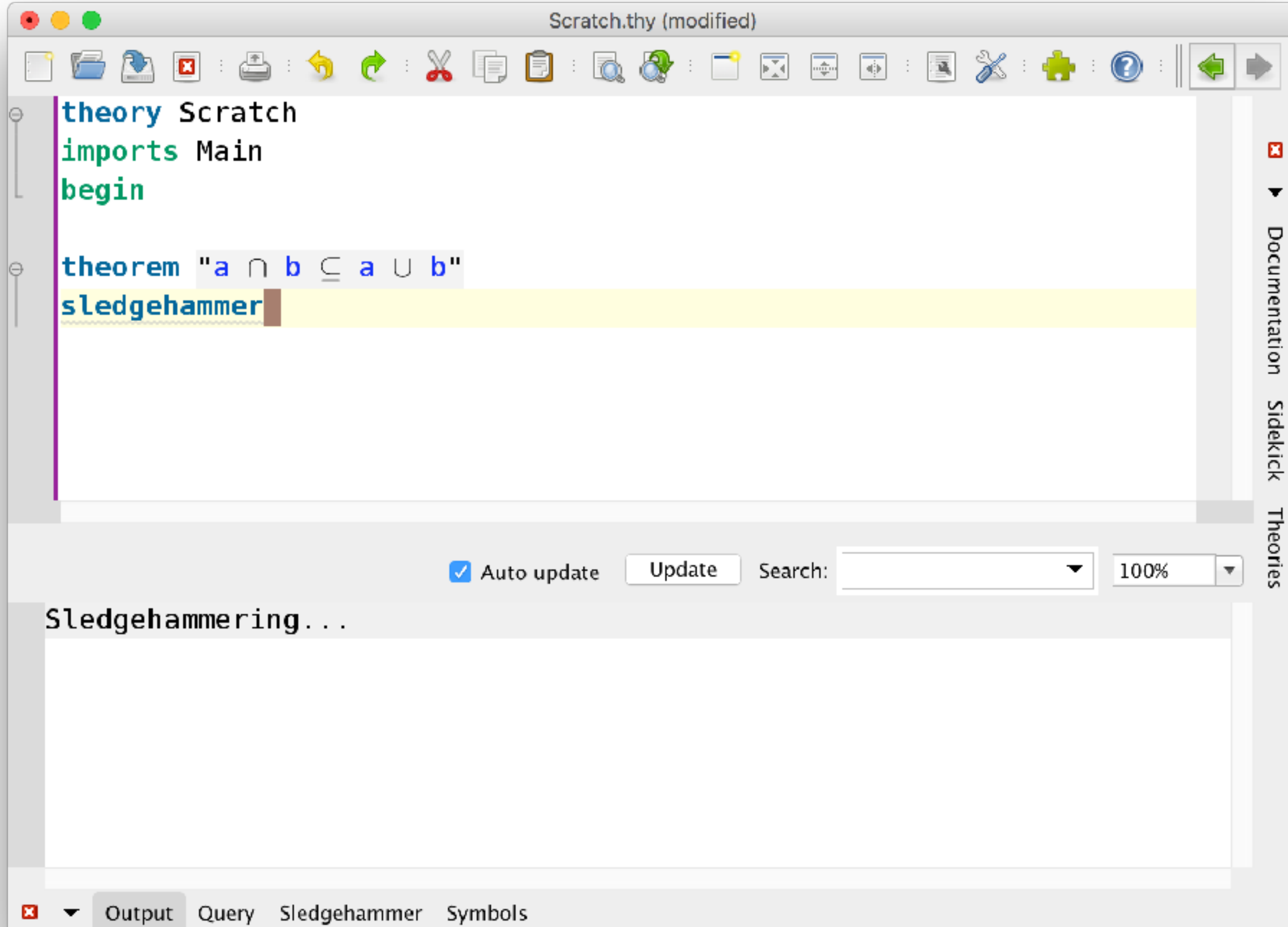
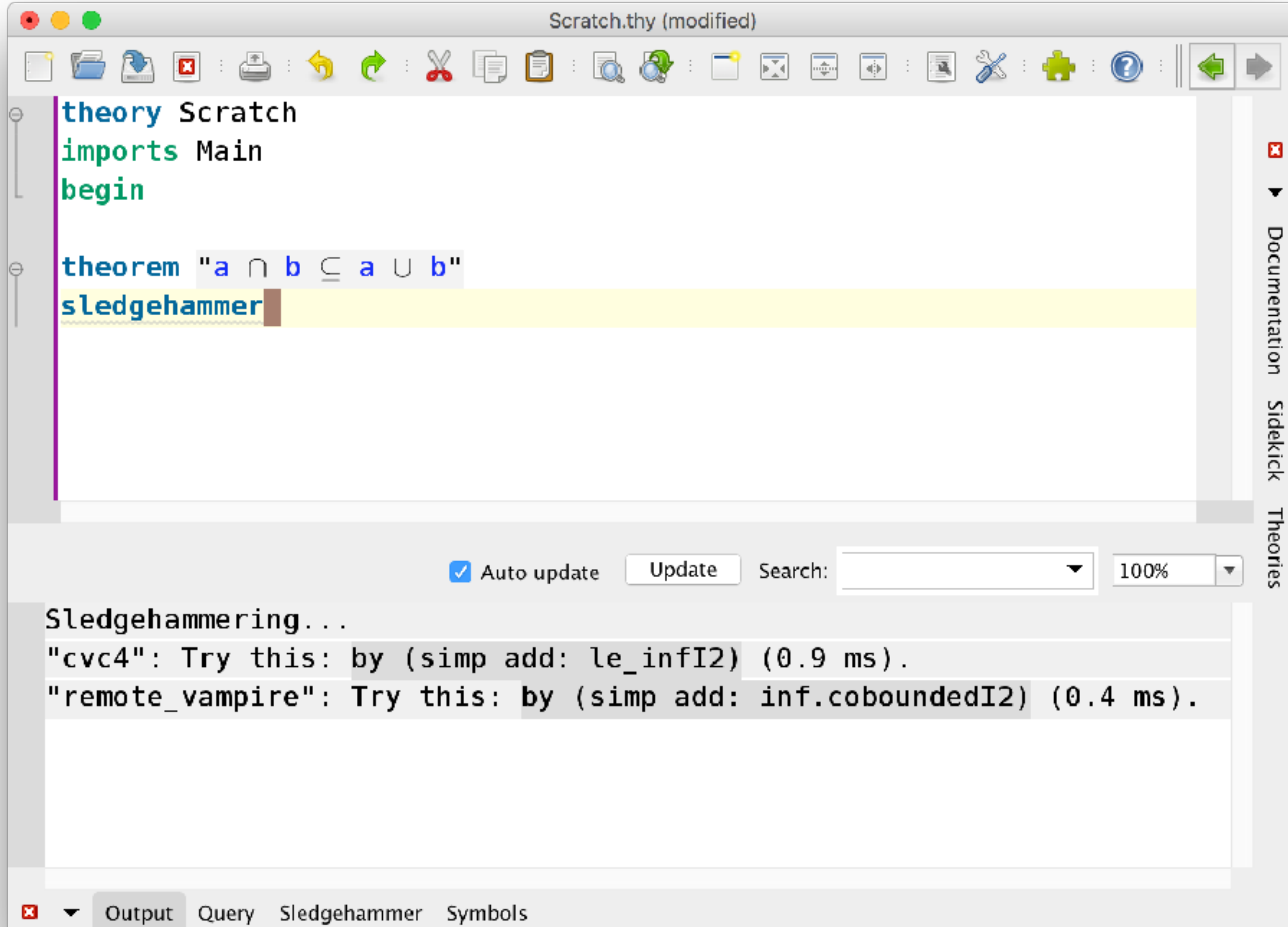
Deductive Program Verification with Vampire

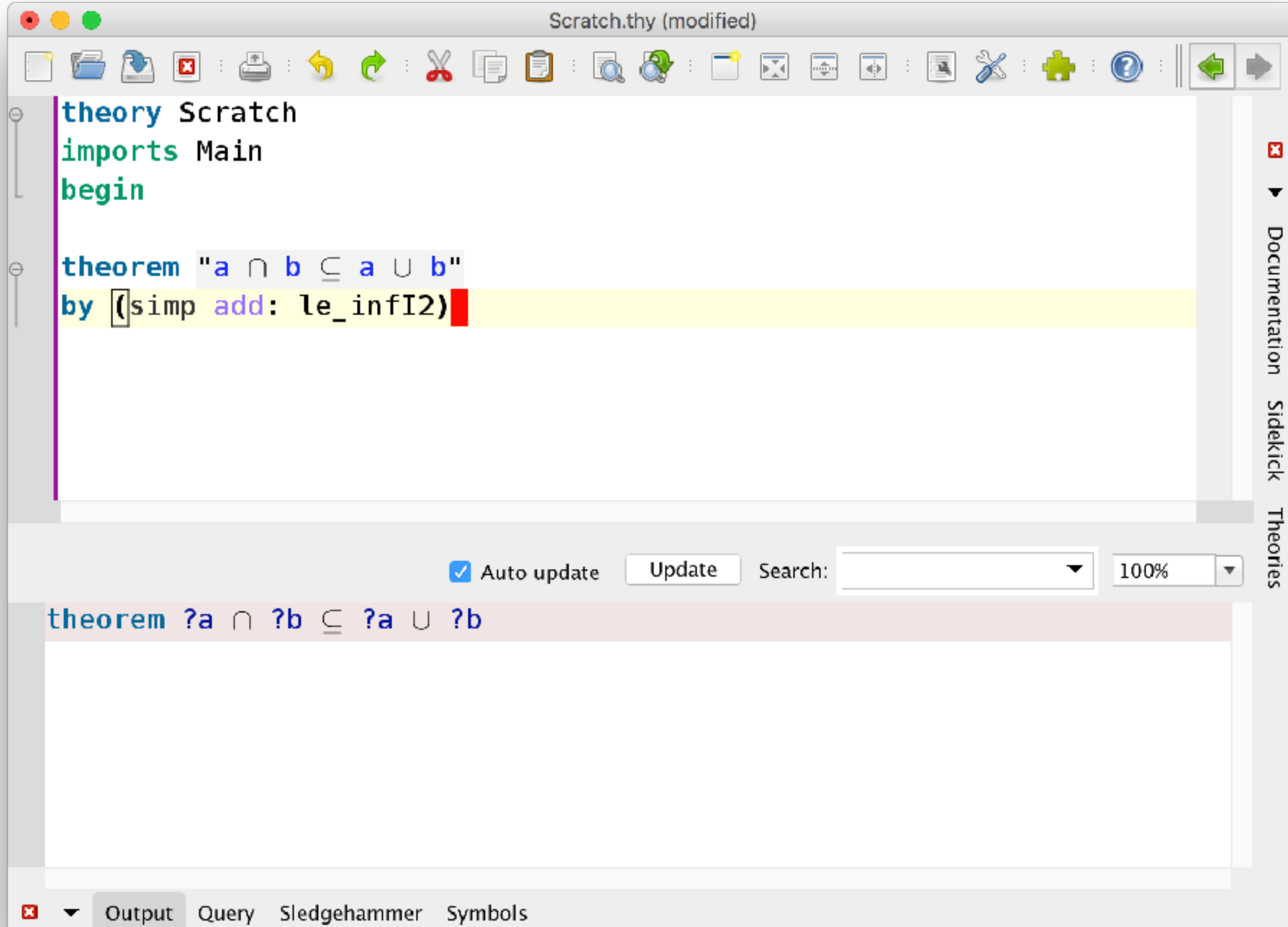
Evgeny Kotelnikov
Chalmers University of Technology
Gothenburg, Sweden

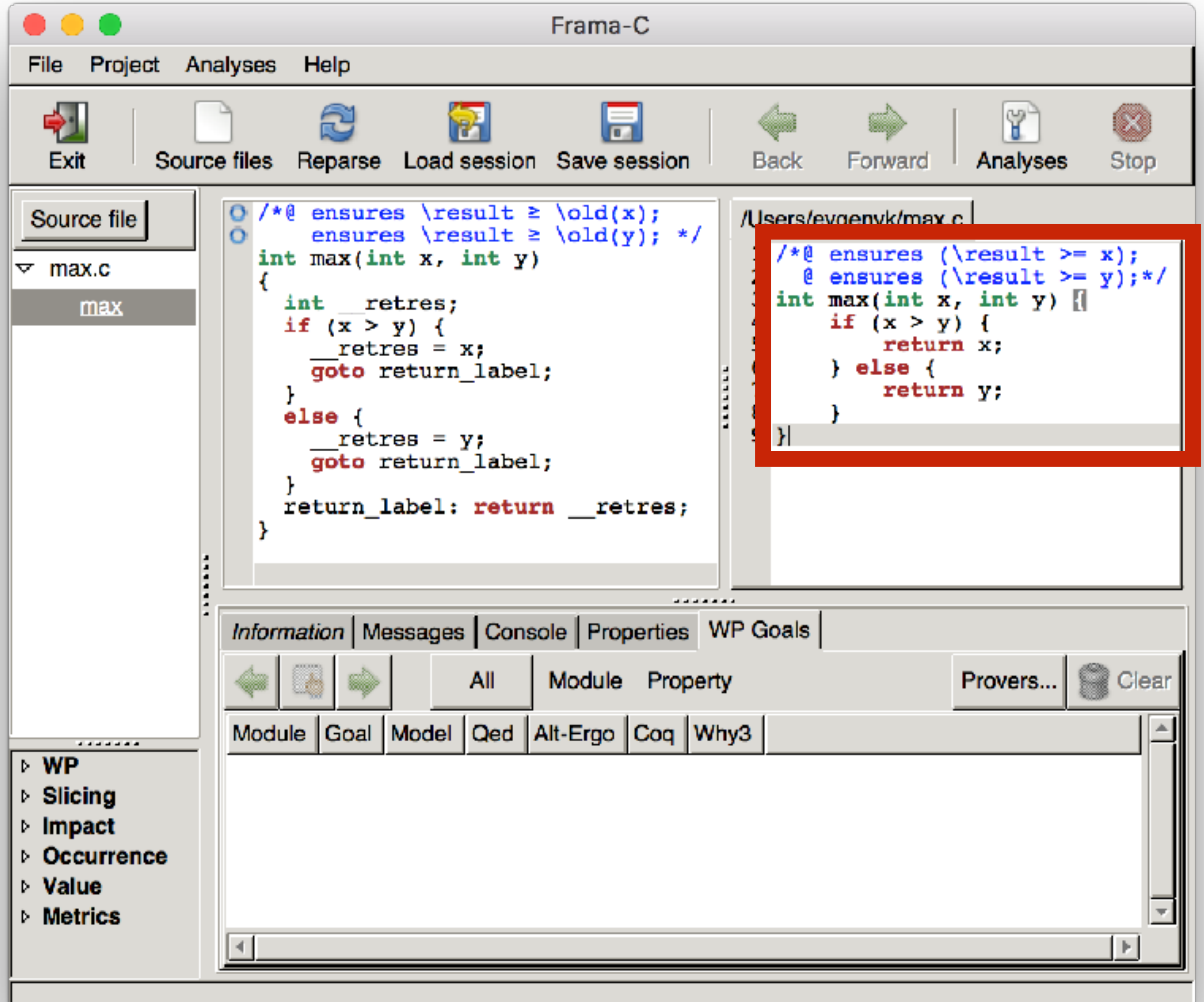

Scratch.thy (modified)
theory Scratch
imports Main
begin
theorem "a ∩ b ⊆ a ∪ b"
Auto update
Update
Search:
100%
proof (prove): depth 0
goal (1 subgoal):
1. a ∩ b ⊆ a ∪ b
Output
Query
Sledgehammer
Symbols
Documentation
Sidekick
Theories

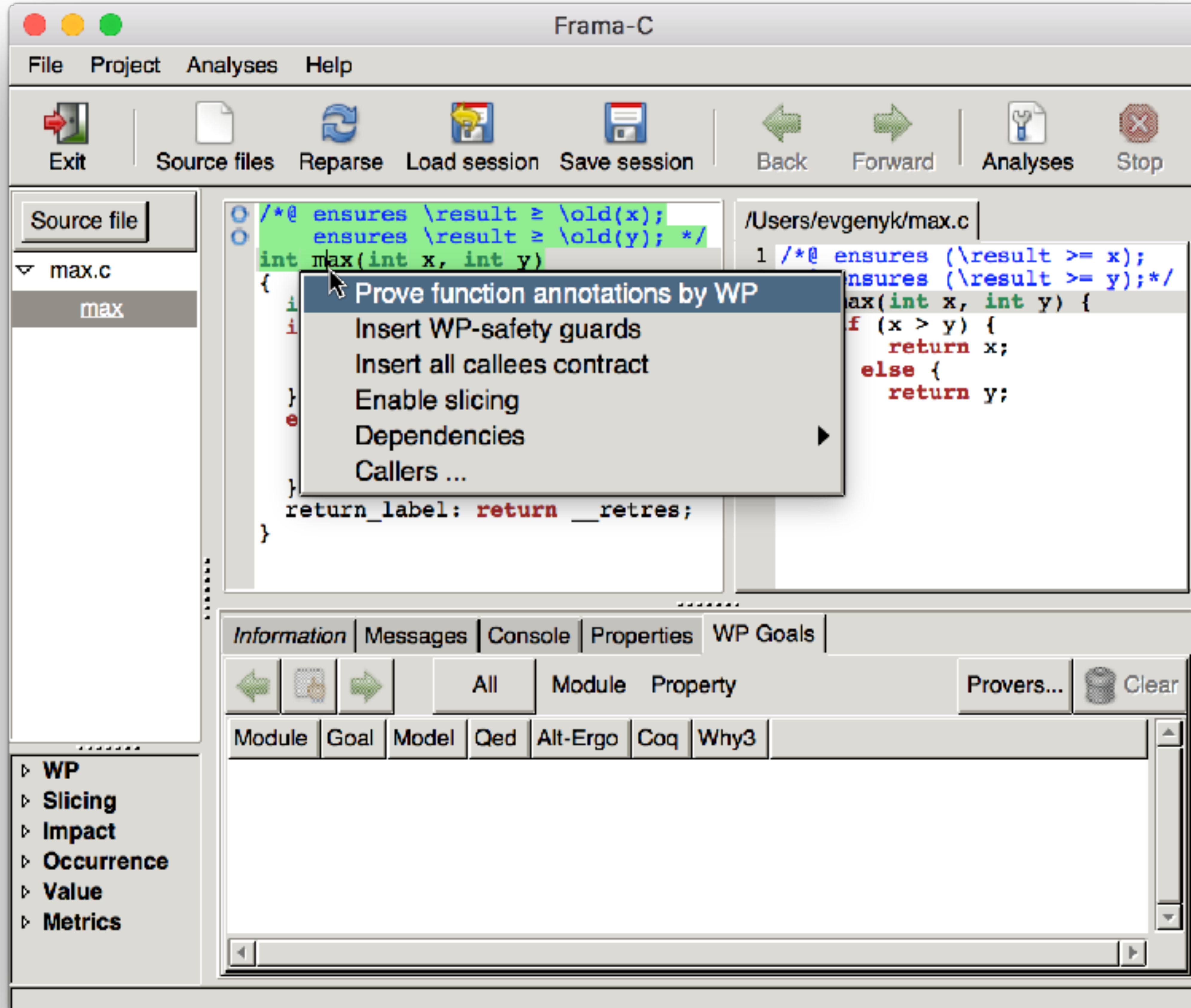

Scratch.thy (modified)
theory Scratch
imports Main
begin
theorem "a ∩ b ⊆ a ∪ b"
sledgehammer
Documentation
Sidekick
Theories
Auto update
Update
Search:
100%
Output
Query
Sledgehammer
Symbols


Scratch.thy (modified)
theory Scratch
imports Main
begin
theorem "a ∩ b ⊆ a ∪ b"
sledgehammer
Auto update
Update
Search:
100%
Sledgehammering...
Output
Query
Sledgehammer
Symbols
Documentation
Sidekick
Theories

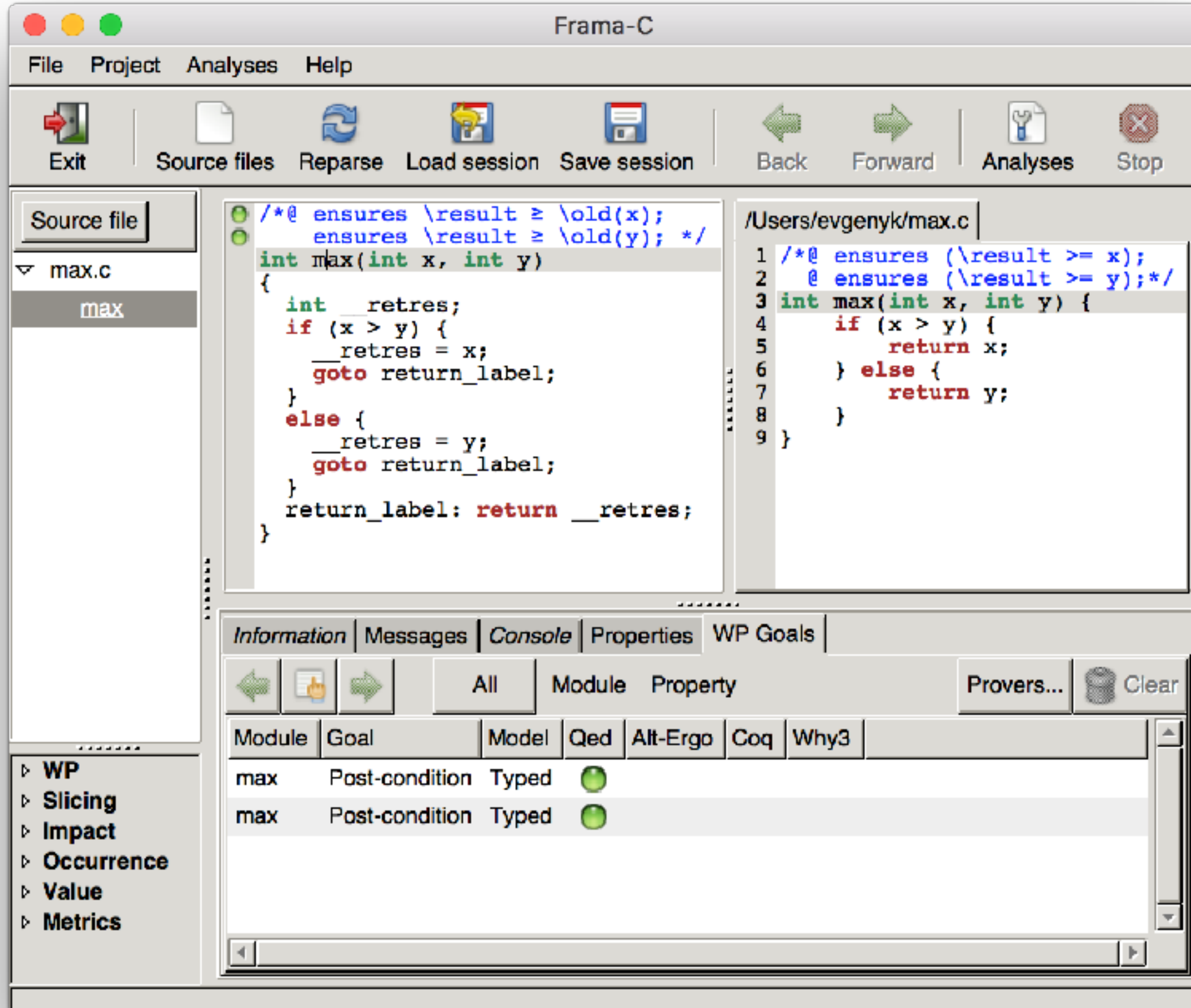

Scratch.thy (modified)
theory Scratch
imports Main
begin
theorem "a ∩ b ⊆ a ∪ b"
sledgehammer
Auto update
Update
Search:
100%
Sledgehammering...
"cvc4": Try this: by (simp add: le_infI2) (0.9 ms).
"remote_vampire": Try this: by (simp add: inf.coboundedI2) (0.4 ms).
Output
Query
Sledgehammer
Symbols
Documentation
Sidekick
Theories

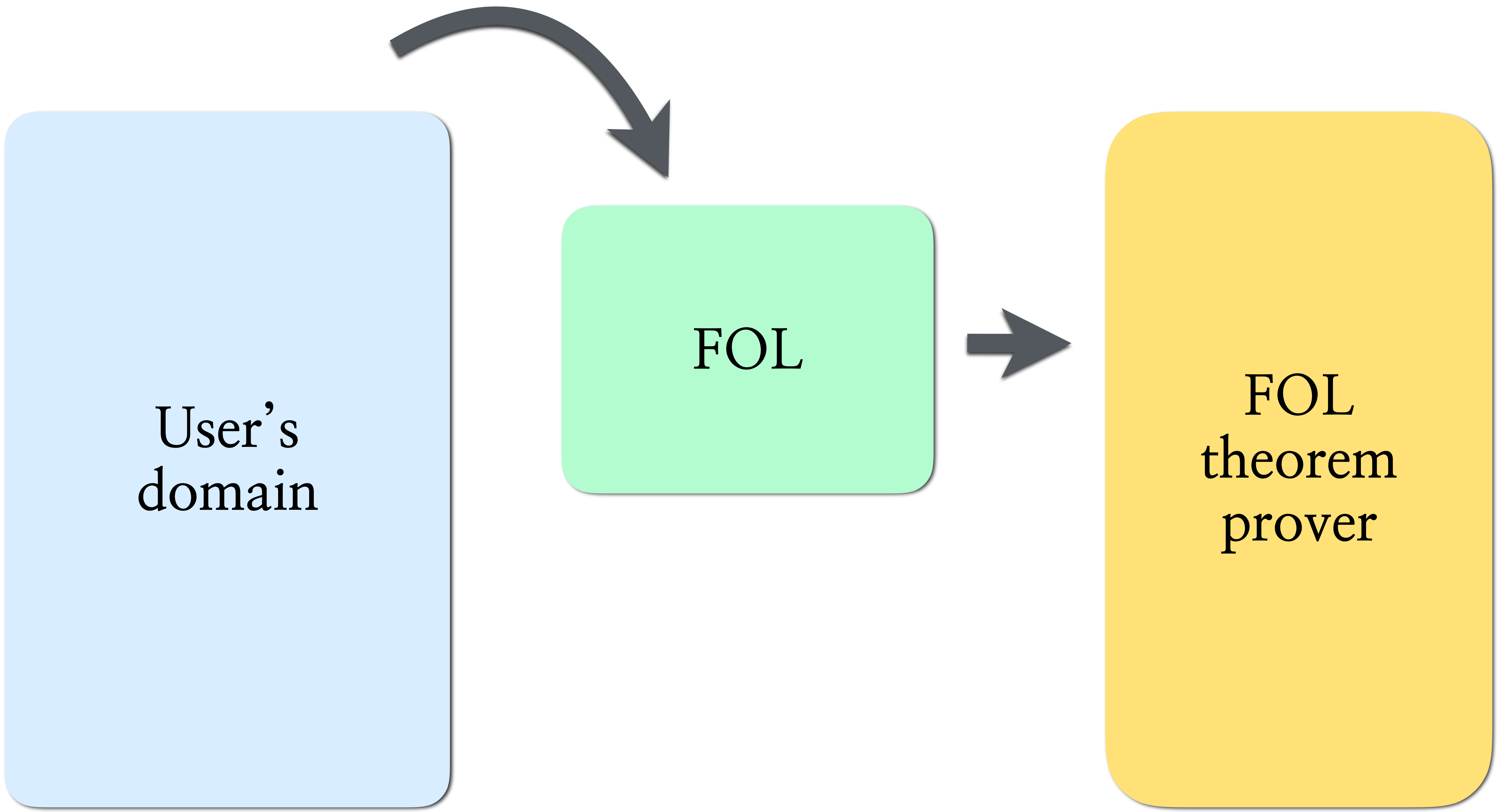

Scratch.thy (modified)
theory Scratch
imports Main
begin
theorem "a ∩ b ⊆ a ∪ b"
by (simp add: le_infI2)
Auto update
Update
Search:
100%
theorem ?a ∩ ?b ⊆ ?a ∪ ?b
Output
Query
Sledgehammer
Symbols
Documentation
Sidekick
Theories


Frama-C
File Project Analyses Help
Exit
Source files
Reparse
Load session
Save session
Back
Forward
Analyses
Stop
Source file
max.c
max
/*@ ensures \result ≥ \old(x);
ensures \result ≥ \old(y); */
int max(int x, int y)
{
int __retres;
if (x > y) {
__retres = x;
goto return_label;
}
else {
__retres = y;
goto return_label;
}
return_label: return __retres;
}
/*@ ensures (\result >= x);
@ ensures (\result >= y);*/
int max(int x, int y) {
if (x > y) {
return x;
} else {
return y;
}
}
Information
Messages
Console
Properties
WP Goals
All
Module
Property
Provers...
Clear
Module
Goal
Model
Qed
Alt-Ergo
Coq
Why3
WP
Slicing
Impact
Occurrence
Value
Metrics

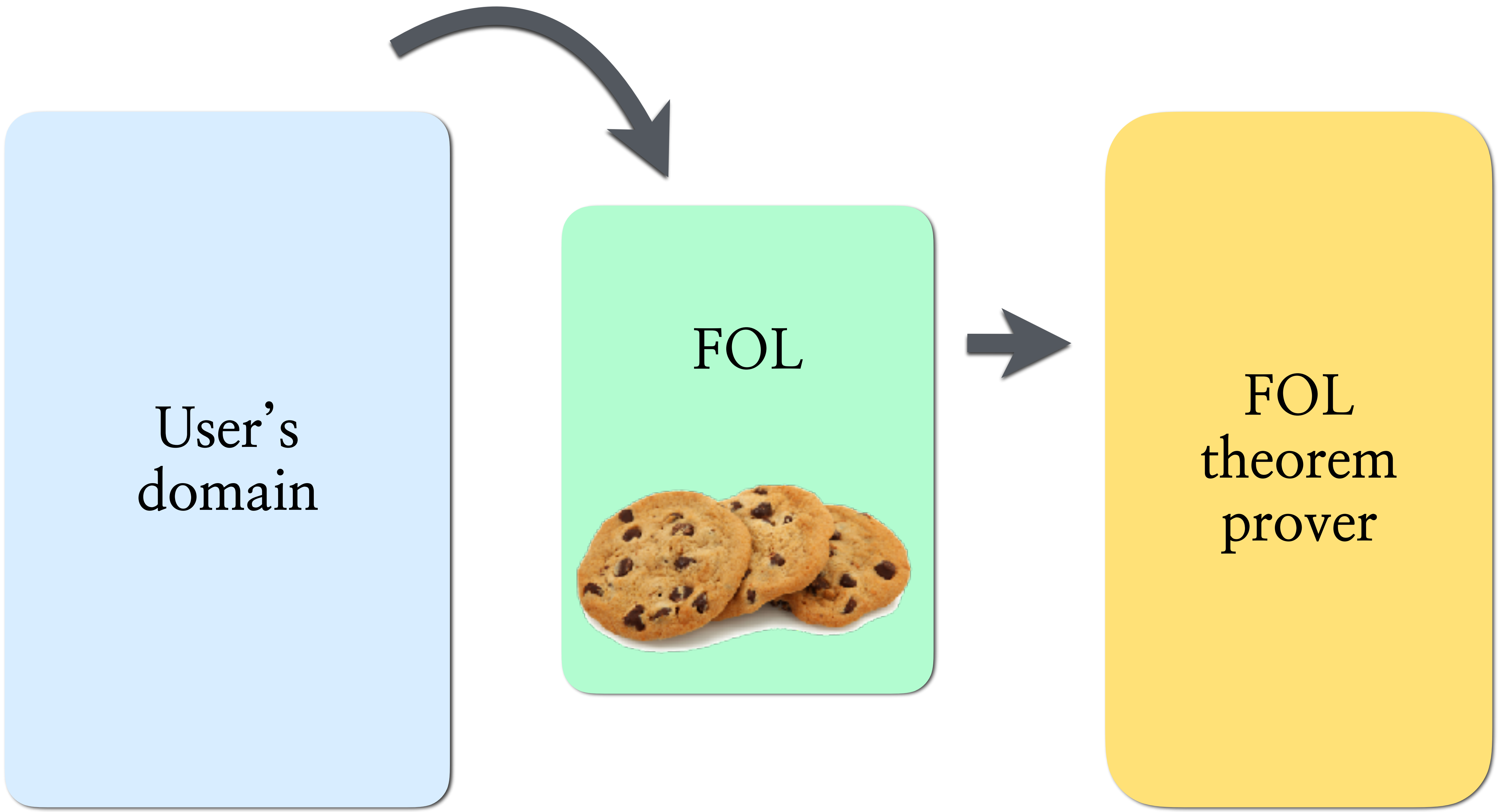


Frama-C
File
Project
Analyses
Help
Exit
Source files
Reparse
Load session
Save session
Back
Forward
Analyses
Stop
Source file
max.c
max
/*@ ensures \result ≥ \old(x);
ensures \result ≥ \old(y); */
int max(int x, int y)
Prove function annotations by WP
Insert WP-safety guards
Insert all callees contract
Enable slicing
Dependencies
Callers ...
return_label: return __retres;
/Users/evgenyk/max.c
1 /*@ ensures (\result >= x);
nsures (\result >= y);*/
ax(int x, int y) {
f (x > y) {
return x;
else {
return y;
WP
Slicing
Impact
Occurrence
Value
Metrics
Information
Messages
Console
Properties
WP Goals
All
Module
Property
Provers...
Clear
Module
Goal
Model
Qed
Alt-Ergo
Coq
Why3


Frama-C
File
Project
Analyses
Help
Exit
Source files
Reparse
Load session
Save session
Back
Forward
Analyses
Stop
Source file
max.c
max
/*@ ensures \result ≥ \old(x);
ensures \result ≥ \old(y); */
int max(int x, int y)
{
int __retres;
if (x > y) {
__retres = x;
goto return_label;
}
else {
__retres = y;
goto return_label;
}
return_label: return __retres;
}
/Users/evgenyk/max.c
1 /*@ ensures (\result >= x);
2 @ ensures (\result >= y);*/
3 int max(int x, int y) {
4 if (x > y) {
5 return x;
6 } else {
7 return y;
8 }
9 }
Information
Messages
Console
Properties
WP Goals
All
Module
Property
Provers...
Clear
Module
Goal
Model
Qed
Alt-Ergo
Coq
Why3
max
Post-condition
Typed
max
Post-condition
Typed
WP
Slicing
Impact
Occurrence
Value
Metrics



User's domain
FOL
FOL theorem prover


User's
domain
FOL
FOL
theorem
prover

```
if (x > y) {
  t = y;
  y = x;
  x = t;
}
d = y - x;
assert d >= 0;
```

- First-class boolean sort

  $(\forall x : bool)(x \vee F(x))$

  $Q((\forall x : \sigma) P(x))$

- Condition expressions

  $\texttt{if}\ \varphi\ \texttt{then}\ \psi_1\ \texttt{else}\ \psi_2$

  $Q(\texttt{if}\ \varphi\ \texttt{then}\ s\ \texttt{else}\ t)$

- Local definitions

  $\texttt{let}\ f(x) = g(x, g(x))\ \texttt{in}\ Q(f(a), f(b))$

- Tuples

  $\texttt{let}\ (a, b) = (b, a)\ \texttt{in}\ f(a, b)$

vim

```
void bubble_sort(int size, int array[]) {
  bool is_sorted;
  do {
    is_sorted = true;
    for (int i = 0; i < size - 1; i++) {
      if (array[i] > array[i + 1]) {
        swap(array[i], array[i + 1]);
        is_sorted = false;
        break;
      }
    }
  } while (!is_sorted);
}
```

$$true \neq false$$

$$(\forall x : bool)(x = true \lor x = false)$$

- First-class boolean sort

  $(\forall x : bool)(x \vee F(x))$

  $Q((\forall x : \sigma)P(x))$

- Condition expressions

  $\texttt{if}\ \varphi\ \texttt{then}\ \psi_1\ \texttt{else}\ \psi_2$

  $Q(\texttt{if}\ \varphi\ \texttt{then}\ s\ \texttt{else}\ t)$

- Local definitions

  $\texttt{let}\ f(x) = g(x, g(x))\ \texttt{in}\ Q(f(a), f(b))$

- Tuples

  $\texttt{let}\ (a, b) = (b, a)\ \texttt{in}\ f(a, b)$

- First-class boolean sort

  $(\forall x : bool)(x \vee F(x))$

  $Q((\forall x : \sigma)P(x))$

- Condition expressions

  $\texttt{if}\ \varphi\ \texttt{then}\ \psi_1\ \texttt{else}\ \psi_2$

  $Q(\texttt{if}\ \varphi\ \texttt{then}\ s\ \texttt{else}\ t)$

- Local definitions

  $\texttt{let}\ f(x) = g(x, g(x))\ \texttt{in}\ Q(f(a), f(b))$

- Tuples

  $\texttt{let}\ (a, b) = (b, a)\ \texttt{in}\ f(a, b)$

- First-class boolean sort

  $(\forall x : bool)(x = true \vee F(x))$

  $Q((\forall x : \sigma)P(x))$

- Condition expressions

  $\texttt{if}\ \varphi\ \texttt{then}\ \psi_1\ \texttt{else}\ \psi_2$

  $Q(\texttt{if}\ \varphi\ \texttt{then}\ s\ \texttt{else}\ t)$

- Local definitions

  $\texttt{let}\ f(x) = g(x, g(x))\ \texttt{in}\ Q(f(a), f(b))$

- Tuples

  $\texttt{let}\ (a, b) = (b, a)\ \texttt{in}\ f(a, b)$

- First-class boolean sort

$$(\forall x : bool)(x = true \vee F(x))$$

$$Q((\forall x : \sigma)P(x))$$

- Condition expressions

$$\texttt{if}\ \varphi\ \texttt{then}\ \psi_1\ \texttt{else}\ \psi_2$$

$$Q(\texttt{if}\ \varphi\ \texttt{then}\ s\ \texttt{else}\ t)$$

- Local definitions

$$\texttt{let}\ f(x) = g(x, g(x))\ \texttt{in}\ Q(f(a), f(b))$$

- Tuples

$$\texttt{let}\ (a, b) = (b, a)\ \texttt{in}\ f(a, b)$$

- First-class boolean sort

$$(\forall x : bool)(x = true \vee F(x))$$

$$Q((\forall x : \sigma)P(x))$$

- Condition expressions

$$(\varphi \Rightarrow \psi_1) \wedge (\neg\varphi \Rightarrow \psi_2)$$

$$Q(\texttt{if } \varphi \texttt{ then } s \texttt{ else } t)$$

- Local definitions

$$\texttt{let } f(x) = g(x, g(x)) \texttt{ in } Q(f(a), f(b))$$

- Tuples

$$\texttt{let } (a, b) = (b, a) \texttt{ in } f(a, b)$$

- First-class boolean sort

$$(\forall x : bool)(x = true \vee F(x))$$

$$Q((\forall x : \sigma)P(x))$$

- Condition expressions

$$(\varphi \Rightarrow \psi_1) \wedge (\neg\varphi \Rightarrow \psi_2)$$

$$Q(\texttt{if } \varphi \texttt{ then } s \texttt{ else } t)$$

- Local definitions

$$\texttt{let } f(x) = g(x, g(x)) \texttt{ in } Q(f(a), f(b))$$

- Tuples

$$\texttt{let } (a, b) = (b, a) \texttt{ in } f(a, b)$$

- First-class boolean sort

  $$(\forall x : bool)(x = true \vee F(x))$$

  $$Q(g_1)$$

- Condition expressions

  $$(\varphi \Rightarrow \psi_1) \wedge (\neg\varphi \Rightarrow \psi_2)$$

  $$Q(g_2)$$

- Local definitions

  $$Q(g_3(a), g_3(b))$$

- Tuples

  $$f(g_4, g_5)$$

$$(Premise_1 \wedge \ldots \wedge Premise_n) \Rightarrow Conjecture$$

$$(Premise_1 \land \ldots \land Premise_n \land \neg Conjecture) \Rightarrow \bot$$

- CNF: $C_1 \wedge \ldots \wedge C_n$

- Clause: $(\forall x_1 : \sigma_1) \ldots (\forall x_n : \sigma_n)(L_1 \vee \ldots \vee L_n)$

- Literal: A or $\neg A$

- Atom: $P(t_1, \ldots, t_n)$ or $t_1 = t_2$

$$(\forall x : \tau)(P(x) \wedge Q(x) \Leftrightarrow R(x, f(x)))$$

$$\downarrow$$

$$\left\{ \begin{array}{l} \neg P(x) \vee \neg Q(x) \vee R(x, f(x)) \\ P(x) \vee \neg R(x, f(x)) \\ Q(x) \vee \neg R(x, f(x)) \end{array} \right\}$$

$$(A_1 \wedge B_1) \vee \ldots \vee (A_n \wedge B_n)$$

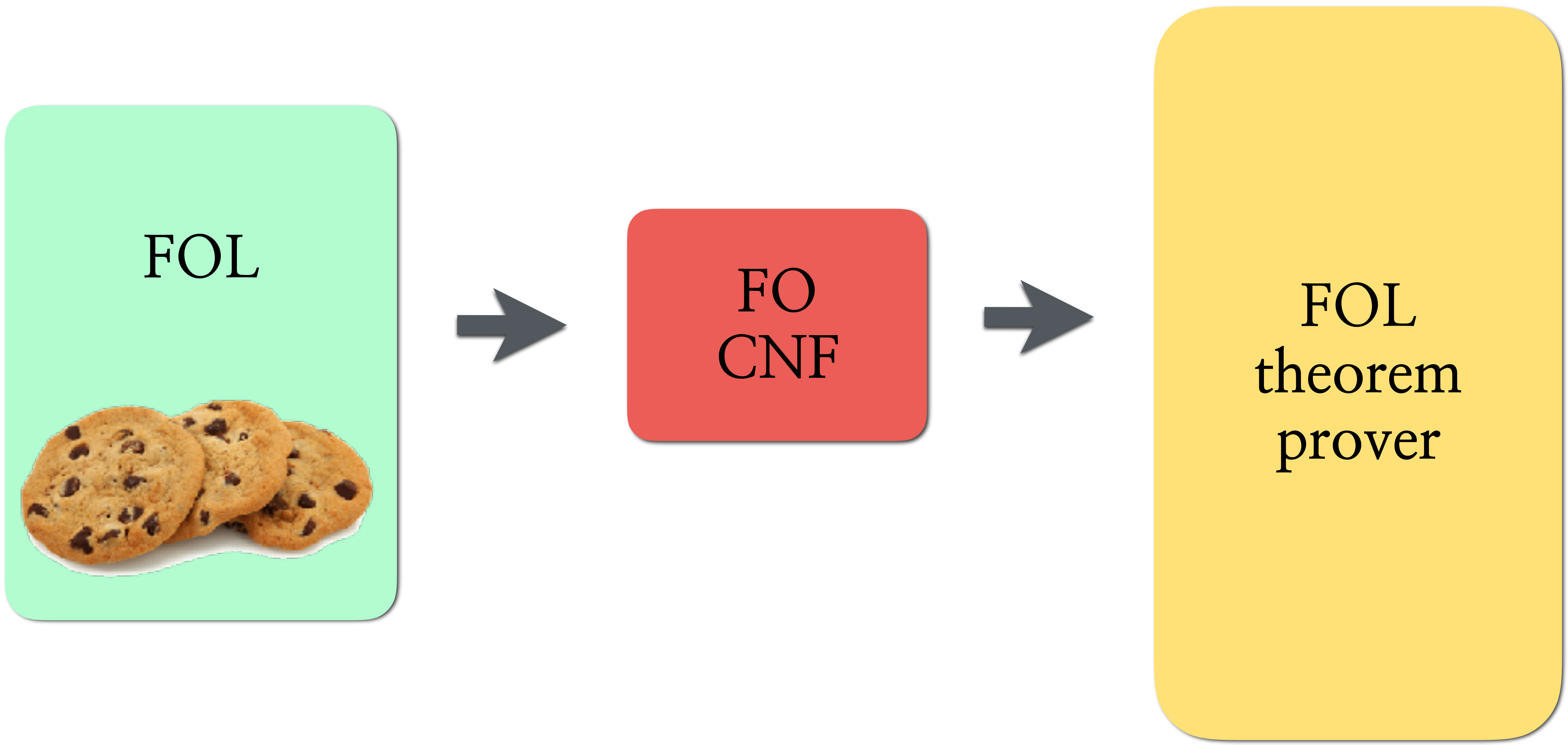
$$(R_1 \vee \ldots \vee R_n) \wedge (R_1 \Leftrightarrow (A_1 \wedge B_1)) \wedge \ldots \wedge (R_n \Leftrightarrow (A_1 \wedge B_n))$$

$$\frac{A \vee C_1 \quad \neg A' \vee C_2}{(C_1 \vee C_2)\theta} \qquad \theta = mgu(A, A')$$

$$\frac{l = r \vee C_1 \quad L[s] \vee C_2}{(L[r] \vee C_1 \vee C_2)\theta} \qquad \theta = mgu(l, s)$$

$$(\forall x : bool)(x = true \vee x = false)$$

$$\frac{x = true \vee x = false \quad y = true \vee y = false}{x = y \vee x = false \vee y = false}$$


FOL
FO
CNF
FOL
theorem
prover

$$\texttt{if } \varphi \texttt{ then } \psi_1 \texttt{ else } \psi_2 \quad \Longrightarrow \quad \left\{ \begin{array}{r} \neg\varphi, \psi_1 \\ \varphi, \psi_2 \end{array} \right\}$$

$$\left\{ \begin{array}{c} Q(x) \vee \neg G(x) \\ \neg\varphi \vee G(s) \\ \varphi \vee \neg G(t) \end{array} \right\}$$

$$Q(\texttt{if}\ \varphi\ \texttt{then}\ s\ \texttt{else}\ t)$$

$$\left\{ \begin{array}{c} \neg\varphi \vee Q(s) \\ \varphi \vee Q(t) \end{array} \right\}$$

$$Q((\forall x : \sigma) P(x))$$

$$Q(\texttt{if } (\forall x : \sigma)P(x) \texttt{ then } \mathit{true} \texttt{ else } \mathit{false})$$

$$(\forall x : bool)(x \vee F(x))$$

$$(\top \vee F(true)) \wedge (\bot \vee F(false))$$

$F(false)$

$$\texttt{let } a = f(b) \texttt{ in } R(a, a)$$

$$\left\{ \begin{array}{c} g = f(b) \\ R(g, g) \end{array} \right\}$$

$$\{\ R(f(b), f(b))\ \}$$

$$\texttt{let}\ P(x) = Q(x) \vee R(x)\ \texttt{in}\ P(a) \wedge P(b)$$

$$\texttt{let}\ (a, b) = (b, a)\ \texttt{in}\ f(a, b)$$

$$\texttt{let}\ t = (b, a)\ \texttt{in}\ f(\pi_1(t), \pi_2(t))$$

```
if (x > y) {
  t = y;
  y = x;
  x = t;
}
d = y - x;
assert d >= 0;
```

```
if (x > y) {
  t = y;
  y = x;
  x = t;
}
d = y - x;
assert d >= 0;
```

let $(x, y, t) =$

if $x > y$ then

let $t = y$ in

let $y = x$ in

let $x = t$ in

(x, y, t)

else

(x, y, t) in

let $d = x - y$

in $d \geq 0$

vim

```
if (x > y) {
  t = y;
  y = x;
  x = t;
}
d = y - x;
assert d >= 0;
```

$$
\begin{array}{l}
\texttt{let}\ (x, y, t) = \\
\quad \texttt{if}\ x > y\ \texttt{then} \\
\qquad \texttt{let}\ t = y\ \texttt{in} \\
\qquad \texttt{let}\ y = x\ \texttt{in} \\
\qquad \texttt{let}\ x = t\ \texttt{in} \\
\qquad (x, y, t) \\
\quad \texttt{else} \\
\qquad (x, y, t)\ \ \texttt{in} \\
\texttt{let}\ d = x - y \\
\quad \texttt{in}\ d \geq 0
\end{array}
$$

```
if (x > y) {
  t = y;
  y = x;
  x = t;
}
d = y - x;
assert d >= 0;
```

let $(x, y, t) =$

  if $x > y$ then

    let $t = y$ in

    let $y = x$ in

    let $x = t$ in

      (x, y, t)

  else

    (x, y, t) in

let $d = x - y$

  in $d \geq 0$

vim

```
if (x > y) {
  t = y;
  y = x;
  x = t;
}
d = y - x;
assert d >= 0;
```

$$\texttt{let}\ (x, y, t) =$$

$$\texttt{if}\ x > y\ \texttt{then}$$

$$\texttt{let}\ t = y\ \texttt{in}$$

$$\texttt{let}\ y = x\ \texttt{in}$$

$$\texttt{let}\ x = t\ \texttt{in}$$

$$(x, y, t)$$

$$\texttt{else}$$

$$(x, y, t)\ \ \texttt{in}$$

$$\texttt{let}\ d = x - y$$

$$\texttt{in}\ d \geq 0$$

vim

```
if (x > y) {
  t = y;
  y = x;
  x = t;
}
d = y - x;
assert d >= 0;
```

$$
\begin{array}{l}
\texttt{let}\ (x, y, t) = \\
\quad \texttt{if}\ x > y\ \texttt{then} \\
\qquad \texttt{let}\ t = y\ \texttt{in} \\
\qquad \texttt{let}\ y = x\ \texttt{in} \\
\qquad \texttt{let}\ x = t\ \texttt{in} \\
\qquad\quad (x, y, t) \\
\quad \texttt{else} \\
\qquad (x, y, t)\ \ \texttt{in} \\
\texttt{let}\ d = x - y \\
\quad \texttt{in}\ d \geq 0
\end{array}
$$

```
% Boolean variables
tff(1, axiom, ![X:$o]: (X | ~X)).

% Formulas as terms
tff(2, axiom, ![X:$o, Y:$o]: (impl(X, Y) <=> (~X | Y))).

% if-then-else
tff(3, axiom, ![X:$int, Y:$int]:
                (max(X, Y) = $ite($greatereq(X, Y), X, Y))).

tff(4, axiom, ![X:$int, Y:$int]: $ite(max(X, Y) = X,
                                      $greatereq(X, Y),
                                      $greatereq(Y, X))).

% let-in
tff(5, axiom, $let(array(I:$int) := $ite(I = 3, 5, array(I)),
                   $sum(array(2), array(3)))).

tff(6, axiom, $let(a := b; b := a, f(a, b))).

~
~
~
~
```