

COMP4011/8011
Advanced Topics in
Formal Methods and Programming Languages
– **Software Verification with Isabelle/HOL** –

Peter Höfner

July 21, 2024

Section 2

Enough Theory!

Getting started with Isabelle

System Architecture

Prover IDE (jEdit) – user interface

HOL, ZF – object-logics

Isabelle – generic, interactive theorem prover

Standard ML – logic implemented as ADT

User can access all layers!

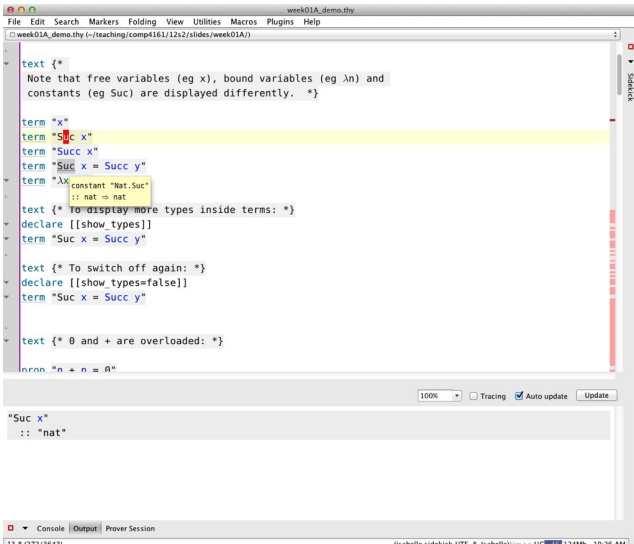
System Requirements

- Linux, **Windows**, or MacOS X (10.8 +)
- Standard ML (PolyML implementation)
- Java (for jEdit)

Pre-made packages for Linux, Mac, and Windows + info on:
<https://proofcraft.systems/isabelle/>

Demo

jEdit/PIDE



The screenshot shows the jEdit/PIDE IDE interface. The main editor displays a Lean script with the following content:

```

text {*
  Note that free variables (eg x), bound variables (eg  $\lambda n$ ) and
  constants (eg Succ) are displayed differently. *}

term "x"
term "Succ x"
term "Succ x"
term "Succ x = Succ y"
term " $\lambda x$  constant "Nat.Suc"
  :: nat  $\rightarrow$  nat

text {* To display more types inside terms: *}
declare [[show_types]]
term "Suc x = Succ y"

text {* To switch off again: *}
declare [[show_types=false]]
term "Suc x = Succ y"

text {*  $\emptyset$  and + are overloaded: *}

prop "n + n =  $\emptyset$ "

```

The output window at the bottom shows the result of the evaluation:

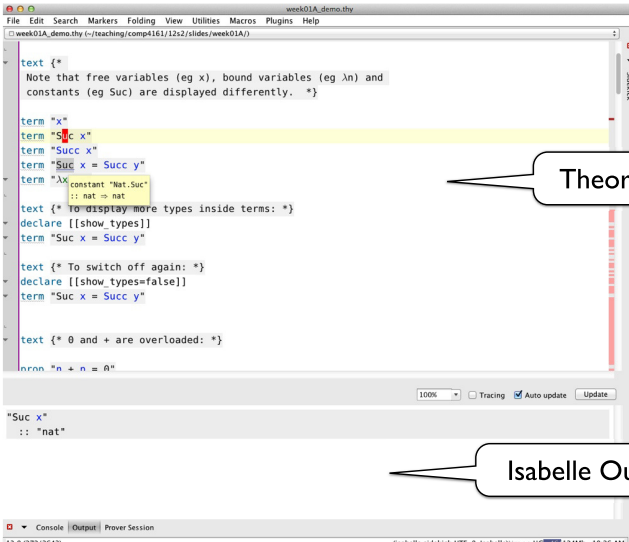
```

"Suc x"
:: "nat"

```

The interface includes a menu bar (File, Edit, Search, Markers, Folding, View, Utilities, Macros, Plugins, Help), a toolbar, and a status bar at the bottom with a console, output, and prover session view.

jEdit/PIDE



The screenshot shows the jEdit/PIDE interface with a theory file open. The editor window displays the following code:

```
text {*  
  Note that free variables (eg x), bound variables (eg  $\lambda n$ ) and  
  constants (eg Suc) are displayed differently. *}  
  
term "x"  
term "Suc x"  
term "Succ x"  
term "Suc x = Succ y"  
term " $\lambda x$  constant \"Nat.Suc\"  
  :: nat  $\rightarrow$  nat"  
  
text {* To display more types inside terms: *}  
declare [[show_types]]  
term "Suc x = Succ y"  
  
text {* To switch off again: *}  
declare [[show_types=false]]  
term "Suc x = Succ y"  
  
text {*  $\emptyset$  and + are overloaded: *}  
prop "n + n = 0"
```

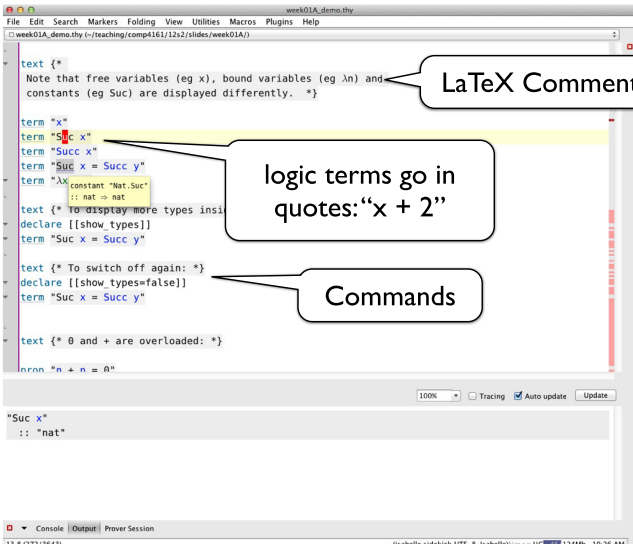
The output window at the bottom shows the result of the compilation:

```
"Suc x"  
:: "nat"
```

Theory File

Isabelle Output

jEdit/PIDE



The screenshot shows the jEdit/PIDE editor interface with a file named `week01A_demo.thy`. The code is as follows:

```

text {*
Note that free variables (eg x), bound variables (eg  $\lambda n$ ) and
constants (eg Suc) are displayed differently. *}

term "x"
term "Suc x"
term "Succ x"
term "Suc x = Succ y"
term "λx constant \"Nat.Suc\"
  :: nat → nat"
text {* To display more types inside
declare [[show_types]]
term "Suc x = Succ y"

text {* To switch off again: *}
declare [[show_types=false]]
term "Suc x = Succ y"

text {* 0 and + are overloaded: *}

prop "n + n = 0"

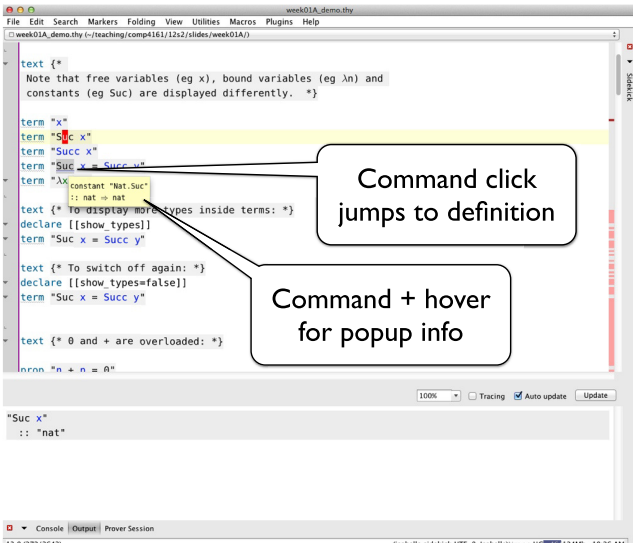
```

Callouts in the image explain the following parts of the code:

- LaTeX Comment:** Points to the `text {*` block containing a comment with LaTeX symbols like λn .
- logic terms go in quotes: "x + 2":** Points to the `term "Suc x"` line, indicating that logic terms are enclosed in double quotes.
- Commands:** Points to the `declare [[show_types=false]]` line, indicating that these are commands used to control the editor's display.

The bottom of the editor shows a preview of the term `"Suc x"` rendered as `:: "nat"`.

jEdit/PIDE



The screenshot shows the jEdit/PIDE editor interface with a file named `week01A_demo.thy`. The editor contains the following code:

```

text {*
Note that free variables (eg x), bound variables (eg λn) and
constants (eg Suc) are displayed differently. *}

term "x"
term "Suc x"
term "Succ x"
term "Suc x = Succ y"
term "λx constant \"Nat.Suc\"
  :: nat ⇒ nat"
text {* To display more types inside terms: *}
declare [[show_types]]
term "Suc x = Succ y"

text {* To switch off again: *}
declare [[show_types=false]]
term "Suc x = Succ y"

text {* 0 and + are overloaded: *}
prop "n + n = 0"
  
```

Two callout boxes provide instructions:

- Command click jumps to definition**: A callout box points to the `"Suc x"` term in the code.
- Command + hover for popup info**: A callout box points to the `"λx constant \"Nat.Suc\" :: nat ⇒ nat"` definition in the code.

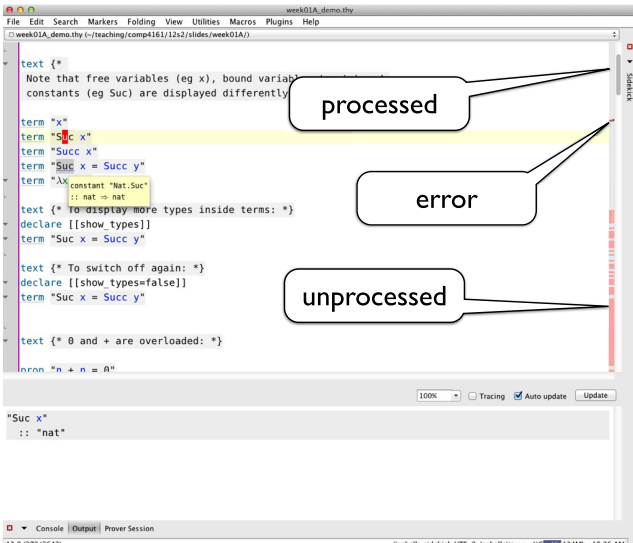
At the bottom of the editor, a popup window displays the definition for `"Suc x"`:

```

"Suc x"
:: "nat"
  
```

The bottom status bar shows the console output and prover session information.

jEdit/PIDE



The screenshot shows the jEdit/PIDE editor interface with a file named 'week01A_demo.thy'. The code is as follows:

```

text {*
Note that free variables (eg x), bound variables (eg x),
constants (eg Suc) are displayed differently

term "x"
term "Suc x"
term "Succ x"
term "Suc x = Succ y"
term "λx constant \"Nat.Suc\"
:: nat ⇒ nat

text {* To display more types inside terms: *}
declare [[show_types]]
term "Suc x = Succ y"

text {* To switch off again: *}
declare [[show_types=false]]
term "Suc x = Succ y"

text {* 0 and + are overloaded: *}
prop "n + n = 0"

```

Annotations in the image:

- processed**: A callout box pointing to the first line of code: `text {*`.
- error**: A callout box pointing to the line `term "Suc x"`, which has a red squiggly line under the `Suc` constructor.
- unprocessed**: A callout box pointing to the line `term "Suc x = Succ y"`, which is not highlighted.

At the bottom of the editor, there is a status bar with '100%' zoom, 'Tracing' unchecked, 'Auto update' checked, and an 'Update' button. Below the editor is a console area with tabs for 'Console', 'Output', and 'Prover Session'. The console shows the output for the term `"Suc x"`:

```

"Suc x"
:: "nat"

```

Exercises

- Download and install Isabelle
- Step through the demo files from the lecture web page
- Write your own theory file, look at some theorems in the library, try 'find_theorems'
- How many theorems can help you if you need to prove something containing the term $\text{Suc}(\text{Suc } x)$?
- What is the name of the theorem for associativity of addition of natural numbers in the library?