Australian National University

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

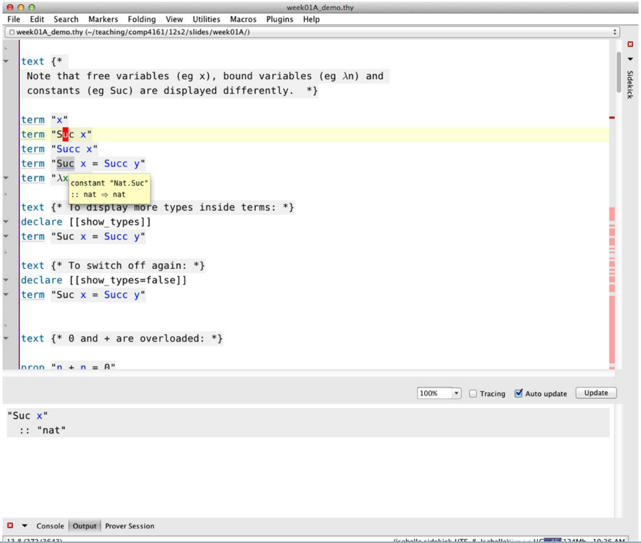
- Linux, **Windows**, or MacOS X (10.8 +)
- Standard ML (PolyML implementation)
- Java (for jEdit)

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

Australian National University

Demo

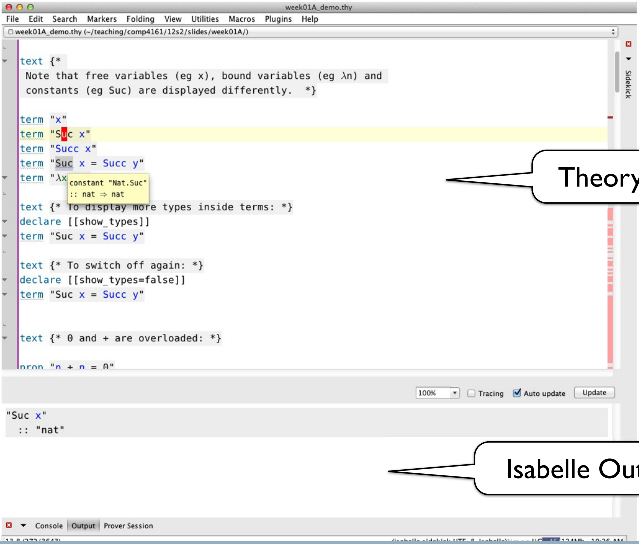
jEdit/PIDE


week01A_demo.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
week01A_demo.thy (~/teaching/comp4161/12s2/slides/week01A/)
Sidekick
100%
Tracing
Auto update
Update
Console
Output
Prover Session

```
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"
```

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

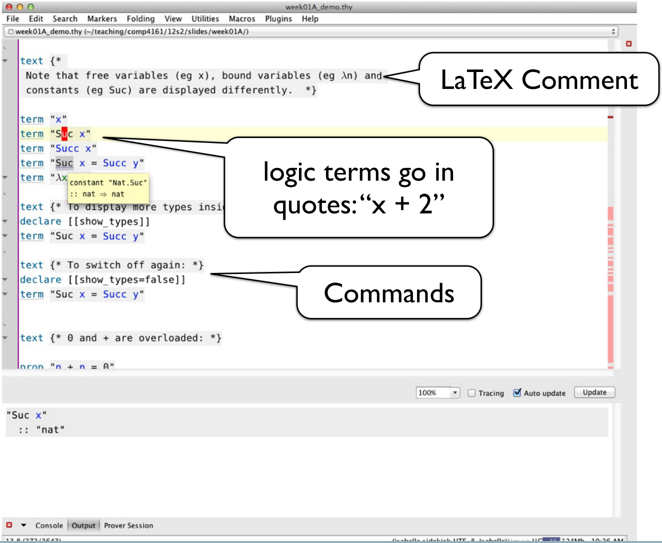
jEdit/PIDE


week01A_demo.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
week01A_demo.thy (~/teaching/comp4161/12s2/slides/week01A/)
Sidekick
100%
Tracing
Auto update
Update
Console
Output
Prover Session
Theory File
Isabelle Output

```
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"
```

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

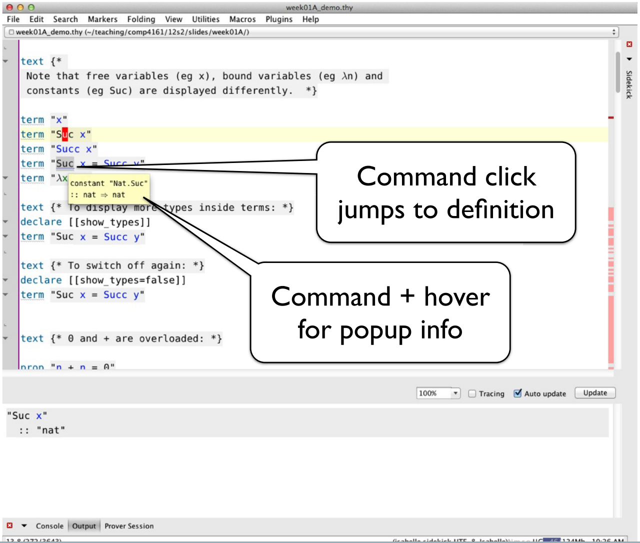
jEdit/PIDE


week01A_demo.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
week01A_demo.thy (~/teaching/comp4161/12s2/slides/week01A/)
LaTeX Comment
logic terms go in quotes: "x + 2"
Commands
100%
Tracing
Auto update
Update
Console Output Prover Session

```
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 insi
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"
```

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

jEdit/PIDE


week01A_demo.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
week01A_demo.thy (~/teaching/comp4161/12s2/slides/week01A/)
Sidekick
100%
Tracing
Auto update
Update
Console
Output
Prover Session

```
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
```

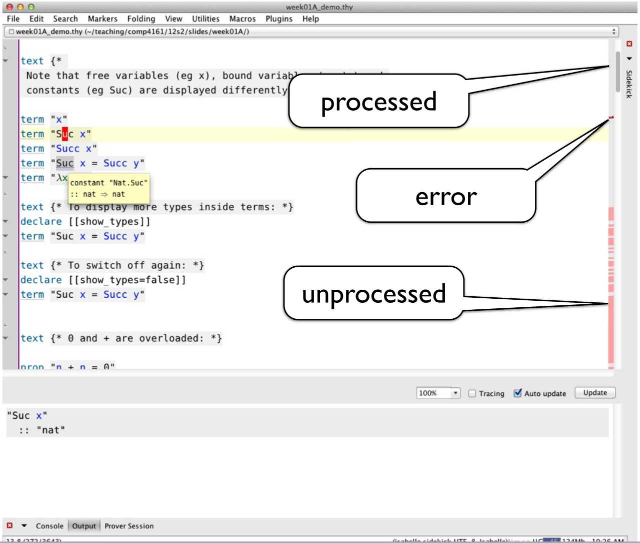
```
text {* To display more types inside terms: *}
declare [[show_types]]
term "Suc x = Succ y"

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

text {* 0 and + are overloaded: *}

prop "n + n = 0"
```

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

Command click jumps to definition

Command + hover for popup info

jEdit/PIDE


week01A_demo.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
week01A_demo.thy (~/teaching/comp4161/12s2/slides/week01A/)
Sidekick
processed
error
unprocessed
100%
Tracing
Auto update
Update
Console
Output
Prover Session

```
text {*
 Note that free variables (eg x), bound variab
 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"
```

```
"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 “Suc(Suc x)”?
- What is the name of the theorem for associativity of addition of natural numbers in the library?