Isabelle/jEdit for seasoned Isabelle users

Isabelle/jEdit NEWS

Makarius Wenzel
Univ. Paris-Sud, LRI

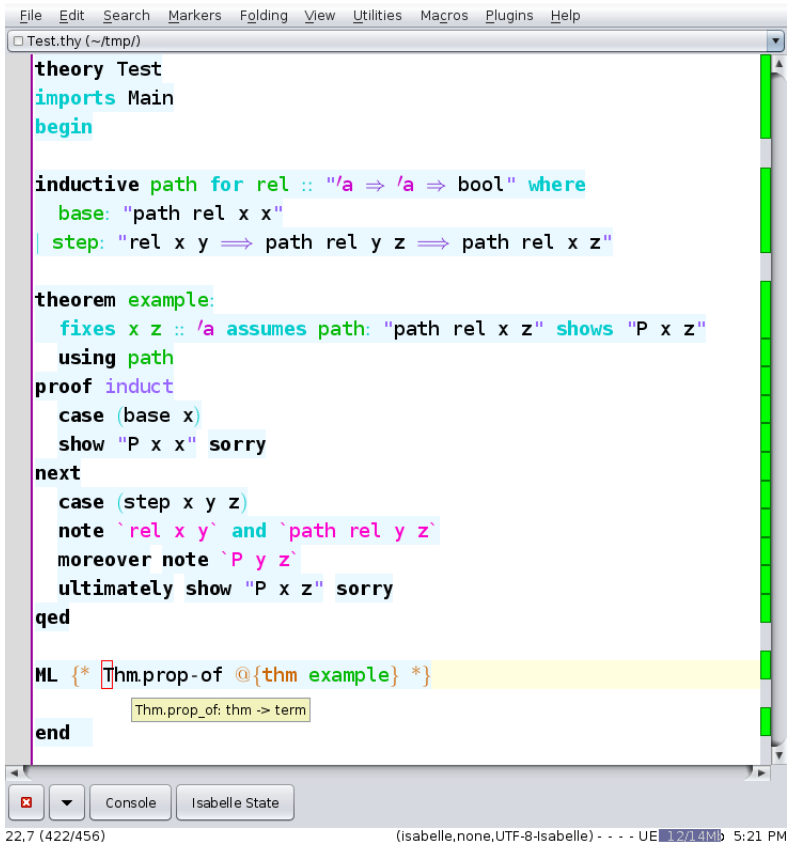
13-Jul-2014

"There is nothing to prove or even to suppose that the Serbian government is accessory to the inducement for the crime, its preparations, or the furnishing of weapons. On the contrary, there are reasons to believe that this altogether out of the question."
(The July Crisis, 13-Jul-1914)

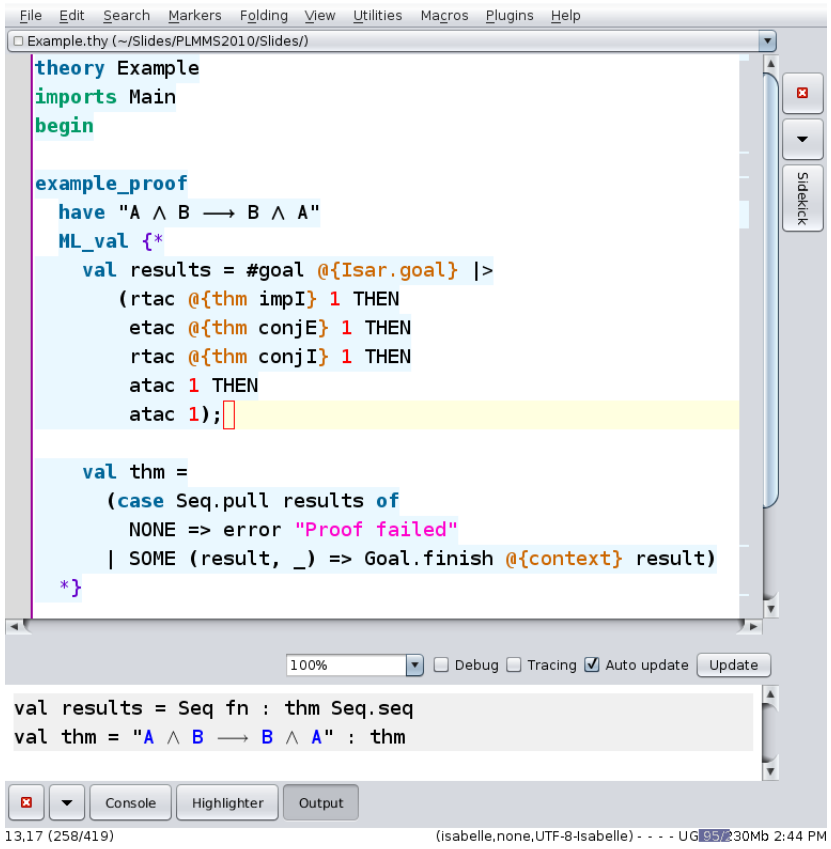
David Fromkin, 2004: *Europe's Last Summer: Who started the Great War in 1914?*

History

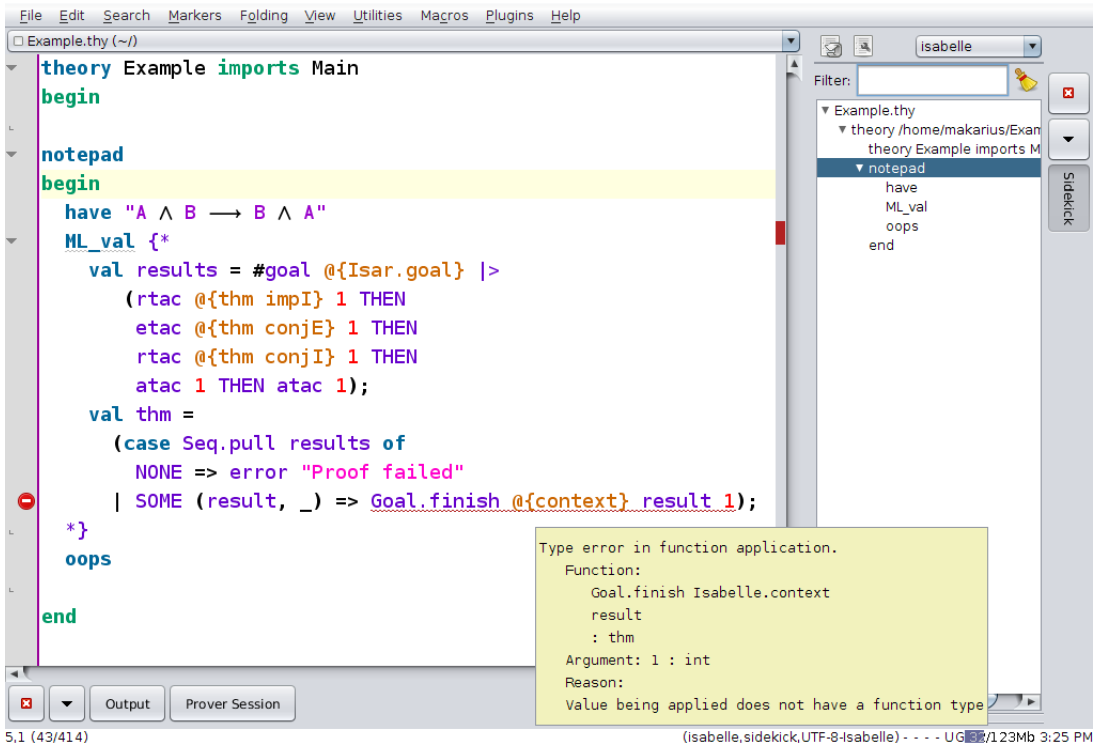
2009 (experimental)



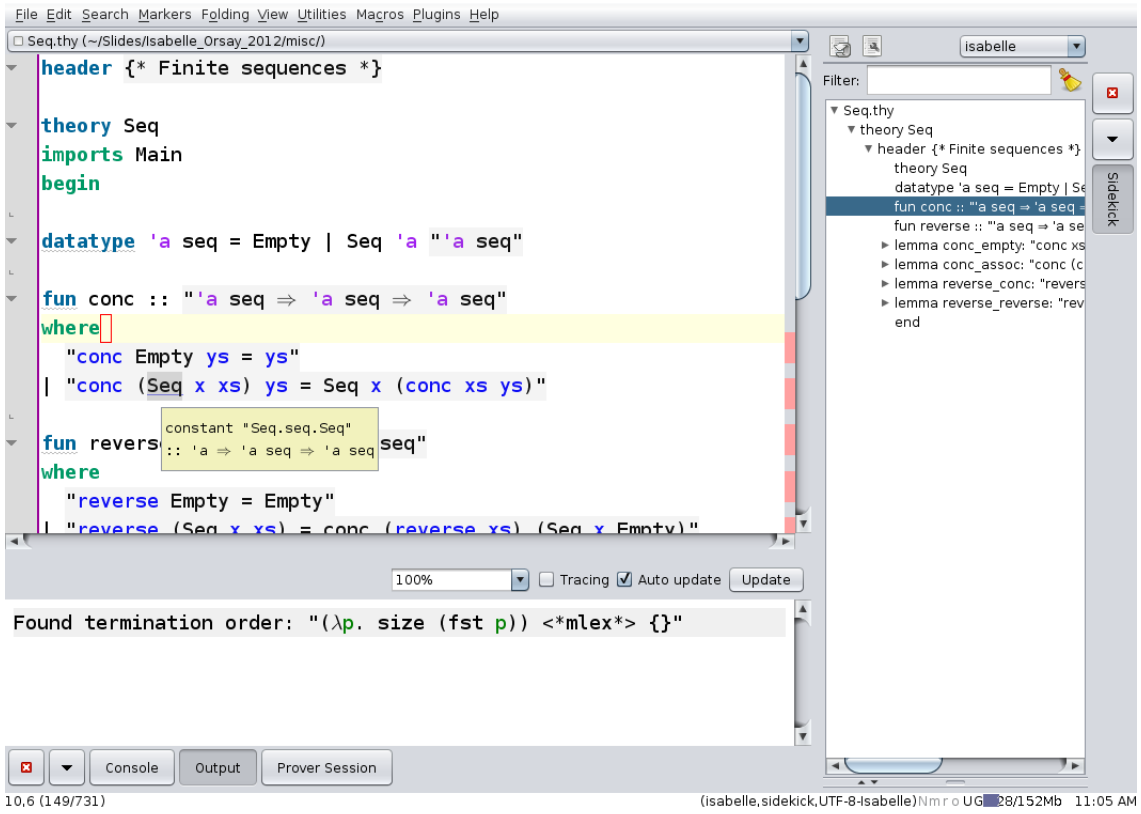
2010 (experimental)



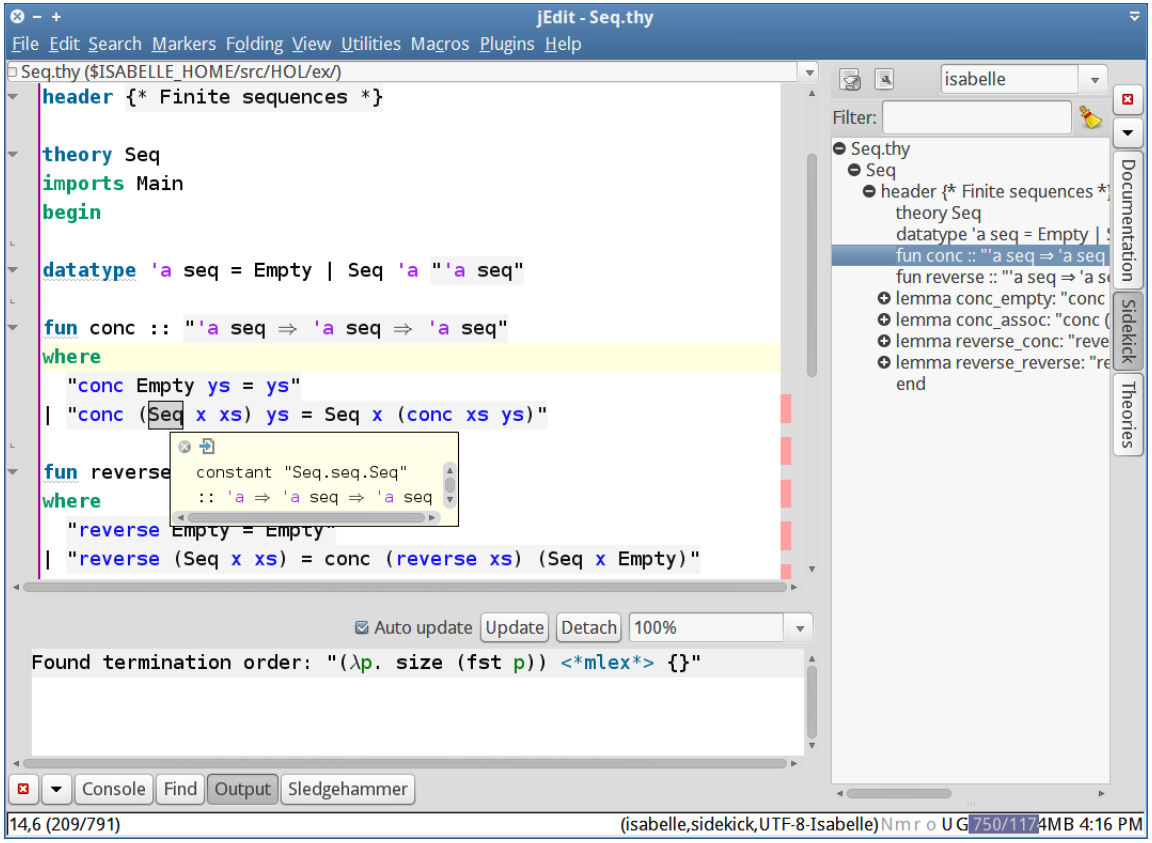
2011 (experimental $\rightsquigarrow$ stable)



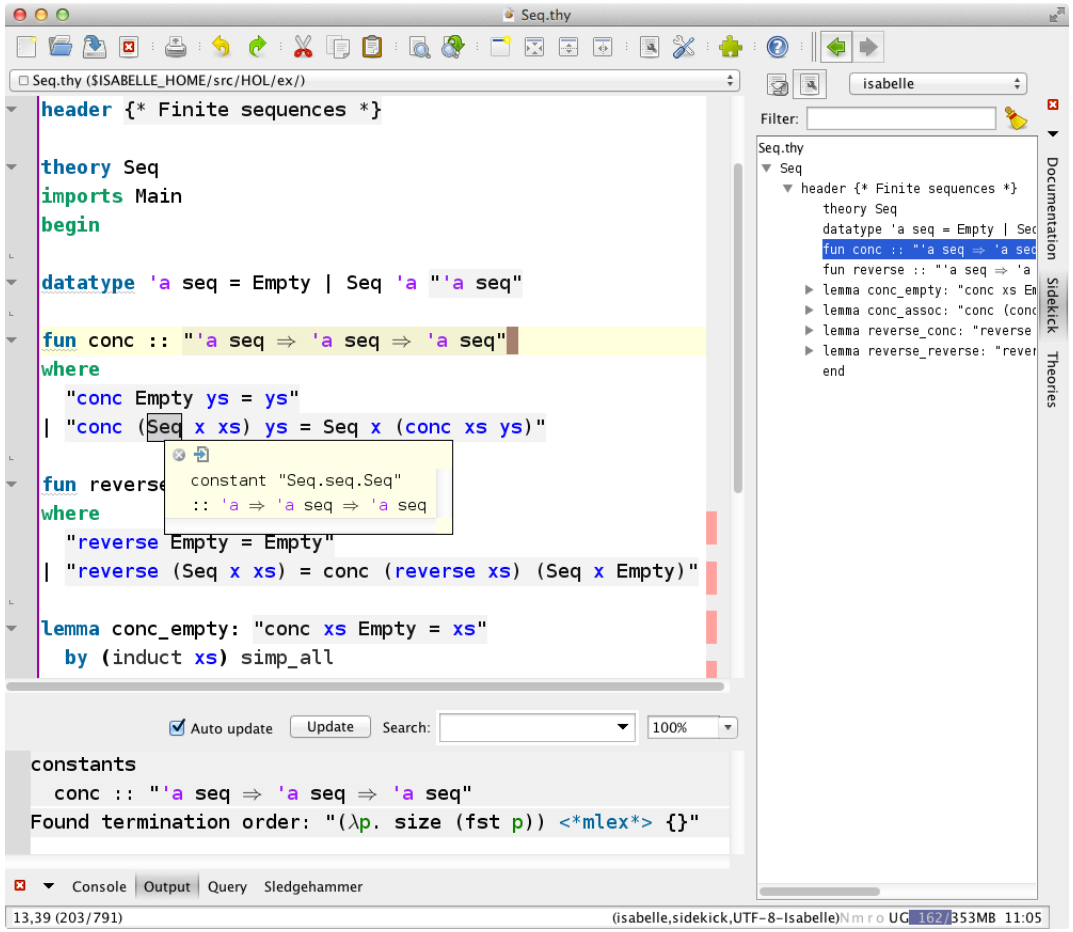
2012 (stable)


File Edit Search Markers Folding View Utilities Macros Plugins Help
Seq.thy (~/Slides/Isabelle_Orsay_2012/misc/)
header {* Finite sequences *}
theory Seq
imports Main
begin
datatype 'a seq = Empty | Seq 'a "'a seq"
fun conc :: "'a seq ⇒ 'a seq ⇒ 'a seq"
where
"conc Empty ys = ys"
| "conc (Seq x xs) ys = Seq x (conc xs ys)"
constant "Seq.seq.Seq"
:: 'a ⇒ 'a seq ⇒ 'a seq
fun revers
seq"
where
"reverse Empty = Empty"
100%
Tracing
Auto update
Update
Found termination order: "(λp. size (fst p)) <*mlex*> {}"
Console
Output
Prover Session
isabelle
Filter:
Seq.thy
theory Seq
header {* Finite sequences *}
theory Seq
datatype 'a seq = Empty | Se
fun conc :: "'a seq ⇒ 'a seq
fun reverse :: "'a seq ⇒ 'a se
lemma conc_empty: "conc xs
lemma conc_assoc: "conc (c
lemma reverse_conc: "revers
lemma reverse_reverse: "rev
end
Sidekick
10,6 (149/731)
(isabelle,sidekick,UTF-8-Isabelle)
28/152Mb
11:05 AM

2013 (stable, rich client)



2014 (stable, filthy rich client)



jEdit 5.1

What is jEdit?

Logo: 

Website: `http://www.jedit.org`

User's guide: `http://www.jedit.org/users-guide`

Official blurb:

"jEdit is a mature programmer's text editor with hundreds [...] of person-years of development behind it. [...] The jEdit core, together with a large collection of plugins is maintained by a world-wide developer team."

Vox populi:

"The legendary jEdit text editor."

Notable jEdit features

- multiple views, buffers, text areas
- dockable windows
- status bar and action bar
- advanced file management: browser, URLs, reload events
- advanced selections: range, rectangular, tall-caret, multi-selections
- copy-paste registers
- persistent markers
- advanced search-replace: hypersearch, multi-file search, search bar
- bracket matching and block selection
- folding and narrowing
- macros and BeanShell scripting
- open-ended plugin interface for JVM languages (Java, Scala)

Notable jEdit plugins

- Console: BeanShell, System shell, Scala toplevel
- Highlight: regular expression "text markers"
- SideKick: tree view
- Navigator (Isabelle2014)
- Isabelle

  but: Isabelle/jEdit is not just another jEdit plugin
  - all-inclusive application bundle
  - different implementation language: Scala instead of Java
  - different intention: Prover IDE instead of plain text editor

Isabelle/jEdit
in Isabelle2013-2

Prover IDE functionality

- File-system access: mixed-platform notation; Isabelle settings from environment; hyperlinks for formal file references
- Text buffers and theories: *Theories* panel; management of editor buffers versus PIDE document-nodes; editor perspective as hint for theory processing; rendering of semantic markup
- Prover output: in-source messages; gutter area; text overview area; *Output* panel as imitation of old-style terminal
- Tooltips and hyperlinks: C-hover idiom to ask for more information; nested tooltips with same recursive structure (including hyperlinks); detachable tooltips and *Info* windows
- Text completion: completion tables from outer syntax and Isabelle symbols; completion popups with minimal key event handling; explicit or implicit completion

- Isabelle symbols: custom encoding; custom font with all required glyphs; various input methods, including *Symbols* panel; control symbols for change of font style
- Automatically tried tools: methods, nitpick, quickcheck, sledgehammer, solve-direct
- Sledgehammer: GUI control panel
- Find theorems: GUI control panel

Miscellaneous tools

- SikeKick: `NEWS` file; alternative parser `isabelle-markup` for PIDE document content
- Timing: separate panel to analyze command transactions
- Isabelle/Scala console: direct access to running PIDE front-end
- Low-level output: side-channels for system diagnostics

Isabelle/jEdit NEWS

Forthcoming Isabelle2014 (August 2014)

- syntactic and semantic completion, including spell-checking
- editor navigation (like web browser)
- auxiliary files within the document model
  - Isabelle/ML files, e.g. direct editing of Isabelle/HOL
  - Standard ML files: IDE for SML'97, independently of theory and proof development
- systematic nesting of sub-languages via text cartouches
  $\langle funny\ \langle quotes\ \langle with \rangle\ arbitrary\ \langle nesting \rangle \rangle \rangle$
- improved platform support: Windows, Mac OS X
  but: continued decay of Linux desktop environments

Syntactic and semantic completion

Completion

Goal: "Do what I mean" for the Prover IDE

Problems:

- timing of GUI events: keyboard input, mouse, popups
  $\longrightarrow$ potential loss of events, deadlocks, confusing display
- need for adequate information from editor and prover
  $\longrightarrow$ asynchronous document-model opens new possibilities,
  but also challenges: non-determinism

Approach:

- interpretation of GUI events wrt. semantic completion context
- particular completion information from prover,
  e.g. failed name lookups

Applications in Isabelle/jEdit

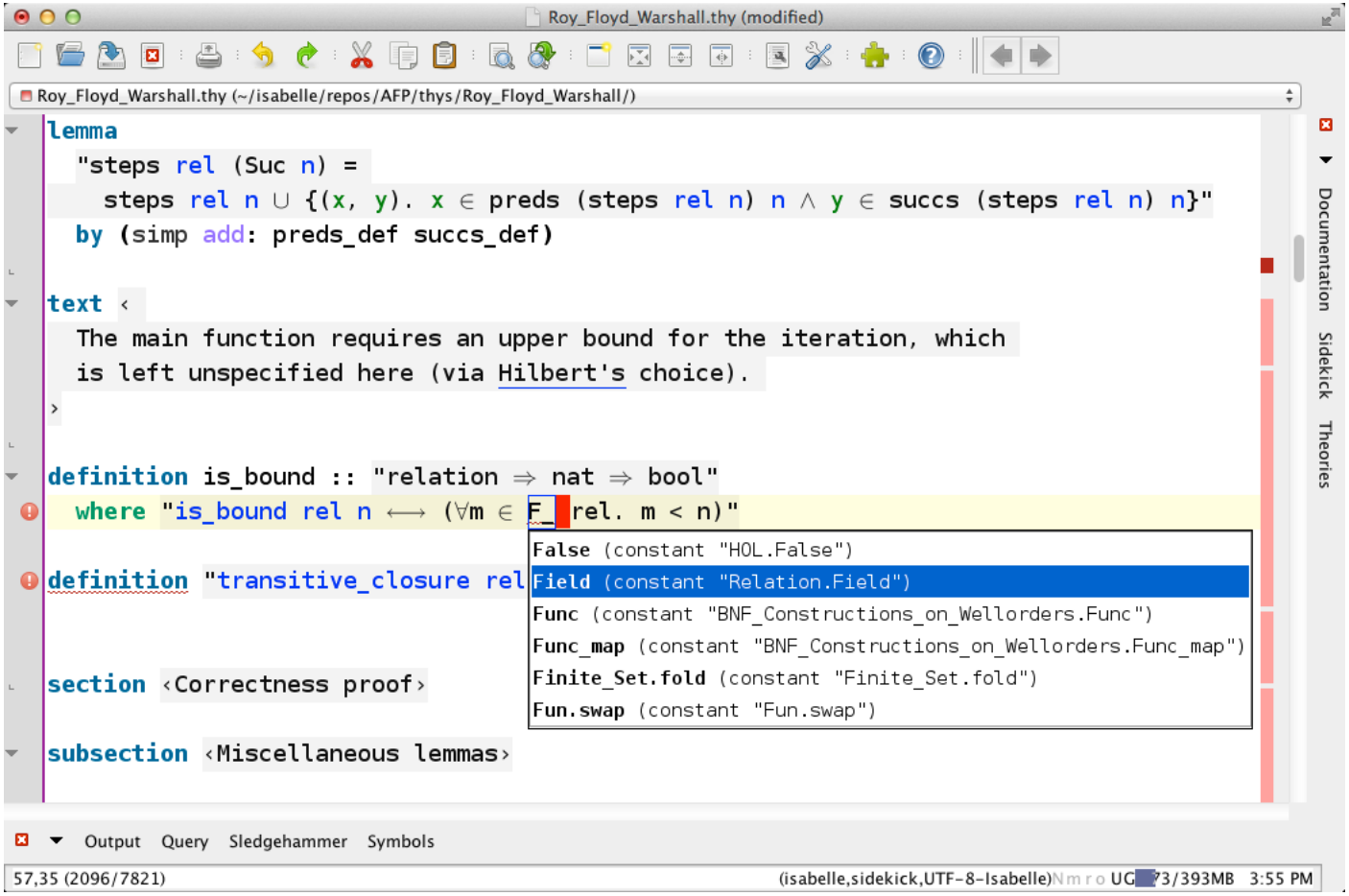
Syntactic completion:

- built-in templates (quotations, antiquotations)
- outer syntax keywords
- Isabelle symbols (--> vs. $\longrightarrow$)

Semantic completion:

- name-space entries
- file-system paths
- spell-checking

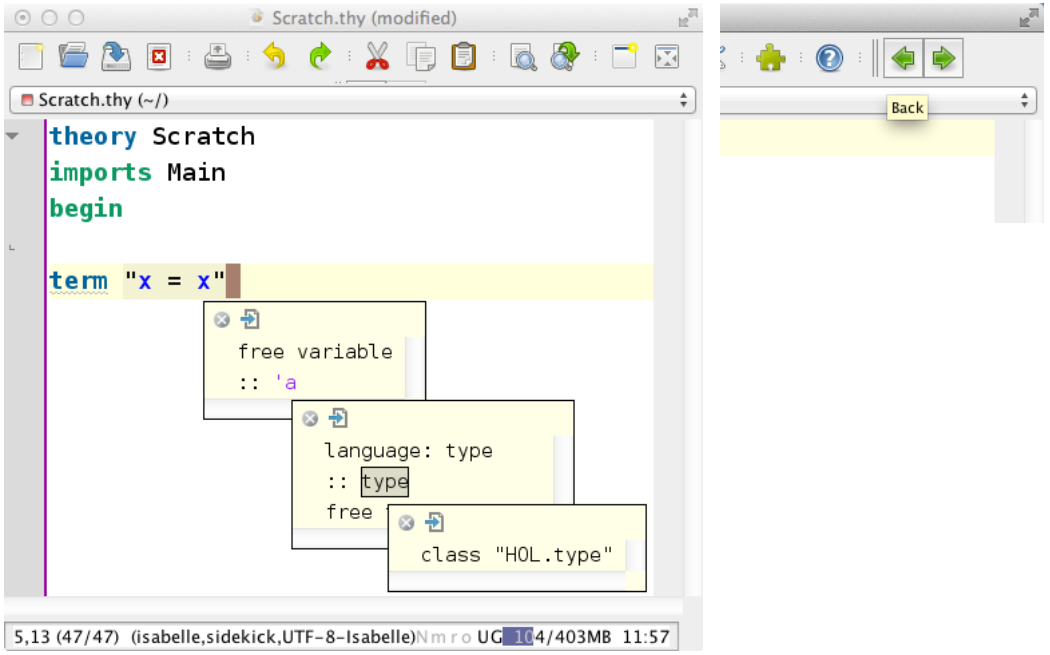
Example: spell-checking and name completion



Editor navigation

Editor navigation

Approach:
browsing PIDE document content via tooltips and hyperlinks



Auxiliary files within the document model

Document blobs

Principle: IDE manages collection of sources and results of processing

So far:

- acyclic graph of document nodes (theories)
- linear chain of command spans within each node

Now also:

- collection of document blobs (uninterpreted byte vectors)
- load commands refer to blobs in value-oriented manner:
  no direct file-system access

Applications in Isabelle/jEdit

Isabelle/ML:

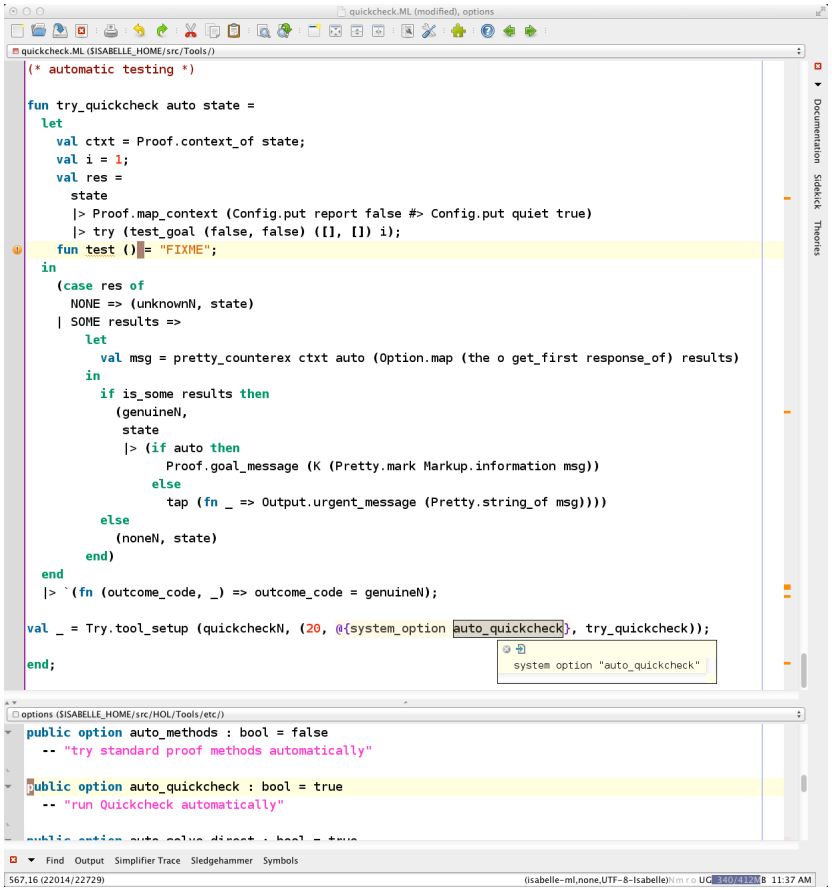
- load command **ML_file** for `.ML` files
- seamless editing of theories together with ML modules
- special tricks (via asynchronous print functions) to allow editing of Isabelle/HOL with only 4 GB

Official Standard ML:

- load command **SML_file** for `.sml` and `.sig` files
- commands **SML_import**, **SML_export** for exchange of toplevel ML bindings

$\longrightarrow$ IDE for SML'97, with or without Isabelle/ML library

Example: editing Isabelle/HOL ML tools



Conclusions

Conclusions

PIDE in 2014:

- Numerous increments towards full-scale Prover IDE.
- More and more integration of development in the environment of Isabelle: theories, documents, add-on tools, embedded languages.
- The more it advances, the higher the ambitions.

Ultimate challenge:

Introducing genuine interaction into ITP

- many conceptual problems
- many technical problems
- many social problems