Paral-ITP Front-End Technologies

Makarius Wenzel
Univ. Paris-Sud, LRI

July 2014

Project **Paral-ITP** meeting

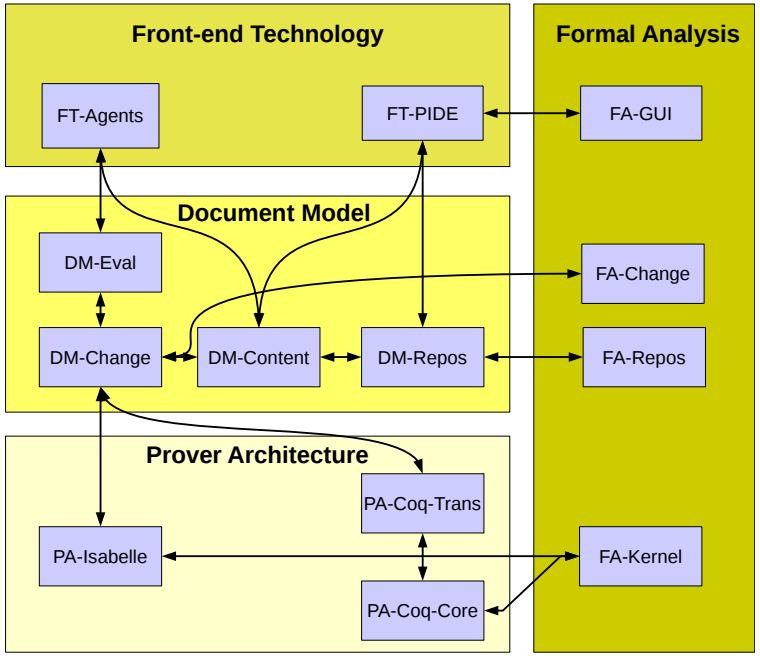
ANR-11-INSE-001

Overview

Relevant work packages

FT-PIDE: Concrete Prover IDE implementations
(integration of prover with editor front-end)

FT-Agents: Tools for augmented prover interaction
(automated provers and disprovers, add-on tools and languages)



Papers and presentations (VSL 2014)

Conference papers:

- M. Wenzel: Asynchronous User Interaction and Tool Integration in Isabelle/PIDE. ITP 2014, Springer LNCS 8558.
- D. Matichuk, M. Wenzel, T. Murray: An Isabelle Proof Method Language. ITP 2014, Springer LNCS 8558.

Workshop presentations:

- M. Wenzel: Tutorial: Isabelle/jEdit for seasoned Isabelle users. Isabelle Workshop 2014, Vienna, July 2014.
- M. Wenzel: Isabelle/jEdit NEWS. Isabelle Workshop 2014, Vienna, July 2014.
- M. Wenzel: System description: Isabelle/jEdit in 2014. UITP 2014, Vienna, July 2014.

Releases

Isabelle2013-2 (December 2013)

- Relaunch of Isabelle2013-1 (November 2013).
- Too little serious testing by actual users!

Isabelle2014 (August 2014)

- `http://isabelle.in.tum.de/website-Isabelle2014-RC0`
- final deliverable for Paral-ITP PA/FT
- PIDE as "filthy rich client" application
- syntactic and semantic completion, including spell-checking
- editor navigation (like web browser)
- auxiliary files within the document model: Isabelle/ML, SML'97
- systematic nesting of sub-languages via text cartouches:
  $\langle funny\ \langle quotes\ \langle with \rangle\ arbitrary\ \langle nesting \rangle \rangle \rangle$
- improved Windows support (MikTeX)

Asynchronous print functions

Asynchronous print functions

Background: READ-EVAL-PRINT within PIDE document-model

Observations:

- cumulative PRINT operations consume more time than EVAL
- PRINT depends on user perspective
- PRINT may diverge or fail
- PRINT augments results without changing proof state
- many different PRINTs may be run independently

Approach:

- manage forked PRINT tasks, depending on document perspective
- optional persistence within document-model
- optional arguments for PRINT: document overlays

Applications in Isabelle/jEdit

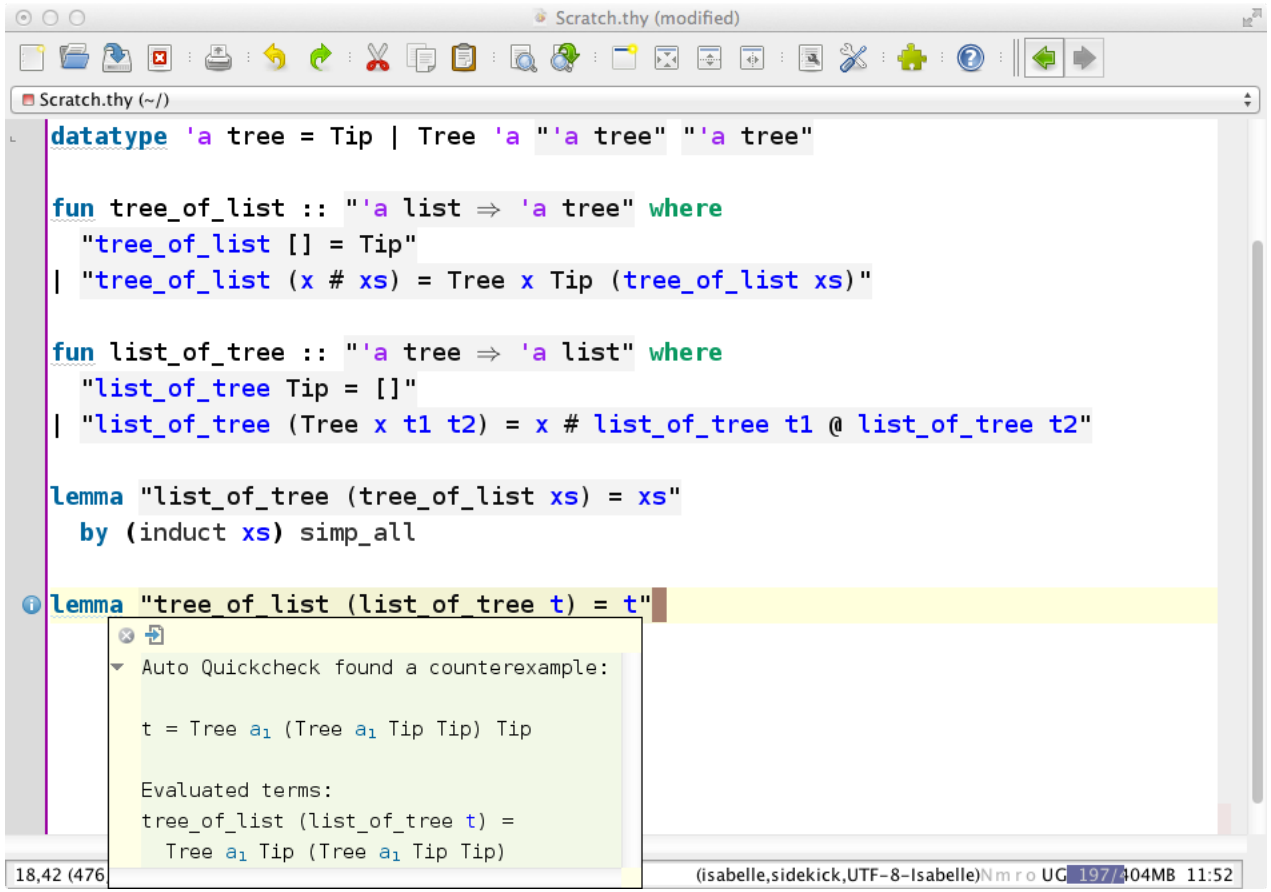
Implicit print functions:

- print proof state
  (printing proof states is slower than most proof steps)
- automatically tried tools
  (provers, disprovers, counter-examples)

Document overlays:

- Query panel
  (find theorems, find constants, print context)
- Sledgehammer panel
  (conventional GUI front-end for asynchronous tool integration)

Example: automatically tried tools



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

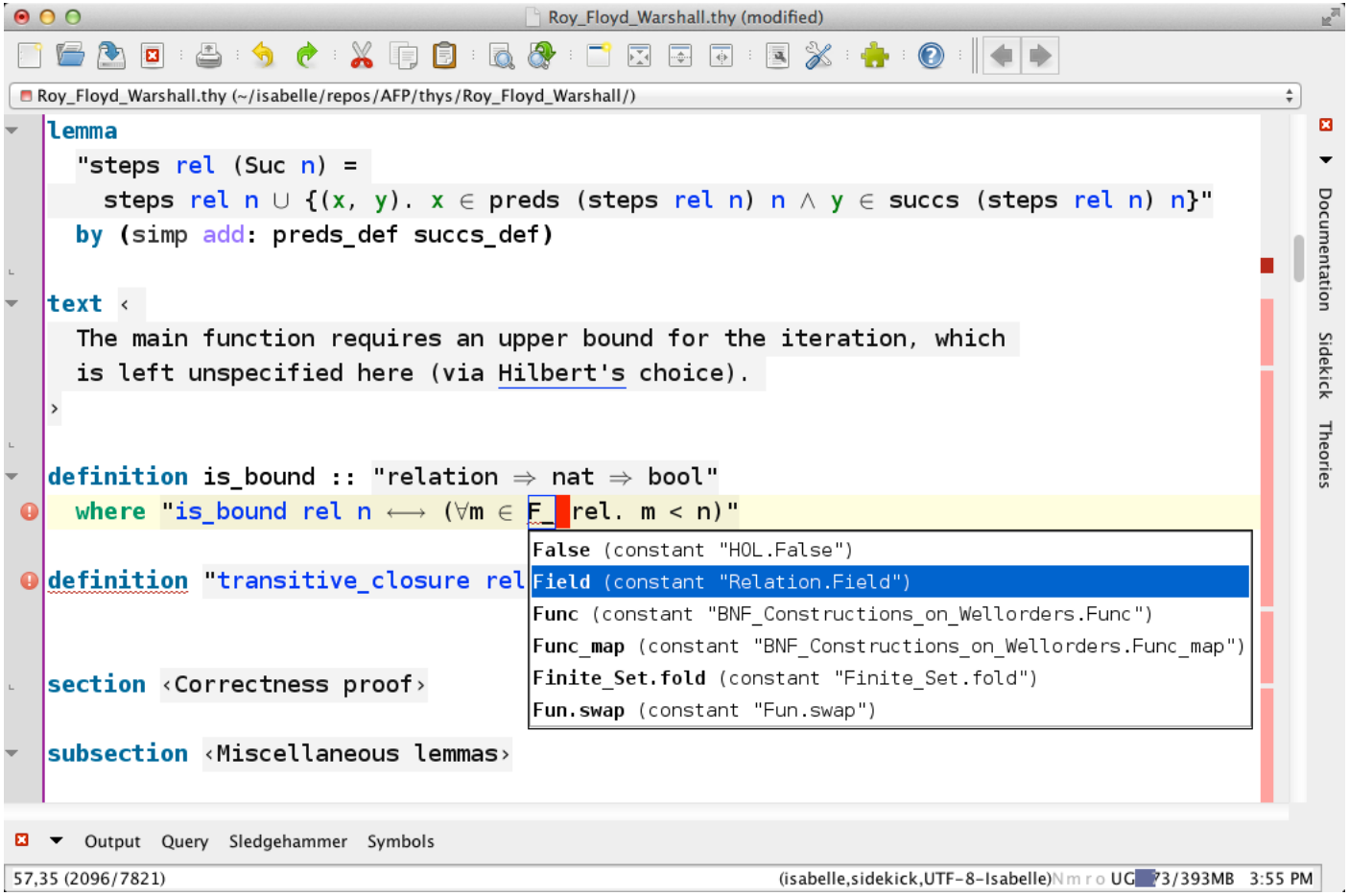
Syntactic completion:

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

Semantic completion:

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

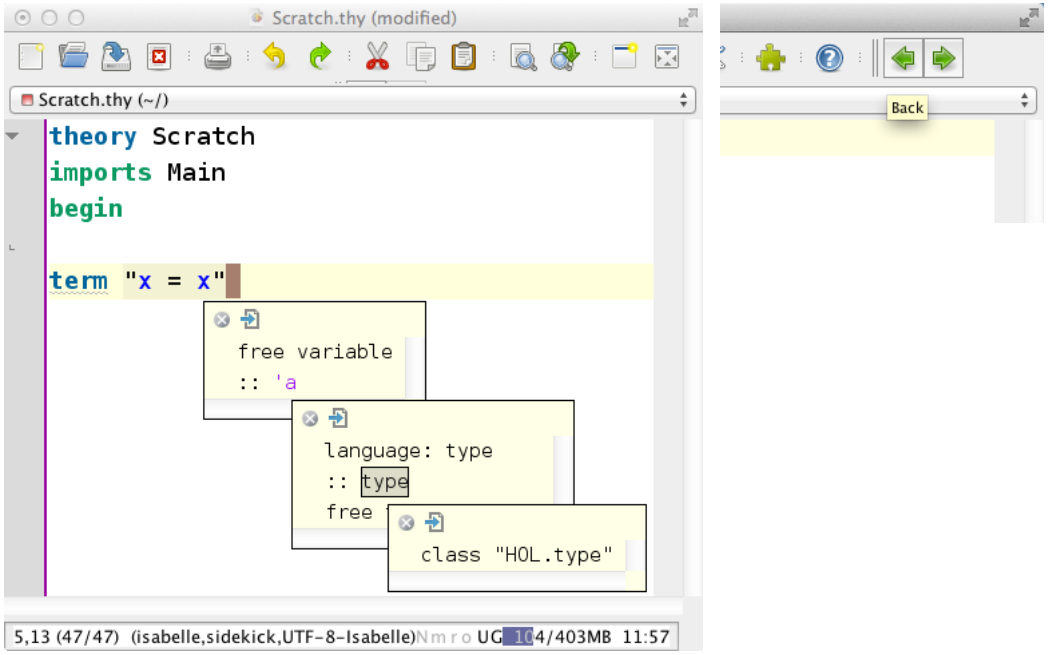
Example: spell-checking and name completion



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:

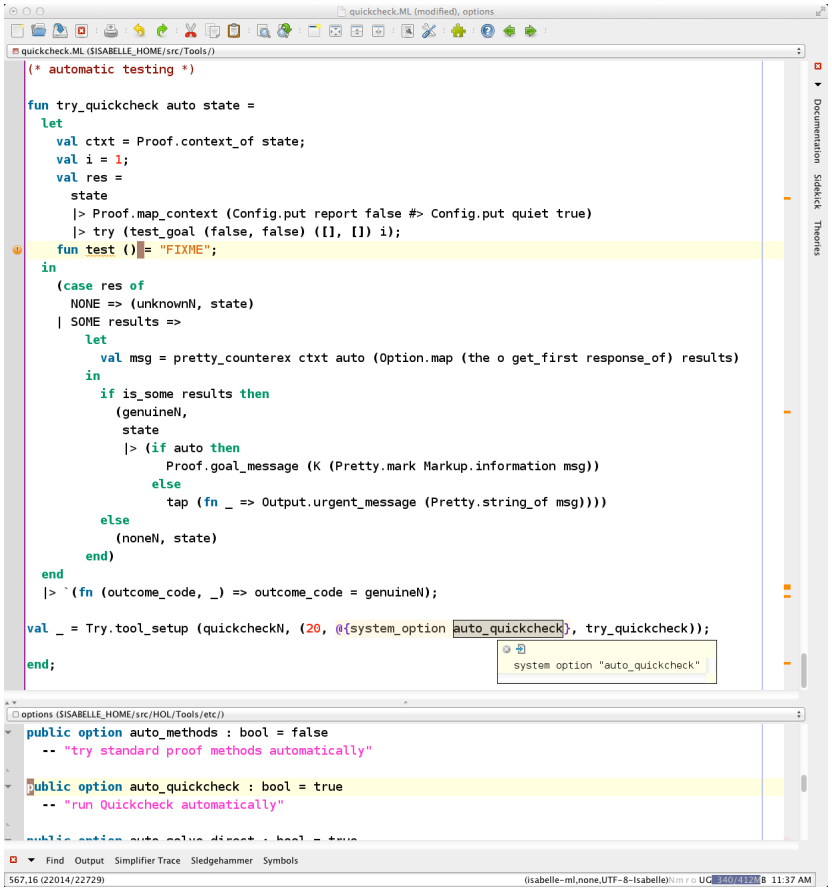
- load command **ML_file** for `.ML` files
- seamless editing of theories together with ML modules
- special tricks (via asynchronous print functions) to allow editing of Isabelle/HOL with only 4 GB

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