Isabelle/jEdit as IDE for domain-specific formal languages and informal text documents

Makarius Wenzel
http://sketis.net

June 2018

Abstract

Isabelle/jEdit is the main application of the Prover IDE (PIDE) framework and the default user-interface of Isabelle, but it is not limited to theorem proving. This presentation explores possibilities to use it as a general IDE for formal languages that are defined in user-space, and embedded into informal text documents. It covers overall document structure with auxiliary files and document antiquotations, formal text delimiters and markers for interpretation (via control symbols). The ultimate question behind this: How far can we stretch a plain text editor like jEdit in order to support semantic text processing, with support by the underlying PIDE framework?

https://sketis.net/wp-content/uploads/2018/05/isabelle-jedit-fide2018.pdf

Introduction

Isabelle — a framework of domain-specific formal languages

Logic:

Isabelle/Pure: Logical framework and bootstrap environment

Isabelle/HOL: Theories and tools for applications

Programming:

Isabelle/ML: Tool implementation (Poly/ML)

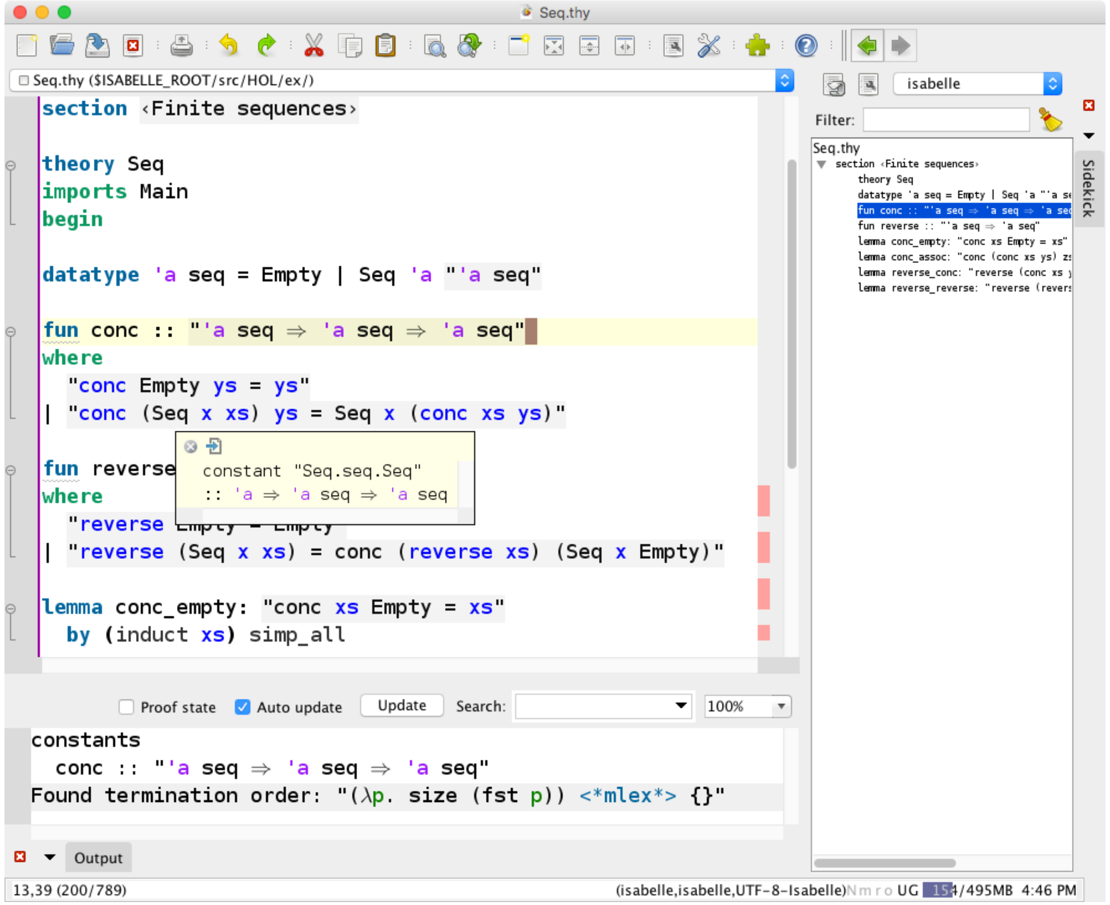
Isabelle/Scala: System integration (JVM)

Proof:

Isabelle/Isar: Intelligible semi-automated reasoning

Document language: $\LaTeX$ type-setting of proof text

Isabelle/jEdit Prover IDE



Seq.thy ($ISABELLE_ROOT/src/HOL/ex/)

```
section ‹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)"

fun reverse
where
  "reverse Empty = Empty
| "reverse (Seq x xs) = conc (reverse xs) (Seq x Empty)"

lemma conc_empty: "conc xs Empty = xs"
  by (induct xs) simp_all
```

```
constant "Seq.seq.Seq"
:: 'a ⇒ 'a seq ⇒ 'a seq
```

Filter:

```
Seq.thy
▼ section ‹Finite sequences›
    theory Seq
    datatype 'a seq = Empty | Seq 'a "'a se
    fun conc :: "'a seq ⇒ 'a seq ⇒ 'a seq
    fun reverse :: "'a seq ⇒ 'a seq"
    lemma conc_empty: "conc xs Empty = xs"
    lemma conc_assoc: "conc (conc xs ys) zs
    lemma reverse_conc: "reverse (conc xs y
    lemma reverse_reverse: "reverse (revers
```

☐ Proof state ☑ Auto update Search: 100%

```
constants
  conc :: "'a seq ⇒ 'a seq ⇒ 'a seq"
Found termination order: "(λp. size (fst p)) <*mlex*> {}"
```

13,39 (200/789) (isabelle,isabelle,UTF-8-Isabelle)N m r o UG 154/495MB 4:46 PM

- asynchronous interaction
- continuous checking
- parallel processing
- scalable applications

Isabelle documents

Document text structure

Markup

- section headings (6 levels like in HTML):
  chapter, **section**, **subsection**, . . . , **subparagraph**
- text blocks: **text**, **txt**, **text_raw**
- raw LaTeX macros (rare)

Markdown

- implicit paragraphs and lists: itemize, enumerate, description

Formal comments

- marginal comments: — $\langle \mathit{text} \rangle$
- canceled text: ***cancel*** $\langle \mathit{text} \rangle$ e.g. $\cancel{b}\cancel{a}\cancel{d}$
- raw LaTeX: ***latex*** $\langle \mathit{text} \rangle$ e.g. $\lim_{n \to \infty} \sum_{i=0}^{n} q^i$

Document antiquotations

full form: `@{`$name\ [options]\ arguments\ \ldots$`}`

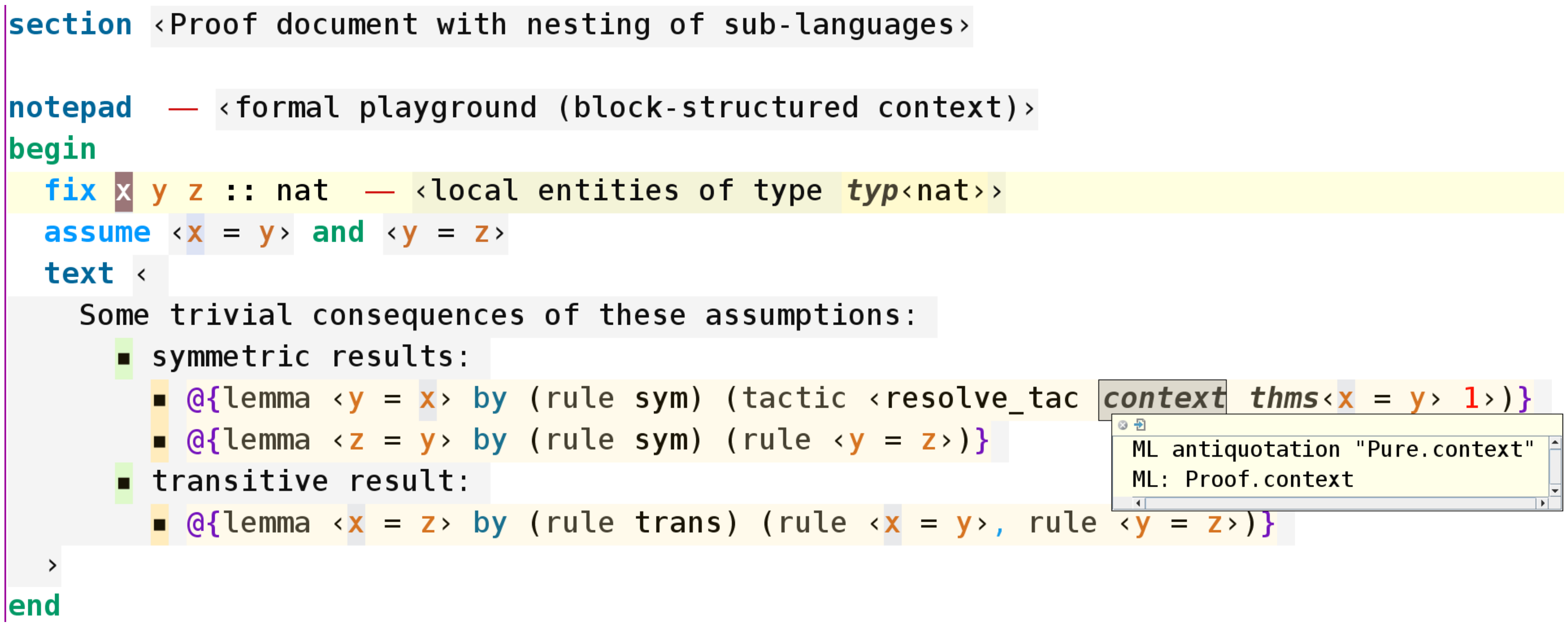
short form:

1. cartouche argument: `\<^`$name$`>`$\langle argument\rangle$
2. implicit standard name: $\langle argument\rangle$
3. no argument: `\<^`$name$`>`

Notable examples:

- *bold*, *emph*, *verbatim*, *footnote*: text styles (with proper nesting)
- *noindent*, *smallskip*, *medskip*, *bigskip*: spacing
- *cite*: formal BibTEX items
- *path*, *file*, *dir*, *url*, *doc*: system resources
- *cartouche*, *theory_text*: self-presentation of Isar
- *action*: jEdit action (interaction)

Example: document with nested sub-languages



```
section ‹Proof document with nesting of sub-languages›

notepad  — ‹formal playground (block-structured context)›
begin
  fix x y z :: nat  — ‹local entities of type typ‹nat››
  assume ‹x = y› and ‹y = z›
  text ‹
    Some trivial consequences of these assumptions:
      ▪ symmetric results:
        ▪ @{lemma ‹y = x› by (rule sym) (tactic ‹resolve_tac context thms‹x = y› 1›)}
        ▪ @{lemma ‹z = y› by (rule sym) (rule ‹y = z›)}
      ▪ transitive result:
        ▪ @{lemma ‹x = z› by (rule trans) (rule ‹x = y›, rule ‹y = z›)}
  ›
end
```

```
ML antiquotation "Pure.context"
ML: Proof.context
```

- markup commands: **section**, **text**
- markdown lists: itemize
- formal comments: — $\langle \mathit{text} \rangle$
- document antiquotations: $@\{\mathit{lemma}\}$

Isabelle/PIDE: Prover IDE

Prover IDE components

Isabelle/jEdit:

- filthy rich client: requires 4–8 GB memory, 2–4 CPU cores
- main example application of the PIDE framework
- default user-interface for Isabelle

Isabelle/PIDE:

- general framework for Prover IDEs based on Scala
- with parallel and asynchronous document processing

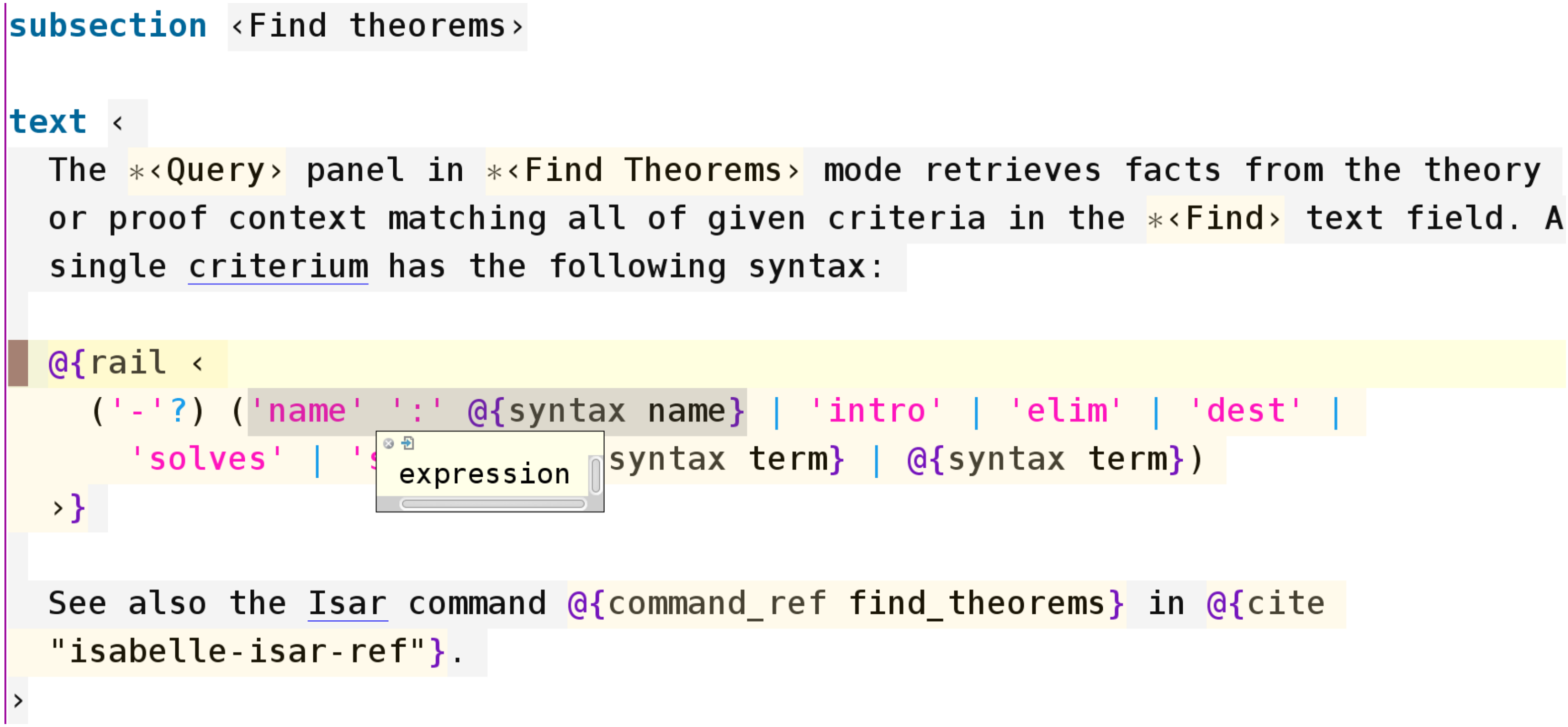
Scala/JVM: http://www.scala-lang.org

- higher-order functional-object-oriented programming

jEdit: http://www.jedit.org

- sophisticated text editor implemented in Java

Example: rail syntax diagrams



```
subsection ‹Find theorems›

text ‹
  The ∗‹Query› panel in ∗‹Find Theorems› mode retrieves facts from the theory
  or proof context matching all of given criteria in the ∗‹Find› text field. A
  single criterium has the following syntax:

  @{rail ‹
    ('-'?) ('name' ':' @{syntax name} | 'intro' | 'elim' | 'dest' |
      'solves' | 's              syntax term} | @{syntax term})
  ›}

  See also the Isar command @{command_ref find_theorems} in @{cite
  "isabelle-isar-ref"}.
›
```

Source: $ISABELLE_HOME/src/Doc/JEdit/JEdit.thy

PIDE document structure (1)

Project directories (tree set): e.g. `Isabelle`, `AFP`

- explicit sub-directories in `ROOTS` files
- explicit session entries in `ROOT` files (reachable set)

Sessions (acyclic graph): e.g. `HOL`, `HOL-Analysis`, `HOL-SPARK`

- options, theories, document files
- potentially a dumped-world image

Theories (acyclic graph): e.g. `Main`, `HOL-Analysis.Lipschitz`

- header **theory** A **imports** $B_1 \ldots B_n$ **begin**
- command keywords (outer syntax)
- arbitrary theory data (ML)

PIDE document structure (2)

Commands (sequence):

- regular commands, e.g. **ML** $\langle val\ a = 1\rangle$ or **definition** $\langle c = t\rangle$ or **lemma** $\langle\varphi\rangle$ **by** $proof\_method$
- load commands, e.g. **ML_file** $\langle a.ML\rangle$

Auxiliary files: path argument to load command

- front-end: management of edits
- back-end: processing of content

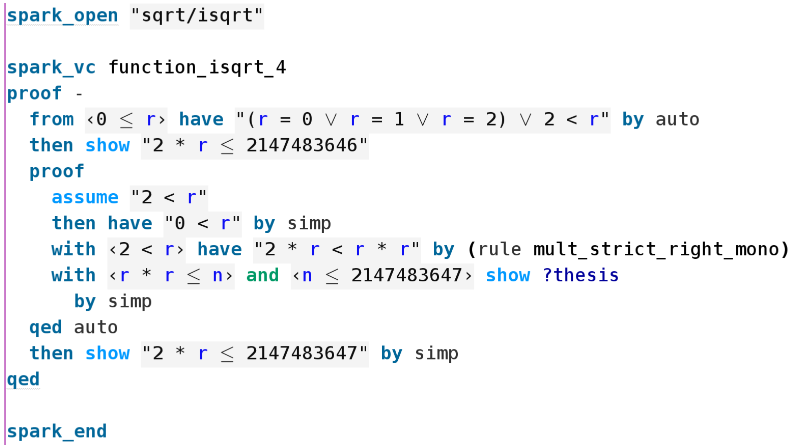
Typical applications: user-defined languages in . . .

1. text cartouche for regular command, e.g. **ML** $\langle val\ a = 1\rangle$
2. text file for load command, e.g. **ML_file** $\langle a.ML\rangle$

Example: Isabelle/HOL-SPARK

Author: Stefan Berghofer, secunet Security Networks AG

- project directory: `Isabelle`
- sessions: `HOL-SPARK`, `HOL-SPARK-Examples`, `HOL-SPARK-Manual`
- theories: e.g. `$ISABELLE_HOME/src/HOL/SPARK/Examples/Sqrt/Sqrt.thy`
- commands: **spark_open**, **spark_vc**, **spark_end**
- auxiliary files: `.siv`, `.fdl`, `.rls` files from external tools (SPARK Examiner and Simplifier by Altran Praxis, Ltd)



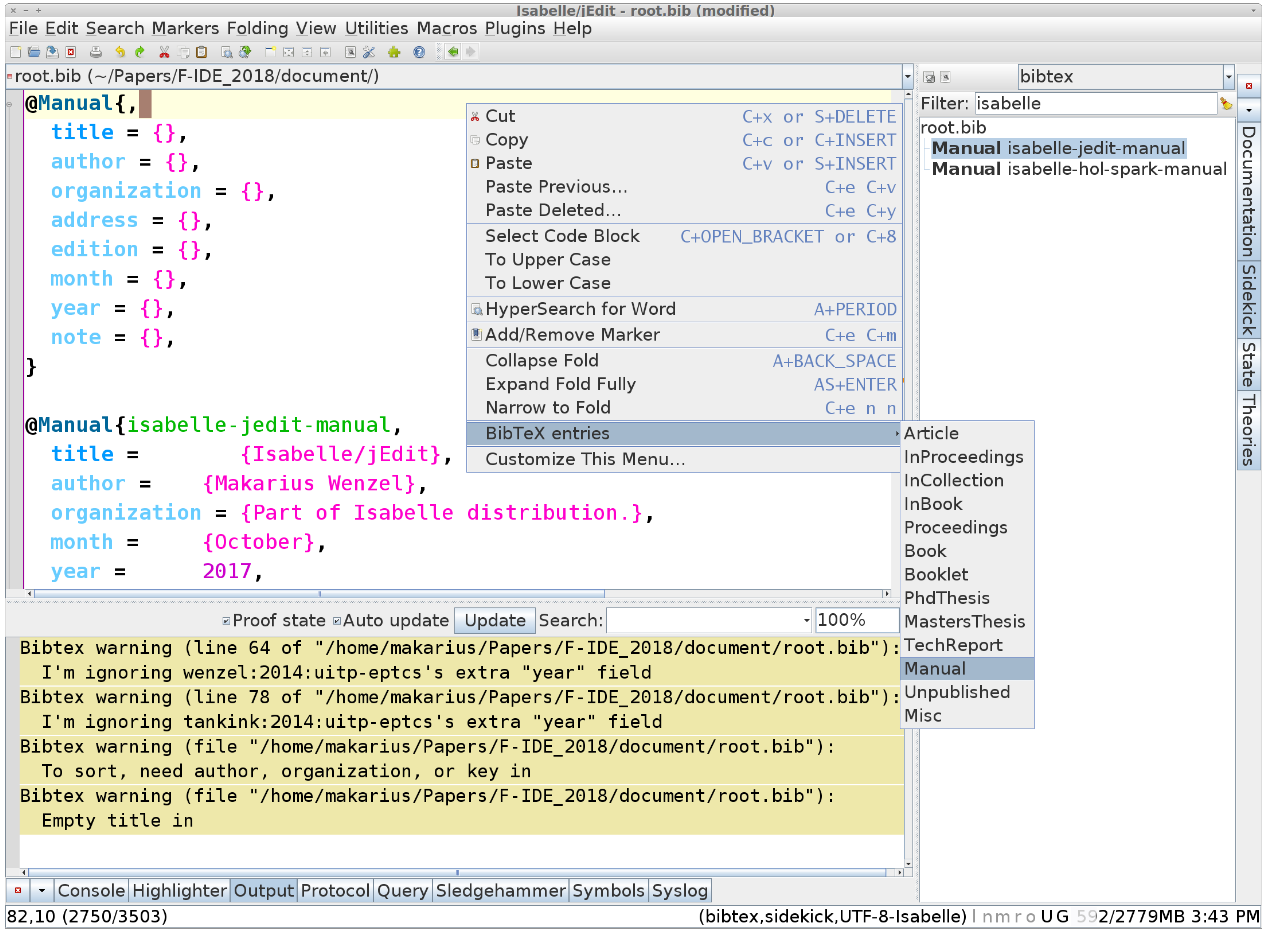
```
spark_open "sqrt/isqrt"

spark_vc function_isqrt_4
proof -
  from ‹0 ≤ r› have "(r = 0 ∨ r = 1 ∨ r = 2) ∨ 2 < r" by auto
  then show "2 * r ≤ 2147483646"
  proof
    assume "2 < r"
    then have "0 < r" by simp
    with ‹2 < r› have "2 * r < r * r" by (rule mult_strict_right_mono)
    with ‹r * r ≤ n› and ‹n ≤ 2147483647› show ?thesis
      by simp
  qed auto
  then show "2 * r ≤ 2147483647" by simp
qed

spark_end
```

Application: BibTeX IDE

Screenshot



Isabelle/jEdit - root.bib (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
root.bib (~/Papers/F-IDE_2018/document/)
bibtex
Filter: isabelle
root.bib
Manual isabelle-jedit-manual
Manual isabelle-hol-spark-manual
Cut C+x or S+DELETE
Copy C+c or C+INSERT
Paste C+v or S+INSERT
Paste Previous... C+e C+v
Paste Deleted... C+e C+y
Select Code Block C+OPEN_BRACKET or C+8
To Upper Case
To Lower Case
HyperSearch for Word A+PERIOD
Add/Remove Marker C+e C+m
Collapse Fold A+BACK_SPACE
Expand Fold Fully AS+ENTER
Narrow to Fold C+e n n
BibTeX entries
Customize This Menu...
Article
InProceedings
InCollection
InBook
Proceedings
Book
Booklet
PhdThesis
MastersThesis
TechReport
Manual
Unpublished
Misc
Documentation Sidekick State Theories
☑Proof state ☑Auto update Update Search: 100%
Console Highlighter Output Protocol Query Sledgehammer Symbols Syslog
82,10 (2750/3503)
(bibtex,sidekick,UTF-8-Isabelle) I n m r o U G 592/2779MB 3:43 PM

```
@Manual{,
  title = {},
  author = {},
  organization = {},
  address = {},
  edition = {},
  month = {},
  year = {},
  note = {},
}

@Manual{isabelle-jedit-manual,
  title =        {Isabelle/jEdit},
  author =    {Makarius Wenzel},
  organization = {Part of Isabelle distribution.},
  month =      {October},
  year =      2017,
```

```
Bibtex warning (line 64 of "/home/makarius/Papers/F-IDE_2018/document/root.bib"):
  I'm ignoring wenzel:2014:uitp-eptcs's extra "year" field
Bibtex warning (line 78 of "/home/makarius/Papers/F-IDE_2018/document/root.bib"):
  I'm ignoring tankink:2014:uitp-eptcs's extra "year" field
Bibtex warning (file "/home/makarius/Papers/F-IDE_2018/document/root.bib"):
  To sort, need author, organization, or key in
Bibtex warning (file "/home/makarius/Papers/F-IDE_2018/document/root.bib"):
  Empty title in
```

Features

- jEdit syntax mode for `.bib` files (BibTeX parser in Isabelle/Scala)
- editor text folds according block structure
- tree-view in the `SideKick` panel (with simple filter)
- context-menu for BibTeX entry types
- syntax highlighting for BibTeX entry fields
- HTML preview similar to LaTeX output
- text antiquotation $@\{cite\}$ with semantic completion and strict checking against `.bib` files in batch-mode
- soft semantic checking of `.bib` files by original `bibtex` tool, with authentic warnings, errors

Implementation: semantic checking

Approach:

- editor: opening file `foo.bib` creates implicit theory context, with load command **bibtex_file** $\langle foo.bib\rangle$
- command **bibtex_file**: Isabelle/ML function invokes Isabelle/Scala method `Bibtex.check_database()` via PIDE protocol
- `Bibtex.check_database()`: precise source positions for tokens, placed on individual lines for `bibtex` input
- scanning BibTeX log for warnings and errors: line positions become token index $\leadsto$ precise source positions with PIDE markup

Conclusions:

1. may pretend that Isabelle understands BibTeX semantics
2. may pretend that BibTeX understands PIDE markup protocol

Conclusions

History and related work

PIDE vs. Proof General Emacs:

- 1998/1999: starting Proof General for Isabelle/Isar
- 2008: thinking beyond the model of “proof scripting”
- 2014: fully native Isabelle/PIDE, no support for Proof General
- Coq is the only remaining Proof General back-end

PIDE vs. mainstream IDEs: e.g. Eclipse, IntelliJ IDEA

- similar in deep checking and rich markup
- dissimilar in built-in functional evaluation model

Isabelle/jEdit 10.0 vs. Isabelle/VSCode 1.0

Isabelle/jEdit: “game engine”

- scalable application
- Java with Swing GUI
- multiple threads
- simple text buffer model
- free-form layered painting (Graphics2D)

Isabelle/VSCode: “smart text editor”

- minimal experiment
- JavaScript with HTML/CSS
- cooperative multitasking
- rich text buffer model
- restricted text decoration model (CSS)

Future work (after 10 years of PIDE)

PIDE technology:

- dynamic session management
- PDF-LaTeX document preparation
- HTML/CSS preview in real-time and high quality

PIDE sociology:

- improve visibility outside of Isabelle community
- motivate tool builders to re-use the Isabelle/PIDE platform