Isabelle technology for the Archive of Formal Proofs with application to MMT

Makarius Wenzel, Augsburg
https://sketis.net

July 2019

Motivation: scalability for Isabelle/AFP

The Archive of Formal Proofs

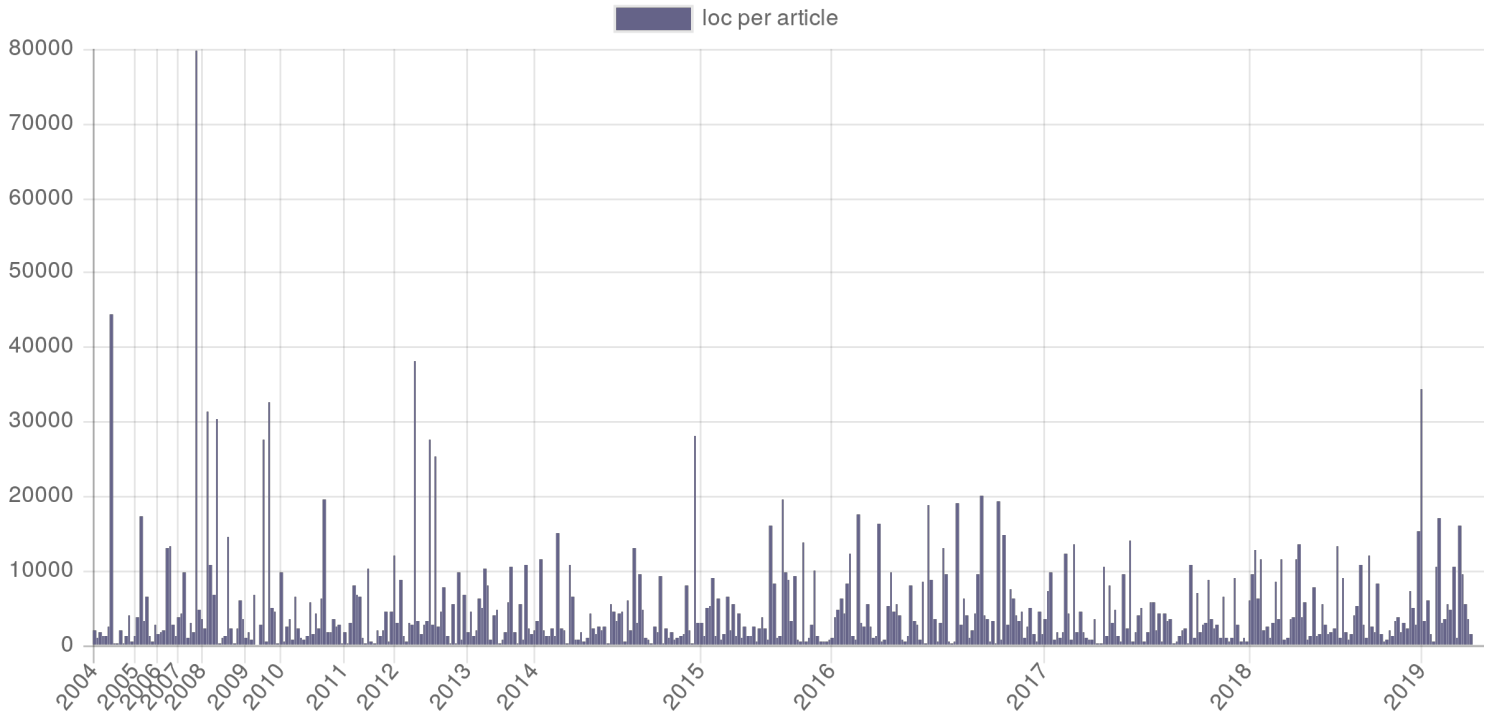
AFP: https://www.isa-afp.org

- repository of formalized mathematics: checked by Isabelle
- scientific journal: reviewed by 5 human editors

Maintenance model:

- everything should always work (most of the time)
- Isabelle changes are pushed through to AFP applications
- demand for fast feedback from build jobs

AFP articles: text size vs. date of first appearance



Practical time scales

Online time: max. 45min ("Paris commuter's constant")

Offline time: max. 2h ("French lunch break")

AFP timing: 8 processes × 8 threads, excluding `very_slow`

- Isabelle with `main` only:
  7.5min elapsed time, 53min CPU time (factor 7.0)
- AFP without `slow` / `large`:
  51min elapsed time, 25h47 CPU time (factor 30.3)
- AFP with `slow` / `large` only:
  50min elapsed time, 12h04 CPU time (factor 14.5)
- Isabelle + AFP:
  1h14 elapsed time, 42h11 CPU time (factor 34.2)

Isabelle technology

What is Isabelle?

Originally: by Larry Paulson (1986/1989)

- logical framework (LF)
- generic proof assistant, e.g. for CTT, FOL, ZF, HOL

After 30 years of evolution:

- software technology for large libraries of formal mathematics
- many sub-systems with "Isabelle/XYZ" naming scheme, e.g.

  Isabelle/HOL: application logic with theories and tools
  Isabelle/ML: internal tool implementation language
  Isabelle/Scala: external system integration language
  Isabelle/PIDE: Prover IDE framework for semantic interaction

Isabelle/HOL

- old-fashioned logic (Church 1940, Gordon 1985)
- highly successful in applications
- classic set-theory with simple types
- many derived mechanisms for specifications and proofs

Note:

- Isabelle/HOL is classic mathematics, not programming
  (but: tutorial "Programming and proving in Isabelle/HOL")
- Isabelle/HOL extensions usually implemented in Isabelle/ML
  ("LCF approach")

Isabelle/ML

- based on Poly/ML: David Matthews (1985)
- rich Isabelle/ML library
- high-end IDE, e.g. for Isabelle itself via `~~/src/Pure/ROOT.ML`
- source-level debugger

Main technology: scalable parallel functional programming

- fast run-time compilation to produce fast machine-code
- shared-memory parallelism (threads/locks or futures)
- stop-the-world garbage collection with internal parallelism
- implicit substructure-sharing of pure values (strings, terms, etc.)
- dumped-world images for fast reloading of semantic state
- compact representation of data on 64 bit hardware:
  32 bit addressing of max. 16 GB heap space

Isabelle/Scala (1)

- based on regular Scala, hosted on Java platform (version 11)
- functional programming style similar to Isabelle/ML
- overlapping parts of libraries with Isabelle/ML

Main technology:

- multi-threaded JVM with parallel garbage collection
- efficient functional programming on the JVM
- access to external databases (notably SQLite, PostgreSQL)
- access to TCP services (notably SSH, HTTP)
- support for Mercurial (the standard SCM for Isabelle + AFP)

Isabelle/Scala (2)

Benefits:

- proper functional programming (with types)
- proper data structures (e.g. acyclic graph for dependencies)
- avoid system "scripts" (e.g. bash, perl, python, ruby)

Isabelle sources:

- Isabelle/Scala: 1.6 MB
- Isabelle/ML/Pure: 2.4 MB

Isabelle/Scala applications:

- Isabelle/jEdit: GUI application (AWT/Swing)
- Isabelle/VSCode: Language Server Protocol server (JSON)
- Isabelle command-line tools (see `isabelle`)

Isabelle/PIDE

- Prover IDE framework, implemented in Scala and ML
- prover as formal document processor (input: edits, output: reports)
- Headless PIDE as interactive object under program control, e.g.
  - export of formal content with access to the internal ML context
  - update of theory sources based on PIDE markup
  - detailed recording of timing information

Applications:

- Isabelle/jEdit: PIDE user-interaction via text editor
- Isabelle/MMT: PIDE document export (OMDoc and RDF/XML)

Application: Isabelle/MMT — OMDoc and RDF/XML from AFP

MMT

MMT: https://uniformal.github.io

- "Meta Meta-Theory" by Michael Kohlhase, Florian Rabe et-al
- OMDoc file-format (based on XML)
- documents with formal, informal, semi-formal content
- MMT sub-projects: importers for various languages
- `mmt.jar`: Scala library with MMT services

Isabelle/MMT

Isabelle/MMT: https://isabelle.sketis.net/Isabelle_MMT_CICM2019

- Isabelle component that incorporates `mmt.jar` into Isabelle/Scala
- command-line tools:
  - `isabelle mmt_build`: build MMT inside Isabelle
  - `isabelle mmt_import`: import content of headless PIDE session into MMT (OMDoc and RDF/XML triples)
  - `isabelle mmt_server`: present imported content via HTTP server of MMT
  - `isabelle mmt`: run interactive MMT shell inside the Isabelle

Implementation of `isabelle mmt`

- command-line arguments like `isabelle build` for selection of sessions (e.g. all of AFP without group `very_slow`)
- headless PIDE session based on Isabelle/Pure (not HOL): provide all session theories as one big edit
- continuous parallel processing of the theory graph
  - finished theories are committed in Scala to produce OMDoc and RDF/XML
  - committed theories are removed eventually (garbage collection)

Resource requirements: for AFP

- ML: 12 CPU cores, 30 GB RAM
- Scala: 2 CPU cores, 30 GB RAM

Isabelle/MMT content

OMDoc content:

- logical foundations: types, consts, facts (but: no proof terms)
- aspects of structured specifications:
  - locales
  - locale interpretations
  - type classes (as locale interpretations)

RDF content:

- Dublin Core Mata data (from formal comments / markers)
- semi-formal document structure (section headings)
- formal status of exported MMT constants
- relations between formal items (e.g. syntactic dependencies)
- physical parameters (e.g. source size, check time)

References

- Andrea Condoluci, Michael Kohlhase, Dennis Müller, Florian Rabe, Claudio Sacerdoti Coen, and Makarius Wenzel: *Relational data across mathematical libraries*, LNAI 11617 (CICM 2019). https://files.sketis.net/CICM-2019-RDF.pdf

Conclusions

Conclusions

Summary:

- Isabelle technology is a mix of several sub-technologies
- Isabelle/Scala can manage all that pretty well
- Isabelle/MMT is a natural application of Isabelle/PIDE,
  to export content from Isabelle/ML via Isabelle/Scala

Future work:

- convergence of batch-builds and PIDE processing
- improved scalability: technology needs to grow with the library