Paral-ITP Front-End Technologies

Makarius Wenzel
Univ. Paris-Sud, LRI

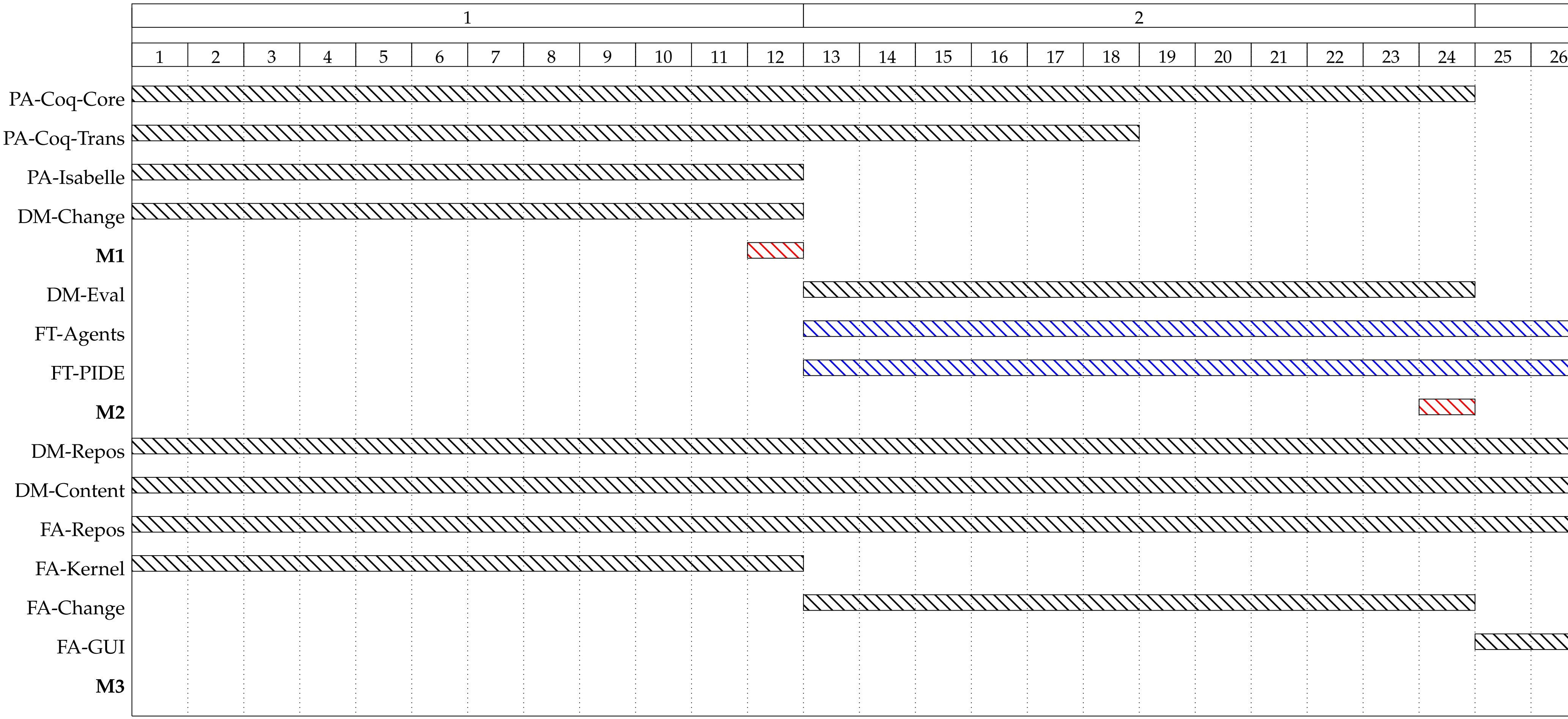
September 2012

AGENCE NATIONALE DE LA RECHERCHE
ANR

Project **Paral-ITP** meeting
ANR-11-INSE-001

Introduction

Le Grand Plan



Isabelle Release

Isabelle2012 (May 2012)

- see `http://isabelle.in.tum.de`
- used in Isabelle/HOL tutorial 14/15-May-2012:
  some experience with fresh users getting exposed to PIDE
- almost routine Windows support:
  already noticeable on `isabelle-users` mailing list
- emerging problem of re-education for Emacs users
  (“HCI” ≈ Habits in Computer Interaction?)

Recent technology upgrades

- Scala 2.9.2 $\surd$
- jEdit 4.5.2 $\surd$ (jEdit 5.0 coming soon)
- Java 1.7.0_06: uniformly on Linux, Mac OS X, Windows $\surd$
- JavaFX 2.2: minimal PIDE integration, some early experiments with HTML5 panel
  (somewhat indirect path: WebKit → JNI → JavaFX → Swing)

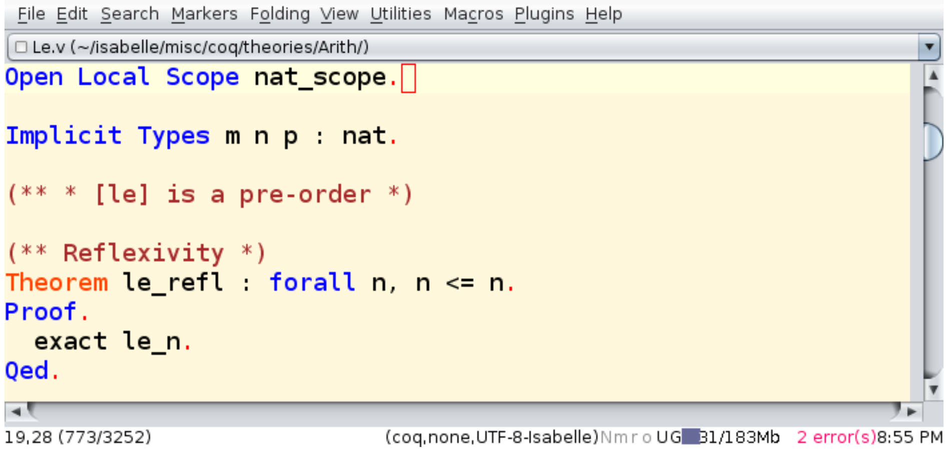
CoqPIDE Experiment
(June 2012)

CoqPIDE approach

- integration of Coq into PIDE, coverage of its main layers
- re-use of CoqIDE architecture to accommodate PIDE
- replacement of Gtk front-end by PIDE document-model operations (internal PIDE protocol in OCaml, using some CoqIDE functions)
- $\approx$ 6 days work, including $\approx$ 2 days struggling with Coq Makefiles
- direct derivative of existing Isabelle sources + some Coq bits

CoqPIDE implementation

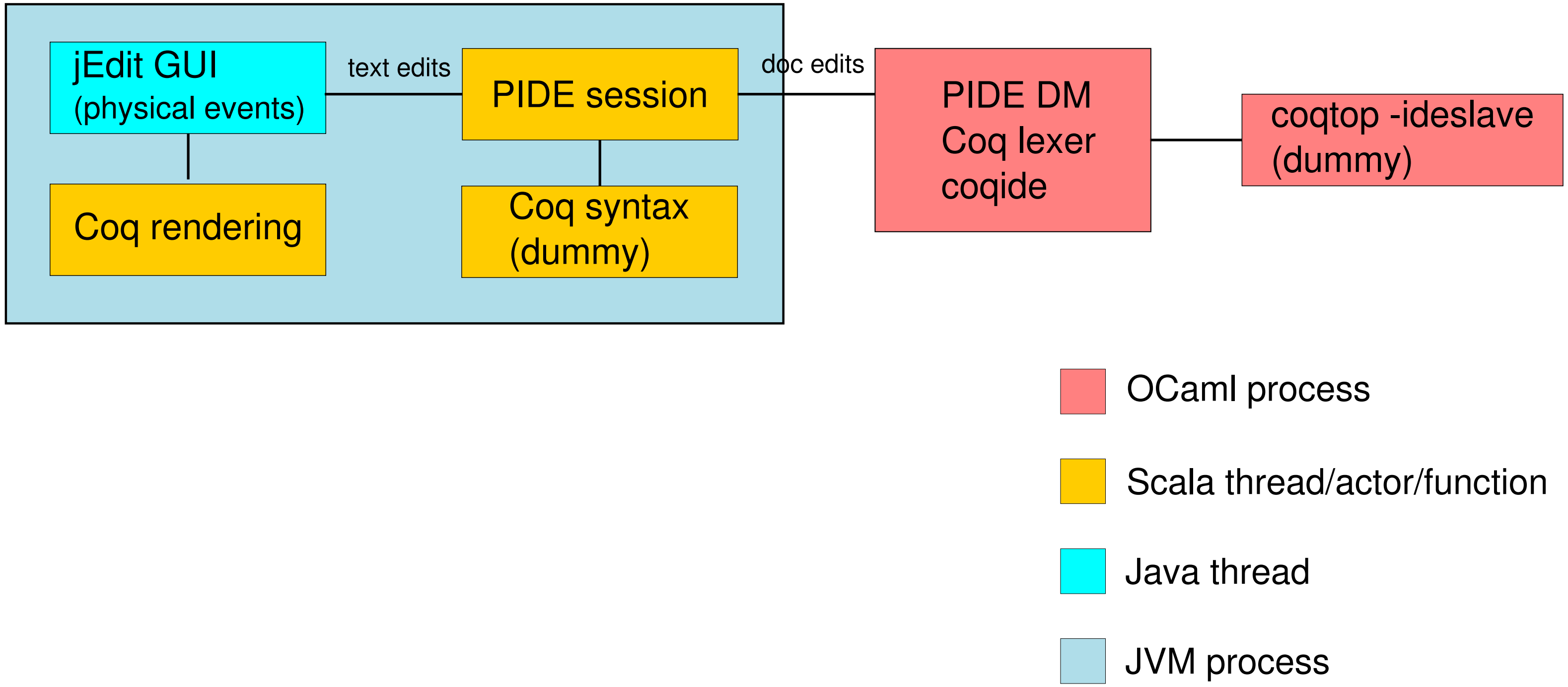
- https://bitbucket.org/makarius/isabelle-coq
  (clone as Isabelle-Coq e.g. revision 07c7fc8ba7bd)
- https://bitbucket.org/makarius/coq-clone
  (clone branch v8.4 e.g. revision 4b806b6e1757)
- See coq-clone/README.PIDE and coq-clone/ide/pide.ml
  (25 kB total; 1.7 kB payload for Coq interpretation of content)


File Edit Search Markers Folding View Utilities Macros Plugins Help
□ Le.v (~/isabelle/misc/coq/theories/Arith/)

```
Open Local Scope nat_scope.

Implicit Types m n p : nat.

(** * [le] is a pre-order *)

(** Reflexivity *)
Theorem le_refl : forall n, n <= n.
Proof.
  exact le_n.
Qed.
```

19,28 (773/3252)
(coq,none,UTF-8-Isabelle)Nmro UG 31/183Mb 2 error(s)8:55 PM

CoqPIDE building blocks



Advances in Isabelle/PIDE
(July + August 2012)

(1) Incremental parsing

Main lessons:

- for usability negative syntax (failure) is more important than positive syntax (success)
- error recovery needs to be modelled explicitly (ancient wisdom of Compiler Construction)
- editor perspective (visible text range) needs to be taken into account for partial parsing (efficiency, reactivity)
- change of perspective (scrolling, open/close of text views) counts as regular edit in the document model
- change of module dependencies (theory headers) affects outer syntax dynamically

(2) Parallel session management

Isabelle build tool:

- outermost hierarchy (tree) for Isabelle sessions (in addition to graph of theories and internal node structure)
- implemented in Isabelle/Scala (module `isabelle.Build`) with additional command-line wrapper (slow startup)
- Live beyond Unix `make` exists!

Application: AFP

Isabelle/AFP:

- $\approx$ 122 sessions with diversity of single-core run-time (3s . . . 1h)
- parameters of fully pervasive parallelism:

8 hardware cores / 16 CPU threads (Intel Xeon with hyperthreading)
4 parallel build jobs (Unix processes)
4 parallel ML worker threads (Isabelle/ML) 4 parallel GC threads (Poly/ML)
parallel theory and proof checking (Isabelle/Isar)

- timing results:

```
Finished LatticeProperties (0:00:15 elapsed time, 0:00:22 cpu time, factor 1.46)
...
Finished JinjaThreads (0:32:59 elapsed time, 1:56:55 cpu time, factor 3.54)
0:36:01 elapsed time, 5:17:18 cpu time, factor 8.80
```

More timings

- 4 processes, 4 threads:

  ```
  Finished LatticeProperties (0:00:15 elapsed time, 0:00:22 cpu time, factor 1.46)
  ...
  Finished JinjaThreads (0:32:59 elapsed time, 1:56:55 cpu time, factor 3.54)
  0:36:01 elapsed time, 5:17:18 cpu time, factor 8.80
  ```

- 2 processes, 8 threads:

  ```
  Finished LatticeProperties (0:00:14 elapsed time, 0:00:23 cpu time, factor 1.64) ...
  Finished JinjaThreads (0:23:04 elapsed time, 2:05:58 cpu time, factor 5.46)
  0:37:50 elapsed time, 5:16:30 cpu time, factor 8.36
  ```

- 1 processes, 8 threads:

  ```
  Finished LatticeProperties (0:00:12 elapsed time, 0:00:18 cpu time, factor 1.50)
  ...
  Finished JinjaThreads (0:18:55 elapsed time, 1:44:08 cpu time, factor 5.50)
  1:04:50 elapsed time, 4:26:18 cpu time, factor 4.10
  ```

- 1 processes, 1 threads:

  ```
  Finished LatticeProperties (0:00:15 elapsed time, 0:00:14 cpu time, factor 0.93) ...
  Finished JinjaThreads (1:06:34 elapsed time, 1:06:33 cpu time, factor 0.99)
  3:20:47 elapsed time, 3:20:47 cpu time, factor 1.00
  ```

(3) Interactive parallel proof checking

History:

- Spring 2008: routine parallel theory and proof checking in batch mode (Isabelle2008 release)
- Summer 2008: commencing PIDE implementation for asynchronous interaction, starting to get beyond speculative talks on workshops and conferences
- Spring 2012: consolidated stable version of asynchronous Isabelle/PIDE (Isabelle2012 release)
- Summer 2012:
  ```
  Tue Sep 04 00:16:03 changeset 61e222517d06
  enable parallel terminal proofs in interaction;
  ```

Some lessons learned

- Human interaction is 1 or 2 orders of magnitude more difficult to handle than batch processing: "monkey at keyboard" produces erratic events, and expects instantaneous feedback in real-time.

- Classic OO-model of text editor (jEdit) poses problems: threads and locks over mutable object state and arrays inadequate for efficient parallel + interactive processing.

  PIDE approach: physical state is detached via true mathematical model in the background, which can travel through parallel time and space efficiently (and correctly).

- Parallel Poly/ML (David Matthews) with its emphasize on large heaps of mainly immutable data greatly helps on the prover side.