

Designing a state transaction machine for Coq

Bruno Barras & *Enrico Tassi*



12 Aug 2012 — Princeton

Roadmap

- 1 The Paral-ITP project
- 2 A State Transaction Machine for Coq
- 3 Demo
- 4 Future work

Paral-ITP in a nutshell

This work has been done in the context of the Paral-ITP project:

- Consortium: LRI, INRIA, CNAM
- Peculiarity: Coq & Isabelle
- Buzzword: parallelization
- Inspiration: Isabelle/Jedit (Wenzel)

Objectives on the Coq side:

- Get rid of the read/eval/print loop:
 - rethink the execution flow of a document
- Towards a document centric prover:
 - design a “document model” /editing API
- Take profit:
 - asynchronous flow between the GUI and the prover
 - parallel processing of independent tasks



Getting rid of the read/eval/print loop

The loop:

- line $n + 1$ requires line n to be fully evaluated

Breaking it:

- `Proof ... Qed` blocks generate opaque proof terms
- opaque proofs are **almost** irrelevant for the type checker

```
1 Require Import Stuff.
2 Section Ex.
3 Variables A B C : Prop.

4 Lemma ex1 : A -> B.
5 Proof.
6 ...
7 Qed.

8 Lemma ex2 : C.
9 Proof.
10 ...
11 apply ex1.
12 ...
13 Qed.

14 End Ex.
```

Getting rid of the read/eval/print loop

The problems:

- `End Ex.` looks at the proof to compute the discharged type
- constraints on `Type` indexes are generated by `Qed` while type checking the proof

```
1 Require Import Stuff.  
2 Section Ex.  
3 Variables A B C : Prop.  
  
4 Lemma ex1 : A -> B.  
5 Proof.  
6 ...  
7 Qed.  
  
8 Lemma ex2 : C.  
9 Proof.  
10 ...  
11 apply ex1.  
12 ...  
13 Qed.  
  
14 End Ex.
```

Getting rid of the read/eval/print loop

The problems:

- `End Ex.` looks at the proof to compute the discharged type
- constraints on `Type` indexes are generated by `Qed` while type checking the proof

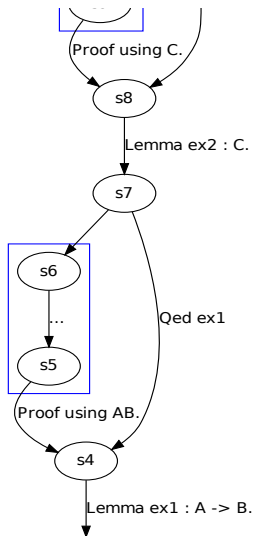
We are lucky:

- `Proof using` vars. to declare the section variables used (in 8.4)
- the addition of `Type` constraints is a commutative operation

```
1  Require Import Stuff.  
2  Section Ex.  
3  Variables A B C : Prop.  
  
4  Lemma ex1 : A -> B.  
5  Proof.  
6  ...  
7  Qed.  
  
8  Lemma ex2 : C.  
9  Proof.  
10 ...  
11 apply ex1.  
12 ...  
13 Qed.  
  
14 End Ex.
```

Getting rid of the read/eval/print loop

The data structure



```

1  Require Import Stuff.
2  Section Ex.
3  Variables A B C : Prop.

4  Lemma ex1 : A -> B.
5  Proof using A B.
6  ...
7  Qed.

8  Lemma ex2 : C.
9  Proof using C.
10 ...
11 apply ex1.
12 ...
13 Qed.

14 End Ex.

```

Demo

File Edit View Navigation TryTactics Templates Queries Compile Windows Help

CoqWs1.v

```
Require Import Arith.Factorial.  
  
Section Demo.  
  
Let n := 9.  
  
Lemma slow : fact n - fact n = 0.  
Proof using n.  
reflexivity.  
Qed.  
  
Lemma fast : fact n - fact n + fact n = fact n.  
Proof using n.  
rewrite slow; reflexivity.  
Qed.  
  
End Demo.  
  
Lemma ongoing : fact 9 > fact 8 + 9.
```

1 subgoal
fact 9 > fact 8 + 9 (1/1)

Demo!

Ready, proving ongoing Line: 19 Char: 37 CoqIDE started

Future/ongoing work

Many things are still needed to take full profit:

- 1 separate compilation revisited
 - .v when compiled generates a .vi and .vp
 - .vi enough data to make **Require** work (fast to generate)
 - .vp delayed tasks (to be run by the .vi interactive user)
 - .vo obtained as .vi + output of tasks in .vp

Future/ongoing work

Many things are still needed to take full profit:

- ① separate compilation revisited
- ② Isabelle/Jedit GUI
 - ▶ share the document model/editing API with Isabelle/Jedit
 - ▶ share most of the Jedit plugin code
 - ▶ fast computation of the DAG (ideally by just parsing)
 - ▶ classification of vernacular commands:
 - ★ branch/merge (**Proof**, **Qed**)
 - ★ global (side)effect (**Hint**)
 - ★ require immediate execution (**Notation**, **Open Scope**)
 - ★ local to the branch (tactics)

Future/ongoing work

Many things are still needed to take full profit:

- ① separate compilation revisited
- ② Isabelle/Jedit GUI
- ③ true concurrency in Coq
 - ▶ farm or slave processes
 - ▶ task and prover status transmission (deltas?)
 - ▶ API available to most Coq internals

Are you interested in a post-doc on these topics?

Future/ongoing work

Many things are still needed to take full profit:

- 1 separate compilation revisited
- 2 Isabelle/Jedit GUI
- 3 true concurrency in Coq
- 4 better tracking of dependencies
 - ▶ recheck only what is really needed

Thanks

Thanks for your attention!