ProofTool: GUI for the GAPT Framework

Mikheil Rukhaia

joint work with C. Dunchev, A. Leitsch, T. Libal, M. Riener, D. Weller *and* B. Woltzenlogel-Paleo.

10th International Workshop on User Interfaces for Theorem Provers,
Bremen, Germany.

July 11, 2012

Outline

Introduction

Main Questions

- What is GAPT?
- Why GAPT?
- What is ProofTool?
- Why ProofTool?

What is GAPT?

- GAPT: General Architecture for Proof Theory.
- Home page: `http://code.google.com/p/gapt/`
- Main purpose: Proof Transformation.
- Important! GAPT is not:
  1) A theorem prover, although it includes one.
  2) Competing with ATP/ITPs, but challenges them.

What is GAPT? (ctd.)

- GAPT implements:
  - Languages: Typed λ-calculus, first- and higher-order logic, formula schemata.
  - Calculi: **LK**, **LKS**, Resolution.
  - Algorithms: Skolemization, unification, β-reduction, etc.
  - Transformations: Reductive cut-elimination, CERES, cut-introduction, etc.
  - Applications: CLI, TAP, ProofTool.

Why GAPT?

- It is generic.
- Includes several projects related to Proof Theory.
- Flexible enough for developers with:
  - varying theoretical background,
  - varying programming experience,
  - skills in different programming paradigms.

What is ProofTool?

- ProofTool is a proof viewer.
- Alternative to Command Line Interface.
- Aim: fully fledged Graphical User Interface.

Why ProofTool?

- Several input/output formats.
- Renders more kind of objects.
- First viewer supporting proof schemata.

Proof Input Language

The Language

- Easily readable by humans and machines.
- Designed for proof schemata, but allows non-recursive proof definitions as well.
- Simple grammar.
- Plain text with ASCII characters.

Grammar

proof *name* proves *sequent*
base {
  id_1 : $rule_1$
    . . .
  id_n : $rule_n$
  root : $rule_{n+1}$
}
step {
  id_1 : $rule_1$
    . . .
  id_m : $rule_m$
  root : $rule_{m+1}$
}

Parser

- Scala combinator parser.
- Simple error handling.
- Include Auto-propositional mode.
- Flexible specification of structural rules.

ProofTool Features and Implementation

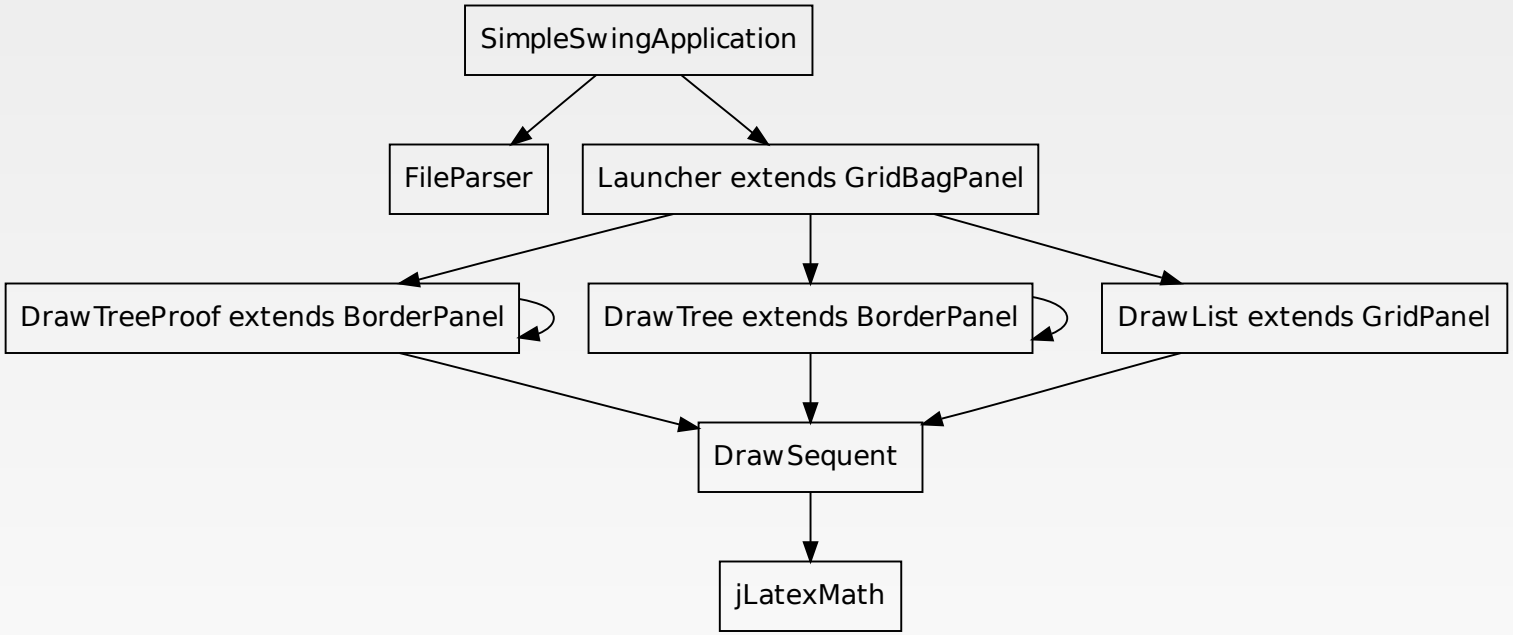
Features

- Parse: .xml, .lks, .gz.
- Export: .xml, .tex, .tptp.
- Basic: Zoom, Scroll.
- General: Search.

Features (ctd.)

- Display: sequent-like proofs, trees, lists.
  - **LK** and **LKS**: Mark cut-ancestors, Extract cut-formulas, Hide sequent context, Hide structural rules.
  - **LK**: Reductive cut-elimination, CERES.
  - **LKS**: Schematic CERES, Compute proof instance.
  - Trees: Hide/Show leaves and branches.

Implementation



Demonstration

Questions?