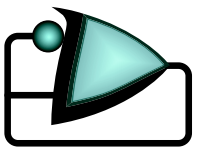


CS101C
Type Theory and Formal Methods

Lecture 4

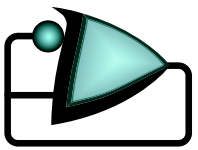
April 9, 2003

MetaPRL

MetaPRL is the latest in the PRL family of systems (λ-PRL, Micro-PRL, NuPRL, MetaPRL) developed over the last 25 years. MetaPRL (called NuPRL-Light at first) project was started by Jason Hickey in 1995. I joined the MetaPRL project in 1998.

MetaPRL contributors include:

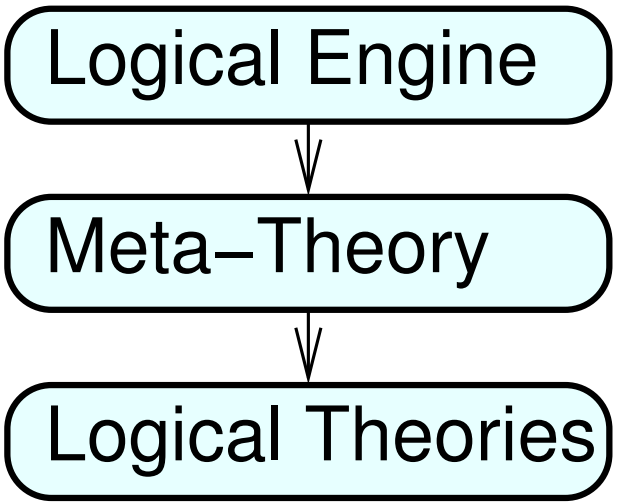
- *Caltech*: Jason Hickey, Aleksey Nogin, Xin Yu, Brian Aydemir, Adam Granicz
- *Cornell*: Robert Constable, Alexei Kopylov, Lori Lorigo, Richard Eaton, Christoph Kreitz, Eli Barzilay
- *CUNY*: Sergei Artemov, Yegor Bryuhov
- *Moscow State University*: Vladimir Krupski
- *Other*: Stephan Schmitt, Carl Witty

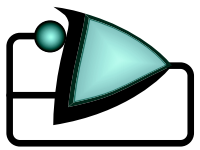


General Theorem Prover Structure

From a very high-level point of view, a modern interactive prover can be divided into three parts:

- The core of the system — its *logical engine* (or *refiner* [Bates 1979]) is in charge of handling *primitive* proof steps.
- *Meta-theory and basic tactics* provide a "support layer"; and
- Axiomatizations of *logical theories*, each potentially equipped with custom proof search, display and other mechanisms.

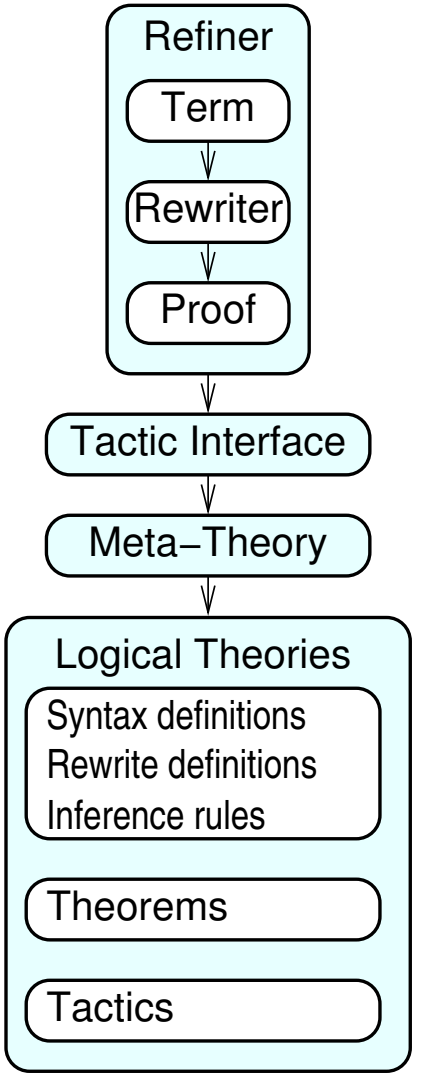


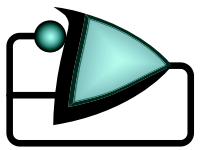


MetaPRL Structure

In MetaPRL refiner:

- `Term` module implements basic logical language.
- `Rewriter` module provides a mechanism for complex syntactical transformations.
- `Proof` accounting module keeps track of what have been proven.





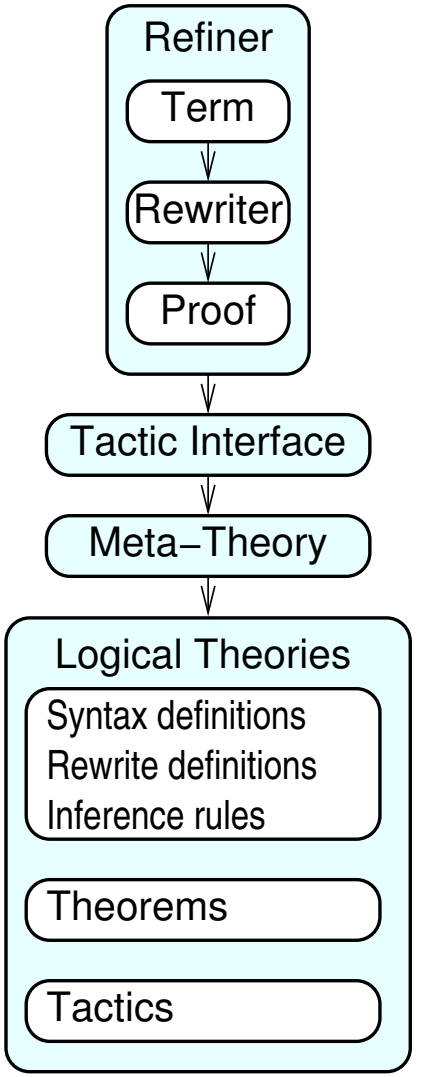
MetaPRL Structure

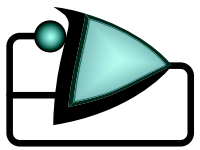
In MetaPRL logical theories

- *Syntax definitions* specify the *language*.
- *Inference rules* define the primitive inferences. Example (axiom):

$$\frac{}{\Gamma;\ A \vdash A}$$

- *Rewrites* define computational and definitional equivalences. Example (β-reduction): $(\lambda x.\, b[x])\ a \longleftrightarrow b[a]$
- *Theorems* provide proofs for derived inference rules and axioms.
- *Tactics* provide theory-specific proof search automation.

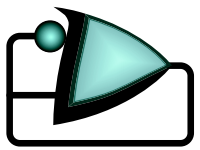




Tactics

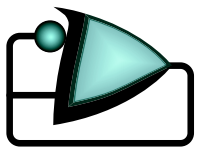
In general, a *tactic* is a program that selects a proof steps for *logical engine* to perform.

- Tactics do not have to be correct
- Tactics are heuristics
- Tactics can potentially fail, make no progress, build a partial proof, or build a complete proof
- A simplest tactic is a single inference rule
- In MetaPRL, tactics are written in OCaml language, using a set of basic combinators (called *tacticals*).



Basic Tacticals

- `tryT` tac tries applying tac (make no progress if tac fails).
- tac_1 `thenT` tac_2 applies tac_1, then on *all* subgoals it applies tac_2. Fails is either tac fails.
- tac_1 `orthenT` tac_2 tries tac_1, then (whether that works or not) on *all* subgoals it applies tac_2.
- tac_1 `orelseT` tac_2 applies tac_1, and if that fails, then applies tac_2.
- `onSomeHypT` (OCaml type: `(int -> tactic) -> tactic`) applies its argument on each hypothesis number (from 1 to number of hypotheses) until first success.



Homework 2

Homework 2 will be posted tonight

Due: Wednesday, Apr 16, at 2PM

Task: Write several proofs in MetaPRL