Fast Tactic–based Theorem Proving

Jason Hickey

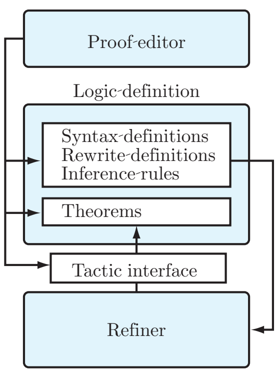
Aleksey Nogin

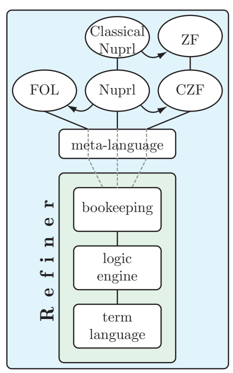
August 15, 2000

Overview of the Talk

- Introduction: What Was Accomplished and Why it is Important.
- MetaPRL Architecture
- Sources of MetaPRL Speed
- Implementation of Term Data Type
- Term Rewriting Engine
- Performance Measurements

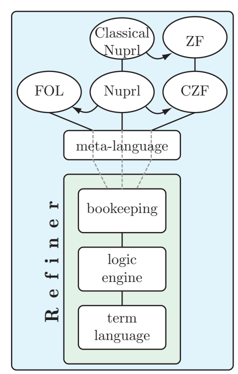
MetaPRL Architecture





- Proof editor provides a user interface -
- A logical library provides definitions, rules, theorems as well as tactics, display forms and other related objects
- The refiner (logical engine) is the core of the system and is its only trusted part.

MetaPRL Refiner



- The term module provides the `term` data type and the basic term operations
- The logical rewriting engine provides higher-level term operations
- The bookkeeping module keeps track of logical validity.

Sources of MetaPRL Speed

- Ocaml programming language
- Efficiency–conscious approach
- Modularity
- Efficient implementation of the term data structure
- Efficient term rewriting engine
- CPU cache efficiency optimizations
- Optimization for failure

*PRL Terms

$$\underbrace{opname}_{operator\ \ name}\ \underbrace{[p_1; \cdots; p_n]}_{parameters}\underbrace{\{\vec{v_1}.t_1; \cdots; \vec{v_m}.t_m\}}_{subterms}$$

Term Components

- an operator name uniquely identifying the logic the operator belongs to
- a list of parameters representing constant values
- a set of subterms with possible variable bindings

Examples

Displayed form	Term
1	`number[1]{}`
$\lambda x.a$	`lambda[]{x. a}`
$(f\ a)$	`apply[]{f; a}`
v	`variable["v"]{}`
$a + b$	`sum[]{a; b}`

Term Module Interface

```
module TermSig = sig
    type opname, param, term, bound_term
    val mk_opname : string list → opname
    val mk_int_param : int → param
    val mk_string_param : string → param
    val mk_term : opname → param list → bound_term list → term
    val mk_bterm : string list → term → bound_term

    val dest_term : term → opname * param list * bound_term list
    val dest_bterm : bound_term → string list * term

    val subst : (string * term) list → term → term
    val alpha_equal : term → term → bool
end
```

Reference implementation:

```
type term = opname * param list * bound_term list
and bound_term = string list * term
```


Terms - the Optimized Implementation

- Free variables sets are saved after they are computed
- Substitutions for unoccuring variables are discarded
- Substitutions are delayed and are being "pushed down" lazily
- Variables and other common terms are represented specially

```
type term = { free_vars :   VarsDelayed
                          | Vars of string set;
              core :    Term of (opname * param list * bound_term list)
                      | Subst of (subst * term)
                      | Var of string
                      | Sequent of sequent }
and subst = (string * term) list
and sequent = ···
```

Term Rewriting Engine

$$(\lambda x.b[x])\ a \to b[a]$$

$$a + (b + a) \to 2 * a + b$$

```
module Rewrite (Term : TermSig) : sig
    type redex_prog, con_prog, state
    exception RewriteError

    val compile_redex : term → redex_prog
    val compile_contractum : redex_prog → term → con_prog

    val apply_redex : redex_prog → term → state
    val build_contractum : con_prog → state → term
end
```


Rewriter Bytecode

All variables are replaced with indices

Examples — redices:

$(\lambda x.b[x])\ a$ becomes
$(\lambda$Bvar1.SOvar2[Bvar1]) SOvar3[]
$a + (b + a)$ becomes
SOvar1[]+(SOvar2[]+SOmatch1[])
$(\lambda x.a[x]) \circ (\lambda x.b[x] + a[b[5]])$ becomes
$(\lambda$Bvar1.SOvar2[Bvar1])$\circ$
$(\lambda$Bvar3.SOvar4[Bvar3]+SOmatch2[SOmatch4[5]]

Examples — contracta:

$b[a]$ becomes
SOvar2[SOvar3]
$2 * a + b$ becomes
2*SOvar1[]+SOvar2[]

Performance Measurements

Configuration	Tactic	Problem size 2	3	4	Memory (Max MB)
Term_std	pigeonT	<0.1	2.53	94.0	126
Term_ds	pigeonT	<0.1	0.71	17.0	20.8
NuPRL-4	pigeonT	0.5	89	>1800	
Term_std	propDecideT	0.3	238	>1800	
Term_ds	propDecideT	0.13	55.0	>1800	
NuPRL-4	propDecideT	21.9	>1800	>1800	

Configuration	Argument value 100	250	400	650
Term_std	0.35	2.05	5.42	16.0
Term_ds	0.42	2.41	6.32	18.4
NuPRL-4	55	330	>1800	>1800

Configuration	Problem GEN	ITT
Term_std	20.4	39.3
Term_ds	14.4	36.6

Pointers to Additional Information

- MetaPRL home page http://metaprl.org/
- 1-week tutorial on *PRL systems (Nuprl-5 and MetaPRL) — Cornell, summer 2001 (probably in June)