CS 671 Automated Reasoning

Meta Reasoning

Object Level versus Meta Level

- **Object level: language for formalizing concepts**
  - Concrete type theoretical expressions: `x`, `2`, `2*x`, `λx.2*x`, ...

  Always a formal language

- **Meta level: describe object level from the outside**
  - Term language: "$\lambda x\,.\,t$ *term if* x *variable and* t *term*"

    x and t are *syntactical meta-variables*
  - Substitution: "$x[t/x] = t$ *and* $y[t/x] = y$ *if* $x{\neq}y$"
  - Evaluation and judgments, validity
  - Sequents, proofs, proof rules, tactics, decision procedures, ...
  - Libraries, theorems, abstractions, display forms, ...

  Often semi-formal: English augmented with formal text

Can We Reason About Meta Level Concepts?

- **Renaming of bound variables does not change meaning**
- **All Nuprl tactics are correct**
- `Arith` **is correct**
  - An arithmetic sequent F is valid iff the corresponding labelled graph has positive cycles
- **A first-order formula F is valid iff JProver can prove it**
  - F has a sequent proof iff there is a matrix proof for F
- **The algorithm extracted from the proof of** `intsqrt_4adic` **runs in logarithmic time**
- **If two record types are syntactically equal up to reordering of labels then they are semantically equal wrt.** $\doteq$
- **F is provable if a certain syntactic transformation of F is**
- **If F has a certain form then tactic *tac* will always prove it**

Meta-reasoning can simplify proof tasks significantly

Formalizing the Meta Level

ML: meta-language as programming language

Express object language as (abstract) data type

```
abstype var                              =  ...

absrectype term                          = (tok # parm list) # bterm list
and        bterm                         = var list # term
  with mk_term (opid,parms) bterms = abs_term((opid,parms),bterms)
  and  dest_term t                       = rep_term t
  and  mk_bterm vars t                   = abs_bterm(vars,t)
  and  dest_bterm bt                     = rep_bterm bt
```

Express proofs and tactics as data types

```
abstype declaration = var # term
lettype sequent     = declaration list # term;;

absrectype proof    = (declaration list # term) # rule # proof list
  with mk_proof_goal decs t = abs_proof((decs,t), ◇,[])
  and  refine r p           = let children   = deduce_children r p
                              and validation= deduce_validation r p
                              in
                                  children, validation
  and  hypotheses p         = fst (fst (rep_proof p))
  and  conclusion p         = snd (fst (rep_proof p))
  and  refinement p         = fst (snd (rep_proof p))
  and  children   p         = snd (snd (rep_proof p))

lettype validation          = proof list -> proof;;
lettype tactic              = proof -> (proof list # validation);;
```

Mixing Object and Meta Level in Nuprl

- **Top loops and proof editor reside at meta level**
- **Object level expressions can be quoted** (use `C-o`)
  - Quoting lifts Nuprl terms to the meta-level
  - Use term editor for editing object level expressions
- **Quoted terms can be arguments of ML functions**
  - Mostly tactics, computation, decomposition, or substitution

. . . but we can't reason about the results

. . . and we can't use ML functions in Nuprl terms

– can't define $\mathtt{R}_1 \mathrel{\hat{=}} \mathtt{R}_2 \quad \equiv \quad \mathtt{sort\text{-}labels(R_1) = sort\text{-}labels(R_2)}$

Can we Reason About the Meta Level?

- **Meta level of Nuprl is not a logic**

  . . . but it has many similarities to type theory

- **One could use type theory to build a meta-logic**

```
Var          ≡  Atom
Parm         ≡  Atom × Atom
Term         ≡  rectype Term = Atom × Parm list × (Var list × Term) list

mk_term opid parms bterms  ≡  < <opid,parms>, bterms>
mk_lambda var t  ≡  mk_term "lambda" [] [[var] t]

Declaration      ≡  Var × Term
Sequent          ≡  Declaration list × Term
Proof            ≡  (Declaration list × Term) × Rule × Proof list
                    ⋮
```

But that involves a lot of double work

– All meta-level constructs (evaluation, tactics, . . .) need to be lifted

– Meta-logic is part of a different (duplicate) object logic as it does not connect to the logic in which it is defined

– We need to formalize the meta logic of that logic as well

How can we reduce double work?

- **Meta-Logical Frameworks**
  - Build logic for meta level first
  - Embed object logic into meta logic
  - Easy to build (Isabelle, Elf/Twelf, HOL, ...)
  - Can handle multiple logics
  - Fast construction of theorem proving tools for new logics
- **Reflection**
  - Bring meta-logic back into the object logic
  - Reasoning about capabilities of its own meta-logic
  - Replace execution of complex tactics by applying meta-theorems
  - More complex but much more powerful

Logical Frameworks

- **Simple logic and proof environment for meta-level**
  - Higher order logic of ∀ ⇒ together with λ-calculus
  - Fast mechanisms for matching, unification, rewriting
- **Represent generic proof theory**
  - Terms, sequents, proofs, rules, tactics, ...
  - Prove generic meta-theorems

$$\forall \mathtt{A,B,C,T_1,T_2.}\ \mathtt{is\_rule(A,B} \vdash \mathtt{C)} \Rightarrow \mathtt{is\_thm(}\vdash \mathtt{T_1)} \Rightarrow \mathtt{is\_thm(}\vdash \mathtt{T_2)}$$
$$\Rightarrow \mathtt{match(A,T_1,}\boldsymbol{\sigma}\mathtt{)} \Rightarrow \mathtt{match(B,T_2,}\boldsymbol{\sigma}\mathtt{)} \Rightarrow \mathtt{is\_thm(}\vdash \boldsymbol{\sigma}\mathtt{(C))}$$

  - Build fast generic proof tactics
- **Define object logic as (inductive) data types**
  - Concrete term language, specific rules
  - Prove that specific logic fits generic theory
  - Build proof tactics specialized to object logic

REFLECTION

- **Represent meta-logic as NUPRL expressions**
  - Data types for terms, sequents, proofs, rules, tactics, ...
  - λ-expressions for substitution, evaluation, refinement, ...
  - Informally prove isomorphism `Term` $\doteq$ `term`, `Proof` $\doteq$ `proof`, ...
- **Express object logic in represented meta logic**
  - λ-expressions for building concrete terms and rules
  - Display forms + color to make embedded logic look like object logic
- **Build hierarchy of levels**
  - Level i is meta level for level i+1
- **Reflection rule links meta level to object level**

$$H \vdash_{i+1} A \qquad \textbf{by } \texttt{reflection } i$$
$$\lceil H \rceil \vdash_{i} \exists \texttt{p:Proof}_{i}.\ \texttt{goal(p) = } \lceil A \rceil$$

  - Use same reasoning apparatus for object and meta level reasoning

Theoretically clean but impractical