Implementing Mathematics

with

The Nuprl Proof Development System

By the PRL Group:

R. L. Constable
S. F. Allen
H. M. Bromley
W. R. Cleaveland
J. F. Cremer
R. W. Harper
D. J. Howe
T. B. Knoblock
N. P. Mendler
P. Panangaden
J. T. Sasaki
S. F. Smith

Computer Science Department
Cornell University
Ithaca, NY 14853

This research supported in part by the National Science Foundation under grant DCR83-03327.

or $\exists A, x, y, B, f, g, h, u, v, w.$
(x, y:$A//B\leftarrow T$ or $A//B\leftarrow T$)
& $A \in \mathtt{U}k$
& $\forall a, a', b, b'.\ B[a, b/x, y] = B[a', b'/x, y] \in \mathtt{U}k$
if $a = a' \in A$ & $b = b' \in A$
& u, v, w are distinct and don't occur in B
& $f \in u$:A->$B[u, u/x, y]$
& $g \in u$:A->v:A->$B[u, v/x, y]$->$B[v, u/x, y]$
& $h \in u$:A->v:A->w:A->$B[u, v/x, y]$->$B[v, w/x, y]$->
$B[u, w/x, y]$

or $\exists j.\ \mathtt{U}j\leftarrow T$ & j is less than k

8.3 The Rules

The Nuprl system has been designed to accommodate the top–down construction of proofs by refinement. In this style one proves a judgement (i.e., a *goal*) by applying a *refinement rule*, thereby obtaining a set of judgements called *subgoals*, and then proving each of the subgoals. The mechanics of using the proof–editing system were discussed in chapter 7. In this section we will describe the refinement rules themselves. First we give some general comments regarding the rules and then proceed to give a description of each rule.

The Form of a Rule

To accommodate the top–down style of the Nuprl system the rules of the logic are presented in the following *refinement* style.

H >> T ext t by *rule*
H_1 >> T_1 ext t_1
⋮
H_k >> T_k ext t_k

The goal is shown at the top, and each subgoal is shown indented underneath. The rules are defined so that if every subgoal is true then one can show the truth of the goal, where the truth of a judgement is to be understood as defined in section 8.1. If there are no subgoals ($k = 0$) then the truth of the goal is axiomatic.

One of the features of the proof editor is that the extraction terms are not displayed and indeed are not immediately available. The idea is that one can judge a term T to be a type and T to be inhabited without explicitly presenting the inhabiting object. When one is viewing T as a proposition this is convenient, as a proposition is true if it is inhabited. If T is being

viewed as a specification this allows one to implicitly build a program which is guaranteed to be correct for the specification. The extraction term for a goal is built as a function of the extraction terms of the subgoals and thus in general cannot be built until each of the subgoals have been proved. If one has a specific term, t, in mind as the inhabiting object and wants it displayed, one can use the explicit intro rule and then show that the type t `in` T is inhabited. The rules have the property that each subgoal can be constructed from the information in the rule and from the goal, exclusive of the extraction term. As a result some of the more complicated rules require certain terms as parameters.

Implicit in showing a judgement to be true is showing that the conclusion of the judgement is in fact a type. We cannot directly judge a term to be a type; rather, we show that it inhabits a universe. An examination of the semantic definition will reveal that this is sufficient for our purposes. Due to the rich type structure of the system it is not possible in general to decide algorithmically if a given term denotes an element of a universe, so this is something which will require proof. The logic has been arranged so the proof that the conclusion of a goal is a type can be conducted simultaneously with the proof that the type is inhabited. In many cases this causes no great overhead, but some rules have subgoals whose only purpose is to establish that the goal is a type, that is, that it is *well–formed*. These subgoals all have the form H `>>` T `in` `U`i and are referred to as *well–formedness* subgoals.

Organization of the Rules

The rules for reasoning about each type and objects of the type will be presented in separate sections. Recall from above that for each judgement of the form H `>>` T `ext` t where the inhabiting object t is left implicit, there is a corresponding explicit judgement H `>>` t `in` T `ext axiom`. As the content of these judgements is essentially the same, the rules for reasoning about them will be presented together. In the preceding chapters, the rules have been classified mainly as introduction or elimination rules, where the introduction rules break down the form of the conclusion of a goal, and the elimination rules use a hypothesis. This is too coarse a classification for the purposes of this section, as the large majority of the rules are introduction rules. Here we will use different criteria for classifying the rules which will hopefully be more illuminating. For each type we will have the following categories of rules:

- *Formation*
  These rules give the conditions under which a canonical type may be judged to inhabit a universe, thus verifying that it is indeed a type.

equality is to be reduced. Among the parameters of some rules are keyword parameters which have the following form:

- `new` $x_1, \cdots, x_n$
  This parameter is used to give new names for hypotheses in the subgoals. In most cases the defaults, which are derived from subterms of the conclusion of the goal, suffice. For technical reasons the same variable can be declared at most once in a hypothesis list, so if a default name is already declared a new name will have to be given. Whenever this parameter is used it must be the case that the names given are all distinct and do not occur in the hypothesis list of the goal.

- `using` T, `over` $z.T$
  These parameters are used when judging the equality of noncanonical forms in types dependent on the principal argument of the noncanonical form. The `using` parameter specifies the type of the principal argument of the noncanonical form. The value should be a canonical type which is appropriate for the particular noncanonical form. The `over` parameter specifies the dependence of the type over which the equality is being judged on the principal argument of the form. Each occurrence of z in T indicates such a dependency. The proof editor always checks that the term obtained by substituting the principal argument for z in T is α–convertible to the type of the equality judgement.

- `at U`i
  The value of this parameter is the universe level at which any type judgements in the subgoals are to be made. The default is `U1`.

Optional Parameters and Defaults

Each rule will be presented in its most general form. However, some of the parameters of a rule may be optional, in which case they will be enclosed by square brackets (`[]`). If a new hypothesis in a subgoal depends on an optional parameter, and in a particular instance of the rule the optional parameter is not given, that new hypothesis will not be added. Such a dependence is usually in the form of a hypothesis specifically referring to an optional `new` name. The `over` parameter discussed above is almost always optional. If it is not given, it is assumed that the type of the equality has no dependence on the principal argument of the noncanonical form.

The issue of default values for variable names arises when the main term of a goal's conclusion contains binding variables. In general, the default values are taken to be those binding variables. For example, the rule for explicitly showing a product to be in a universe is

$H, x{:}A, y{:}B$ >> t in T[<x,y>/z]

Again this is the general pattern for rules of this type.

Hidden Assumptions

For certain rules, we need to be able to control the free variables occuring in the extract term. The mechanism used to achieve this is that of *hidden* hypotheses. A hypothesis is hidden when it is displayed enclosed in square brackets. At the moment the only place where such hypotheses are added is in a subgoal of the set elim rule. The intended meaning of a hypothesis being hidden is that the name of the hypothesis cannot appear free in the extracted term; that is, that it cannot be used computationally. Accordingly, a hidden hypothesis cannot be the object of an `elim` or `hyp` rule. For the rules for which the extract term is the trivial term `axiom`, the extract term contains no free variable references and so all restrictions on the use of hidden hypotheses can be removed. The editor will remove the brackets from any hidden hypotheses in displaying a goal of this form.

Shortcuts in the Presentation

With the exception of one of the direct computation rules, each of the rules has the property that the list of hypotheses in a subgoal is an extension of the hypothesis list of the goal. To highlight the new hypotheses and to save space, we will show only the new hypotheses in the subgoals. Also, we will not explicitly display trivial extraction terms, that is, extraction terms which are just `axiom`.

ATOM

formation

1. H >> Ui ext atom by intro atom
2. H >> atom in Ui by intro

canonical

3. H >> atom ext "···" by intro "···"
4. H >> "···" in atom by intro

where '···' is any sequence of prl characters.

5. H >> atom_eq(a;b;t;t') in T by intro
   >> a in atom
   >> b in atom
   a=b in atom >> t in T
   (a=b in atom)->void >> t' in T

computation

6a. H >> atom_eq(a;a;t;t')=t'' in T by reduce 1
   >> t=t'' in T

where a is a canonical token term.

6b. H >> atom_eq(a;b;t;t')=t'' in T by reduce 1
   >> t'=t'' in T

where a and b are different canonical token terms.

INT

formation

1. H >> Ui ext int by intro int
2. H >> int in Ui by intro

canonical

3. H >> int ext c by intro c
4. H >> c in int by intro

where c must be an integer constant.

noncanonical

5. H >> $-t$ in int
   >> t in int
6. H >> int ext m op n by intro op
   >> int ext m
   >> int ext n
7. H >> m op n in int by intro
   >> m in int
   >> n in int

where op must be one of +,-,*,/, or mod.

8. H, x:int, H' >> T ext ind($x; y, z.t_d; t_b; y, z.t_u$) by elim x new z[,y]
   y:int, y<0, z:T[y+1/x] >> T[y/x] ext t_d
   >> T[0/x] ext t_b
   y:int, 0<y, z:T[y-1/x] >> T[y/x] ext t_u

The optional new name must be given if x occurs free in H'.

9. H >> ind($e; x, y.t_d; t_b; x, y.t_u$) in $T[e/z]$
   by intro [over $z.T$] [new u, v]
   >> e in int
   u:int, u<0, v:T[u+1/z] >> $t_d[u, v/x, y]$ in $T[u/z]$
   >> t_b in T[0/z]
   u:int, 0<u, v:T[u−1/z] >> $t_u[u, v/x, y]$ in $T[u/z]$
10. H >> int_eq($a; b; t; t'$) in T by intro
    >> a in int
    >> b in int
    a=b in int >> t in T
    (a=b in int)->void >> t' in T

11. H >> less(a;b;t;t') in T by intro
    >> a in int
    >> b in int
    a<b >> t in T
    (a<b)->void >> t' in T

computation

12a. H >> ind(nt;x,y.t_d;t_b;x,y.t_u) = t in T by reduce 1 down
    >> t_d[nt,(ind(nt+1;x,y.t_d;t_b;x,y.t_u))/x,y] = t in T
    >> nt<0

12b. H >> ind(zt;x,y./t_d;t_b;x,y.t_u) = t in T by reduce 1 base
    >> t_b = t in T
    >> zt = 0 in int

12c. H >> ind(nt;x,y.t_d;t_b;x,y.t_u) = t in T by reduce 1 up
    >> t_u[nt,(ind(nt-1;x,y.t_d;t_b;x,y.t_u))/x,y] = t in T
    >> 0<nt

13a. H >> int_eq(a;a;t;t') = t'' in T by reduce 1
    >> t=t'' in T

13b. H >> int_eq(a;b;t;t') = t'' in T by reduce 1
    >> t' = t'' in T

where a and b are canonical int terms, and $a \neq b$.

14a. H >> less(a;b;t;t') = t'' in T by reduce 1
    >> t=t'' in T

where a and b are canonical int terms such that $a < b$.

14b. H >> less(a;b;t;t') = t'' in T by reduce 1
    >> t'=t'' in T

where a and b are canonical int terms such that $a \geq b$.

LIST

formation

1. H >> Ui ext A list by intro list
   >> Ui ext A
2. H >> A list in Ui by intro
   >> A in Ui

canonical

3. H >> A list ext nil by intro nil at Ui
   >> A in Ui
4. H >> nil in A list by intro at Ui
   >> A in Ui
5. H >> A list ext $h.t$ by intro .
   >> A ext h
   >> A list ext t
6. H >> $a.b$ in A list by intro
   >> a in A
   >> b in A list

noncanonical

7. $H,x{:}A$ list,H' >> T ext list_ind($x;t_b;u,v,w.t_u$)
   by elim x new w,u[,v]
   >> T[nil/x] ext t_b
   $u{:}A,v{:}A$ list,$w{:}T[v/x]$ >> $T[u.v/x]$ ext t_u
8. H >> list_ind($e;t_b;x,y,z.t_u$) in $T[e/z]$
   by intro [over $z.T$] using A list [new u,v,w]
   >> e in A list
   >> t_b in T[nil/z]
   $u{:}A,v{:}A$ list,$w{:}T[v/z]$
   >> $t_u[u,v,w/x,y,z]$ in $T[u.v/z]$

computation

9a. H >> list_ind(nil;$t_b;u,v,w.t_u$) = t in T by reduce 1
   >> t_b = t in T

9b. H >> list_ind($a.b;t_b;u,v,w.t_u$) = t in T by reduce 1
   >> $t_u[a,b,$list_ind($b;t_b;u,v,w.t_u$)$/u,v,w]$ = t in T

UNION

formation

1. H >> Ui ext $A|B$ by intro union
   >> Ui ext A
   >> Ui ext B
2. H >> $A|B$ in Ui by intro
   >> A in Ui
   >> B in Ui

canonical

3. H >> $A|B$ ext inl(a) by intro at Ui left
   >> A ext a
   >> B in Ui
4. H >> inl(a) in $A|B$ by intro at Ui
   >> a in A
   >> B in Ui
5. H >> $A|B$ ext inr(b) by intro at Ui right
   >> B ext b
   >> A in Ui
6. H >> inr(b) in $A|B$ by intro at Ui
   >> b in B
   >> A in Ui

noncanonical

7. $H,z{:}A|B,H'$ >> T ext decide(z;$x.t_l$;$y.t_r$) by elim z [new x,y]
   $x{:}A,z$=inl(x) in $A|B$ >> T[inl(x)/z] ext t_l
   $y{:}B,z$=inr(y) in $A|B$ >> T[inr(y)/z] ext t_r
8. H >> decide(e;$x.t_l$;$y.t_r$) in T[e/z]
   by intro [over $z.T$] using $A|B$ [new u,v]
   >> e in $A|B$
   $u{:}A$, e=inl(u) in $A|B$ >> t_l[u/x] in T[inl(u)/z]
   $v{:}B$, e=inr(v) in $A|B$ >> t_r[v/y] in T[inr(v)/z]

computation

9a. H >> decide(inl(a);$x.t_l$;$y.t_r$) = t in T by reduce 1
   >> t_l[a/x] = t in T
9b. H >> decide(inr(b);$x.t_l$;$y.t_r$) = t in T by reduce 1
   >> t_r[b/y] = t in T

FUNCTION

formation

1. H >> Ui ext x:A->B by intro function A new x
   >> A in Ui
   x:A >> Ui ext B
2. H >> x:A->B in Ui by intro [new y]
   >> A in Ui
   y:A >> B[y/x] in Ui
3. H >> Ui ext A->B by intro function
   >> Ui ext A
   >> Ui ext B
4. H >> A->B in Ui by intro
   >> A in Ui
   >> B in Ui

canonical

5. H >> x:A->B ext \y.b by intro at Ui [new y]
   y:A >> B[y/x] ext b
   >> A in Ui
6. H >> \x.b in y:A->B by intro at Ui [new z]
   z:A >> b[z/x] in B[z/y]
   >> A in Ui

noncanonical

7. H,f:(x:A->B),H' >> T ext t[f(a)/y] by elim f on a [new y]
   >> a in A
   y:B[a/x], y=f(a) in B[a/x] >> T ext t
8. H,f:(x:A->B),H' >> T ext t[f(a)/y] by elim f [new y]
   >> A ext a
   y:B >> T ext t

   The first form is used when x occurs free in B, the second when it doesn't.

9. H >> f(a) in B[a/x] by intro using x:A->B
   >> f in x:A->B
   >> a in A

PRODUCT

formation

1. H >> Ui ext x:A#B by intro product A new x
   >> A in Ui
   x:A >> Ui ext B
2. H >> x:A#B in Ui by intro [new y]
   >> A in Ui
   y:A >> B[y/x] in Ui
3. H >> Ui ext A#B by intro product
   >> Ui ext A
   >> Ui ext B
4. H >> A#B in Ui by intro
   >> A in Ui
   >> B in Ui

canonical

5. H >> x:A#B ext <a,b> by intro at Ui a [new y]
   >> a in A
   >> B[a/x] ext b
   y:A >> B[y/x] in Ui
6. H >> <a,b> in x:A#B by intro at Ui [new y]
   >> a in A
   >> b in B[a/x]
   y:A >> B[y/x] in Ui
7. H >> A#B ext <a,b> by intro
   >> A ext a
   >> B ext b
8. H >> <a,b> in A#B by intro
   >> a in A
   >> b in B

noncanonical

9. H,z:(x:A#B),H' >> T ext spread(z;u,v.t) by elim z new u,v
   u:A,v:B[u/x],z=<u,v> in x:A#B >> T[<u,v>/z] ext t
10. H >> spread(e;x,y.t) in T[e/z]
    by intro [over z.T] using w:A#B [new u,v]
    >> e in w:A#B
    u:A,v:B[u/w],e=<u,v> in w:A#B >> t[u,v/x,y] in
T[<u,v>/z]

QUOTIENT

formation

1. H >> Ui ext $(x,y):A//E$ by intro quotient A,E new z
   >> A in Ui
   $x:A,y:A$ >> E in Ui
   $x:A$ >> $E[x,x/x,y]$
   $x:A,y:A,E[x,y/x,y]$ >> $E[y,x/x,y]$
   $x:A,y:A,z:A,E[x,y/x,y]$, $E[y,z/x,y]$ >> $E[x,z/x,y$
2. H >> $(u,v):A//E$ in Ui by intro new x,y,z
   >> A in Ui
   $x:A,y:A$ >> $E[x,y/u,v]$ in Ui
   $x:A$ >> $E[x,x/u,v]$
   $x:A,y:A,E[x,y/u,v]$ >> $E[y,x/u,v]$
   $x:A,y:A,z:A,E[x,y/u,v]$, $E[y,z/u,v]$ >> $E[x,z/u,v$

canonical

3. H >> $(x,y):A//E$ ext a by intro at Ui
   >> $(x,y):A//E$ in Ui
   >> A ext a
4. H >> a in $(x,y):A//E$ by intro at Ui
   >> $(x,y):A//E$ in Ui
   >> a in A

noncanonical

5. $H,u:(x,y):A//E,H'$ >> t=t' in T by elim u at Ui [new
   $v:A,w:A$ >> $E[v,w/x,y]$ in Ui
   >> T in Ui
   $v:A,w:A,E[v,w/x,y]$ >>
   $t[v/u]$ = $t'[w/u]$ in $T[v/u]$

equality

6. H >> $(x,y):A//E$ = $(u,v):B//F$ in Ui by intro [new r
   >> $(x,y):A//E$ in Ui
   >> $(u,v):B//F$ in Ui
   >> A = B in Ui
   A=B in Ui,$r:A,s:A$ >> $E[r,s/x,y]$ -> $F[r,s/u,v]$
   A=B in Ui,$r:A,s:A$ >> $F[r,s/u,v]$ -> $E[r,s/x,y]$

SET

formation

1. H >> Ui ext $\{x{:}A|B\}$ by intro set A new x
   >> A in Ui
   $x{:}A$ >> Ui ext B
2. H >> $\{x{:}A|B\}$ in Ui by intro [new y]
   >> A in Ui
   $y{:}A$ >> $B[y/x]$ in Ui
3. H >> Ui ext $\{A|B\}$ by intro set
   >> Ui ext A
   >> Ui ext B
4. H >> $\{A|B\}$ in Ui by intro
   >> A in Ui
   >> B in Ui

canonical

5. H >> $\{x{:}A|B\}$ ext a by intro at Ui a [new y]
   >> a in A
   >> $B[a/x]$ ext b
   $y{:}A$ >> $B[y/x]$ in Ui

   All hidden hypothesis in H become unhidden in the second subgoal.

6. H >> a in $\{x{:}A|B\}$ by intro at Ui [new y]
   >> a in A
   >> $B[a/x]$
   $y{:}A$ >> $B[y/x]$ in Ui
7. H >> $\{A|B\}$ ext a by intro
   >> A ext a
   >> B ext b

   All hidden hypotheses in H become unhidden in the second subgoal.

8. H >> a in $\{A|B\}$ by intro
   >> a in A
   >> B ext b

noncanonical

9. H,u:{x:A|B},H' >> T ext (\y.t)(u) by elim u at Ui [new y]
   y:A >> B[y/x] in Ui
   y:A,[B[y/x]],u=y in A >> T[y/u] ext t

   Note that the second new hypotheses of the second subgoal is hidden.

equality

10. H >> {x:A|B} = {y:A'|B'} in Ui by intro [new z]
    >> A = A' in Ui
    z:A >> B[z/x] -> B'[z/y]
    z:A >> B'[z/y] -> B[z/x]

EQUALITY

formation

1. H >> Ui ext a_1=$\cdots$=a_n in A by intro equality A n
   >> A in Ui
   >> A ext a_1
   $\vdots$
   >> A ext a_n

   The default for n is 1.

2. H >> (a_1=$\cdots$=a_n in A) in Ui by intro
   >> A in Ui
   >> a_1 in A
   $\vdots$
   >> a_n in A

canonical

3. H >> axiom in (a in A) by intro
   >> a in A

4. H,x:T,H' >> x in T by intro

   This rule doesn't work when T is a set or quotient term, since intro will invoke the equality rule for the set or quotient type, respectively. In any case, the equality rule can be used.

UNIVERSE

canonical

1. H `>> U`i `ext U`j `by intro universe U`j
2. H `>> U`j `in U`i `by intro`

where $j < i$. Note that all the formation rules are intro rules for a universe type.

noncanonical

Currently there are no rules in the system for analyzing universes. At some later date such rules may be added.

MISCELLANEOUS

hypothesis

1. $H, x{:}A, H'$ >> A' ext x by hyp x

   where A' is α-convertible to A

sequence

2. H >> T ext $(\backslash x_1.\cdots(\backslash x_n.t)(t_n))\cdots)(t_1)$
   by seq $T_1,\ldots,T_n$ [new $x_1,\ldots,x_n$]
   >> T_1 ext t_1
   $x_1{:}T_1$ >> T_2 ext t_2
   $\vdots$
   $x_1{:}T_1,\ldots,x_n{:}T_n$ >> T ext t

lemma

3. H >> T ext t[term_of(*theorem*)/x] by lemma *theorem* [new x]
   $x{:}C$ >> T ext t

   where C is the conclusion of the complete theorem *theorem*.

def

4. H >> T ext t by def *theorem* [new x]
   x:term_of(*theorem*) = *ext-term* in C >> T ext t

   where C is the conclusion of the complete theorem, *theorem*, and *ext-term* is the term extracted from that theorem. [10]

explicit intro

5. H >> T ext t by explicit intro t
   >> t in T

cumulativity

6. H >> T in Ui by cumulativity at Uj
   >> T in Uj

   where $j < i$

[10] This rule introduces very strong interproof dependencies. A proof using this rule depends not only on C but also on the way C is proved.