Computation and Deduction

Lecture 15: Natural Deduction
March 4, 1997

1. Natural Deduction
2. Local Reduction
3. Congruences

Predicate Logic

```
i : type.  % individuals
o : type.  % formulas
%name i T S
%name o A B C

and    : o -> o -> o.  %infix right 11 and
imp    : o -> o -> o.  %infix right 10 imp
or     : o -> o -> o.  %infix right 11 or
not    : o -> o.       %prefix 12 not
true   : o.
false  : o.
forall : (i -> o) -> o.
exists : (i -> o) -> o.
```

Natural Deduction I

```
nd : o -> type.  % deductions
%name nd D E

andi    : nd A -> nd B -> nd (A and B).
andel   : nd (A and B) -> nd A.
ander   : nd (A and B) -> nd B.

impi    : (nd A -> nd B) -> nd (A imp B).
impe    : nd (A imp B) -> nd A -> nd B.

oril    : nd A -> nd (A or B).
orir    : nd B -> nd (A or B).
ore     : nd (A or B) -> (nd A -> nd C) -> (nd B -> nd C)
            -> nd C.

noti    : ({p:o} nd A -> nd p) -> nd (not A).
note    : nd (not A) -> {C:o} nd A -> nd C.
```

Natural Deduction II

```
truei   : nd (true).

falsee  : nd (false) -> nd C.

foralli : ({a:i} nd (A a)) -> nd (forall A).
foralle : nd (forall A) -> {T:i} nd (A T).

existsi : {T:i} nd (A T) -> nd (exists A).
existse : nd (exists A) -> ({a:i} nd (A a) -> nd C) -> nd C.
```

Local Reductions I

```
==>L : nd A -> nd A -> type.  %name ==>L L
%infix none 14 ==>L

redl_andl   : (andel (andi D E)) ==>L D.
redl_andr   : (ander (andi D E)) ==>L E.

redl_imp    : (impe (impi D) E) ==>L (D E).
```

Local Reductions II

```
redl_orl    : (ore (oril D) E1 E2) ==>L (E1 D).
redl_orr    : (ore (orir D) E1 E2) ==>L (E2 D).

redl_not    : (note (noti D) C E) ==>L (D C E).

redl_forall : (foralle (foralli D) T) ==>L (D T).

redl_exists : (existse (existsi T D) E) ==>L (E T D).
```

Congruences, Conjunction

```
==> : nd A -> nd A -> type.                %name ==> R
%infix none 14 ==>

red_local    : D ==>L D'
                 -> D ==> D'.

% Conjunction
red_andi1    : D1 ==> D1'
                 -> (andi D1 D2) ==> (andi D1' D2).
red_andi2    : D2 ==> D2'
                 -> (andi D1 D2) ==> (andi D1 D2').
red_andel    : D ==> D'
                 -> (andel D) ==> (andel D').
red_ander    : D ==> D'
                 -> (ander D) ==> (ander D').
```

Congruences, Implication

```
% Implication
red_impi      : ({u:nd A} u ==> u -> (D u) ==> (D' u))
                  -> (impi D) ==> (impi D').
red_impe1     : D1 ==> D1'
                  -> (impe D1 D2) ==> (impe D1' D2).
red_impe2     : D2 ==> D2'
                  -> (impe D1 D2) ==> (impe D1 D2').
```

Conguences, Quantification

```
% Universal Quantification
red_foralli  : ({a:i} (D a) ==> (D' a))
                 -> (foralli D) ==> (foralli D').
red_foralle  : D ==> D'
                 -> (foralle D T) ==> (foralle D' T).

% Existential Quantification
red_existsi  : D ==> D'
                 -> (existsi T D) ==> (existsi T D').
red_existse1 : D1 ==> D1'
                 -> (existse D1 D2) ==> (existse D1' D2).
red_existse2 : ({a:i} {u:nd (A a)} (D2 a u) ==> (D2' a u))
                 -> (existse D1 D2) ==> (existse D1 D2').
```