Computation and Deduction

Lecture 18: Sequent Calculus and Natural Deduction
Thursday, March 13, 1997

1. Sequent Calculus

Sequent Calculus I

```
hyp  : o -> type.  % Hypotheses (left)
conc : o -> type.  % Conclusion (right)
%name hyp H
%name conc R

axiom : (hyp A -> conc A).
```

Sequent Calculus II

```
% Conjunction
andr  : conc A
         -> conc B
         -> conc (A and B).

andl1 : (hyp A -> conc C)
         -> (hyp (A and B) -> conc C).

andl2 : (hyp B -> conc C)
         -> (hyp (A and B) -> conc C).
```

Sequent Calculus III

```
% Implication
impr  : (hyp A -> conc B)
         -> conc (A imp B).

impl  : conc A
         -> (hyp B -> conc C)
         -> (hyp (A imp B) -> conc C).
```

Sequent Calculus IV

```
% Disjunction
orr1  : conc A
         -> conc (A or B).

orr2  : conc B
         -> conc (A or B).

orl   : (hyp A -> conc C)
         -> (hyp B -> conc C)
         -> (hyp (A or B) -> conc C).
```

Sequent Calculus V

```
% Truth
truer : conc (true).
% no truel

% Falsehood
% no falser
falsel : (hyp (false) -> conc C).

% Negation
notl   : ({p:o} hyp A -> conc p)
          -> conc (not A).

notr   : conc A
          -> (hyp (not A) -> conc C)
```

Sequent Calculus VI

```
% Universal Quantification
forallr : ({a:i} conc (A a))
           -> conc (forall A).

foralll : {T:i} (hyp (A T) -> conc C)
           -> (hyp (forall A) -> conc C).

% Existential Quantification
existsr : {T:i} conc (A T)
           -> conc (exists A).

existsl : ({a:i} hyp (A a) -> conc C)
           -> (hyp (exists A) -> conc C).
```

Introductions and Eliminations I

```
ndi : nd A -> type.  % Introduction deductions
nde : nd A -> type.  % Elimination deductions
%name ndi I
%name nde E

%%% Coercions

% Closing the gap
ndi_nde : ndi D <- nde D.

% Using a lemma
% Deductions without this rule are in normal form
nde_ndi : nde D <- ndi D.
```

Introductions and Eliminations II

```
% Conjunction
ndi_andi  : ndi (andi D E)
              <- ndi D
              <- ndi E.

nde_andel : nde (andel D)
              <- nde D.

nde_ander : nde (ander D)
              <- nde D.
```

Introductions and Eliminations III

```
% Implication
ndi_impi  : ndi (impi D)
              <- ({u:nd A} nde u -> ndi (D u)).

nde_impe  : nde (impe D E)
              <- nde D
              <- ndi E.
```