Computation and Deduction

Lecture 19: Cut Elimination
Tuesday March 18, 1997

1. Soundness of Sequent Calculus
2. Completeness of Sequent Calculus
3. Cut Elimination

Soundness of Sequent Calculus I

```
ssdc : conc A -> nd A -> type.
ssdh : hyp A -> nd A -> type.

ss_axiom : ssdc (axiom H) D
             <- ssdh H D.

ss_cut : ssdc (cut S1 S2) (D2 D1)
          <- ssdc S1 D1
          <- ({h:hyp A} {u:nd A}
                ssdh h u
                -> ssdc (S2 h) (D2 u)).
```

Soundness of Sequent Calculus II

```
% Conjunction

ss_andr : ssdc (andr S1 S2) (andi D1 D2)
           <- ssdc S1 D1
           <- ssdc S2 D2.

ss_andl1 : ssdc (andl1 S1 H) (D1 (andel D2))
            <- ({h:hyp A} {u:nd A}
                  ssdh h u
                  -> ssdc (S1 h) (D1 u))
            <- ssdh H D2.

ss_andl2 : ssdc (andl2 S1 H) (D1 (ander D2))
            <- ({h:hyp B} {u:nd B}
                  ssdh h u
                  -> ssdc (S1 h) (D1 u))
            <- ssdh H D2.
```

Soundness of Sequent Calculus III

```
% Disjunction

ss_orr1 : ssdc (orr1 S1) (oril D1)
           <- ssdc S1 D1.

ss_orr2 : ssdc (orr2 S2) (orir D2)
           <- ssdc S2 D2.

ss_ore  : ssdc (orl S1 S2 H) (ore D3 D1 D2)
           <- ({h1:hyp A1} {u1:nd A1}
                 ssdh h1 u1 -> ssdc (S1 h1) (D1 u1))
           <- ({h2:hyp A2} {u2:nd A2}
                 ssdh h2 u2 -> ssdc (S2 h2) (D2 u2))
           <- ssdh H D3.
```

Completeness of Sequent Calculus I

```
scp : nd A -> conc A -> type.

% Conjunction
scp_andi : scp (andi D1 D2) (andr S1 S2)
            <- scp D1 S1
            <- scp D2 S2.

scp_andel : scp (andel D1)
             (cut S1 ([h:hyp (A and B)]
                        andl1 ([h1:hyp A] axiom h1) h))
             <- scp D1 S1.

%{ Eta-contracted alternative (also posible below)
scp_andel : scp (andel D1) (cut S1 (andl1 (axiom)))
             <- scp D1 S1.
}%
```

Completeness of Sequent Calculus II

```
scp_impi : scp (impi D1) (impr S1)
            <- ({u:nd A} {h:hyp A}
                  scp u (axiom h) -> scp (D1 u) (S1 h)).

scp_impe : scp (impe D1 D2)
            (cut S1 ([h:hyp (A imp B)]
                       impl S2 ([h1:hyp B] axiom h1) h))
            <- scp D1 S1
            <- scp D2 S2.
```

Admissibility of Cut, Axioms

```
ca : {A:o} conc A -> (hyp A -> conc C) -> conc C -> type.

%% Axiom Conversions

ca_axiom_l : ca A (axiom H) E (E H).

ca_axiom_r : ca A D ([h:hyp A] axiom h) D.
```

Admissibility of Cut, Essential Conversions

```
ca_and1 : ca (A1 and A2) (andr D1 D2)
           ([h:hyp (A1 and A2)] andl1 (E1 h) h) F
           <- ({h1:hyp A1}
                 ca (A1 and A2) (andr D1 D2)
                 ([h:hyp (A1 and A2)] E1 h h1) (E1' h1))
           <- ca A1 D1 E1' F.

ca_imp  : ca (A1 imp A2) (impr D2)
           ([h:hyp (A1 imp A2)] impl (E1 h) (E2 h) h) F
           <- ca (A1 imp A2) (impr D2) E1 E1'
           <- ({h2:hyp A2}
                 ca (A1 imp A2) (impr D2)
                 ([h:hyp (A1 imp A2)] E2 h h2) (E2' h2))
           <- ca A1 E1' D2 D2'
           <- ca A2 D2' E2' F.
```

Admissibility of Cut, Commutative Conversions

```
cal_andl1  : ca A (andl1 D1 H) E (andl1 D1' H)
               <- {h1:hyp B1} ca A (D1 h1) E (D1' h1).

cal_impl   : ca A (impl D1 D2 H) E (impl D1 D2' H)
               <- ({h2:hyp B2} ca A (D2 h2) E (D2' h2)).

car_andr : ca A D ([h:hyp A] andr (E1 h) (E2 h)) (andr E1' E2')
             <- ca A D E1 E1'
             <- ca A D E2 E2'.
```