# Linear Logic

Lecture 11: Tactics and Tacticals
February 20, 1998

1. Rewriting tactics

2. Goal-reduction tactics

# Generic Tactic Signature

```
signature TACTICS =
sig
  exception Fail
  type 'a tac

  (* infixr THEN ORELSE *)

  val THEN : 'a tac * 'a tac -> 'a tac
  val ID : 'a tac                              (* unfailing *)
  val ORELSE : 'a tac * 'a tac -> 'a tac
  val FAIL : 'a tac
  val TRY : 'a tac -> 'a tac                   (* unfailing *)
  val REPEAT : 'a tac -> 'a tac                (* unfailing *)
end;
```

## Rewriting Tactics

```
signature TACTICS1 = TACTICS where type 'a tac = 'a -> 'a

structure Tactics1 :> TACTICS1 =
struct

  exception Fail
  type 'a tac = 'a -> 'a                        (* may raise Fail *)

  infixr THEN ORELSE

  fun (f THEN g) x = g(f(x))
  fun ID x = x
  fun (f ORELSE g) x = f x handle Fail => g x
  fun FAIL x = raise Fail
  fun TRY f x = (f ORELSE ID) x
  fun REPEAT f x = TRY (f THEN (REPEAT f)) x
end;
```

# Conjunctive Goal Structures

```
signature GOAL =
sig

  datatype 'a goal =
    Success
  | Conj of 'a goal * 'a goal
  | Prim of 'a

  val conj : 'a goal * 'a goal -> 'a goal

end;
```

## Goal-Reduction Tactics

```
signature TACTICS2 =
sig
  structure Goal : GOAL
  include TACTICS where type 'a tac = 'a -> 'a Goal.goal
  val map : ('a -> 'a Goal.goal)
            -> 'a Goal.goal -> 'a Goal.goal
end;
```

# Implementation of Goal-Reduction Tactics

```
functor Tactics2 (structure Goal : GOAL) : TACTICS2 =
struct
  structure Goal = Goal
  exception Fail
  type 'a tac = 'a -> 'a Goal.goal          (* may raise Fail *)
  infixr THEN ORELSE
  fun map g (Goal.Success) = Goal.Success
    | map g (Goal.Conj(G1,G2)) = Goal.conj (map g G1, map g G2)
    | map g (Goal.Prim(x)) = g x
  fun (f THEN g) x = map g (f x)
  fun ID x = Goal.Prim x
  fun (f ORELSE g) x = f x handle Fail => g x
  fun FAIL x = raise Fail
  fun TRY f x = (f ORELSE ID) x
  fun REPEAT f x = TRY (f THEN (REPEAT f)) x
end;
```