1 What is a decision tree?

Scott, Ramsey 2000:

A decision tree is a pattern-matching automaton in which every state except the initial state has a unique predecessor.

Patterns and subject trees:

In functional languages, a pattern matcher takes a value and identifies the first of a list of patterns that matches the value. We call the value a subject tree or term. Patterns and subject trees are defined recursively. A subject tree is a constructor applied to a (possibly empty) list of subject trees. A pattern is either a variable or a constructor applied to a (possibly empty) list of patterns. A linear pattern is one in which no variable appears more than once. ML requires that patterns be linear, and our match compiler works only with linear patterns.

Let's define an ML datatype for our trees.

```
datatype 'a tree = MATCH of 'a * (path * name) list
                 | TEST of path * 'a edge list * 'a tree option
withtype 'a edge = constructor * 'a tree

(* From 106: *)

type register
type arity = int
type labeled_constructor = Pattern.vcon * arity
type pat = Pattern.pat
datatype 'a tree = TEST      of register * 'a edge list * 'a tree option
                 | LET_CHILD of (register * int) * (register -> 'a tree)
                 | MATCH     of 'a * register Env.env
and      'a edge = E of labeled_constructor * 'a tree

datatype path = REGISTER of register | CHILD of register * int
  (* in order to match block slots, children should be numbered from 1 *)

type constraint = path * pat
  (* (pi, p) is satisfied if the subject subtree at path pi matches p *)

datatype 'a frontier = F of 'a * constraint list
  (* A frontier holds a set of constraints that apply to the scrutinee.

     A choice's initial frontier has just one contraint: [(root, p)],
     where root is the scrutinee register and p is the original pattern
     in the source code.

     A choice is known to match the scrutinee if its frontier
     contains only constraints of the form (REGISTER t, VAR x).
     These constraints show in which register each bound name is stored.

     The key operation on frontiers is *refinement* (called ‘project‘
     in the paper).  Refing revises the constraints under the assumption
     that a given register holds an application of a given labeled_constructor
   *)
```