CSE 341: Programming Languages

Dan Grossman

Fall 2004

Lecture 10— Mutual Recursion, Equivalence and Syntactic Sugar

Mutual Recursion

We haven't yet seen how multiple functions can recursively call each other? (Why can't we do this with what we have?)

ML uses the keyword `and` to provide different *scope* rules. Example:

```
fun even i = if i=0 then true  else odd (i-1)
and odd  i = if i=0 then false else even (i-1)
```

Roughly extends the binding form for functions from `fun f1 x1 = e1` to `fun f1 x1 = e1 and f2 x2 = e2 and ... and fn xn = en`.

Actually, you can have `val` bindings too, but bindings being defined are in scope only inside function bodies. (Why?)

Syntax gotcha: I always forget you write `and fi xi = ei`, not `and fun fi xi = ei`.

Mutual Recursion Idioms

1. Encode a state machine (see `product_sign` example)
   - Sometimes easier to understand than explicit state values.
2. Process mutually recursive types, example:

```
datatype webtext = Empty
                 | Link of webpage * string * webtext
                 | Word of string * webtext
and webpage = Found of string * webtext
            | Unfound of string
```

A function "crawl for word" is inherently mutually recursive. (You could make a datatype for "webtext or webpage", but that's ugly.)

Problem: the Web has *cycles*, which (sigh) is a common need for mutation in ML.

Unproblem: When crawling, we don't want cycles (use Unfound if we have seen the page before).