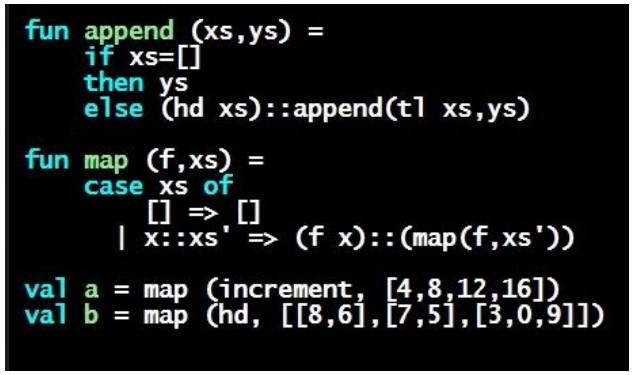

Programming Languages

Dan Grossman
2013

Pattern-Matching So Far: Precisely

Careful definitions

When a language construct is "new and strange," there is *more* reason to define the evaluation rules precisely…

… so let's review datatype bindings and case expressions "so far"

- *Extensions* to come but won't invalidate the "so far"

Datatype bindings

```
datatype t = C1 of t1 | C2 of t2 | … | Cn of tn
```

Adds type `t` and constructors `Ci` of type `ti->t`

- `Ci v` is a value, i.e., the result "includes the tag"

Omit "`of t`" for constructors that are just tags, no underlying data

- Such a `Ci` is a value of type `t`

Given an expression of type `t`, use *case expressions* to:

- See which variant (tag) it has
- Extract underlying data once you know which variant

Datatype bindings

```
case e of p1 => e1 | p2 => e2 | … | pn => en
```

- As usual, can use a case expressions anywhere an expression goes
  - Does not need to be whole function body, but often is
- Evaluate **`e`** to a value, call it **`v`**
- If **`pi`** is the first *pattern* to *match* **`v`**, then result is evaluation of **`ei`** in environment "extended by the match"
- Pattern **`Ci(x1,…,xn)`** matches value **`Ci(v1,…,vn)`** and extends the environment with **`x1`** to **`v1`** … **`xn`** to **`vn`**
  - For "no data" constructors, pattern **`Ci`** matches value **`Ci`**