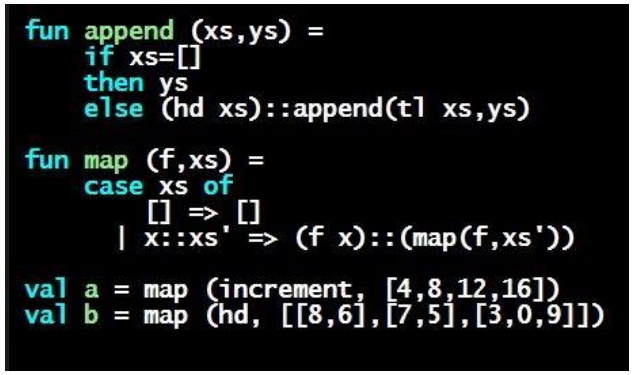

Programming Languages

Dan Grossman

Introducing Lists

Lists

- Despite nested tuples, the type of a variable still "commits" to a particular "amount" of data

In contrast, a list:

  – Can have any number of elements
  – But all list elements have the same type

Need ways to *build* lists and *access* the pieces…

Building Lists

- The empty list is a value:

```
[]
```

- In general, a list of values is a value; elements separated by commas:

```
[v1,v2,…,vn]
```

- If **`e1`** evaluates to **`v`** and **`e2`** evaluates to a list **`[v1,…,vn]`**, then **`e1::e2`** evaluates to **`[v,…,vn]`**

```
e1::e2 (* pronounced "cons" *)
```

Accessing Lists

Until we learn pattern-matching, we will use three standard-library functions

- **`null e`** evaluates to **`true`** if and only if **`e`** evaluates to **`[]`**
- If **`e`** evaluates to **`[v1,v2,…,vn]`** then **`hd e`** evaluates to **`v1`**
  - (raise exception if **`e`** evaluates to **`[]`**)
- If **`e`** evaluates to **`[v1,v2,…,vn]`** then **`tl e`** evaluates to **`[v2,…,vn]`**
  - (raise exception if **`e`** evaluates to **`[]`**)
  - Notice result is a list

Type-checking list operations

Lots of new types: For any type **`t`**, the type **`t list`** describes lists where all elements have type **`t`**

- Examples: **`int list`** **`bool list`** **`int list list`**
  `(int * int) list` **`(int list * int) list`**

- So **`[]`** can have type **`t list`** list for *any* type
  - SML uses type **`'a list`** to indicate this ("quote a" or "alpha")
- For **`e1::e2`** to type-check, we need a **`t`** such that **`e1`** has type **`t`** and **`e2`** has type **`t list`**. Then the result type is **`t list`**
- **`null : 'a list -> bool`**
- **`hd : 'a list -> 'a`**
- **`tl : 'a list -> 'a list`**