CSE 341 AA: Section 1

Porter Jones
pbjones@cs.washington.edu
Office Hours: Wednesdays 8:30 - 10:30am
Type Synonyms

- Another name for a type
  - The two types become completely interchangeable
- Can make referring to types more convenient and readable
- Will be important for modularity later in the course
- Example:

```haskell
type date = int * int * int
```
datatype in SML

- Introduces a new type name, distinct from all existing types
- Example:

```ml
datatype color = RED | GREEN | BLUE

type mix = color * color
```
Type Generality

fun append_string_lists (xs, ys) =
  if null xs
  then ys
  else (hd xs) :: append_string_lists (tl xs, ys)

val append_string_lists = fn : 'a list * 'a list -> 'a list

We may have expected the type:

string list * string list -> string list

But the SML type checker gave us the more general type:

'a list * 'a list -> 'a list
Type Generality Rule

- A type $t_1$ is more general than the type $t_2$ if you can take $t_1$, replace its type variables consistently, and get $t_2$
  - “Consistently” means that you replace each 'a, 'b, etc. with the same type

The type

\[
'a \text{ list} \times 'a \text{ list} \to 'a \text{ list}
\]

is more general than

\[
\text{string list} \times \text{string list} \to \text{string list}
\] ('a can be replaced by string)

but it is not more general than

\[
\text{string list} \times \text{int list} \to \text{string list}
\] ('a can’t be both string and int)
Equality Types

- The double quoted variable arises from use of the = operator
  - We can use = on most types like int, bool, string, tuples (that contain only “equality types”)
  - Functions and real are not ”equality types”
- Generality rules work the same, except substitution must be some type which can be compared with =
- You can ignore warnings about “calling polyEqual”

``` ML
fun triple_equal (a, b, c) = 
    a = b andalso b = c
val triple_equal = fn : ''a * ''a * ''a -> bool
```
Syntactic Sugar: if-then-else

case x of
    true => "chocolate"
| false => "huckleberry"

if x then "chocolate" else "huckleberry"

- The two expressions above are equivalent, we could use either of them interchangeably.
- We choose to use if-then-else because it looks much nicer (it’s sweet like sugar!!!) but it isn’t functionally necessary if we have case expressions.