Today’s Agenda

• Type synonyms
• Type generality
• Equality types
• Syntactic sugar
Type Synonyms

• What does `int * int * int` represent?
• In HW1 we called it a date
• Wouldn’t it be nice to reflect this representation in the source code itself?

```
type date = int * int * int
```
**type vs datatype**

- **datatype** introduces a new type name, distinct from all existing types
  
  ```
  datatype suit = Club | Diamond | Heart | Spade
  datatype rank = Jack | Queen | King | Ace
  | Num of int
  ```
  
- **type** is just another name
  
  ```
  type card = suit * rank
  ```
Type Synonyms

Why?
• For now, just for convenience
• It doesn’t let us do anything new

Later in the course we will see another use related to modularity.
Type Generality

Let’s revisit the “append” function...

```haskell
fun append (xs, ys) =
    if null xs
    then ys
    else hd xs :: append(tl xs, ys)
```
Type Generality

• We would expect
  \[
  \text{string list } \ast \text{ string list } \rightarrow \text{ string list}
  \]

• But the type checker found
  \[
  \text{'a list } \ast \text{'a list } \rightarrow \text{'a list}
  \]

• Why is this OK?
More General Types

• The type
  `a list * `a list -> `a list
  is **more general** than the type
  string list * string list list -> string list list
  and “can be used” as any **less general** type, such as
  int list * int list -> int list

• But it is **not** more general than the type
  int list * string list list -> int list
The Type Generality Rule

The “more general” rule:

A type \( t_1 \) is more general than the type \( t_2 \) if you can take \( t_1 \), replace its type variables \textit{consistently}, and get \( t_2 \)

What does \textit{consistently} mean?
Equality Types

Write a function called Contains that takes a value and a list and returns true if the value is in the list...

What type will Contains have?
Equality Types

Let’s take a look at Contains...

```haskell
fun contains(x, xs) =
    if null xs
    then false
    else (hd xs = x) orelse contains(x, tl xs)
```
Equality Types

• The double quoted variable arises from use of the \( = \) operator
  • We can use \( = \) on most types like \texttt{int}, \texttt{bool}, \texttt{string}, tuples (that contain only “equality types”)
  • Functions and \texttt{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”
Syntactic Sugar
Syntactic Sugar

- Tuples are actually Records with fields 1~n
- If-then-else is implemented as syntactic sugar for a case statement
If-then-else

• We’ve just covered case statements
• How could we implement if-then-else?

    case x of
    true => "apple"
    | false => "banana"

    if x then "apple" else "banana"
Adventures in pattern matching

• Shape example
• Function-pattern syntax if we get to it