Today’s Agenda

- Type synonyms
- Polymorphism and type generality
- Equality types
- Some sugar!
  - Do we really need “if/then/else”?

Type Synonyms

- What does \texttt{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?

```plaintext
type date = int * int * int
```

\textbf{type} vs \textbf{datatype}

- \texttt{datatype} introduces a new type name, distinct from all existing types

```plaintext
datatype suit = Club | Diamond | Heart | Spade
datatype rank = Jack | Queen | King | Ace | Num of int
```

- \texttt{type} is just another name

```plaintext
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

Write a function that appends two string lists...
Type Generality

• We would expect

  \texttt{string list * string list \to string list}

• But the type checker found

  \texttt{'a list * 'a list \to 'a list}

• Why is this OK?

More General Types

• The type

  \texttt{'a list * 'a list \to 'a list}

  is \textbf{more general} than the type

  \texttt{string list * string list \to string list}

  and “can be used” as any less general type, such as

  \texttt{int list * int list \to int list}

• But it is \textbf{not} more general than the type

  \texttt{int list * string list \to 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 \textbf{consistently}, and get \( t_2 \)

What does \textbf{consistently} mean?

Equality Types

Write a list contains function...

Equality Types

• The double quoted variable arises from use of the \( = \) operator

  • We can use \( = \) on most types like \texttt{int, bool, 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 \texttt{polyEqual}”

Syntactic Sugar

• 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?

```haskell
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