Today’s Agenda

- Type Synonyms
- Type Generality
- Equality Types
- More 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?

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

Type Generality

Write a function that appends two string lists...

Type vs datatype

- `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
```

- `type` is just another name

```plaintext
type card = suit * rank
```

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

• We would expect

```string list * string list -> string list```

• But the type checker found

```'a list * 'a list -> '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 -> string 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 -> 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 consistently, and get \( t_2 \)

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 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”

Syntactic Sugar

• If-then-else is implemented as syntactic sugar for a case statement.
• Function-pattern-case syntax