Updated: 2021-02-07

3.6
ADT Algebra

Abstract Data Types

- Basically, an ADT is simply three things:
  - *Data structure;*
  - *Algorithm;*
  - *Invariant*.
- An invariant is something that must be true when the ADT is in a steady state.
  - For example, the data structure of a *Map* is simply a sequence of key-value pairs.
  - The invariant requires that there are no duplicates among keys.
  - Otherwise, a *Map* would be identical with a sequence of key-value pairs.

But, first, a little algebra

- I'd like to cover something that isn't unique to functional programming but is more relevant than with other paradigms.
- Suppose I am defining a method *m* as follows:
  - `def m(x: Int): String = ???`
- I may not know much about what to put instead of ???. But here's what I do know:
  - it must involve *x*. Why?
  - it must yield a *String*. Why?
  - it *should* not involve anything else, apart from *this* (if *m* is an instance method). Why?
  - in practice, we are allowed to use identifiers from Objects but we should only do so if it's completely obvious why we're doing it. For example we might reference *Math.PI* or *System.out.println*.
  - in particular, we should avoid using any "global" values unless they are totally obvious.

ADT algebra

- So, what are the operators that we can apply to an ADT? That's to ask what is its *algebra*?
- That's going to depend obviously on what the ADT does and whether or not its an immutable ADT.
- What good is an immutable ADT? I hear you ask.
- Well, we can easily just copy an existing ADT and make some modifications to it if we want the equivalent of mutation. This sounds like a lot of copy perhaps, but actually it isn't.
- Let's start out with a nice simple ADT: *List[A]*.
  - BTW, *List* is an abstract class which implements the *Seq* trait. Its two concrete implementations are :: and *Nil*.

Dreaming up all possible signatures

- Let’s start with no parameters: What possible return types are there?
  - Unit
  - Boolean
  - Int
  - String
  - A
  - (A, A)
  - Option[A]
  - List[A]

No parameters

- What names and semantics might we give these?
  - Unit: ? There doesn't seem to be an obvious candidate;
  - Boolean: *isEmpty;*
  - Int: *length;*
  - String: *toString*;
  - A: *head* [but what does it do on an empty list? See below: *headOption*];
  - (A, A): *minmax*? [but we would need a way of comparing *A*s for that to work];
  - Option[A]: *headOption*;
  - List[A]: *tail*;

One function parameter:

- What possible return types are there with parameter *A=>X*?
  - Unit: *foreach* [where *X* is *Unit*];
  - Boolean: *exists* [where *X* is *Boolean*];
  - Int: *count* [where *X* is *Boolean*];
  - Option[A]: *find* [where *X* is *Boolean*];
  - List[B]: *map* [where *X* is *B*];
  - List[B]: *flatMap* [where *X* is *List[B]*];
  - (List[A], List[A]): *partition* [where *X* is *Boolean*];