Algebraic Structores

CS 152 (Spring 2020)

Harvard University

Tuesday, April 7, 2020

Announcements

- HW2: Grades available in Gradescope
- HW3: grading in progress...
- HW4: due Apr 14
- HW5: will be released Apr 14, due May 1, and combine previous HW5 and HW6.
- Survey: by the end of Wednesday Apr 8
  - `https://forms.gle/FM7mb9n4Gbze14Js6`

Today, we will learn about

- Type constructors
  - Lists, Options
- Alegebraic structures
  - Monoids
  - Functors
  - Monads
- Alegebraic structures in Haskell

Type Constructors

- A *type constructor* creates new types from existing types
  - E.g., product types, sum types, reference types, function types, ...

Lists

- Assume CBV λ-calc with booleans, fixpoint operator $\mu x\!:\!\tau.\ e$

$$
\begin{array}{lrl}
\text{Expressions} & e ::= & \cdots \mid [\,] \\
 & & \mid e_1 :: e_2 \mid \text{isempty? } e \mid \text{head } e \\
 & & \mid \text{tail } e \\
\text{Values} & v ::= & \cdots \mid [\,] \mid v_1 :: v_2 \\
\text{Types} & \tau ::= & \cdots \mid \tau\ \textbf{list} \\
\text{Eval contexts} & E ::= & \cdots \mid E :: e \mid v :: E \\
 & & \mid \text{isempty? } E \mid \text{head } E \mid \text{tail } E
\end{array}
$$

List inference rules

$$\frac{}{\text{isempty? } [\,] \longrightarrow \textbf{true}} \qquad \frac{}{\text{isempty? } v_1 :: v_2 \longrightarrow \textbf{false}}$$

$$\frac{}{\text{head } v_1 :: v_2 \longrightarrow v_1} \qquad \frac{}{\text{tail } v_1 :: v_2 \longrightarrow v_2}$$

$$\frac{}{\Gamma \vdash [\,] : \tau \textbf{ list}} \qquad \frac{\Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau \textbf{ list}}{\Gamma \vdash e_1 :: e_2 : \tau \textbf{ list}}$$

$$\frac{\Gamma \vdash e : \tau \textbf{ list}}{\Gamma \vdash \text{isempty? } e : \textbf{bool}} \qquad \frac{\Gamma \vdash e : \tau \textbf{ list}}{\Gamma \vdash \text{head } e : \tau} \qquad \frac{\Gamma \vdash e : \tau \textbf{ list}}{\Gamma \vdash \text{tail } e : \tau \textbf{ list}}$$

$$\begin{aligned} \text{append} \triangleq\ & \mu f : \tau \textbf{ list} \to \tau \textbf{ list}.\ \lambda a : \tau \textbf{ list}.\ \lambda b : \tau \textbf{ list}. \\ & \textbf{if } \text{isempty? } a \textbf{ then } b \textbf{ else } (\text{head } a) :: (f\ (\text{tail } a)\ b) \end{aligned}$$

Options

Expressions	$e ::= \cdots \mid \text{none} \mid \text{some } e \mid \text{case } e_1 \text{ of } e_2 \mid e$
Values	$v ::= \cdots \mid \text{none} \mid \text{some } v$
Types	$\tau ::= \cdots \mid \tau\ \textbf{option}$
Eval contexts	$E ::= \cdots \mid \text{some } E \mid \text{case } E \text{ of } e_2 \mid e_3$

Monoids

Monoid examples

Functors

Functor examples

Monad

Option monad

Algebraic structures in Haskell

- https://www.haskell.org/
- Pure functional language
- Call-by-need evaluation (aka lazy evaluation)
- Type classes: mechanism for ad hoc polymorphism
  - Declares common functions that all types within class have
  - We will use to express algebraic structures in Haskell

Monoid

Monad

Using Monads

Why Monads?

- Monads are *very* useful in Haskell
- Haskell is pure: no side effects
- But side effects useful!
- **Monadic types cleanly and clearly express side effects computation may have**
- Monads force computation into sequence
- Monads as type classes capture underlying structure of computation
  - Reusable readable code that works for any monad