Midterm 2 Review Notes

Monday, November 4, 2019 6:55 PM

Chapter 11 - Options

- have an empty case? don't want to return null/error

```
type 'a option
  | None
  | Some 'a
```

can pattern match on data type

Chapter 12: Unit / Sequencing Commands

- only value of unit is ()
- unit inputs if a function has no inputs
- unit output if the function has no output, only does sideeffects

; can be used to string commands together (embedded)

;; open a module / run a command at top level

; separate elements of list / record and sequence commands

Chapter 13: Records

- types w/ named fields <ID> : <Type>

For example type rgb = { r : int ; g : int ; b : int }

let white = { r=255, g=255, b=255 }

with keyword let red = { black with r=255 }

Chapter 14 Mutable State / Aliasing

Keyword mutable - value may be updated

- use "<-" to assign new value

Aliasing - having 2 variables point to the same address on the heap (updating one effects the other)

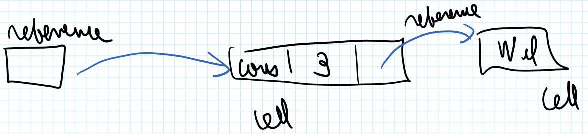
Chapter 15 ASM

- how to model programs in the mutable state
  - accounts for location of data

Workspace	Stack	Heap
keeps track of the commands / expression computer is currently simplifying	keeps track of bindings that map identifiers to values	computer memory specifies abstractly where data structure

commands/expression computer is currently simplifying	indentifiers to values new thing added when let expressions are simplified also contain partially simplified expressions waiting for function calls	specifies abstractly where data structure reside/how they interact with eachother

- primitive values are stored on the stack
- more complicated values are stored on the heap
    - address/reference stored on stack



- records have a box for each field
    - mutable fields are double boxed

- Referential equality (==)
    - 2 things point to same item on heap (aliases)
- Structural equality (=)
    - 2 things are structurally equivalent
    - are the arguments equal?

Chapter 16 Linked structures, queues

- 2 data types
    - one stores head / tail of queue
    - one to store data for internal nodes
- Queue invariants

head is none, tail is none
tail is none, head is none
there is a cycle

tail is not reachable from head
tail doesn't point to last element

Tail call optimization
- don't keep everything on the stack

~~.1 + recursive call~~

- add another argument to loop, update loop call each time instead

Chapter 17 Local State

- local, mutable state (counter for example)
- have functions that relate directly to state

- new copy saved on heap that points to state each time a new state is created

- basis for creating an object (has fields and methods)

type = {

Methods

}

constructor =

fields { record }
methods { record }

'a ref type

ref e means { contents = e }
!e ↓ e.contents
e := v ↓ e.contents ⇐ v

Chapter 18 End of OCaml / GUI

- gctx
  - module w/ type gctx
  - represents contextual info required to draw widget
  - translate coordinates and moves origin to bottom left corner
- Widgets
  - repaint
    - asks widget to redraw itself
  - size
    - reports the widgets size
  - event handler
    - how to handle if a user interacts w/ the widget

can create a widget tree with h and v pair, border, and widgets (eventually everything must be just one larger widget)

- Event loop
  - process user generated events (key presses, mouse clicks, mouse moves, and drags)
  - must pass event down widget tree to proper sub-widget
- State widget using a record
  - need a controller to update it

- event listeners/notifiers check if certain events interact w/ a specific widget
- Button
  - is label widget w/ label and notifier controller
    - can change state of the button

Chapter 19 Java

- Ocaml naturally better at functional, Java naturally better at Object Oriented
- by default everything in java is mutable
- objects are initialized to null value
- can use interfaces to separate type from object

Chapter 20 Java V Ocaml

- also 2 types of equality
  - == is referential equality (or structural for basic types)
  - .equals is structural equality
- Static Vs Dynamic Methods

↓	↓
associated w/ a class	associated w/ an instance of a class
can be determined at compile time	determined at run time

Chapter 21 Arrays

- collection of ordered data, can be indexed
- first index is 0
- .length returns length (always iterate while i < length)
- size is immutable but entries are mutable

Multidimensional Arrays
- array of arrays (String [][])
- can be different lengths

Chapter 22 Java ASM

- almost everything is mutable (even on stack)

this (fields also

Chapter 22 Java ASM

- almost everything is mutable (even on stack)
- null reference
- method bodies are stored in class table
- heap values only contain arrays/ objects

static fields also in class table

Chapter 23 Subtyping, Extension, Inheritance

- Interface contains specifications for an object
- class implements those specifications

key word, must follow interface

- Subtyping
  - super type may not be able to access subtypes methods
- a class can implement multiple interfaces
- Extend
  - and interface can extend another interface by adding extra methods
  - a class can extend or inherit from 1 other class
    - copies fields and methods of super class
- Super keyword
  - calls the superclass constructor
- All classes extend from the object class
- Static types vs. Dynamic classes

because of subtyping can have more than 1 valid static type

  - static type (declaration on left of =) compile time
  - dynamic type (type of object created on right of =) runtime

Chapter 24 Java ASM w/ Dynamic Methods

- dynamic dispatch, evaluation of method calls
- controlled by dynamic class
- class table models inheritance tree
- dynamic method call, this points to instance of object
- static methods can't use this keyword / non static methods/fields