

CSE 130 : Fall 2011

Programming Languages

Lecture 9: Deconstructing ML *The Anatomy of a Prog. Lang.*

Ranjit Jhala
UC San Diego



Key components of a lang

- Units of computation
- Types
- Memory model

Thursday, November 10, 2011

Thursday, November 10, 2011

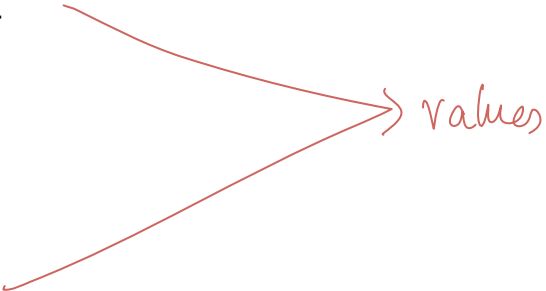
Units of computation

In OCaml

Thursday, November 10, 2011

Thursday, November 10, 2011

In OCaml

- Expressions that evaluate to values
 - **Everything** is an expression
 - int, bool, real
 - if-then-else
 - let-in
 - match
 - fun x -> x+1
 - e1 e2
- 

In Java/Python/C++/C/...

In Java/Python/C/C++

- Store and update commands *statements*
 $x := 2 * y$
- Message sends
 $obj.setX(12)$



In Java/Python/C/C++

- Store and update commands *statements*
 $x := 2 * y$
- Message sends
 $obj.setX(12)$

X → "pumpkin"

Y → 3.142

Z → [1,2,3]



In Java/Python/C/C++

statements

- Store and update commands

$x := 2 * y$

- Message sends

$obj.setX(12)$



In Java/Python/C/C++

statements

- Store and update commands

$x := 2 * y$

- Message sends

$obj.setX(12)$



Thursday, November 10, 2011

Thursday, November 10, 2011

In Prolog

In Prolog

Cent (Carnitas)

- Logical facts
- Inference rules



$? Delirious(X)$

Thursday, November 10, 2011

Thursday, November 10, 2011

In Prolog

Cat (Carnitas)

- Logical facts
- Inference rules

Mexican(CARNITAS)

"Fact"

Food(CARNITAS)

"Fact"



? Delizioso(X)

In Prolog

Cat (Carnitas)

- Logical facts
- Inference rules

Mexican(CARNITAS)

"Fact"

Food(CARNITAS)

"Fact"



Mexican(X) $\wedge$ Food(X) $\rightarrow$ Delicious(X)

"Rule"

? Delizioso(X)

In Prolog

Cat (Carnitas)

- Logical facts
- Inference rules

Mexican(CARNITAS)

"Fact"

Food(CARNITAS)

"Fact"



Mexican(X) $\wedge$ Food(X) $\rightarrow$ Delicious(X)

"Rule"

? Delizioso(X)

In Prolog

Cat (Carnitas)

- Logical facts
- Inference rules

Mexican(CARNITAS)

"Fact"

Food(CARNITAS)

"Fact"



Mexican(X) $\wedge$ Food(X) $\rightarrow$ Delicious(X)

"Rule"

Delicious(CARNITAS)

? Delizioso(X)

"Fact"

Memory/Data model

aka: what do variables refer to?

Data model in functional langs

- Environment of bindings (phonebook)

X	3
Y	"abc"
Z	[1, 2, 3]

Thursday, November 10, 2011

Thursday, November 10, 2011

Data model in functional langs

- Environment of bindings (phonebook)

X	3
Y	"abc"
Z	[1, 2, 3]
x	"cat"

- **Never change a binding**
 - add new bindings at the **end** of phonebook

Data model in functional langs

- Variables = names in phonebook
- Most recent entry looked up during eval

Thursday, November 10, 2011

Thursday, November 10, 2011

Data model in functional langs

- Variables = names in phonebook
- Most recent entry looked up during eval

Closures:

Data model in functional langs

- Variables = names in phonebook
- Most recent entry looked up during eval

Closures:

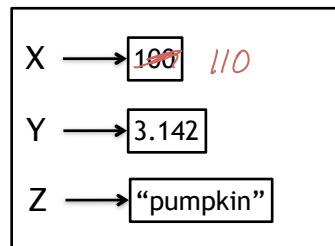
- Environment “frozen” in function value
 - behavior of function cannot be changed later
 - easier reasoning, debugging

Thursday, November 10, 2011

Thursday, November 10, 2011

Data model in Imp/OO langs

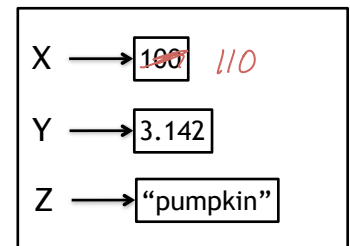
$X = 5$



Data model in Imp/OO langs

- Variables are named cells in memory

$X = 5$



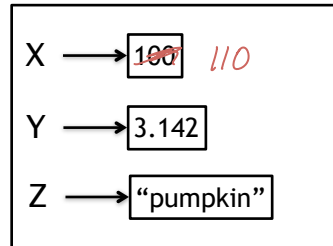
Thursday, November 10, 2011

Thursday, November 10, 2011

Data model in Imp/OO langs

- Variables are **named cells in memory**
Can **change** them by **assigning** into them

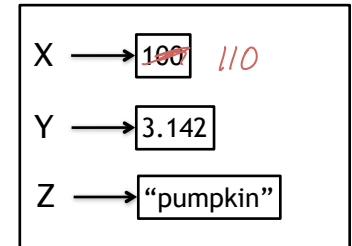
$X = 5$



Data model in Imp/OO langs

- Variables are **named cells in memory**
Can **change** them by **assigning** into them

$X = 5$



Thursday, November 10, 2011

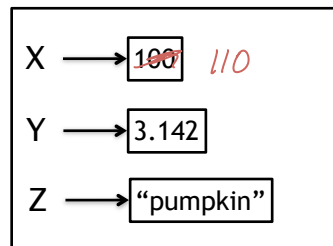
Thursday, November 10, 2011

Data model in Imp/OO langs

- Variables are **named cells in memory**
Can **change** them by **assigning** into them

$X = 5$

- Variables = **names** of objects on the heap

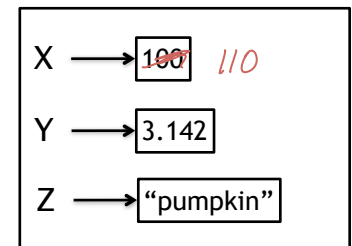


Data model in Imp/OO langs

- Variables are **named cells in memory**
Can **change** them by **assigning** into them

$X = 5$

- Variables = **names** of objects on the heap



Thursday, November 10, 2011

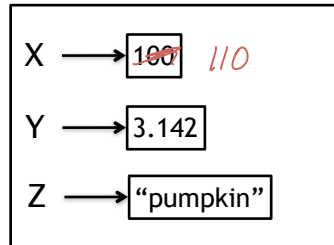
Thursday, November 10, 2011

Data model in Imp/OO langs

- Variables are **named cells in memory**
Can **change** them by **assigning** into them

- Variables = **names** of objects on the heap

- $x = x + 10$



Data model in Prolog

- Variables = **unknowns** to solve for

Data model in Prolog

- Variables = **unknowns** to solve for

Mexican(CARNITAS)

Food(CARNITAS)

$8 \text{ X Mexican(X) } \wedge \text{ Food(X) } \rightarrow \text{ Delicious(X)}$

Delicious(**Y**)?

Data model in Prolog

- Variables = **unknowns** to solve for

Mexican(CARNITAS)

Food(CARNITAS)

$8 \text{ X Mexican(X) } \wedge \text{ Food(X) } \rightarrow \text{ Delicious(X)}$

Delicious(**Y**)?

Q: What is delicious?

A: CARNITAS!

Types

Types

- Used to classify things created by the programmer
- Classification used to check what can be done with/to those things

Thursday, November 10, 2011

Thursday, November 10, 2011

In OCaml: Static typing

- Types assigned statically (at compile time)
- **Without computing values**
- Rules say when expressions are type-correct

$$\frac{e1 : T1 \rightarrow T2 \quad e2 : T1}{e1 \ e2 : T2}$$

In OCaml: Static typing

- How to reuse code for different types?
 - parametric types: $\overline{'a * 'b} \rightarrow 'b * 'a$
 - implicit for-all
- Type “discovered” (inferred) automatically from code
 - less burden on the programmer

Thursday, November 10, 2011

Thursday, November 10, 2011

In Python: Dynamic typing

- Types assigned to values/objects
as they are computed, ie: dynamically
- Before op is executed,
check that operands are compatible with op

*def f(x).
 x.m()

f(a)*

Thursday, November 10, 2011

In Python: Dynamic typing

In Python: Dynamic typing

- More programs are accepted by compiler
i.e. More flexible

Thursday, November 10, 2011

In Python: Dynamic typing

- More programs are accepted by compiler
i.e. More flexible

`[1, "abc", 1.8, [ "efg", 20]]`

Thursday, November 10, 2011

In Python: Dynamic typing

- More programs are accepted by compiler
i.e. More flexible

```
[1, "abc", 1.8, [ "efg", 20]]
```

```
let x = if b then 1 else "abc"  
let y = if b then x + 1 else x ^ "efg"
```

Thursday, November 10, 2011

In Python: Dynamic typing

- More programs are accepted by compiler
i.e. More flexible

```
[1, "abc", 1.8, [ "efg", 20]]
```

```
let x = if b then 1 else "abc"  
let y = if b then x + 1 else x ^ "efg"
```

Thursday, November 10, 2011

In Python: Dynamic typing

- More programs are accepted by compiler
i.e. More flexible

```
[1, "abc", 1.8, [ "efg", 20]]
```

```
let x = if b then 1 else "abc"  
let y = if b then x + 1 else x ^ "efg"
```

Thursday, November 10, 2011

In Python: Dynamic typing

- More programs are accepted by compiler
i.e. More flexible

```
[1, "abc", 1.8, [ "efg", 20]]
```

```
let x = if b then 1 else "abc"  
let y = if b then x + 1 else x ^ "efg"
```

Thursday, November 10, 2011

In Python: Dynamic typing

- More programs are accepted by compiler
i.e. More flexible

```
[1, "abc", 1.8, [ "efg", 20]]
```

```
let x = if b then 1 else "abc"  
let y = if b then x + 1 else x ^ "efg"
```

Dynamic vs. Static, OO vs. Func

	Statically typed	Dynamically typed
OO		
Functional		

Thursday, November 10, 2011

Thursday, November 10, 2011

Dynamic vs. Static, OO vs. Func

	Statically typed	Dynamically typed
OO	Java	Python, Smalltalk
Functional	Ocaml, Haskell	Lisp/Scheme

Polymorphism

- PL that is polymorphic + dynamically typed ?

Thursday, November 10, 2011

Thursday, November 10, 2011

Polymorphism

- PL that is polymorphic + dynamically typed ?
- Every dynamically typed PL is polymorphic
 - functions simply work on **any** datatype that can be operated on at runtime

Polymorphism

- PL that is polymorphic + dynamically typed ?
- Every dynamically typed PL is polymorphic
 - functions simply work on **any** datatype that can be operated on at runtime

Thursday, November 10, 2011

Thursday, November 10, 2011

Polymorphism

- PL that is polymorphic + dynamically typed ?
- Every dynamically typed PL is polymorphic
 - functions simply work on **any** datatype that can be operated on at runtime
- Explicit polymorphism in statically typed PL
 - assign at compile time a general polymorphic type

Final words on functional programming

Thursday, November 10, 2011

Thursday, November 10, 2011

What's the point of all this?

Advantages of functional progs

- Functional programming more concise
“one line of lisp can replace 20 lines of C” (quote from <http://www.ddj.com/dept/architect/184414500?pgno=3>)
- Recall reverse function:

```
let reverse = fold (::) [];;
```
- How many lines in C, C++?

Thursday, November 10, 2011

Thursday, November 10, 2011

Better reasoning about Programs

- No side effects
 - Same inputs return same outputs
- So, can safely **reorder** computations
- So, can also **parallelize** computations

An example: Map/Reduce

- Scalable data processing:
 - **MapReduce** programming model
 - Hadoop [Yahoo], Hive [Facebook], Dryad [MSR]
- Programming model based on **map**, **fold**
 - apply a function on each data element (**map**)
 - aggregate the results (**fold**)
- Functional) easy parallelism) scalable
- For more, take **Prof. Vahdat's CSE 124**

Thursday, November 10, 2011

Thursday, November 10, 2011

So what?

- Form the authors: “Inspired by similar primitives in LISP and other languages”
<http://labs.google.com/papers/mapreduce-osdi04-slides/index-auto-0003.html>
- Programmers who only know Java/C/C++ would probably not have come up with this idea

Remember



Remember



“Free your mind”
-Morpheus

Say hello to Python

Recap of the course so far

- 4+ weeks of functional with OCaml
- Next: 3 weeks of OO with Python

OO at the highest level

- What is OO programming?

OO at the highest level

- What is OO programming?
- Answer:
 - Objects
 - message sends
 - dynamic dispatch

Just to whet your appetite

- Say we have objects, like `cars`, `ducks`, `pig`, `cell_phones`
- Say we have a message name: `make_some_noise`

Just to whet your appetite

- Each object has its own implementation for `make_some_noise`: these are traditionally called methods.
 - `car`: vroom vroom,
 - `pig`: oink oink,
 - `duck`: quack quack
- Can send `make_some_noise` to any object. Depending on the actually run-time object, we'll get a different noise!

Thursday, November 10, 2011

Oh btw...

- What's the difference between message and method...

Thursday, November 10, 2011

Oh btw...

- What's the difference between message and method...
- Message is just the name of the message, method is the implementation
- Message is the “interface”/“prototype” of the method.

Thursday, November 10, 2011

OO programming

- Message:
 - the name of an operation
- Method:
 - the implementation of an operation
- Dynamic dispatch:
 - dynamic type of object determines
 - which method is run for given message send
- These are the core ideas of OO

Thursday, November 10, 2011

This brings us to Python...

- We'll use Python as our vehicle for OO programming
- Fun and useful language
- Let's compare with OCaml along some of the dimensions we saw last time

OCaml/Python comparison

	ML	Python
PL paradigm		
Basic unit		
Types		
DataModel		

Thursday, November 10, 2011

Thursday, November 10, 2011

Python

- Python has a very relaxed philosophy
 - if something "can be done" then it is allowed.
- Dynamic types + Everything is an object
 - very flexible
 - very intuitive code

No static types

- **No static type system** to "prohibit" operations.
- No more of that OCaml compiler giving you hard-to-decipher error messages!

Thursday, November 10, 2011

Thursday, November 10, 2011

No static types: but what instead?

- **Dynamic typing**
- At runtime, every "operation" is translated to a method call on the appropriate object. If the object supports the method, then the computation proceeds.
- Duck-typing: if it looks like a duck, quacks like a duck, then it is a duck!

Dynamic typing

- This loose, comfortable, free-style, philosophy is at the heart of python.
- But... beware, you can get burned with this flexibility...
- Q: how many times did OCaml complain to you statically about something that was **NOT** a bug?

Thursday, November 10, 2011

Thursday, November 10, 2011

Similarities to ML

- Uniform model
 - everything is an object, including functions
- Pass functions around
 - functions are objects!
- Supports functional programming
 - map and fold

Other cool things about Python

- A lot of stuff that you may first think is a "language feature" is actually just translated under the hood to a method call...
- Very widely used, supported.
- Has libraries for all sorts of things.

Thursday, November 10, 2011

Thursday, November 10, 2011

Ok, let's start playing with Python!

- Like Perl, python is a "managed" or "interpreted" language that runs under the python environment, i.e. not compiled to machine code.
- Makes it convenient to rapidly write, check-in and test code!

Ways to run Python code

- At an interactive Python prompt
 - like "read-eval-print" loop of ML
- As shell scripts
- As stand-alone programs
 - run from the shell.

Thursday, November 10, 2011

Let's fire it up!

- Ok, let's give it a try...
- See lecture-10.py file for the rest...

Thursday, November 10, 2011

Thursday, November 10, 2011