CS 476 – Programming Language Design

William Mansky

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Java-Like Language: Semantics

- Values: ints, objects

- How should we represent an object?

Name	Value
x	3
y	5
getx()	return x;
sety(n)	y := n;

Point(x = 3, y = 5)

Java-Like Language: Semantics

- Values: ints, objects

- How should we represent an object?

Point(x = 3, y = 5)   Square(side = 4)   Item(type = “book”, len = 200)

- In general: $C(x_1 = v_1, \ldots, x_n = v_n)$ where C is the object’s class, v_1 is the value of its field x_1, etc.
  - Including fields inherited from superclasses!
- We can also write $C(fs)$ where fs is a map from fields to their values

Java-Like Language: Semantics

$$\frac{(e_1, \rho) \Downarrow v_1 \quad \dots \quad (e_n, \rho) \Downarrow v_n \quad (\text{fields}(\Gamma, C) = \tau_1\, f_1, \dots, \tau_n\, f_n)}{(\mathtt{new}\, C(e_1, \dots, e_n), \rho) \Downarrow C(f_1 = v_1, \dots, f_n = v_n)}$$

$$\frac{\qquad\qquad\qquad}{(e.f, \rho) \Downarrow}$$

- Exercise: Fill in the rule to give semantics for field access.

Java-Like Language: Semantics

$$\frac{(e_1, \rho) \Downarrow v_1 \quad \dots \quad (e_n, \rho) \Downarrow v_n \quad (\text{fields}(\Gamma, C) = \tau_1\ f_1, \dots, \tau_n\ f_n)}{(\texttt{new}\ C(e_1, \dots, e_n), \rho) \Downarrow C(f_1 = v_1, \dots, f_n = v_n)}$$

$$\frac{(e, \rho) \Downarrow C(f_1 = v_1, \dots, f_n = v_n)}{(e.f_i, \rho) \Downarrow v_i}$$

Java-Like Language: Semantics

$$\frac{(e_1, \rho) \Downarrow v_1 \quad \ldots \quad (e_n, \rho) \Downarrow v_n \qquad (\text{fields}(\Gamma, C) = \tau_1\ f_1, \ldots, \tau_n\ f_n)}{(\texttt{new}\ C(e_1, \ldots, e_n), \rho) \Downarrow C(f_1 = v_1, \ldots, f_n = v_n)}$$

$$\frac{(e, \rho) \Downarrow C(fs) \qquad (fs(f) = v)}{(e.f, \rho) \Downarrow v}$$

Java-Like Language: Contexts

- In IMP, the type context Γ stored function *signatures* and the runtime environment ρ stored function *definitions*

$$\frac{\Gamma[x_1 \mapsto \tau_1, \dots, x_n \mapsto \tau_n] \vdash c : \text{ok}}{\Gamma \vdash \tau\ f(\tau_1\ x_1, \dots, \tau_n\ x_n)\{\ c\ \} : \Gamma[f \mapsto \tau(\tau_1\ x_1, \dots, \tau_n\ x_n)]}$$

$$\frac{}{(\tau\ f(\tau_1\ x_1, \dots, \tau_n\ x_n)\{\ c\ \}, \rho) \to (\texttt{skip}, \rho[f \mapsto (x_1, \dots, x_n)\{c\}])}$$

- In Java, both typing and semantics might need the whole class declaration

Java-Like Language: Contexts

- In IMP, the type context Γ stored function *signatures* and the runtime environment ρ stored function *definitions*
- In Java, both typing and semantics might need the whole class declaration

$$\frac{(e_1, \rho) \Downarrow v_1 \quad \dots \quad (e_n, \rho) \Downarrow v_n \qquad (\text{fields}(\Gamma, C) = \tau_1\ f_1, \dots, \tau_n\ f_n)}{(\texttt{new}\ C(e_1, \dots, e_n), \rho) \Downarrow C(f_1 = v_1, \dots, f_n = v_n)}$$

- Java uses type information at runtime!
  - Every object is tagged with its class in memory
  - Used to find fields, figure out which version of a method to call, etc.
- Vs. IMP, C, etc., where types disappear at runtime

Functions: Semantics of Calls

$$\frac{(e_1, \rho) \Downarrow v_1 \quad \ldots \quad (e_n, \rho) \Downarrow v_n \quad (\rho(f) = (x_1, \ldots, x_n)\{\, c \,\})}{\begin{array}{c}(x = f(e_1, \ldots, e_n), k, \rho) \to \\ (c, (\rho, x) :: k, \rho[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n])\end{array}}$$

- Evaluate the arguments $e_1, \ldots, e_n$
- Look up f in ρ
- Execute the body of f and produce a return value
- Assign the return value to x

OO: Semantics of Methods

$$\frac{(e_1, \rho) \Downarrow v_1 \quad \dots \quad (e_n, \rho) \Downarrow v_n \quad (\rho(f) = (x_1, \dots, x_n)\{\, c \,\})}{\begin{array}{c}(x = e.m(e_1, \dots, e_n), k, \rho) \rightarrow \\ (c, (\rho, x) :: k, \rho[x_1 \mapsto v_1, \dots, x_n \mapsto v_n])\end{array}}$$

- Evaluate the arguments $e_1, \dots, e_n$
- Look up f in ρ
- Execute the body of f and produce a return value
- Assign the return value to x

OO: Semantics of Methods

$$\frac{(e_1, \rho) \Downarrow v_1 \quad \dots \quad (e_n, \rho) \Downarrow v_n \quad (\rho(f) = (x_1, \dots, x_n)\{\, c \,\})}{(x = e.m(e_1, \dots, e_n), k, \rho) \rightarrow (c, (\rho, x) :: k, \rho[x_1 \mapsto v_1, \dots, x_n \mapsto v_n])}$$

- Evaluate the arguments $e_1, \dots, e_n$ and the object e
- Look up m in the methods of e's class
- Execute the body of m (with `this` set to e) and produce a return value
- Assign the return value to x

OO: Semantics of Methods

$$\frac{(e,\rho) \Downarrow C(fs) \quad (e_1,\rho) \Downarrow v_1 \quad \dots \quad (e_n,\rho) \Downarrow v_n \quad (\rho(f) = (x_1, \dots, x_n)\{\, c \,\})}{(x = e.m(e_1, \dots, e_n), k, \rho) \rightarrow (c, (\rho, x) :: k, \rho[x_1 \mapsto v_1, \dots, x_n \mapsto v_n])}$$

- Evaluate the arguments $e_1, \dots, e_n$ and the object e
- Look up m in the methods of e's class
- Execute the body of m (with `this` set to e) and produce a return value
- Assign the return value to x

OO: Semantics of Methods

$$\frac{(e,\rho) \Downarrow C(fs) \quad (e_1,\rho) \Downarrow v_1 \quad \dots \quad (e_n,\rho) \Downarrow v_n \\ (\text{methods}(\Gamma, C) = \cdots, \tau\ m(\tau_1\ x_1, \dots, \tau_n\ x_n)\{\ c\ \})}{(x = e.m(e_1, \dots, e_n), k, \rho) \to \\ (c, (\rho, x) :: k, \rho[x_1 \mapsto v_1, \dots, x_n \mapsto v_n])}$$

- Evaluate the arguments $e_1, \dots, e_n$ and the object e
- Look up m in the methods of e's class
- Execute the body of m (with **`this`** set to e) and produce a return value
- Assign the return value to x

OO: Semantics of Methods

$$\frac{(e, \rho) \Downarrow C(fs) \quad (e_1, \rho) \Downarrow v_1 \quad \dots \quad (e_n, \rho) \Downarrow v_n \qquad (\text{methods}(\Gamma, C) = \cdots, \tau\, m(\tau_1\, x_1, \dots, \tau_n\, x_n)\{\, c\, \})}{(x = e.m(e_1, \dots, e_n), k, \rho) \to (c, (\rho, x) :: k, \rho[\texttt{this} \mapsto C(fs), x_1 \mapsto v_1, \dots, x_n \mapsto v_n])}$$

- Evaluate the arguments $e_1, \dots, e_n$ and the object e
- Look up m in the methods of e's class
- Execute the body of m (with **`this`** set to e) and produce a return value
- Assign the return value to x

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Java-Like Language: Casts

```
Building build(Building model);

class School extends Building

School s = new School();
s2 = w.build((Building) s);
((School) s2).getCourse();
```

- Exercise: When can we cast from one class to another?

Java-Like Language: Casts

```
Building build(Building model);

class School extends Building

School s = new School();
s2 = w.build((Building) s);
// upcast from School to Building

((School) s2).getCourse();
// downcast from Building to School
```

Java-Like Language: Casts

- Upcast: always safe, doesn't do anything
- Downcast: safe only if object actually has the right type – we might not know until runtime
- Other casts: ??

```
B b = new B();
A a = (A) b;
```

Java-Like Language: Casts

- Upcast: always safe, doesn't do anything
- Downcast: safe only if object actually has the right type – we might not know until runtime
- Other casts: ??

```
B b = new B();
A a = (A) ((Object) b);
```

Java-Like Language: Casts

- Upcast: always safe, doesn't do anything
- Downcast: safe only if object actually has the right type – we might not know until runtime
- Other casts: never work, but users can write them anyway

```
B b = new B();
A a = (A) ((Object) b);
```

$$\frac{\Gamma \vdash e : D \quad D <: C \text{ or } C <: D}{\Gamma \vdash (C)\, e : C}$$

Java-Like Language: Casts

- Upcast: always safe, doesn't do anything
- Downcast: safe only if object actually has the right type
- At runtime, we know the object's specific type!

$$\frac{(e,\rho) \Downarrow C(fs) \quad C <: D}{\big((D)\, e\,, \rho\big) \Downarrow C(fs)}$$

$$\frac{(e,\rho) \Downarrow C(fs) \quad \text{not } C <: D}{\big((D)\, e\,, \rho\big) \Downarrow \text{ClassCastException}}$$

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Java-Like Language: Syntax

CL ::= `class` <id> `extends` <id> { *T* <id>`;` … `;` *T* <id>`;` *M* … *M* }

M ::= *T* <id>`(`*T* <id>`,` … `,` *T* <id>`)`{ *C* }

P ::= *CL* … *CL*

E ::= <#> | *E* **+** *E* | <id> | … | *E*`.`<id>

C ::= <id> **=** *E* | … | <id> **=** *E*`.`<id>`(`*E*`,` … `,` *E*`)`
| <id> **=** `new` <id>`(`*E*`,` … `,` *E*`)`

T ::= `int` | <id>

Java-Like Language: Syntax

CL ::= `class` <id> `extends` <id> { *T* <id>; …; *T* <id>; *M* … *M* }

M ::= *T* <id>(*T* <id>, …, *T* <id>){ *C* }

P ::= *CL* … *CL*

E ::= <#> | *E* + *E* | <id> | … | *E*.<id>

C ::= <id> = *E* | … | <id> = *E*.<id>(*E*, …, *E*)
| <id> = `new` <id>(*E*, …, *E*) | *E*.<id> = *E*

T ::= `int` | <id>

Objects vs. Values

- We said “objects are values”

Objects:	Values:
Point(x = 3, y = 5)	5, true, etc.
can be stored in variables	can be stored in variables
have pieces that can change	can’t change
different objects can have the same values in them	if the value is the same, they’re equal

Objects vs. Values

- We said “objects are values”, but they’re also like variables!

Objects:	Values:
Point(x = 3, y = 5)	5, true, etc.
can be stored in variables	can be stored in variables
have pieces that can change	can’t change
different objects can have the same values in them	if the value is the same, they’re equal

Java-Like Language: Mutable Objects

- Our language so far has no field-set operation!

```
A a1 = new A(3, 5);
A a2 = a1;
a1.x = 4;
int result = a2.x;
```

- Exercise: What should the value of `result` be?

Java-Like Language: Mutable Objects

- Our language so far has no field-set operation!

```
A a1 = new A(3, 5);
A a2 = a1;
a1.x = 4;
int result = a2.x; // should be 4
```

Java-Like Language: Mutable Objects

- Our language so far has no field-set operation!

```
A a1 = new A(3, 5);
A a2 = a1;
a1.x = 4;
int result = a2.x; // should be 4
```

{a1 = A(x = 3, y = 5)}

Java-Like Language: Mutable Objects

- Our language so far has no field-set operation!

```
A a1 = new A(3, 5);          {a1 = A(x = 3, y = 5)}
A a2 = a1;                   {a1 = A(x = 3, y = 5), a2 = A(x = 3, y = 5)}
a1.x = 4;
int result = a2.x; // should be 4
```

Java-Like Language: Mutable Objects

- Our language so far has no field-set operation!

```
A a1 = new A(3, 5);        {a1 = A(x = 3, y = 5)}
A a2 = a1;                 {a1 = A(x = 3, y = 5), a2 = A(x = 3, y = 5)}
a1.x = 4;                  {a1 = A(x = 4, y = 5), a2 = A(x = 3, y = 5)}
int result = a2.x; // should be 4
```

Java-Like Language: Mutable Objects

- Our language so far has no field-set operation!

```
A a1 = new A(3, 5);            {a1 = r1, r1 -> A(x = 3, y = 5)}
A a2 = a1;                     {a1 = r1, a2 = r1, r1 -> A(x = 3, y = 5)}
a1.x = 4;                      {a1 = r1, a2 = r1, r1 -> A(x = 4, y = 5)}
int result = a2.x; // should be 4
a1 = new A(6, 7);              {a1 = r2, a2 = r1, r1 -> A(x = 4, y = 5),
                                r2 -> A(x = 6, y = 7)}
```

- Two-level model: variables hold references, references point to values

Java-Like Language: Mutable Objects

- Split the environment ρ into two levels
- Program state is now a tuple (c, k, ρ, σ) where:
  - c is the currently executing command
  - k is the call stack
  - ρ is the environment, mapping variables to either *primitive values* (int, bool) or *references*
  - σ is the *store*, mapping *references* to object values

Java-Like Language: Semantics

$$\frac{(e, \rho) \Downarrow C(fs) \quad (fs(f) = v)}{(e.f, \rho) \Downarrow v}$$

Java-Like Language: Semantics

$$\frac{(e, \rho, \sigma) \Downarrow r \quad \sigma(r) = C(fs) \quad (fs(f) = v)}{(e.f, \rho, \sigma) \Downarrow v[i]}$$

$$\frac{(\rho(x) = v)}{(x, \rho, \sigma) \Downarrow v}$$

$$\frac{(e, \rho, \sigma) \Downarrow v}{(x = e, \rho, \sigma) \rightarrow (\texttt{skip}, \rho[x \mapsto v], \sigma)}$$

Java-Like Language: Semantics

$$\frac{(e_1, \rho) \Downarrow v_1 \quad \dots \quad (e_n, \rho) \Downarrow v_n \quad (\text{fields}(\Gamma, C) = \tau_1\ f_1, \dots, \tau_n\ f_n)}{(\texttt{new}\ C(e_1, \dots, e_n), \rho) \Downarrow C(f_1 = v_1, \dots, f_n = v_n)}$$

Java-Like Language: Semantics

$$\frac{(e_1, \rho, \sigma) \Downarrow v_1 \;\; \dots \;\; (e_n, \rho, \sigma) \Downarrow v_n \quad (\text{fields}(\Gamma, C) = \tau_1\, f_1, \dots, \tau_n\, f_n) \quad \left(r \notin \text{dom}(\sigma)\right)}{(x = \texttt{new}\; C(e_1, \dots, e_n), \rho, \sigma) \to (\texttt{skip}, \rho[x \mapsto r], \sigma[r \mapsto C(f_1 = v_1, \dots, f_n = v_n)])}$$

$$\frac{(e, \rho, \sigma) \Downarrow r \quad \left(\sigma(r) = C(fs)\right) \quad (e_1, \rho, \sigma) \Downarrow v}{(e.f \texttt{=} e_1, \rho, \sigma) \to (\texttt{skip}, \rho, \sigma[r \mapsto C(fs[f \mapsto v])])}$$

Java-Like Language: Mutable Objects

Code	ρ (variables)	σ (memory)
`A a1 = new A(3, 5);`	{a1 = r1}	{r1 -> A(x = 3, y = 5)}
`A a2 = a1;`	{a1 = r1, a2 = r1}	{r1 -> A(x = 3, y = 5)}
`a1.x = 4;`	{a1 = r1, a2 = r1}	{r1 -> A(x = 4, y = 5)}
`int result = a2.x; // should be 4`		
`a1 = new A(6, 7);`	{a1 = r2, a2 = r1}	{r1 -> A(x = 4, y = 5), r2 -> A(x = 6, y = 7)}

- Two-level model: variables hold references, references point to values

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Homework 5 Overview

- Syntax: types, expressions, commands, declarations
- Records in OCaml
- Field and method lookup
- type_of and typecheck_cmd