

Modular Multiple Dispatch with Multiple Inheritance

Sukyoung Ryu

Joint work with Eric Allen, Joseph Hallett, Victor Luchangco and Guy Steele

Sun Microsystems Laboratories

March 12, 2007

Fortress Overloading Goal

- No *ambiguous* nor *undefined* calls at run time
- **M**odular **M**ultiple dispatch with **M**ultiple inheritance

Modern Object-Oriented Languages

- Good for data extensibility
- Bad for function extensibility

```
trait Flower
  color():String
end
object Rose extends Flower
  color() = "Red"
end
object Lily extends Flower
  color() = "White"
end
```

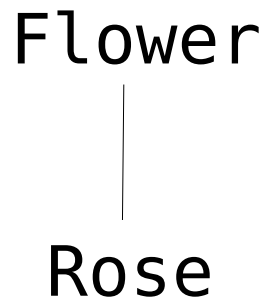
Overloaded Top-Level Functions

- Multiple declarations with the same name for function extensibility

```
color(r: Rose) = "Red"  
color(r: Lily) = "White"
```

- Dynamic dispatch selects the most specific definition at run time

Dynamic Dispatch



```
countRoses(x: Flower, y: Flower) = 0  
countRoses(x: Flower, y: Rose) = 1  
rose: Flower = Rose  
countRoses(rose, rose)
```

- Single dispatch: 0
- Multiple dispatch: 1

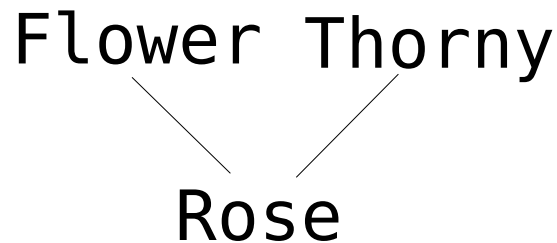
Multiple Dispatch and Ambiguity

Flower
|
Rose

```
countRoses(x: Flower, y: Flower) = 0  
countRoses(x: Flower, y: Rose) = 1  
countRoses(x: Rose, y: Flower) = 1  
rose: Flower = Rose  
countRoses(rose, rose)
```

- Ambiguous call!

Multiple Inheritance and Ambiguity



```
trait Flower end
trait Thorny end
object Rose extends { Flower, Thorny } end

toString(x: Flower) = "Flower"
toString(x: Thorny) = "Thorny"

toString(Rose)
```

- Ambiguous call!

Modular Check for Ambiguity

- Want static restrictions.
- Most restrictions require whole program analysis.
- Millstein and Chambers present modular restrictions, but no *multiple inheritance*.

Language

- Components
 - > Import other components but modularly checked
- Traits `trait Flower end`
 - > Multiple inheritance of code without fields
- Objects `object Rose extends Flower end`
 - > Leaves of type hierarchy containing fields
- Exclusive types `trait Animal excludes Flower end`
 - > No object is a subtype of excluding traits.
 - > Objects exclude one another.

Language (Continued)

- Top-level functions

```
color(r: Rose) = "Red"  
color(r: Lily) = "White"
```

- (Functional) Methods: explicit **self** parameter in the parameter list, rather than an implicit **self** parameter before the method name

```
trait Matrix excludes Vector  
  opr ·(self, other: Vector): Matrix  
  opr ·(other: Vector, self): Matrix  
end  
  
v · M + M · v
```

Overloading Rules

- Compare overloaded declarations pairwise.
- If any rule holds then a valid overloading:
 - > Exclusion Rule
 - > Subtype Rule
 - > Meet Rule

Exclusion Rule

- Parameter types exclude each other.

```
trait Animal excludes Flower end
eat(who: Animal, what: Flower): Boolean
eat(who: Flower, what: Animal): Boolean
```

Subtype Rule

- Parameter type of one declaration is a subtype of the other.
- Return types must also be in subtype relation.

```
characteristic(x: Flower): Object  
characteristic(x: Rose): Thorny
```

Meet Rule for Functions

Flower



Rose

countRoses(*x*: Flower, *y*: Rose) = 1

countRoses(*x*: Rose, *y*: Flower) = 1

countRoses(*x*: Rose, *y*: Rose) = 2

- Exists a declarations that is more specific than both.

Meet Rule for Methods

- Treating methods like functions is too restrictive.

```
trait Flower
  name(self)
end
trait Thorny
  name(self)
end
object Rose extends { Flower, Thorny }
  name(self) = "Rose"
end
```

Meet Rule for Methods

- Ambiguity due to **self** parameter position

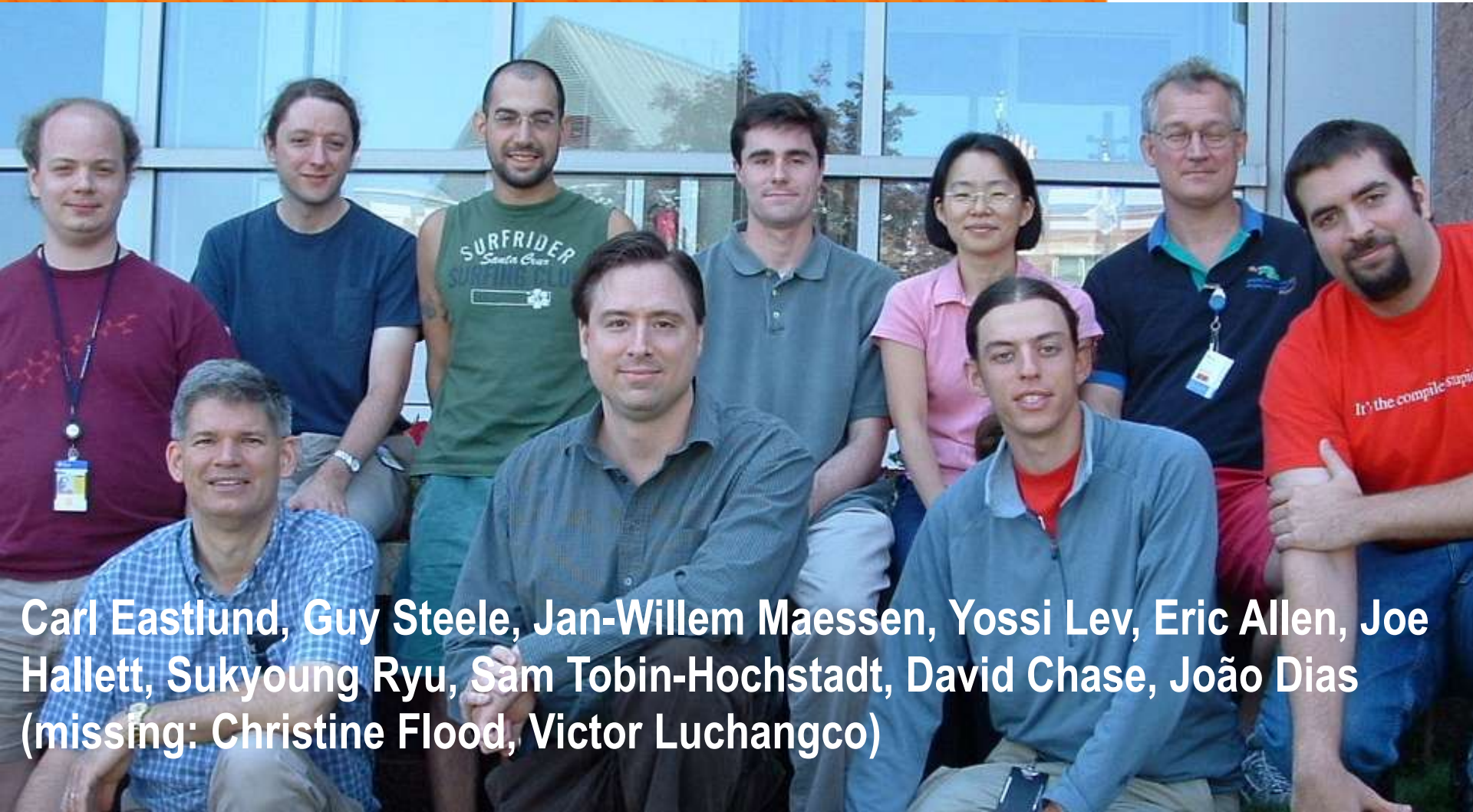
```
object Rose extends Flower
  countRoses(self, l: Lily)
end
object Lily extends Flower
  countRoses(r: Rose, self)
end
countRoses(Rose, Lily)
```

- **self** parameters must be in the same position.

Fortress Overloading

- Object-oriented language with overloaded top-level functions for functionality extension
- Multiple implementation inheritance
- Multiple dispatch for overloaded functionals
- Modular checks to prevent ambiguous or undefined calls at run time

<http://research.sun.com/projects/plrg>



Carl Eastlund, Guy Steele, Jan-Willem Maessen, Yossi Lev, Eric Allen, Joe Hallett, Sukyoung Ryu, Sam Tobin-Hochstadt, David Chase, João Dias
(missing: Christine Flood, Victor Luchangco)