Polymorphism in languages and its implications
(Smalltalk, Modula-3, Java, and C++)

Amer Diwan

Outline

• Not much to say about ad-hoc polymorphism in languages
• Case studies from inclusion polymorphism
  – “Smalltalk”
  – Modula-3
  – Java
  – C++
• Case study from parametric polymorphism
  – SML
Smalltalk

• Example:
  myThings ← Bag new
  myThings add (Point new)
  myThings add (Rectangle new)

• Point and Rectangle may be unrelated classes (i.e.,
  common supertype is Object)

Example (cont.)

• myThings do: [ :oneThing | oneThing print]
  Above code works on any collection of any classes
• “Implicitly” the items in the collection must have
  a “print” method
Discussion

• Does Smalltalk’s polymorphism fits in with Cardelli and Wegner’s classification?

Run-time representation
(from Mitchell’s book)
How to implement method dispatch?

• Steps:
  – Get class object
  – Get method dictionary
  – Search method dictionary
    • If found, invoke code
    • If not found, continue search in superclass
    • If reached “Object” then “method not understood”

Finishing up the example
How to improve the performance of method dispatch?

• Search is very expensive. What to do?

Modula-3

• Polymorphism is a direct consequence of subtyping
• Example:
  
  TYPE printableBag =
  OBJECT
  values: ARRAY OF printableThing;
  METHODS add(v: printableThing) …
  VAR myThings: printableBag;
  myThings.Add(new point); myThings.Add(new Rect)
Example (Cont.)

- FOR i = 0 TO LAST(myThings.values) DO
  myThing.values[i].print()
- How is this different from Smalltalk?
- What kind of polymorphism is this in Cardelli and Wegner’s terminology?

Visualizing object types in Modula-3

Subtyping restricted to simple inclusion for objects
Discussion

• How to represent in Modula-3?
  – value moveX = fun(p: Point, dx: Int)(returns Point)
    p.x = p.x + dx; p

    (returns P)
    p.x = p.x + dx; p

How to implement method dispatch?

• Can do it like Smalltalk but can we exploit
  – Static typing?
    • Possible types of object is constrained statically
  – Single inheritance?
    • Exactly one immediate supertype
V-Tables

- Idea:
  - Prepend the methods of a supertype to a subtype
  - A method T::m appears in the same position in all T's subclasses

V-tables (cont.)

Construct a v-table for each class

v-tables are typically part of the type descriptor
V-tables (cont.)

t->g() becomes
vp = t->vtable_ptr

gaddr = *(vp+g’s offset)

(*gaddr())

How to improve the performance of method dispatch?
C++

- Similar to Modula-3 except for
  - virtual/non-virtual distinction
- Multiple inheritance

Visualizing object types in C++

Multiple inheritance allows classes that partially overlap
Challenges with multiple inheritance

- **Conflicts:**
  - e.g., what if two inherited methods have the same name?

- **Implementation**
  - class C : A, B { … }
    C *pc; B* pb;
    pc = new C;
    pb = (B*)pc;

Implementation issues

- Need to adjust pointers when casting, invoking methods, comparing, ...

```c
if (pc == pb) {
    pb = (B*)pc;
    pc = (C*)pb;
    if (pc == pb) { … }

pc
```
An example involving virtual functions

- class A { virtual void f(); }
  class B { virtual void f(); virtual void g(); }
  class C : A, B { void f() {...this... } }
  B *pb = new C;
pb->f();
- Challenge?

Example continued

Now we can invoke a virtual function after casting a C object to B.
But the body of f() expects “this” to be a C object not a B object!
Example continued

\[\text{pb} \rightarrow f() \Rightarrow \]
\[\text{vt} = \text{pb} \rightarrow \text{vtbl}[\text{index}(f)]\]
\[(\ast \text{vt} \rightarrow \text{fct})(B\ast)((\text{char} \ast)\text{pb} + \text{vt} \rightarrow \text{delta})]\n
How to speed up method dispatch?
Discussion

• How to represent in C++?
  – value moveX = fun(p: Point, dx: Int)(returns Point)
    p.x = p.x + dx; p

    (returns P)
    p.x = p.x + dx; p

C++ and bounded quantification

• Multiple inheritance does not result in bounded
  universal quantification.
• But it provides a richer type system for expressing
  subtyping
Java

• Similar to Modula-3 except for `invokeinterface`
• Example:
  ```java
  interface hasAPrint { void print(); }
  interface hasASize { void size(); }
  class text implements hasAPrint {
    void print() {...} ... ;
  }
  class list implements hasASize {
    void size(...) {...}
  }
  class figure implements hasASize, hasAPrint {
    void size(...) {...}; void print(...) {...} ...
  }
  hasAPrint anobj; anobj.print();
  What’s the problem here?
  ```

Implementing `invokeinterface`

• **Problem:** `print` method may have different offsets in each class implementing `hasAPrint`
• **Solution:**
  – Can implement `invokeinterface` using Smalltalk-like run-time method search
  – (better idea) Have v-tables for interfaces that are hashed using (class, interface name) pair
Visualizing object types in Java

Multiple inheritance of interfaces

But single inheritance of implementations (classes)

How to handle the mismatch?

Single inheritance of classes

And many other possibilities...
... but awkward
Discussion

• Does Java’s multiple inheritance of interfaces require the “deltas” needed in C++?

The big picture

<table>
<thead>
<tr>
<th>Dictionary lookup</th>
<th>V-table lookup</th>
<th>V-table + Slower hash lookup</th>
<th>V-table lookup + adjustments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untyped + Single inheritance</td>
<td>Single inheritance</td>
<td>Single inheritance + interfaces</td>
<td>Multiple inheritance</td>
</tr>
<tr>
<td>Inclusion polymorphism</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discussion

• Languages implement inclusion polymorphism very differently
  – Different performance
  – Different expressiveness
• Why did different languages make different decisions?

Next lecture

• Parametric polymorphism case study: SML
• Reading: SML document, Bob Harper (link from class web page)