AspectJ as a Polyglot extension

- the frontend of abc -
Roadmap

• What is Polyglot?

• Brief overview of the AspectJ extension

• Sketch of disambiguation of “this” in ITDs

• Summary
What is Polyglot?

An extensible Java compiler

Sample extensions:

- Jif: Java information flow and program partitioning
- PolyJ 2.0: Java with parameterized types
- JMatch: Abstract iterable pattern matching for Java
- Jx: Nested inheritance in Java
- Jedd: BDD-based analyses
- JPred: Practical predicate dispatch

Produced by Andrew Myers, Nate Nystrom *et al.* at Cornell
How does Polyglot do it?

- Structured as a series of visitors
- Each visitor pass rewrites AST; about 15 such visitors
- Rigorous use of interfaces and factories makes it easy to change type system, environment, ... 
- Delegates for overriding members of non-final AST classes (cf. intertype decls)
The AspectJ extension

Like any other Polyglot extension, five new packages:

- **AST**: new ast nodes (89 classes)
- **Extension**: overrides of existing Java AST nodes (13 classes)
- **Parse**: new lexer and grammar (2 files)
- **Types**: new types and type system (8 classes)
- **Visit**: new passes (35 classes)

- Includes Java/AspectInfo separator
- Many AST classes in pointcut language are light-weight
- The tricky bits are the type rules for ITDs, and the separator into Java & AspectInfo
Example: intertype scope rules

```java
public class A {
    int x;
    class B { int x; }
}

aspect Aspect {
    static int x;
    static int y;
    int A.B.foo() {
        class C {
            int x = 3;
            int bar() {return x + A.this.x;}
        }
        return this.x + (new C()).bar() + y;
    }
}
```
Example: intertype scope rules

```java
public class A {
    int x;
    class B { int x; }
}

aspect Aspect {
    static int x;
    static int y;
    int A.B.foo() {
        class C {
            int x = 3;
            int bar() {return x + A.this.x;}
        }
        return this.x + (new C()).bar() + y;
    }
}
```

need to disambiguate field references:
- may be a reference to aspect fields,
- local class fields,
- or host (=target) of intertype declaration

Rules:
- no explicit receiver? if it was introduced into environment by the host, give it “this” from host.
- explicit “this” or “super”? if there is no qualifier and we're not inside a local class, it refers to the host. If there is a qualifier Q, and there is no enclosing instance of type Q nested inside the ITD, it refers to the host if the host has an enclosing instance of type Q.
How to disambiguate “this”

- Extend context type in Polyglot
- Test to determine whether this refers to host
- Override disambiguate for Polyglot this.
New context type

```java
public interface AJContext extends Context {
    Context pushHost(ClassType ct, boolean declaredStatic);
    // called when entering itd
    ClassType hostClass(); // return target of current itds
    boolean inInterType(); // are we inside an intertype declaration?
    boolean nested(); // are we inside a local class in an intertype declaration?

    // other itd-related members...
    boolean varInHost(String name);
    boolean methodInHost(String name);
    ClassType findFieldScopeInHost(String name);
    ClassType findMethodScopeInHost(String name) throws SemanticException;
    // ... more for advice and declare decls ...
}
```
Does “this” refer to host of ITD?

types.AJTypeSystem_c

```java
public boolean refHostOfITD(AJContext c, Typed qualifier) {
    if (!c.inInterType())  // if not inside an ITD, cannot refer to a host
        return false;
    if (qualifier == null)  // if there is no qualifier
        return !c.nested();  // it refers to the host if we're not in a local class
    else  // otherwise look for enclosing instance in host
        return c.hostClass().hasEnclosingInstance(qualifier.type().toClass());
}
```
Override disambiguate

extension.AJSpecial_c (Special is the Polyglot class to represent “this”):

```java
public Node disambiguate(AmbiguityRemover ar) throws SemanticException {
    AJContext c = (AJContext) ar.context();
    AJTypeSystem ts = (AJTypeSystem) ar.typeSystem();
    if (!ts.refHostOfITD(c, qualifier())) {
        // this is an ordinary special, it does not refer to the host
        return super.disambiguate(ar);
    } else {
        // this is a host special
        AJNodeFactory nf = (AJNodeFactory) ar.nodeFactory();
        HostSpecial_c hs = (HostSpecial_c) nf.hostSpecial(position, kind,
                                                               qualifier, ((AJContext)c).hostClass());
        return hs.type(type()).disambiguate(ar);
    }
}
```
Frontend summary

- Extensible in all dimensions:
  - syntax, type system, visitors
- Potential merge problems with pure Java compiler only occur in extension dir and type system
- Extensions to abc have same structure as abc itself