-tokyo tech-

TOKYO INSTITUTE OF TECHNOLOGY

Predicate dispatch for Aspect-Oriented Programming

Shigeru Chiba Tokyo Institute of Technology

VMIL 2008

—токуо tech

TOKYO INSTITUTE OF TECHNOLOGY

Understanding AOP

- What are primitive operations of AOP?
 - pointcut and advice?
- Primitives of OOP
 - dynamic method dispatch
 - instances
 - inheritance

2

токүо Тесн

TOKYO INSTITUTE OF TECHNOLOGY

OOP v.s. AOP

- Dynamic method dispatch
 OOP
 - function calls are virtual.
- Virtual Join Points

AOP

- [Bockish et al, SPLAT '06]
- [Haupt & Schippers, ECOOP '07]
- Join points are virtual
 - function calls, field accesses, ... and more

TOKYO TECH

TOKYO INSTITUTE OF TECHNOLOG

A way of overriding

- OOP
 - single dispatch, multi-dispatch, and predicate dispatch.
- AOP
 - more complex dispatch than predicate dispatch.
 - how?

-TOKYO TECH

TOKYO INSTITUTE OF TECHNOLOGY

TOKYO INSTITUTE OF TECHNOLOGY

This work

- What predicates do we need for AOP?
 - if we emulate AOPby predicate dispatch + open classes.
 - This would be beneficial for understanding AOP and the differences from OOP.

-rokyo tech

TOKYO INSTITUTE OF TECHNOLOGY

Predicate dispatch

- M. Ernst et al. [1998], Millstein [2004], ...
- Jpred
 - type, equality, linear arithmetic

```
class TypeCheck {
  Type check(Env e, Node n) when n@IfStmnt { ... }
  Type check(Env e, Node n) when n@WhileStmnt { ... }
  Type check(Env e, Node n) when n == null { ... }
}
```

6

такуа несн Logging aspect

• needs context-dependent dispatch.

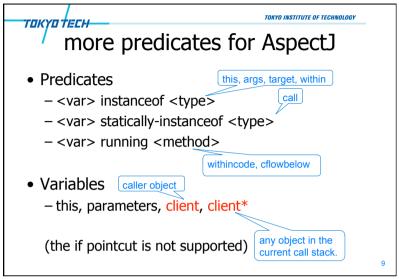
TOKYO TECH

TOKYO INSTITUTE OF TECHNOLOGY

Crosscutting concerns

- need to determine a method by global contexts.
 - within and withincode
 - e.g. call(* Rect.move(..)) && within(Window)
 - cflow
 - who is the caller?

Jpred determines a method by only local contexts.





Drawbacks

- Jpred does not provide these predicates because JPred statically and modularly checks:
 - Exhaustiveness
 - no "message not understood" error happens during runtime.
 - No ambiguity
 - no "multiple methods are applicable" error happens during runtime.

10

Exhaustiveness

- No "message not understood" error
 - Open classes requires global analysis.
 - all aspects must be given at compile time.
 - client/client* (cflow) disable static check.
 - Or very conservative analysis (require the "base/default" method)

AspectJ's approach

TOKYO TECH

TOKYO INSTITUTE OF TECHNOLOGY

No Ambiguity

- No "multiple methods are applicable" error
 - Determine the most specific method.
 - JPred determines it by logical implication among predicates.
 - AspectJ determines it by the precedence relations among aspects.

токүо тесн

TOKYO INSTITUTE OF TECHNOLOGY

A predicate representing precedence

- deployed(<aspect>)
 - deployed(A) is true if the aspect A is deployed.
 - deployed(A) implies deployed(B)if A has a higher precedence than B.

For normal methods,

- deployed(null) is always true
- deployed(A) always implies deployed(null)

13

TOKYO TECH

TOKYO INSTITUTE OF TECHNOLOGY

Modularly checkable?

- Yes for JPred.
- No for AOP?
 - Only deploy(<aspect>) is used for determining
 - All aspects must be given at compile time.
 - client/client* (cflow) requires very conservative analysis.

14

-такүа Тесн

TOKYO INSTITUTE OF TECHNOLO

Generic advices

- Pattern match
 - * (wildcard), + (subclasses), .. (any parameters)
 - e.g. javassist.CtClass+, Shape.set*(..)
- Solution
 - Multi-mixin [Apel 2005]
 - We can deal with an aspect as a template.
 A template instance is supplied to each class.
 - How to emulate proceed(), args(), …??

15

такүа Тесн-

TOKYO INSTITUTE OF TECHNOLOG

Code example

• Class with pattern matching

- Predicates
 - target(java.util.Map+)
 - methodName(put*)

TOKYO TECH

TOKYO INSTITUTE OF TECHNOLOGY

Aspect instances

- In AspectJ,
 - an aspect is instantiated. An advice is executed on that instance.
 - Solution
 - Overriding methods can be forwarders to an aspect instance.

17

-tokyo tech-

TOKYO INSTITUTE OF TECHNOLOGY

Summary

- AOP is as OOP with:
 - an extended method dispatching mechanism
 - global-contexts v.s. modular type check
 - Future work
 - compare AOP with other language constructs
 - FOP, Classbox, eJava, JPred, ..
 - design a better AOP language
 - we can borrow ideas from other languages