Graph Rewriting Techniques for Modeling AOP Semantics- influence on software evolution

Alon Amsel @ EVA

03/05/2004

Presentation Overview

- Motivation
- Model Overview
  - Base program representation
  - Advice
  - Pointcuts
  - Weaving
- Evolution
  - Aspect-oriented refactoring
  - Pointcut evolution

Motivation

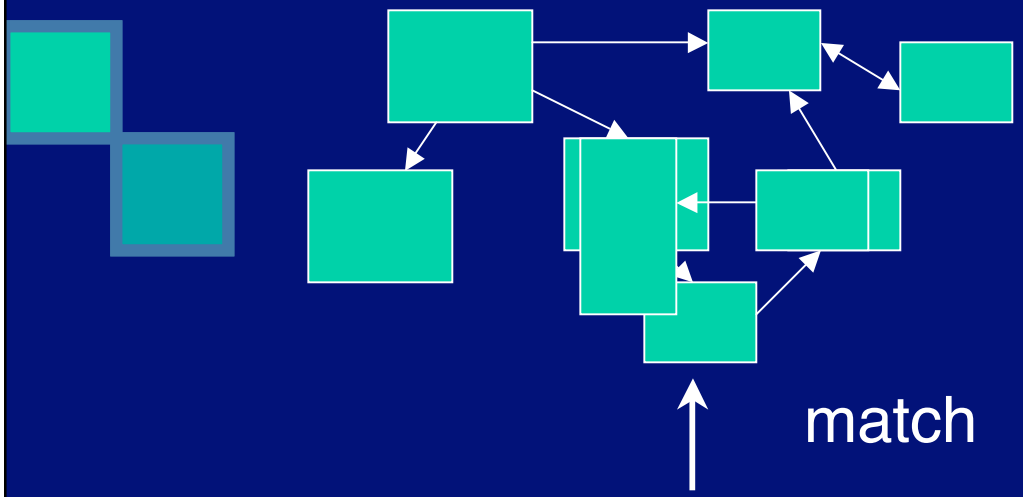
- Need of Formal Models
  - Understandability
  - Comprehensivity
  - Predictability
  - Clean semantics
  - AOP supporting tools
    - Support for refactorings
    - Verification
- $\prod$/ λ - calculus are fine but …
  - Advantages of visual languages

[Heckel,Engels]

Overview of the Model

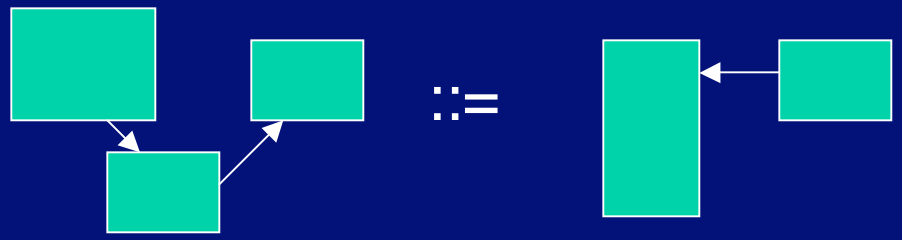
- Dynamic Program Representation
- Based on Graph (Rewriting) semantics for Object Orientation [Grogono,…]
- GR-based advice
- GR based pointcut matching
- GR based weaving [Aßmann]

Graph Rewriting in a nutshell



match

Rule:

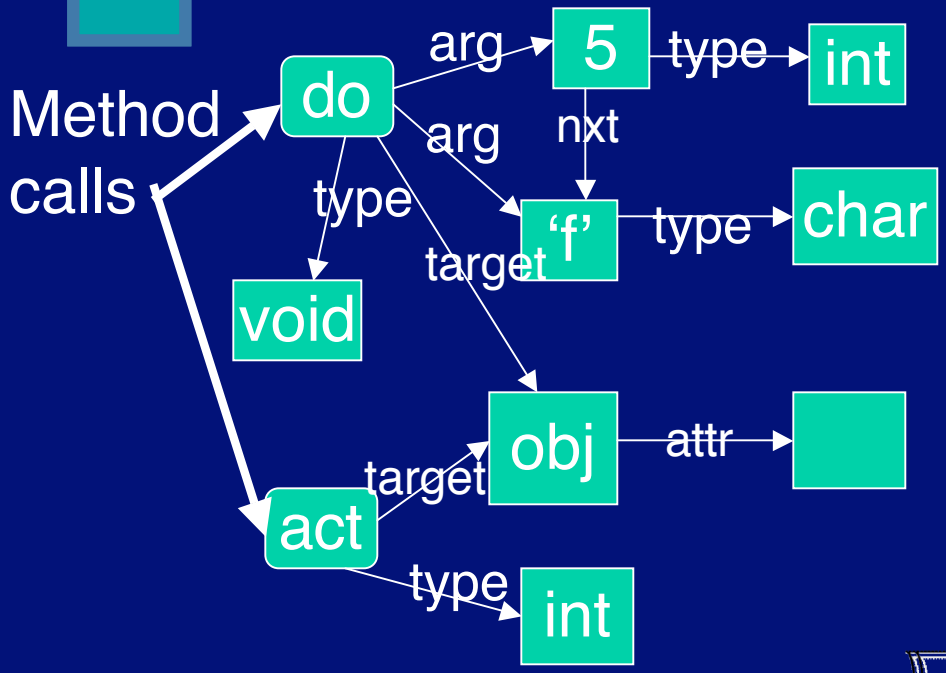


::=

Left Hand Side

Right Hand Side

Object Orientation & Graph Representation



Object Orientation and GRS (continued)

- Methods = rewriting rules
- Program is specified by a set of graph rewriting rules

Advice: *Spectative Aspects* [Katz]

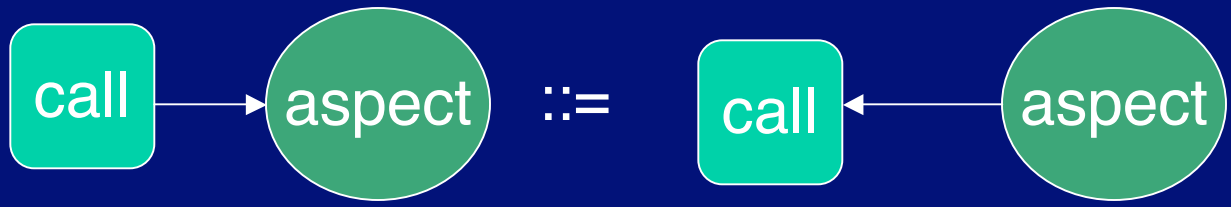
- Before and After in Pairs
- Each specifies a rule
- *Aspect Node*
- Connect to call node
- *Aspect View* :
  - call node
  - Aspect node
  - Aspect data
  - Edges connecting them

Advice: *Invasive/Modifying Aspects*

- *Aspect View* : direct neighbors of Aspect Node + edges
- Connect aspect node to all objects it interacts directly with

Before Advice

Subrule of every preliminary rule



Usually only precondition checking

After Advice (*Updating Rule)*

Subgraph of Left Hand Side of every updating rule

Includes around advice
To be composed with base program

So far…

- Base Program
  - Set of rewriting rules
- Aspects
  - Set of (pairs of) rewriting rules
- What next?
  - Which rules to combine
  - How to combine them

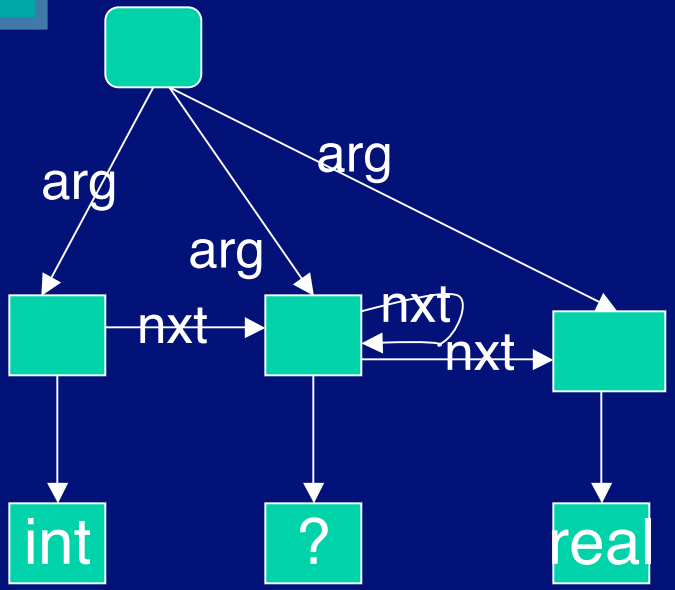
Pointcuts

- Pointcuts are specified graphically
- Each *pointcut graph* has same structure as in BP-rules
- Wildcards allowed
- Pair of pointcut graphs specify *pointcut rule*

Pointcut Matching

- Pattern matching
- Pointcuts match rules, not graphs!
- After the matching:
  - Aspect Node connects to corresponding call nodes
  - Rules are now ready to be instrumented

Example pointcut rule



This pointcut graph matches every RHS graph

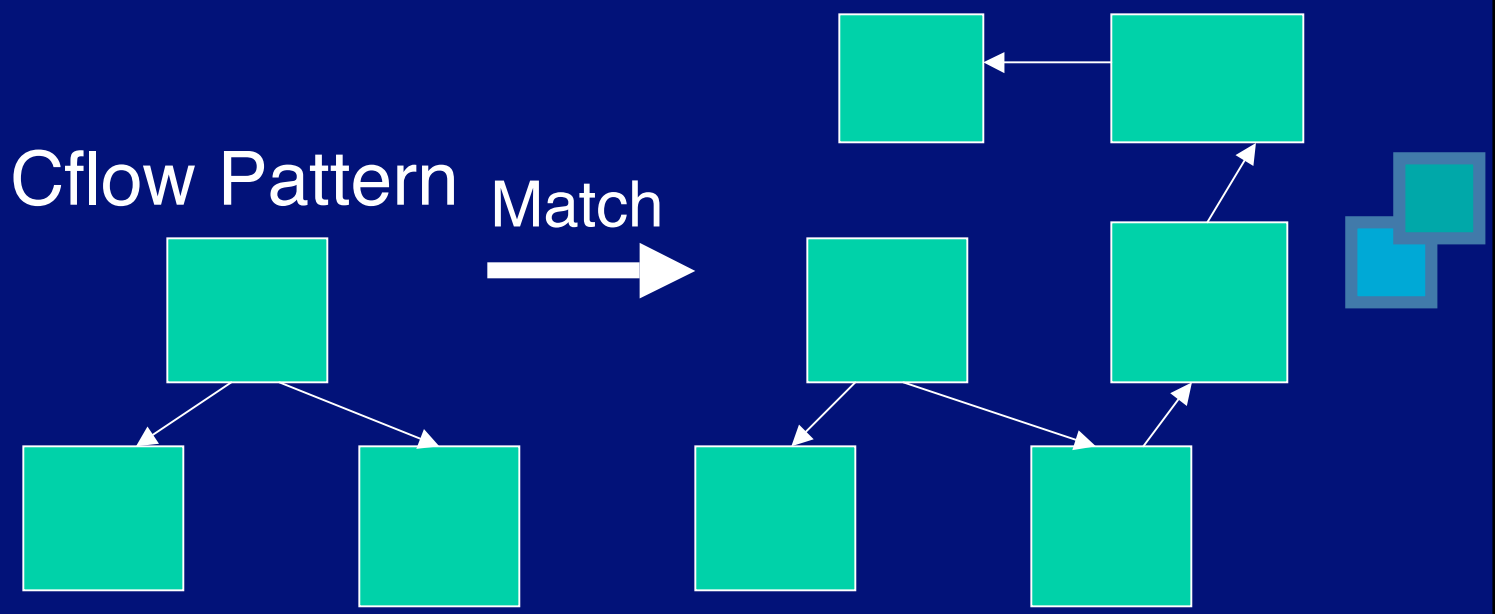
::= ⊤

args(int,..,real)

Expressiveness of Pointcut Rules

- AspectJ's pointcuts
  - call (method name & return type)
  - within,args,target
  - execution not fully defined yet
  - -> low level statically determinable pointcuts

Dynamic Pointcut Matching?



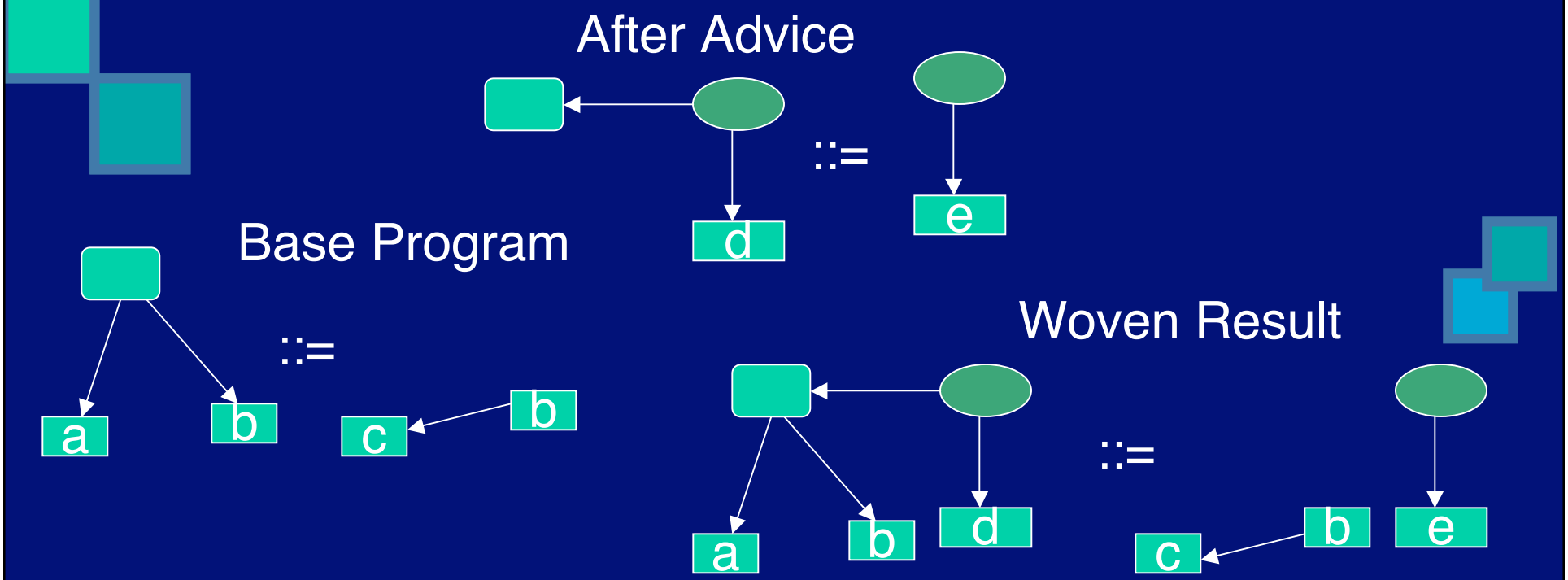
Dynamic Pointcut Matching?

- Rule Application part of larger configuration graph
- Rule definitions have to be bypassed

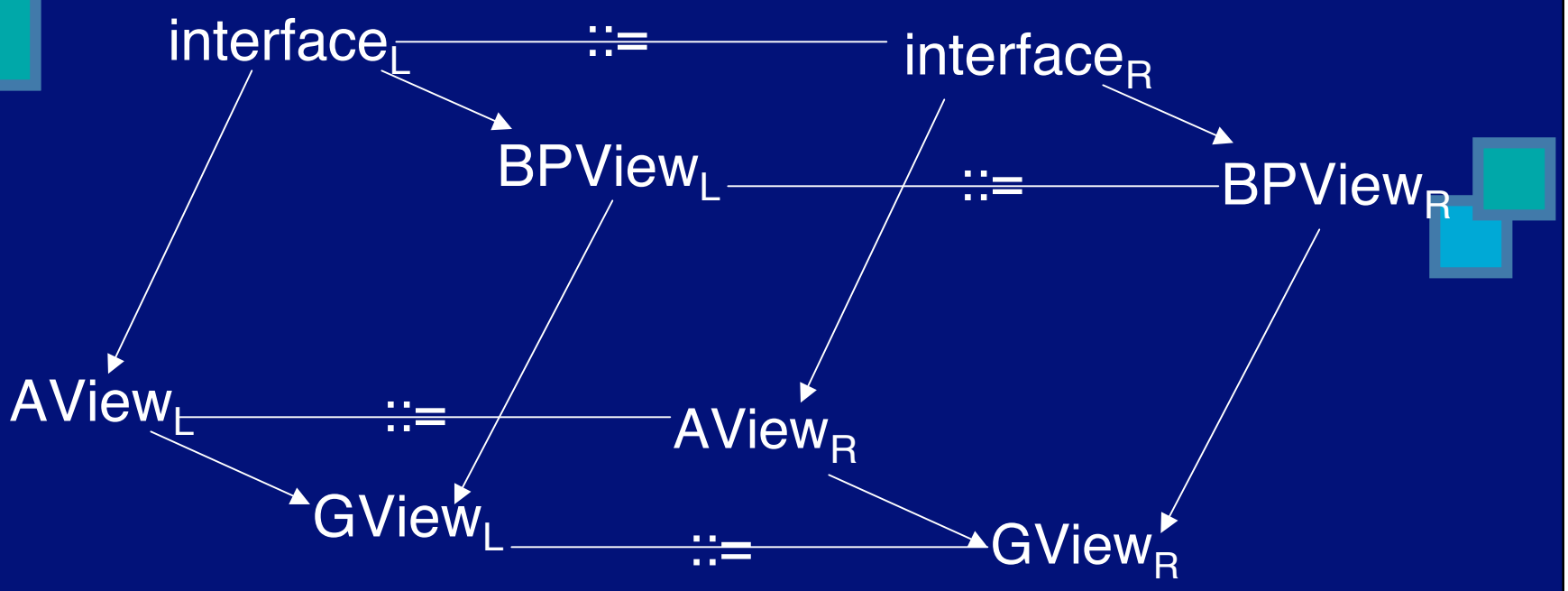
Weaving = Synchronous Composition of Rewriting Rules

- Base Program
- Advice
- Overlap => all rules to be applied simultaneously
- Gluing Rules over an Interface
- Weaving more aspects: in turn on top of BP

Weaving Example



Interface: spectative aspects closer look



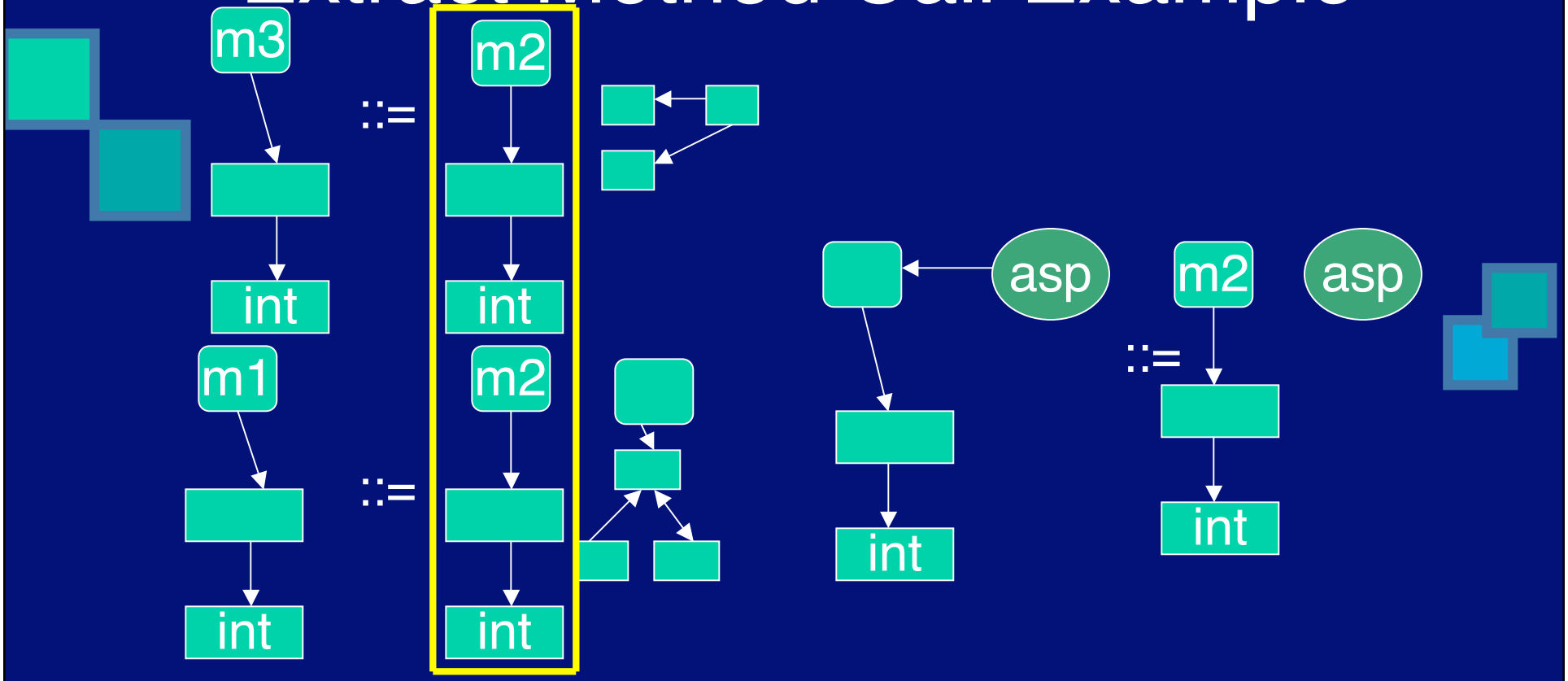
Model Implementation

- Graph Rewriting Tool ATOM^3 [de Lara]
  - A Tool for Multiformalism Meta-Modeling
  - +Intended for simulation of dynamical systems
  - +Allows definition of own diagrams, metamodels and transformations
  - -Not very stable (version 2.2)
  - -Python programming language
- Still buggy implementation
- No Aspect Code Generation yet

Aspect-Oriented Refactoring: two topics in software evolution

- Extracting code into an aspect / improving the aspect structure
- Modifying code into which aspects are woven
  - Satisfactory pointcut declaration

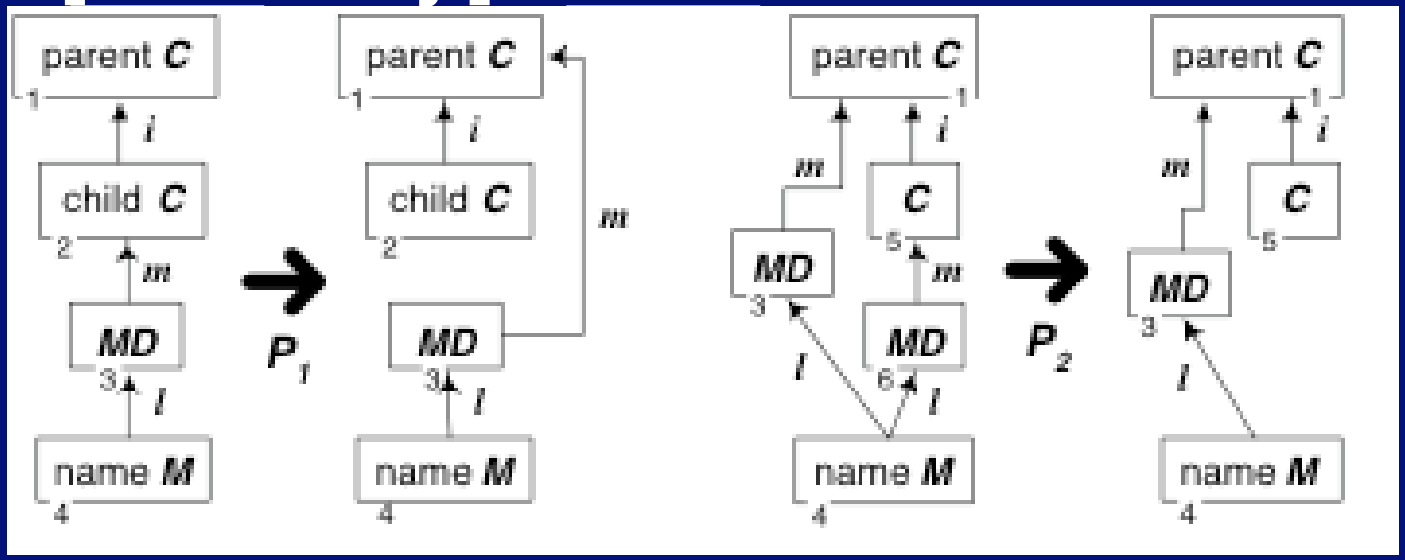
Extract Method Call Example



Are Pointcut Graphs Satisfactory?

- Refactoring -> do pointcuts still include the desired join points?
- Depending on how well they are used
- +expressive and intuitive
- Pull-up method AspectJ problem

Pull-up method Problem [Barzilay]



[Van Eetvelde]

AspectJ Pointcut: call(void B.f())

References

- Aβmann, Ludwig: Aspect weaving by graph rewriting
- Barzilay et al: Call and execution semantics in AspectJ
- Grogono: Graph semantics for OOP
- Heckel,Engels: Graph Transformation and Visual Modeling Techniques
- Katz: Aspects and Superimposition
- Van Eetvelde,Janssens: A Hierarchical Program Representation for Refactoring