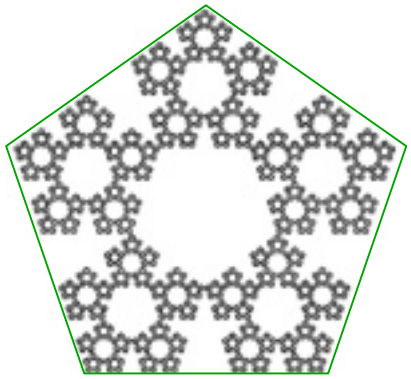
2nd Belgian-Dutch workshop on
Software Evolution



BENEVOL 2004

8-9 July 2004
University of Antwerp
Belgium

UMH

Problem statement

- More and better tool support needed for software evolution
  - traceability management
  - version control (e.g., software merging)
  - impact analysis
  - change propagation
  - consistency maintenance
  - model transformation
  - co-evolution
  - analysing release histories
  - a "theory of software evolution"

- Formalisms can be helpful for some of these tools

Critical pair analysis of graph transformations

for software refactoring

Tom Mens
Service de Génie Logiciel
Université de Mons Hainaut
http://www.umh.ac.be/~genlog

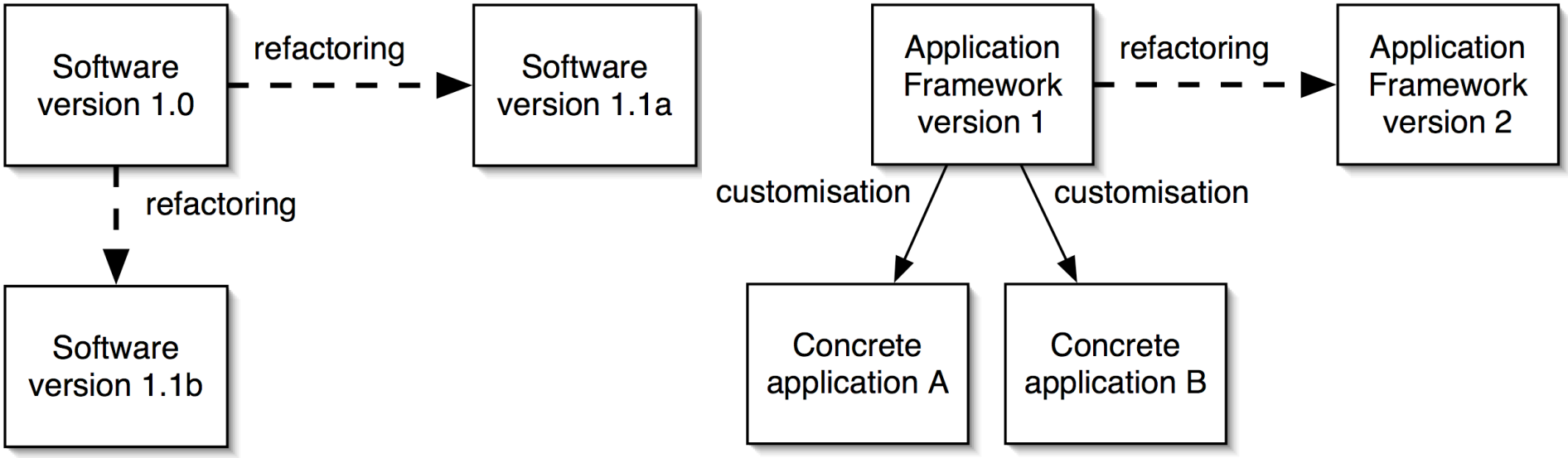
Case study: Graph transformation

- Formalism based on
  - graphs: to represent software entities
  - graph transformation: to represent software evolution
  - offers many theoretical results that can help during analysis
    - type graph, negative application conditions, parallel and sequential (in)dependence, confluence, critical pair analysis

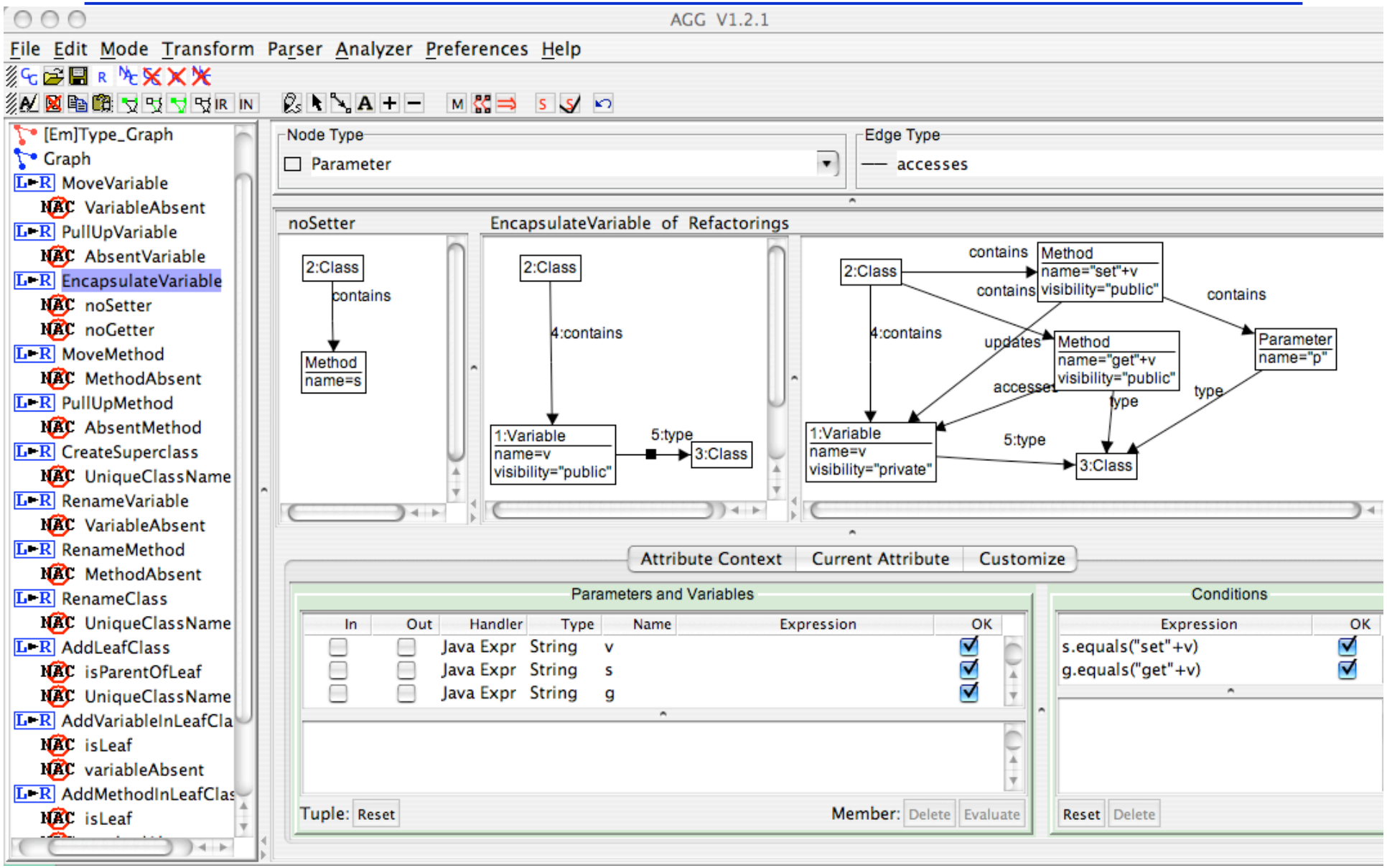
- Experiment: use graph transformation theory to detect and resolve structural conflicts when refactorings are applied in parallel
  - Use AGG tool for experiments
  - in collaboration with Gabi Taentzer, Berlin

Case study: Graph transformation

- Two concrete scenarios

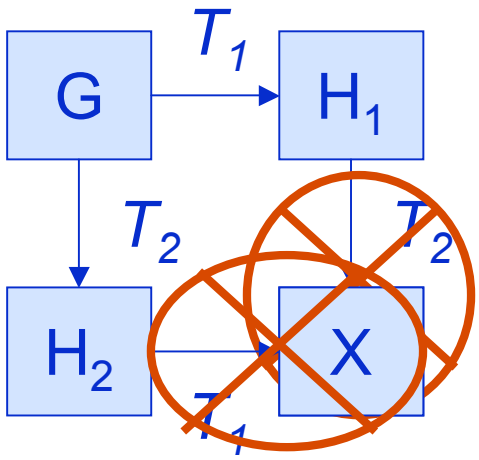


Case study: AGG



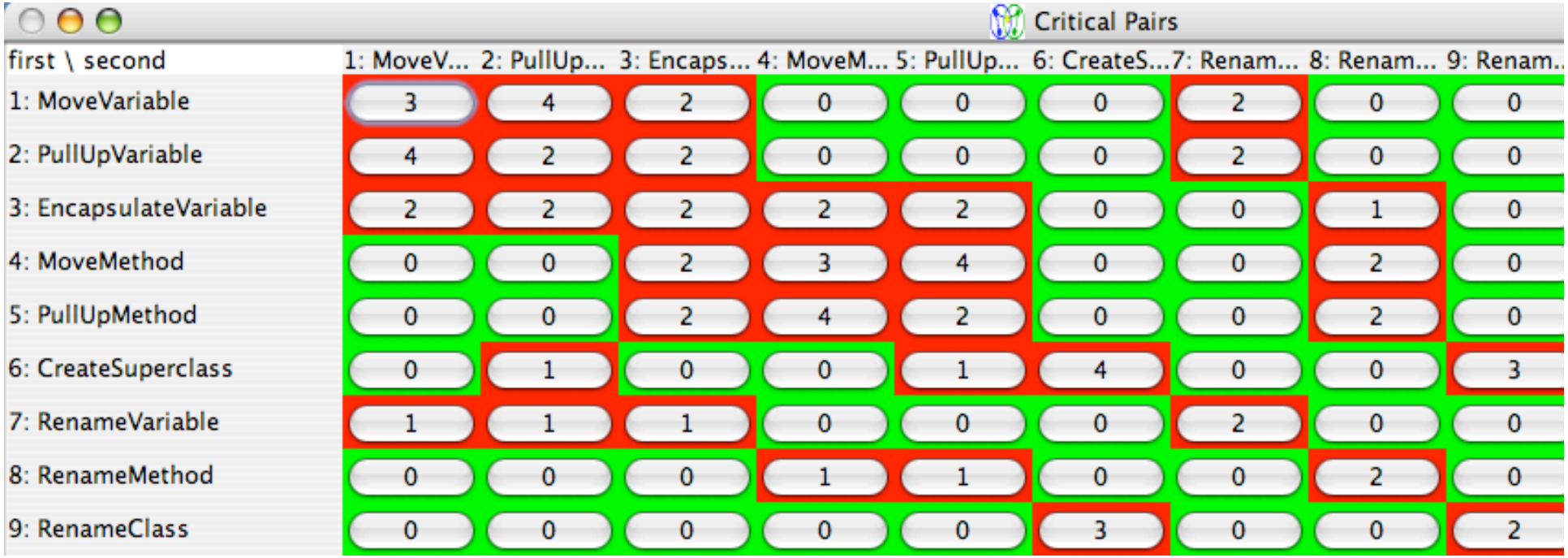
Case study: critical pair analysis

- Use critical pair analysis in *AGG*
  - T_1 and T_2 form a *critical pair* if
    - they can both be applied to the same initial graph *G* but
    - applying T_1 prohibits application of T_2 and/or vice versa



Case study: parallel refactorings

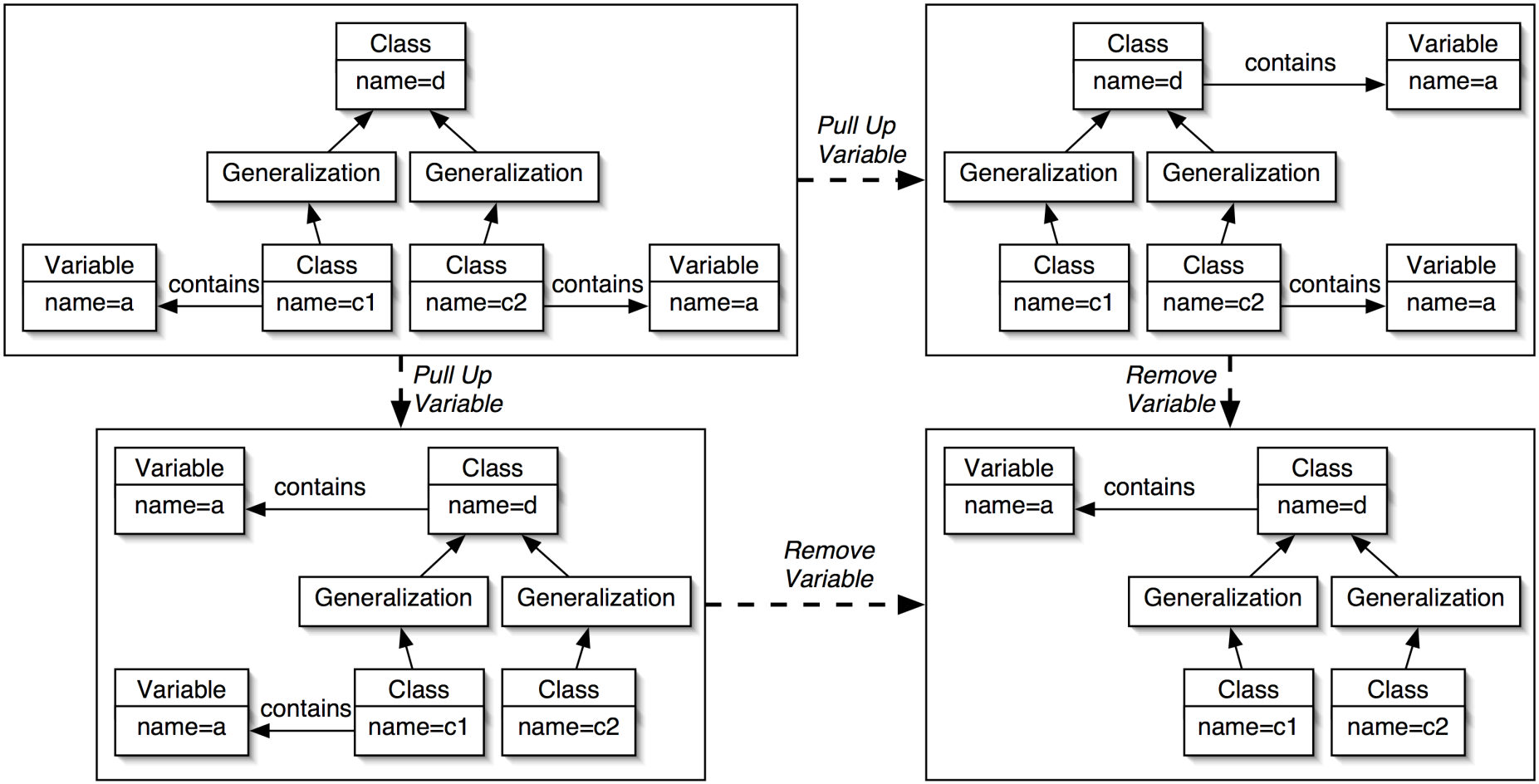
- Compute critical pairs for 9 representative refactorings



first \ second	1: MoveV...	2: PullUp...	3: Encaps...	4: MoveM...	5: PullUp...	6: CreateS...	7: Renam...	8: Renam...	9: Renam..
1: MoveVariable	3	4	2	0	0	0	2	0	0
2: PullUpVariable	4	2	2	0	0	0	2	0	0
3: EncapsulateVariable	2	2	2	2	2	0	0	1	0
4: MoveMethod	0	0	2	3	4	0	0	2	0
5: PullUpMethod	0	0	2	4	2	0	0	2	0
6: CreateSuperclass	0	1	0	0	1	4	0	0	3
7: RenameVariable	1	1	1	0	0	0	2	0	0
8: RenameMethod	0	0	0	1	1	0	0	2	0
9: RenameClass	0	0	0	0	0	3	0	0	2

Case study: parallel refactoings

- Perform confluence analysis to resolve detected conflicts

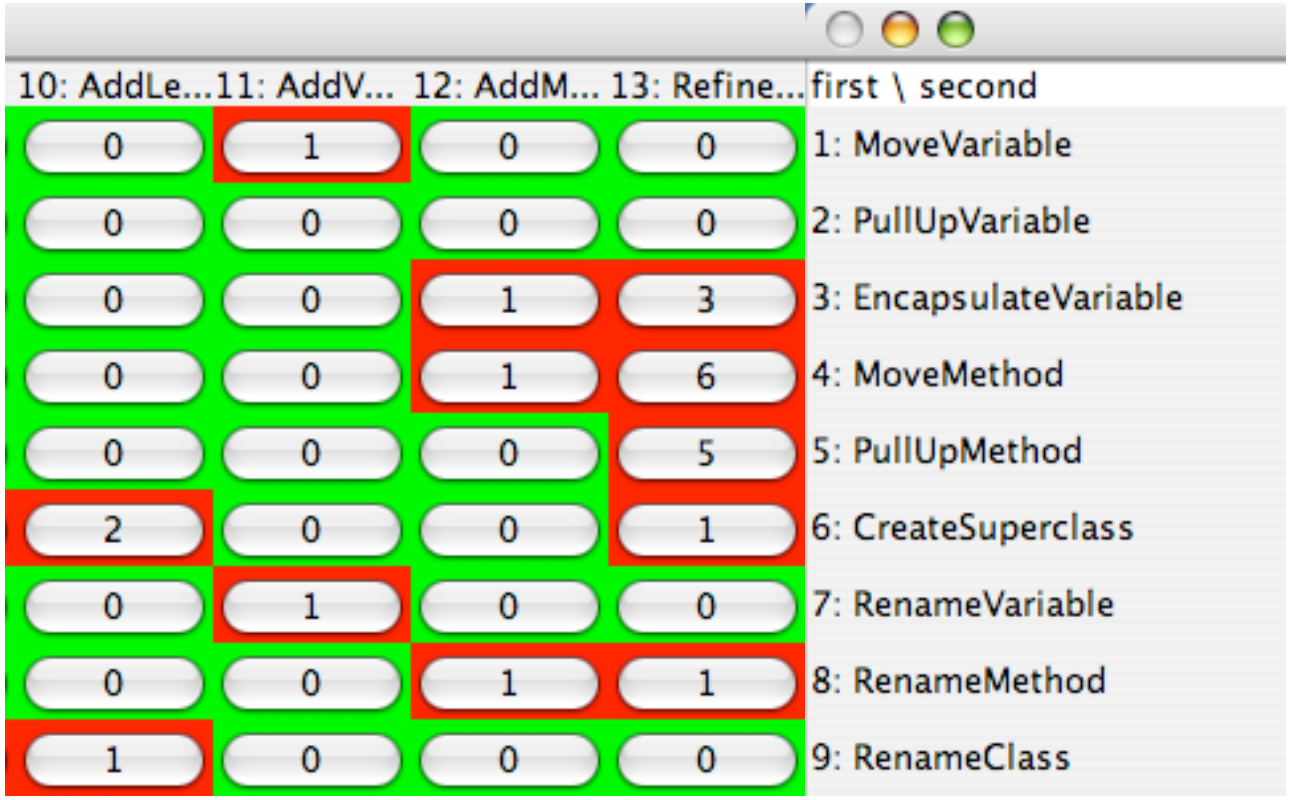


Case study: parallel refactoings

- To do
  - Improve performance of critical pair analysis algorithm
  - Find out to which extent conflict resolution can be automated
  - Reduce set of critical pairs
    - e.g. by taking into account transitive closure of inheritance
  - Investigate distinction between symmetric and asymmetric conflicts

Case study: framework customisation

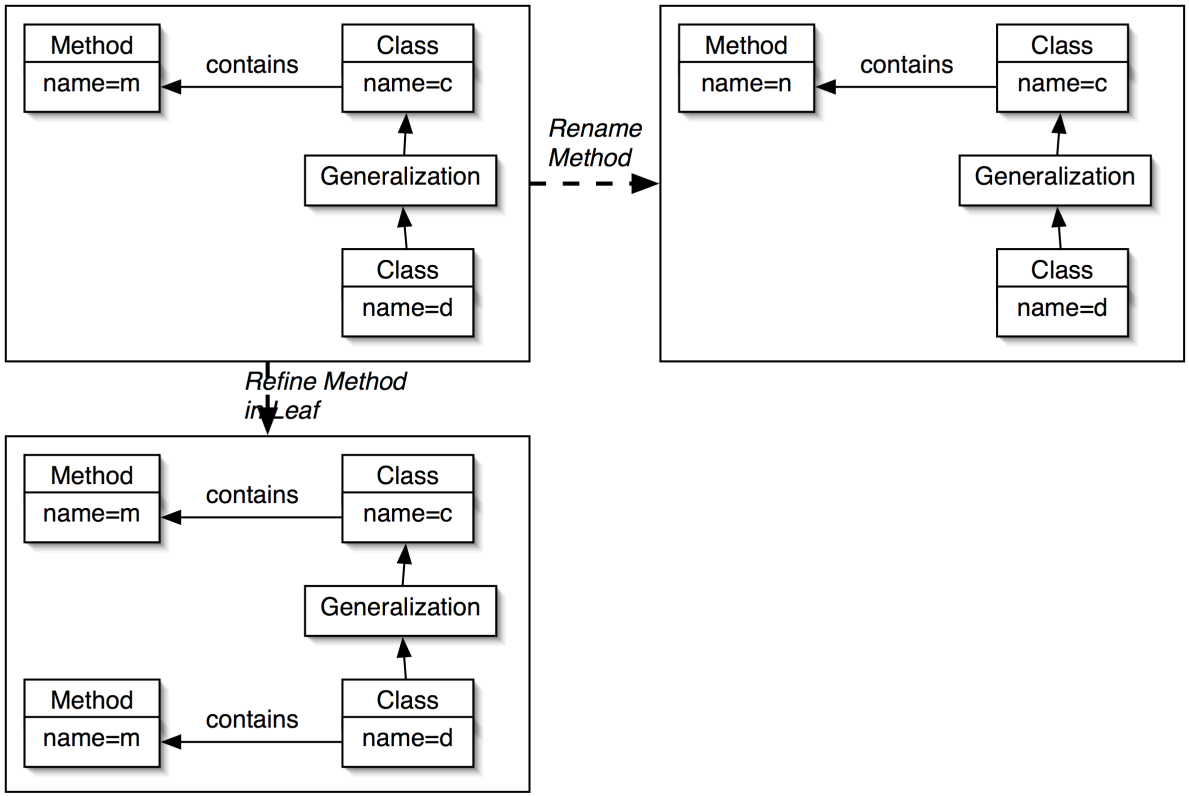
- Customisation conflicts due to framework refactoring



10: AddLe...	11: AddV...	12: AddM...	13: Refine...	first \ second
0	1	0	0	1: MoveVariable
0	0	0	0	2: PullUpVariable
0	0	1	3	3: EncapsulateVariable
0	0	1	6	4: MoveMethod
0	0	0	5	5: PullUpMethod
2	0	0	1	6: CreateSuperclass
0	1	0	0	7: RenameVariable
0	0	1	1	8: RenameMethod
1	0	0	0	9: RenameClass

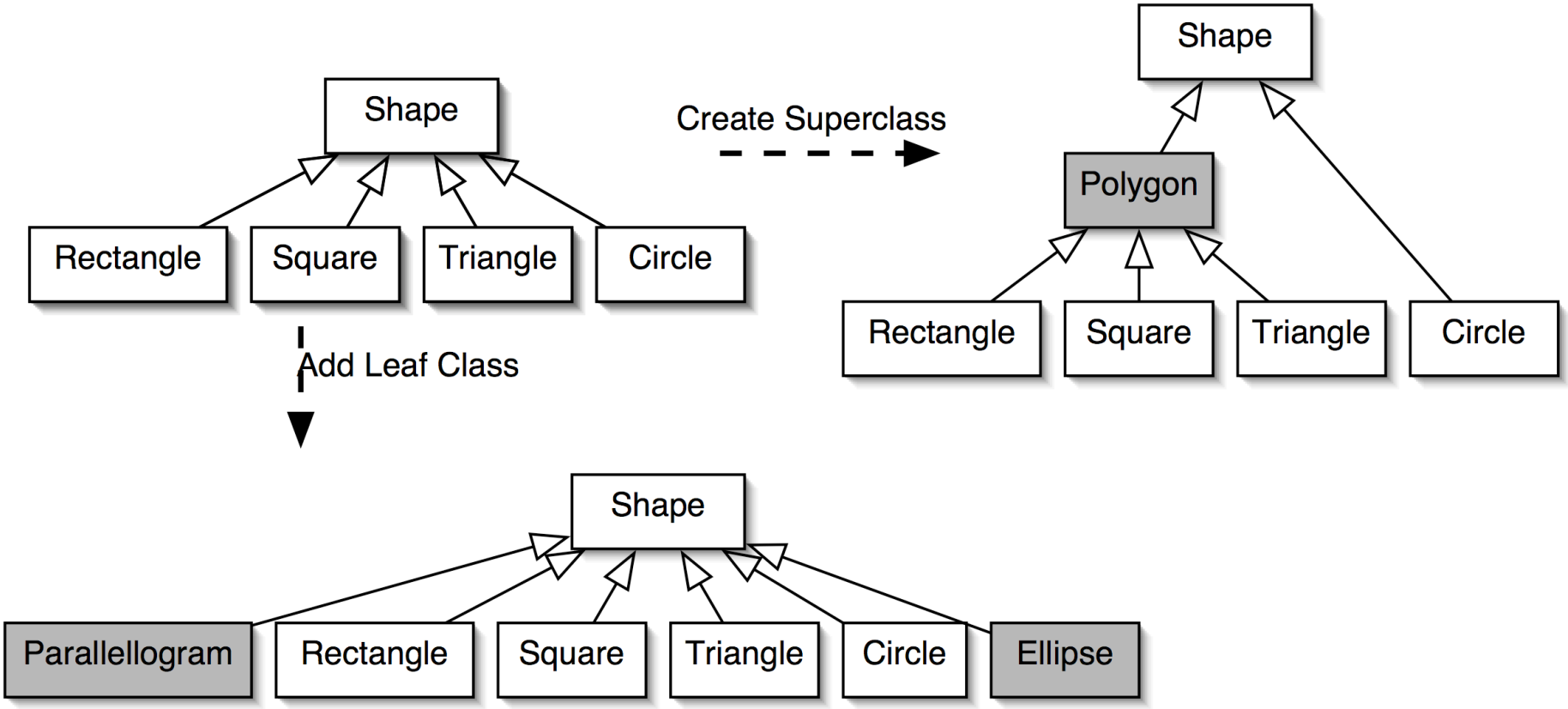
Case study: framework customisation

- Customisation conflicts due to framework refactoring



Case study: Open question

- How to deal with semantic conflicts?



Case study: another potential scenario

- Use some tool to detect "bad smells"
  - opportunities for refactoring
  - can be used to propose a list of possible refactorings that can be applied in the same context
    - cf. Mens&Tourwé, CSMR 2003 and IWPSE 2003
- Critical pair analysis can be used to
  - identify which of the refactorings in this list are in conflict
  - suggest a non-conflicting sequence of refactorings that removes the detected bad smells

Formal foundations for software evolution

Tom Mens

Software Engineering Lab
University of Mons-Hainaut
http://www.umh.ac.be/~genlog

Example: Refactoring formalisms

- **Question**
  - which formalisms can be used to improve tool support for refactoring?
- **Answers**
  - Graph transformation
  - Logic formalisms
    - description logic, fuzzy logic, temporal logic, ...
  - Software metrics
  - Formal concept analysis
  - Program slicing
  - Denotational semantics

Fundamental Research Questions

- possible uses of graph transformation to assist with refactoring ?
  - How to (de)compose refactorings ?
  - How to detect and resolve conflicts due to refactorings ?
    - critical pair analysis
  - How to deal with co-evolution ?
    - triple (quadruple) graph grammars
  - How to guarantee "behaviour preserving" ?
  - How to guarantee "structure improving" ?

Fundamental Research Questions

- other formalisms to assist with refactoring?
  - formal concept analysis
  - program slicing
  - description logics
  - ...
- What is behaviour ? Behaviour preserving ?
  - real-time systems (time); embedded systems (power & memory); safety critical systems (liveness, ...)
  - What are good program invariants ? How to express them ?
- What is structure ? Structure improving ?
  - How to measure impact/effect of refactoring on software quality ?
- Co-evolution
  - How to address consistency maintenance and change propagation ?
    - code ⇔ design ⇔ architecture ⇔ requirements
  - How to refactor at higher abstraction levels ?
    - UML models, design patterns, architectures, components

Practical Questions

- How to measure complexity of refactorings ?
  - Comparing different refactorings in same formalism
  - Comparing same refactoring in different formalisms

  - computational complexity of preconditions
  - computational complexity of applying the refactoring
  - readability/understandability of the refactoring
- How can we determine where and why to refactor ?
  - bad smells
- Where does refactoring fit in the development process ?
- How to combine refactoring with other techniques ?
  - design patterns, application frameworks, aspect-oriented programming, generative programming, …

Opportunities for collaboration

- Applying refactorings to UML models
  - Fits in the MDA model transformation context
  - Addresses theoretical and practical aspects
    - Theoretical
      - deciding on an appropriate formalism ; subset of UML ; definition of behaviour
    - Practical
      - developing tools / plug-ins for model refactoring

- Opportunities
  - Suggest as a topic for ERCIM Strategy 2004
  - Propose a small-scale European project (possible with support from ERCIM)
    - academic partners: UA, UMH, CWI, ... ?
    - industrial partners ?