Analysis Techniques for Graph Transformation Systems

Gabriele Taentzer

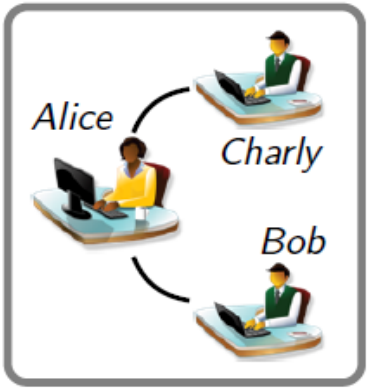
Philipps-Universität Marburg
Germany

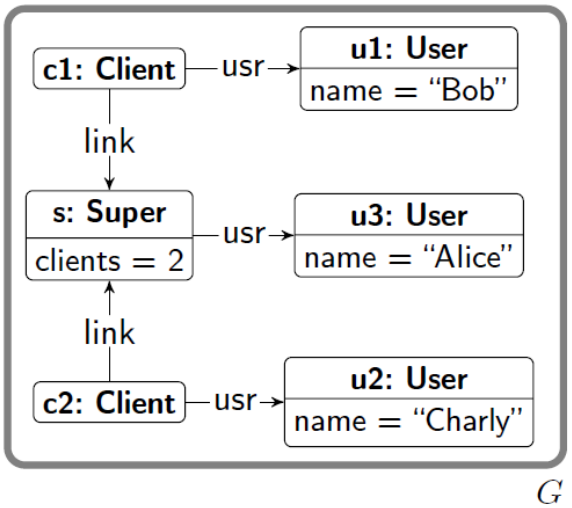
Overview: Main Analysis Techniques

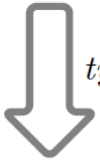
- Type correctness
  - Are all graphs of the transformation system type correct?
- Functional behavior
  - Termination: Does the transformation system terminate?
  - Confluence (Determinism): For each input graph, does the transformation system yield a deterministic result?
- Property checking
  - Invariants: Do all graphs of the transformation system satisfy specified graph-specific properties?
  - Temporal properties: Does the transformation system fulfill specified temporal properties?

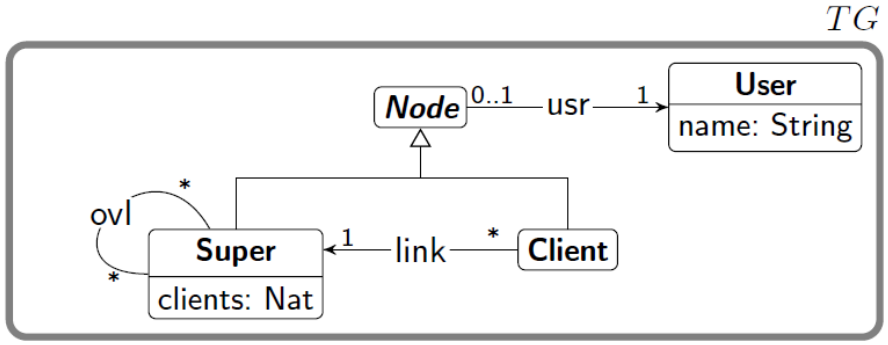
Short introduction to graph transformation

- Running example:
  VoIP networks and their dynamic reconfiguration

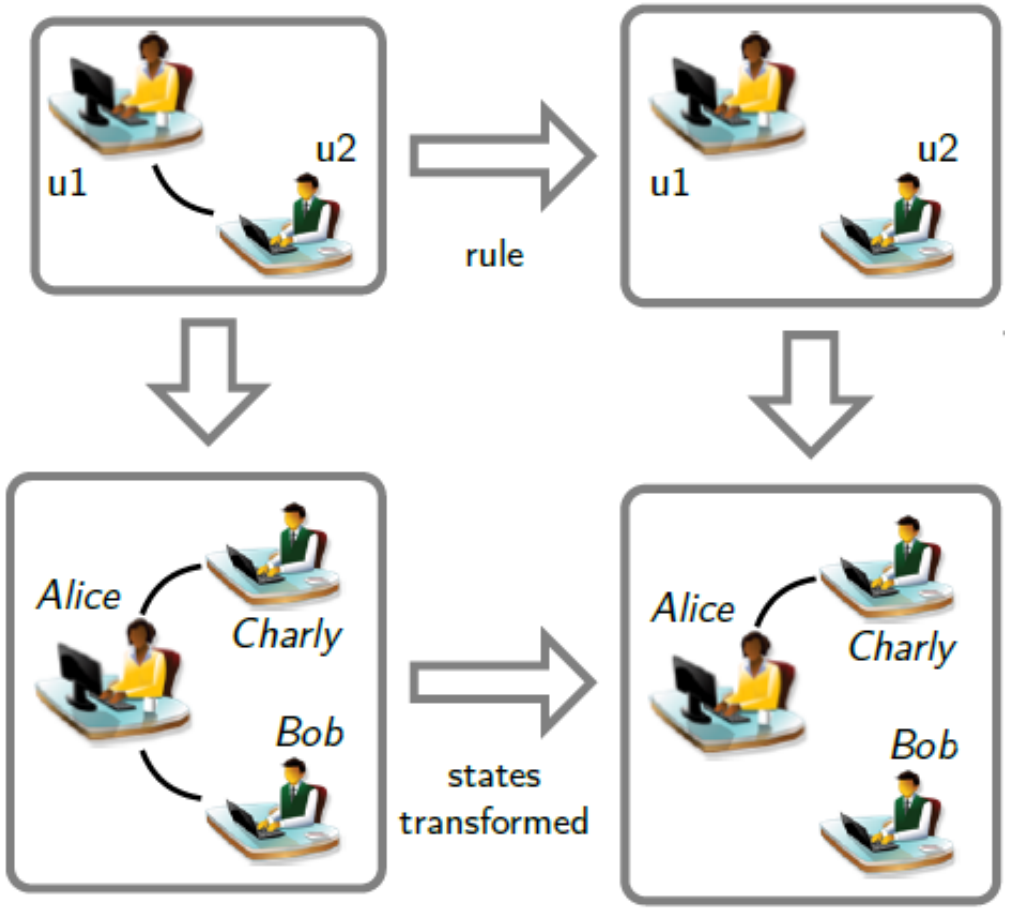




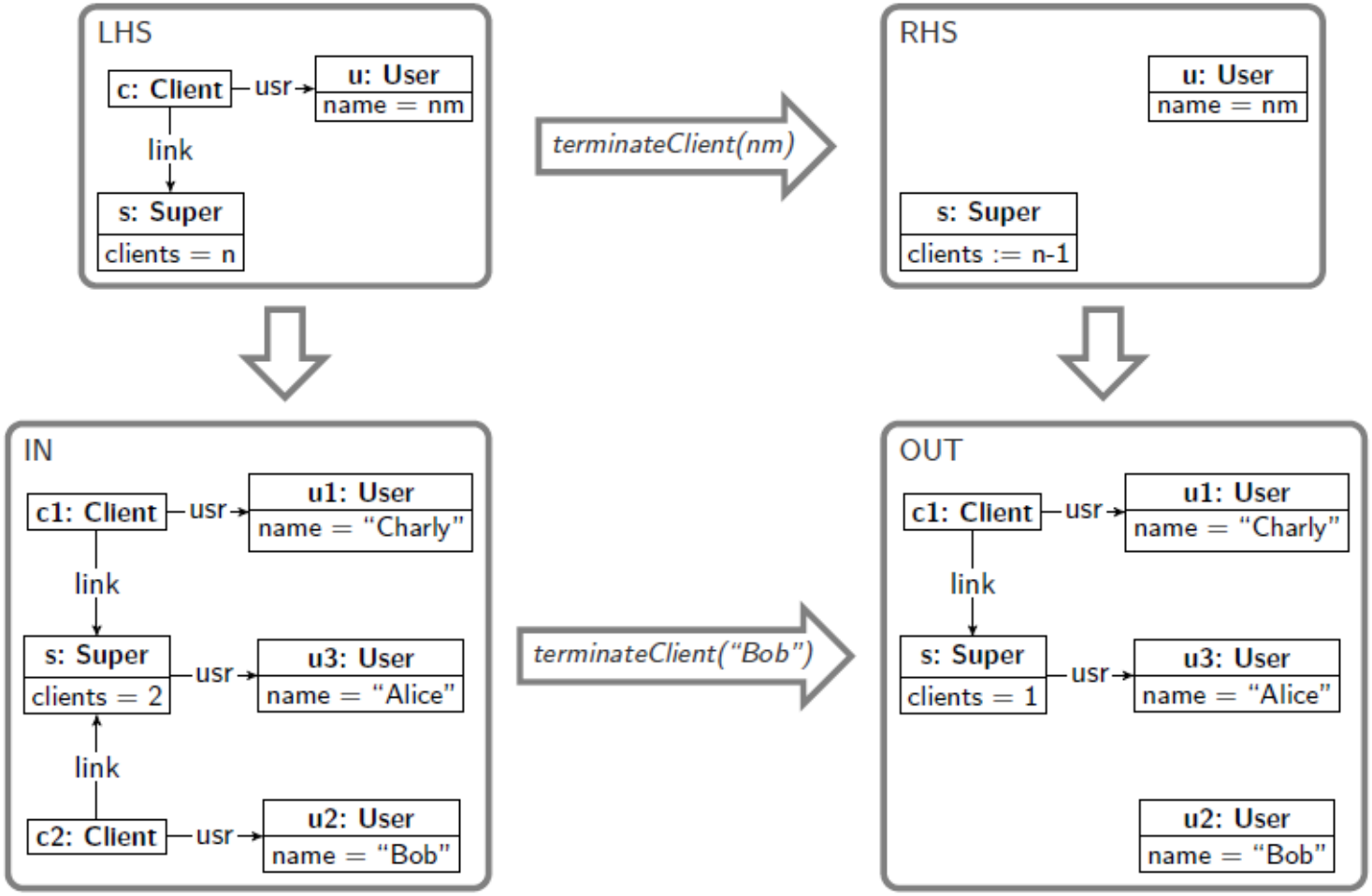




Example: Rule-based Model Change



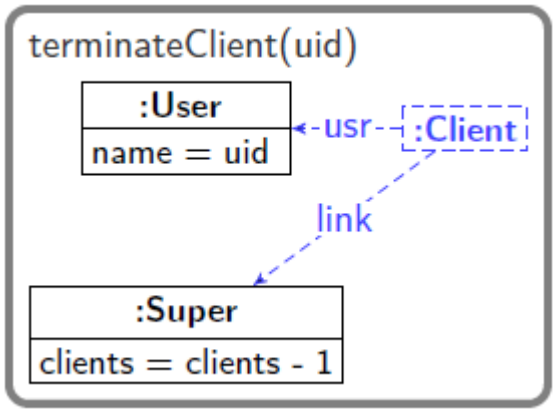
Example: Graph Transformation Step



Each result graph has to conform to the type graph.

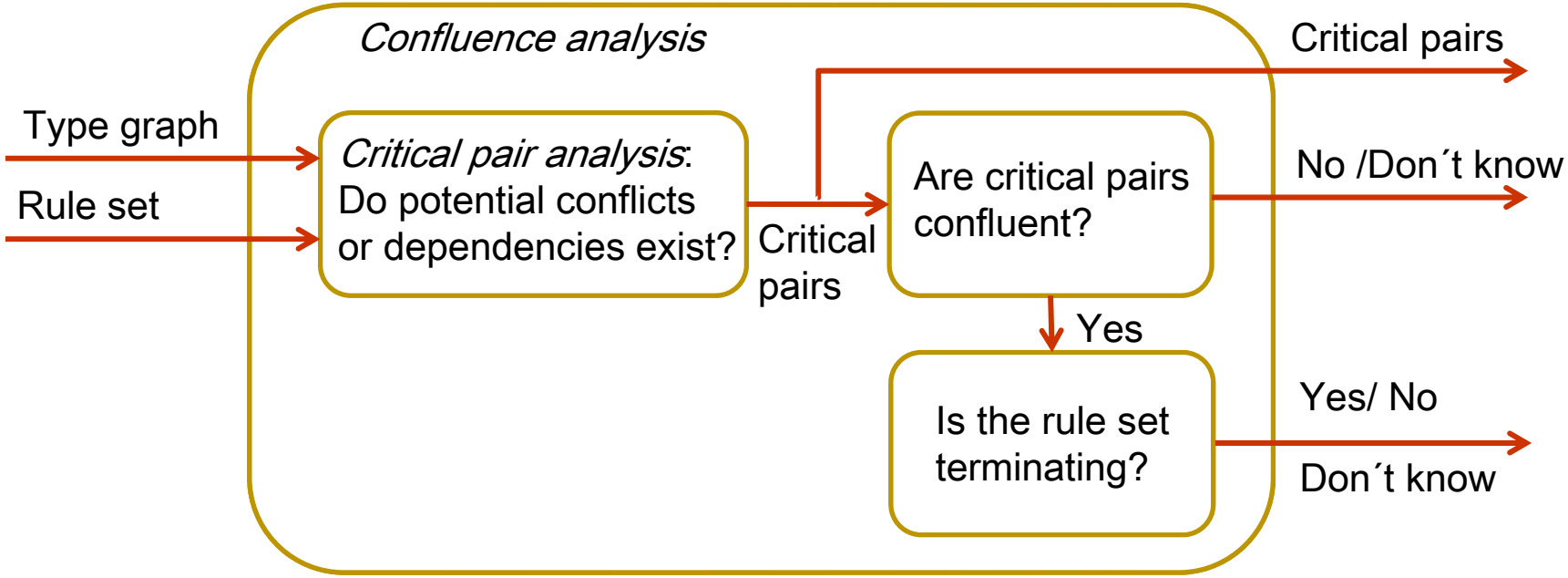
Integrated Rule Representation

Integrated format:



Confluence Analysis

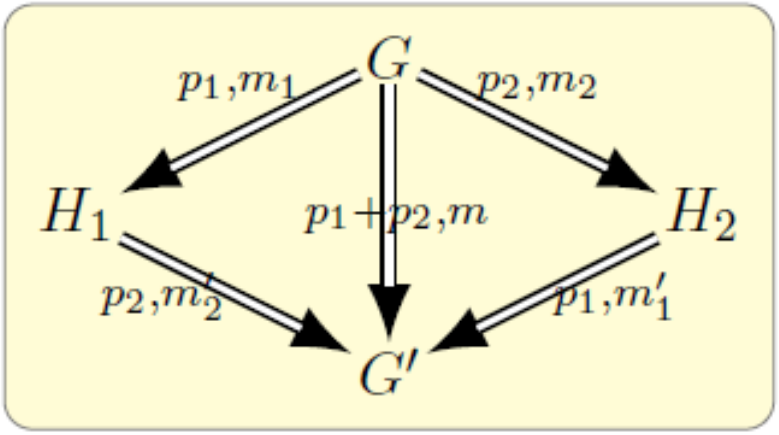
Critical pairs report potential conflicts and dependencies in minimal contexts.



Church-Rosser Property

Church-Rosser Property:

- Two parallel independent graph transformation steps G ➔$_{p1}$ H1 and G ➔$_{p2}$ H2 can be completed by graph transformations H1 ➔$_{p2}$ G‘ and H2 ➔$_{p1}$ G‘.
- Rules p1 and p2 can also be applied in parallel yielding graph transformation G ➔$_{p1+p2}$ G‘.



Independence of Rules

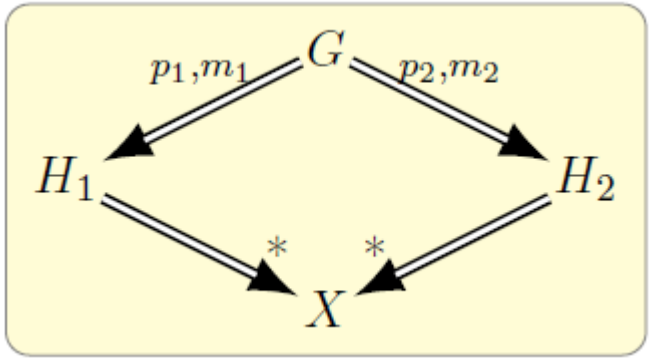
- Two rules are parallel independent if their LHS can overlap in preserved elements only.
- Rule B is sequentially independent of rule A if the LHS of B can overlap in the RHS of A in preserved elements only.

Rules *newClient* and *linkClient* are parallel independent.
They are not sequentially independent.

Conflict Analysis by Critical Pairs

- A critical pair describes a minimal conflicting situation that may occur in transformation system.
- A transformation system is locally confluent if all its critical pairs are strictly confluent.

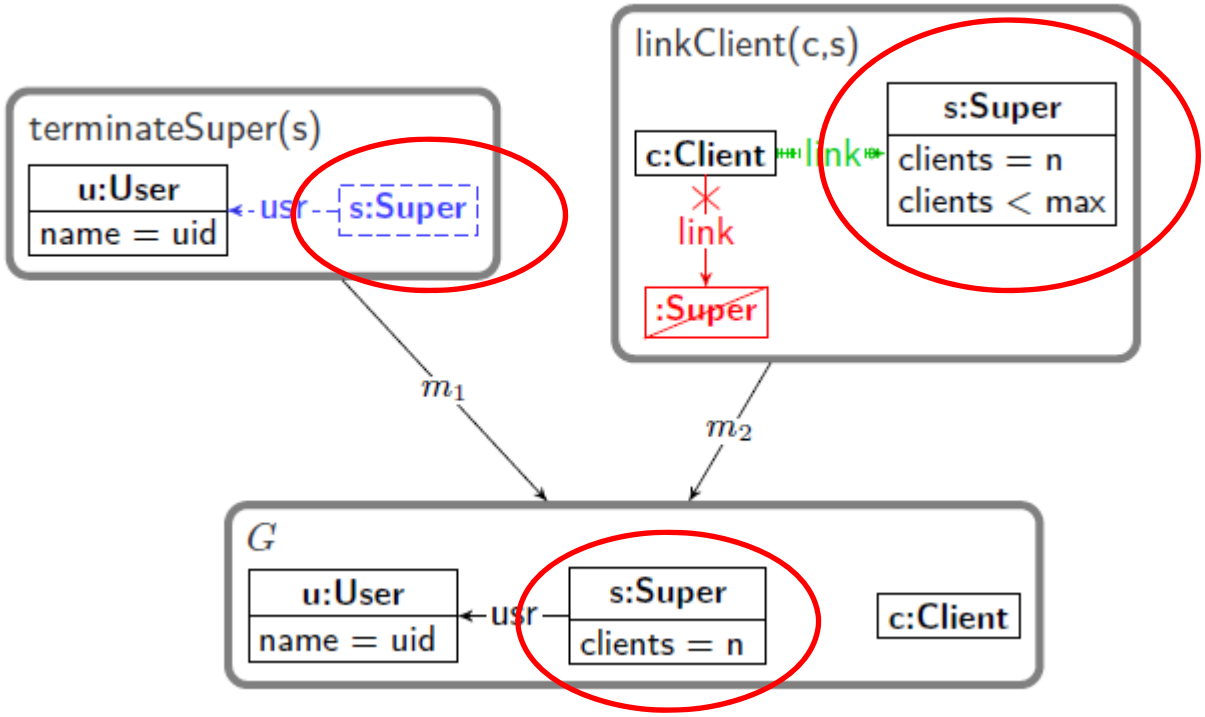
Local confluence:



Strict confluence requires that commonly preserved elements of conflicting transformations are not deleted by completing ones.

Kinds of conflicts

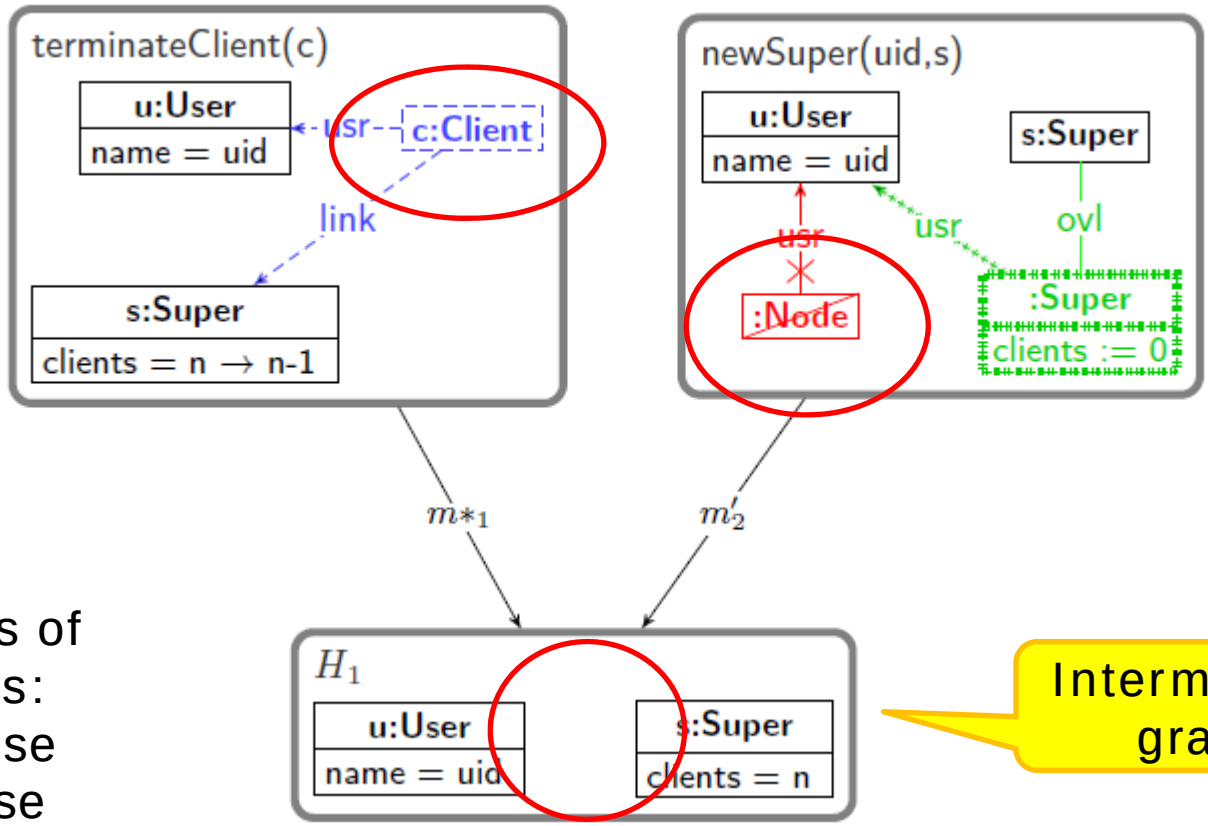
Delete/use conflict: Rule A deletes an element used by rule B.



Further kinds of conflicts:
Produce/forbid: Rule A produces an element that is forbidden by rule B.
Change/use: Rule A changes an attribute value used by rule B.

Kinds of dependencies

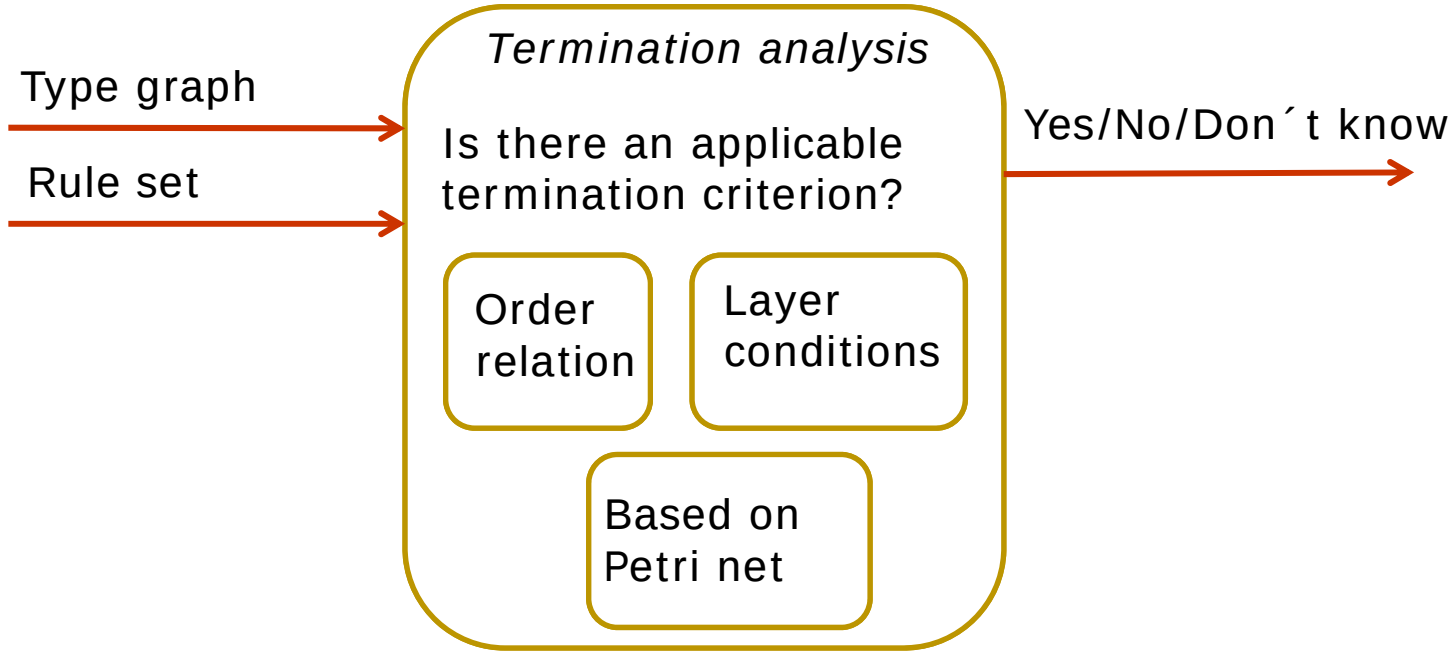
Delete/forbid dependency:
Rule A deletes an element being forbidden by rule B.



Further kinds of dependencies:
- Produce/use
- Change/use

Termination analysis

Does the transformation system terminate?

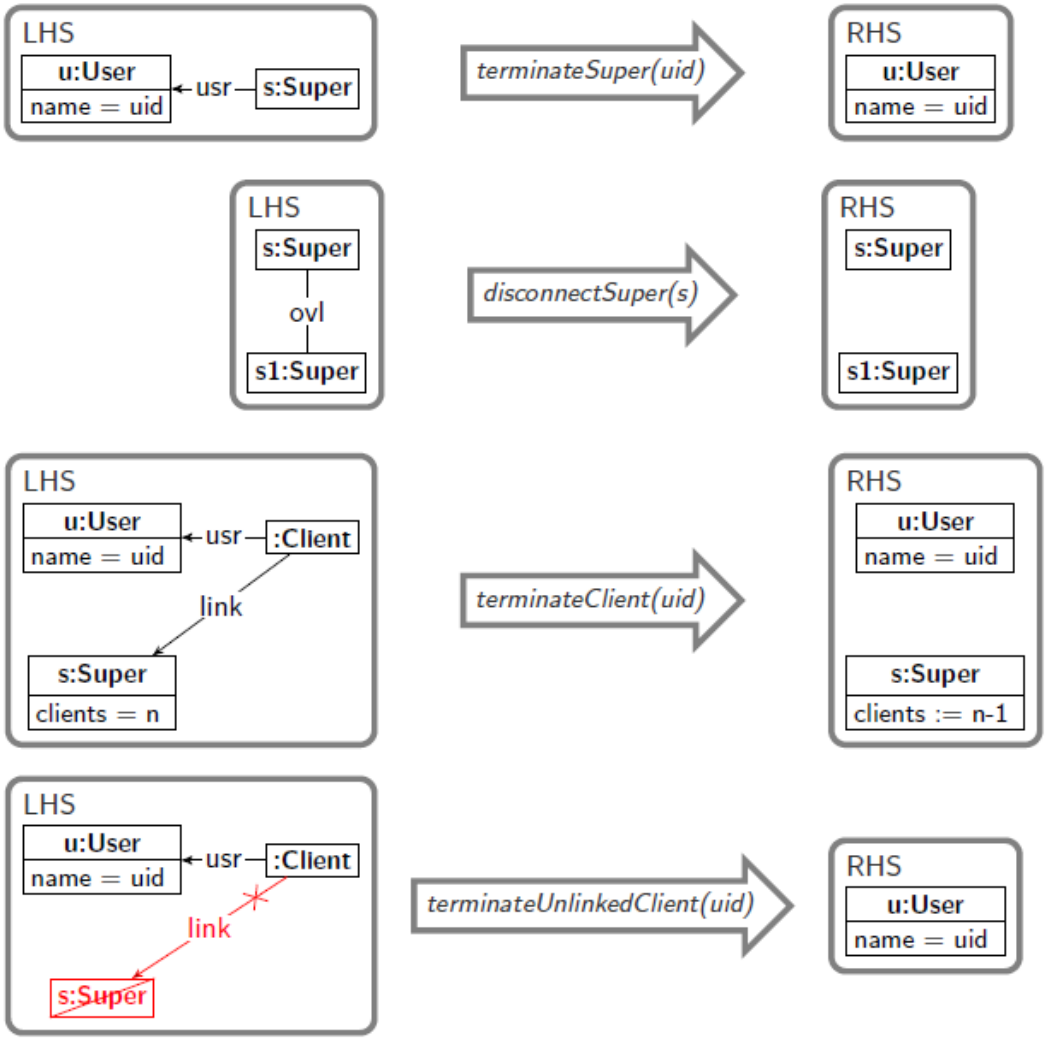


Order relation

- Define a relation > where L > R is true for all rules.
- Example:

  Step-by-step deletion of networks
  - s: num. of Super nodes
  - c: num. of Client nodes
  - o: num. of ovl edges
  - t = s + c + o
  - L > R wrt. t

  Rule application terminates.



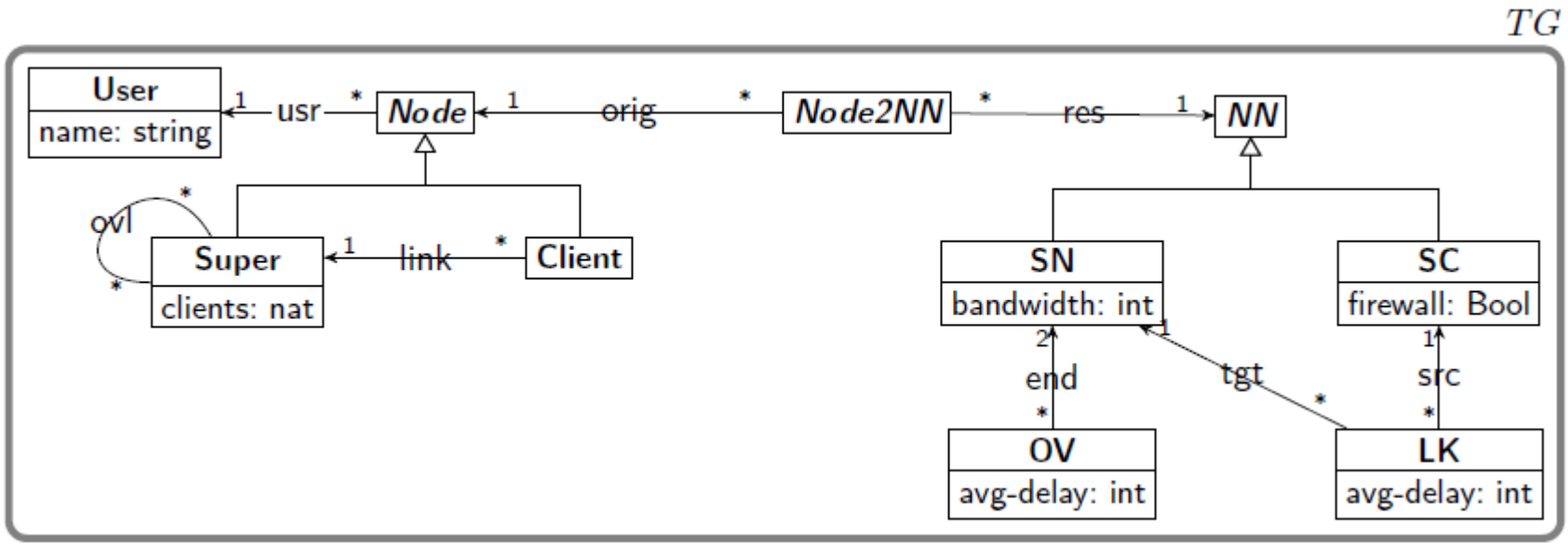
Layer conditions

- What if some rules are non-deleting?

  Try to split rule set into layers such that layer for layer is applied one after the other.

Deletion Layer Conditions	Non-deletion Layer Conditions
If k is a deletion layer,	If k is a non-deletion layer,
1. r with $rl(r) = k$ is deleting at least one element, 2. $0 \leq cl(t) \leq dl(t) \leq k_0$ for all $t \in TG$, 3. r deletes an element of type $t \Rightarrow dl(t) \leq rl(r)$ and 4. r creates an element of type $t \Rightarrow cl(t) > rl(r)$.	1. r with $rl(r) = k$ is non-deleting, 2. all NACs of r can be consistently embedded into its RHS. 3. $x \in L \Rightarrow cl(type(x)) \leq rl(r)$ and 4. r creates an element of type $t \Rightarrow cl(t) > rl(r)$.

Example: Model translation

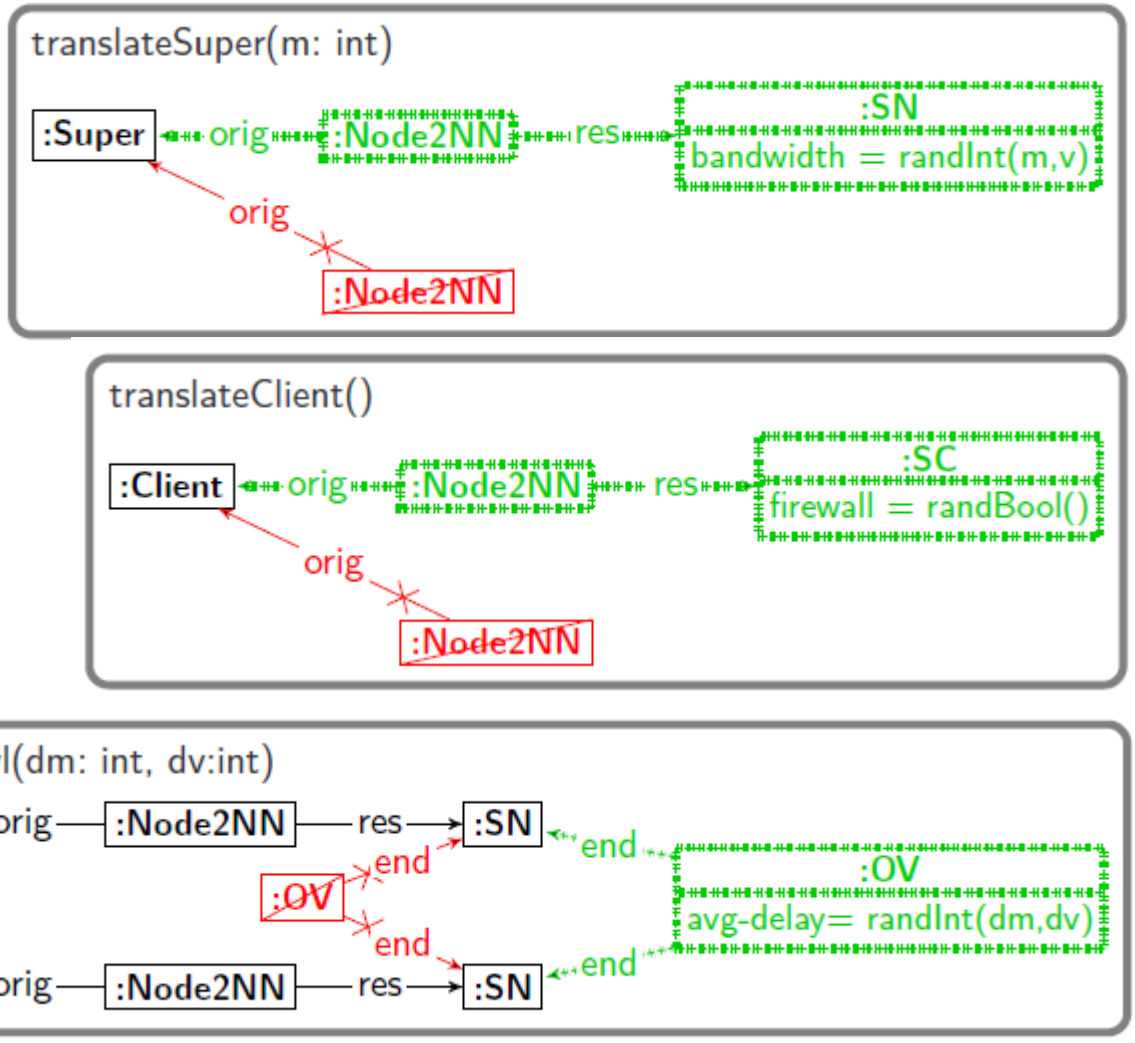


Creation and deletion layers for types

Orig type	$cl(t)$	$dl(t)$	Trace type	$cl(t)$	$dl(t)$	Result type	$cl(t)$	$dl(t)$
Super	0	3	Node2NN	1	3	SN	1	4
Client	0	3				SC	1	4
User	0	3				LK	2	4
						OV	2	4

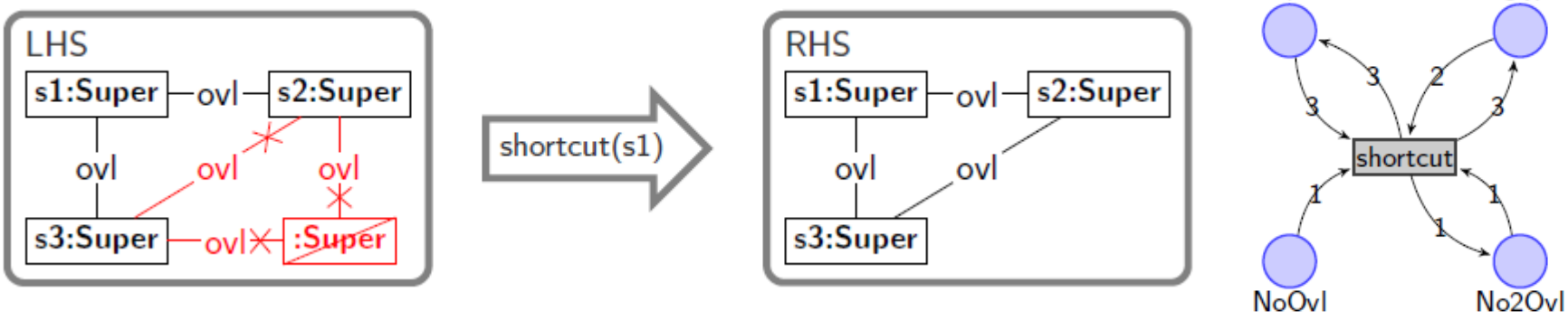
Example: Model Translation

Rule	Layer
translateSuper	0
translateClient	0
translateLink	1
translateOvl	1



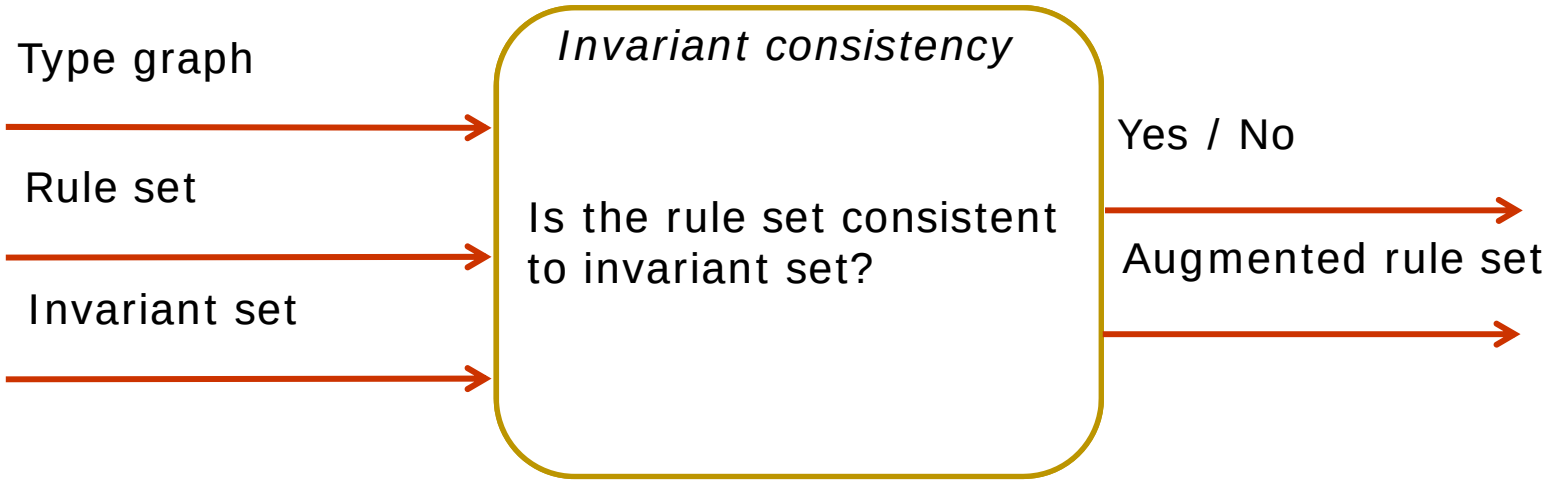
Termination based on Petri nets

- Graph transformation system → Petri net
  - Rule → transition
  - Element type → place
  - Special places for neg. appl. conditions
- If Petri net runs out of tokens after finitely many steps, the transformation system terminates.



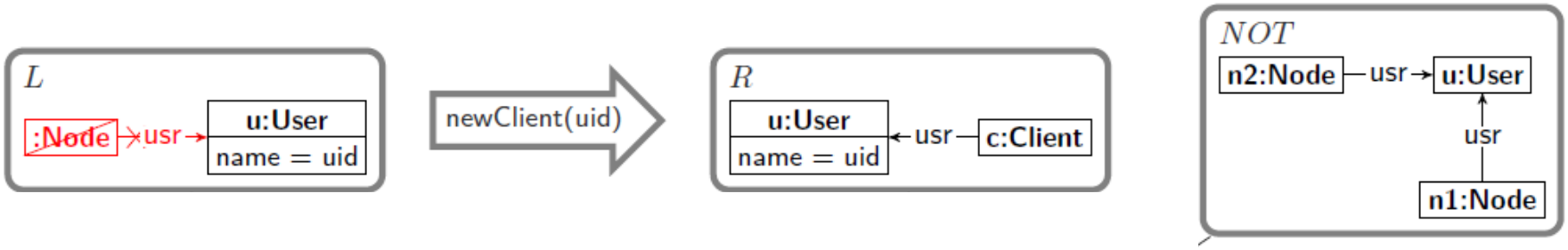
Invariant checking

Do all graphs of the transformation system satisfy specified graph-specific properties?



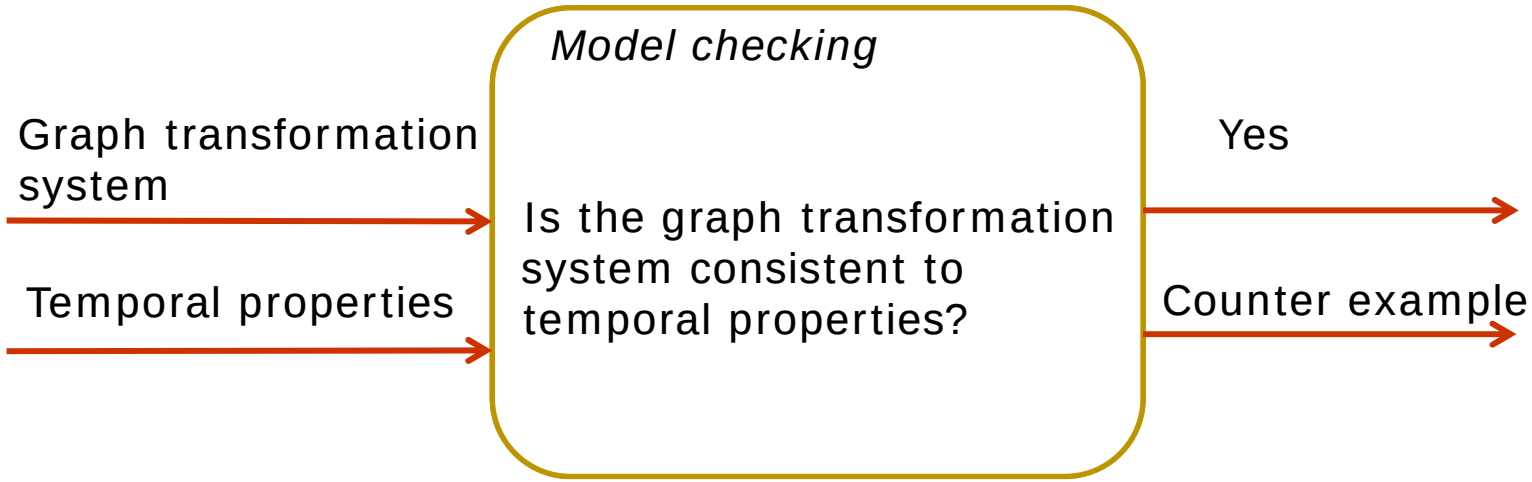
Example: Invariant checking

- Given: Rule, invariant
- Construct post-conditions by overlappings with RHS
- Construct pre-conditions by inverse rule application
- If no new pre-conditions are constructed, the rule is consistent.



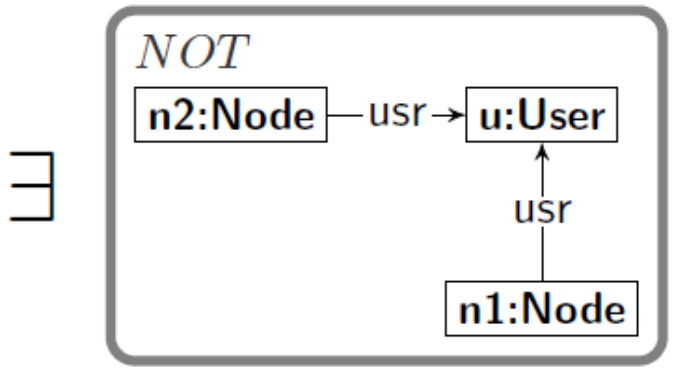
Checking of Temporal Properties

Does the graph transformation system satisfy specified temporal properties?

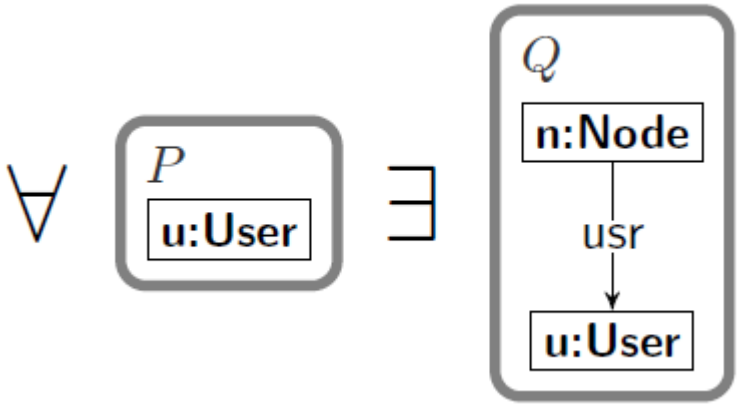


Temporal properties

- *Safety property:* A property that should always hold on every execution path.

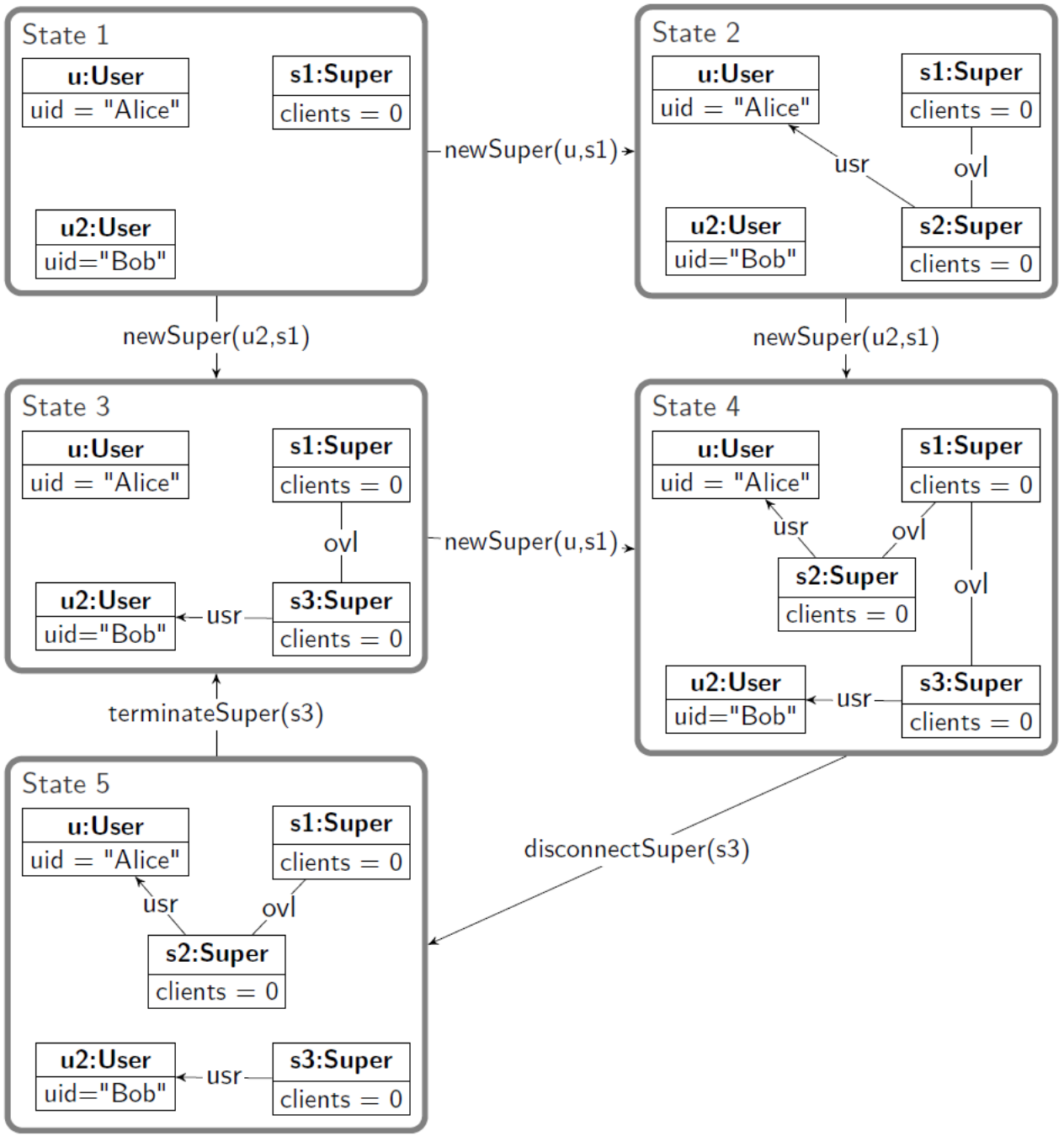


- *Reachability property:* A property which be reachable on at least one path.



Transition System

Graph transformation system as transition system:



Model checking of graph transformation systems

- Graphs: Labeled / typed and attributed
- Rules: limited node creation / unlimited
- Matching: injective / non-injective
- Verification:
  - Groove: extensible library, symmetry recognition by graph isomorphisms, no preprocessing
  - ViaTra: translation to SPIN, symmetry recognition by reusable object ids
  - Henshin: translation to mCRL2, symmetry recognition by graph isomorphisms or object ids
- Experiments: at dining philosophers example and others
  - Groove: stores just difference, can handle larger problems, not so fast
  - ViaTra: stores entire graphs, ca. 10 times faster

Tool support

- Termination

- Confluence

- Invariant checking

PROCON

- Temporal property checking

Conclusion

- Analysis techniques for algebraic graph transformation systems:
  - Type correctness
  - Functional behavior: Termination and confluence
  - Property checking: Invariants and temporal properties
- Different kinds of tool support

- Future work:
  - Extension to advanced transformation concepts
  - Further analysis techiques such as slicing
  - Comparison with analysis techniques of other model transformation approaches