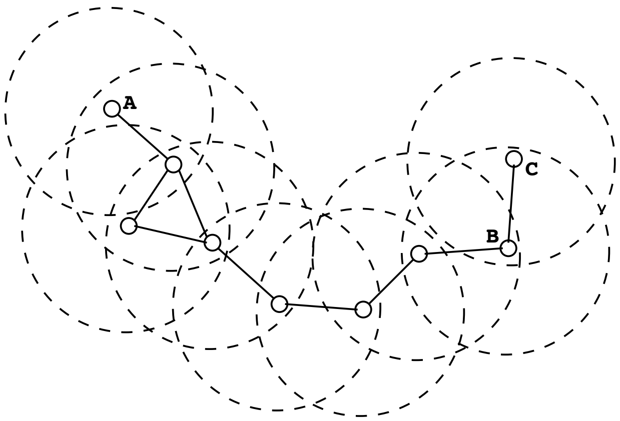


Graph Grammar Modeling and Verification of Ad Hoc Routing Protocols

Mayank Saksena, **Oskar Wibling**, and Bengt Jonsson

Summary

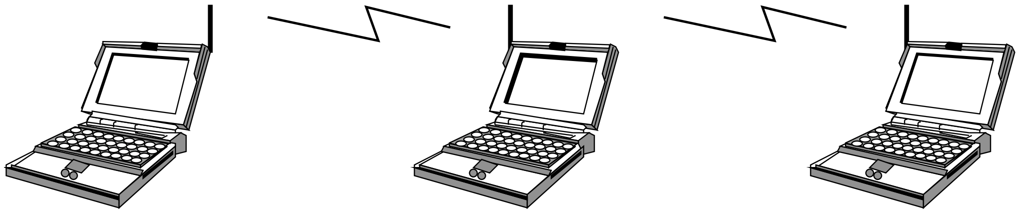
- Developed tool for verifying safety properties
  - Symbolic backward reachability analysis of graph grammars
    - `http://www.it.uu.se/research/group/mobility/adhoc/gbt` (tool + example models)
- Verified loop freedom of DYMO
  - Ad hoc routing protocol

Outline

- Introduction
  - Ad hoc networking, properties
- The DYMO protocol
- Graph grammars
- Modeling
- Verification
  - GBT tool
- Conclusions, future work

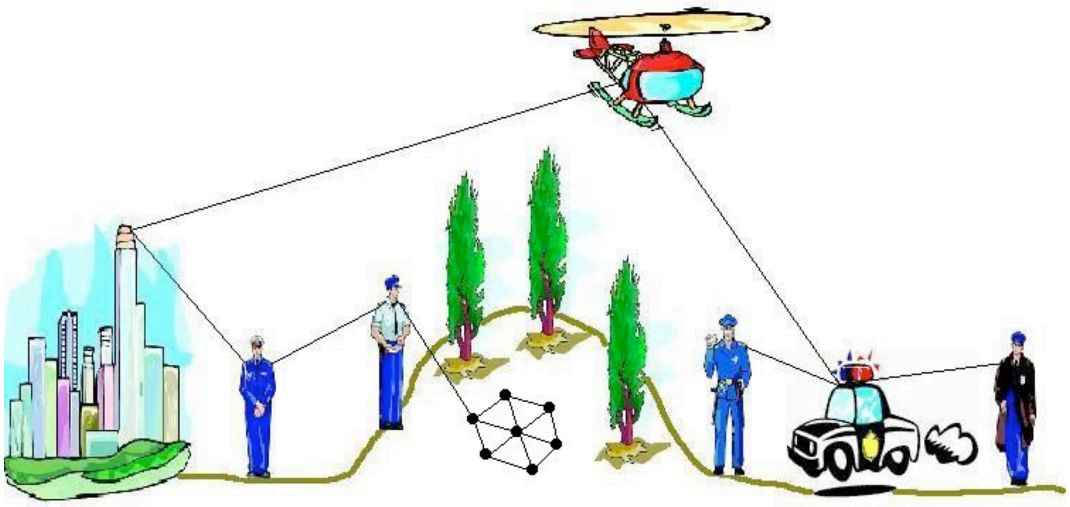
Mobile ad hoc networks

- Mobile ad hoc network (MANET)
  - No wireless infrastructure
    - Build own structure dynamically, multi-hop
  - Behavior
    - Nodes change location
    - Connectivity fluctuates
    - Power and bandwidth limited



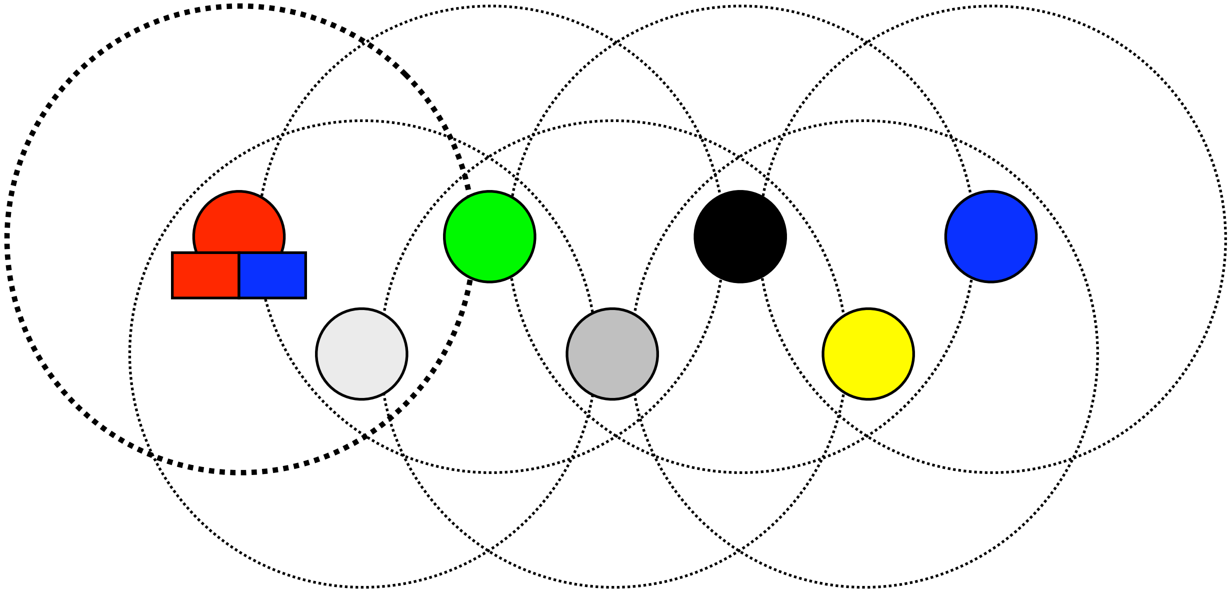
Mobile ad hoc networks

- Usage areas
  - Lack of infrastructure
  - Search and rescue teams, military operations
  - Spontaneous networking in mobile communities
  - Sensor networks



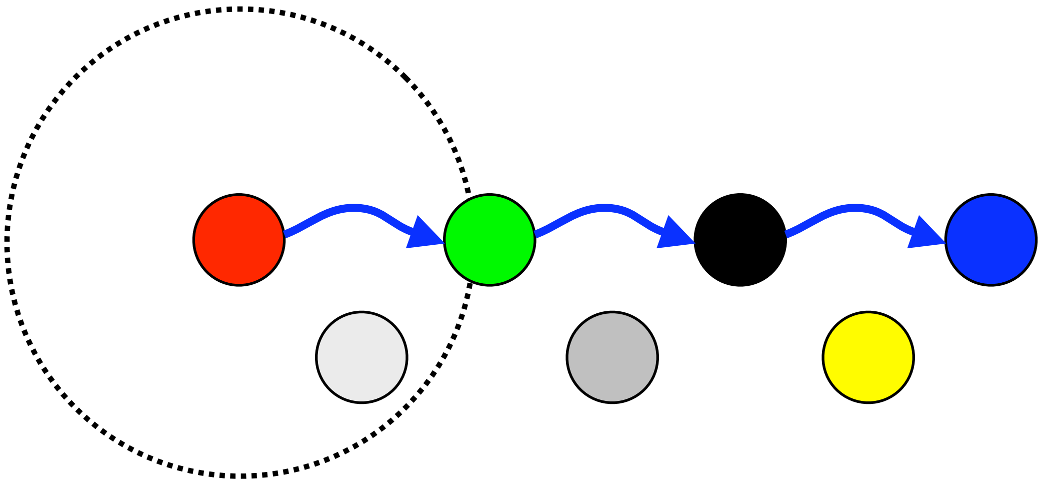
Ad hoc routing protocols

- Set up forwarding paths
- Extend communication range
  - Save power, overcome obstacles, avoid interference



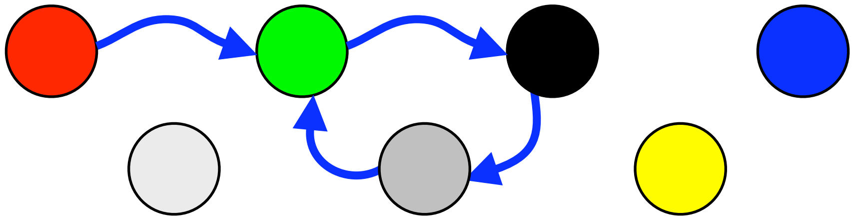
Ad hoc routing protocols

- Set up forwarding paths
- Extend communication range
  - Save power, overcome obstacles, avoid interference



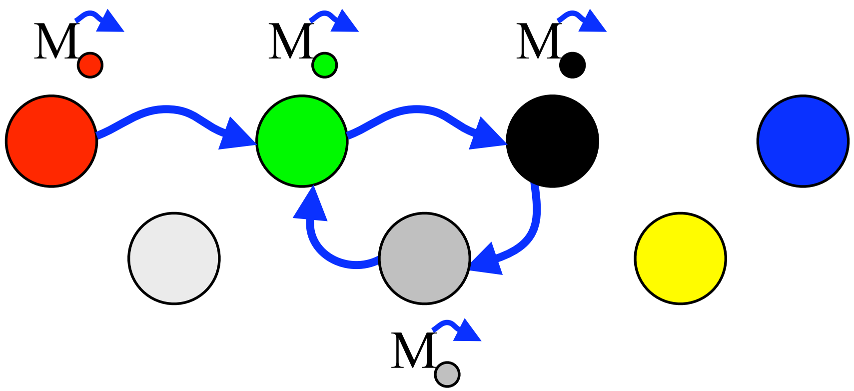
Properties of ad hoc protocols

- Correct routing
  - The protocol sets up "usable" routes
  - Loop freedom
    - Forwarded packets do not enter a loop



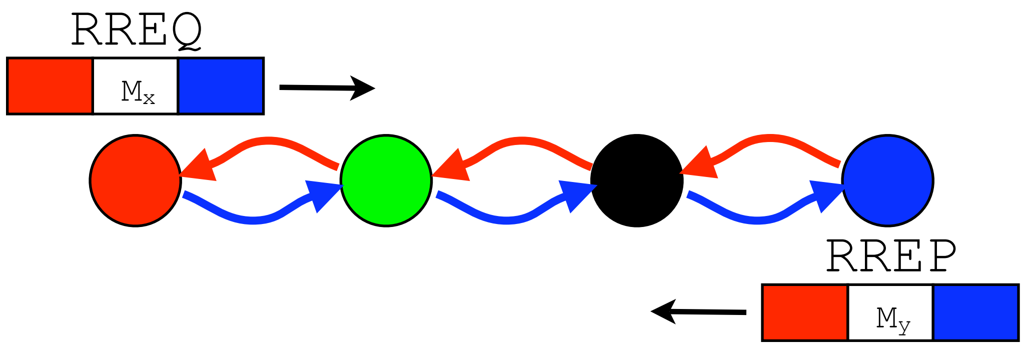
Properties of ad hoc protocols

- Correct routing
  - The protocol sets up “usable” routes
  - Loop freedom
    - Forwarded packets do not enter a loop
    - Can be checked by condition on route metric



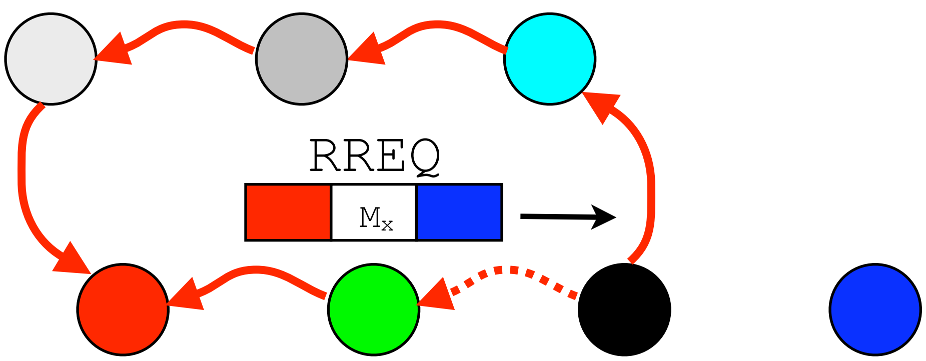
DYMO

- Dynamic MANET On-demand (DYMO) routing protocol
  - Nodes have sequence number and route table
  - Route table entry: $<$ `target, next, metric` $>$
  - Metric: $<$ `-sequence number, hop count` $>$
- Routing messages used for updates



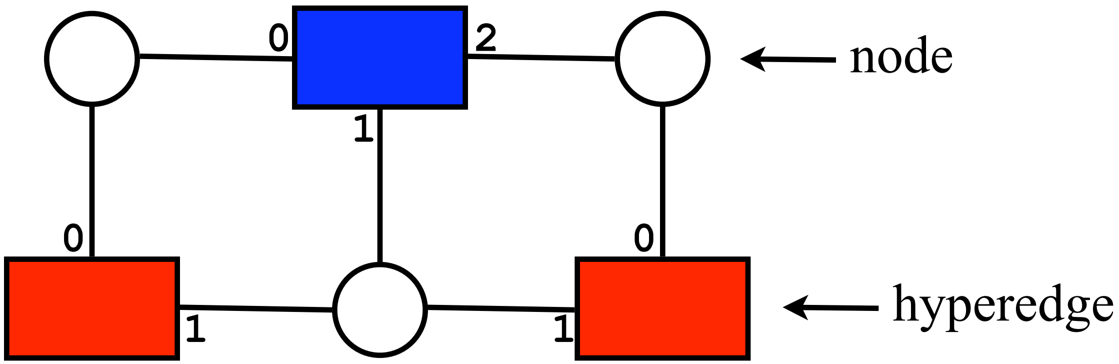
DYMO

- If there is previous entry
  - Compare and replace if “better”
    - Fresher data, shorter path
  - Specification contains update rules



Graph grammars

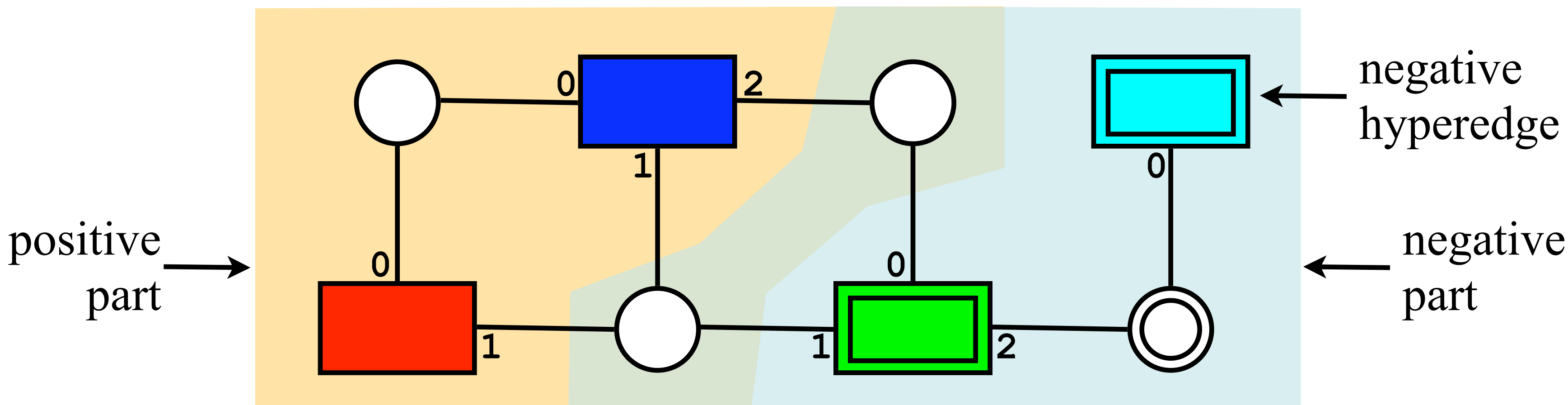
- We model a system using graph grammars
  - Initial hypergraph + hypergraph rewriting rules
  - Example hypergraph:



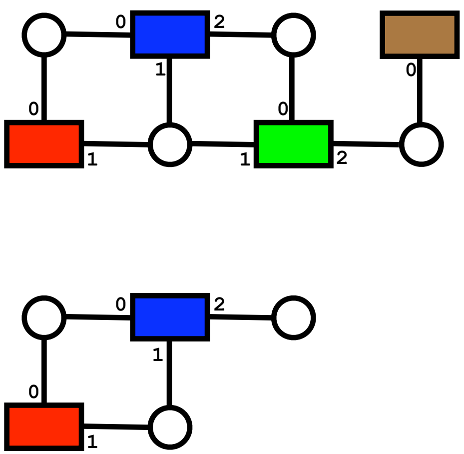
Hypergraphs vs patterns

- Configuration represented by hypergraph
- Patterns - constraints on configurations
  - Represent sets of hypergraphs
  - Positive part
    - Must be subgraph of included hypergraphs
  - Negative parts
    - Must **not** match included hypergraphs

Hypergraphs vs patterns

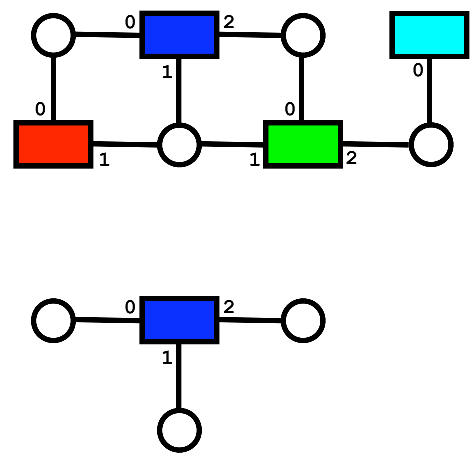


Included



...

Not included



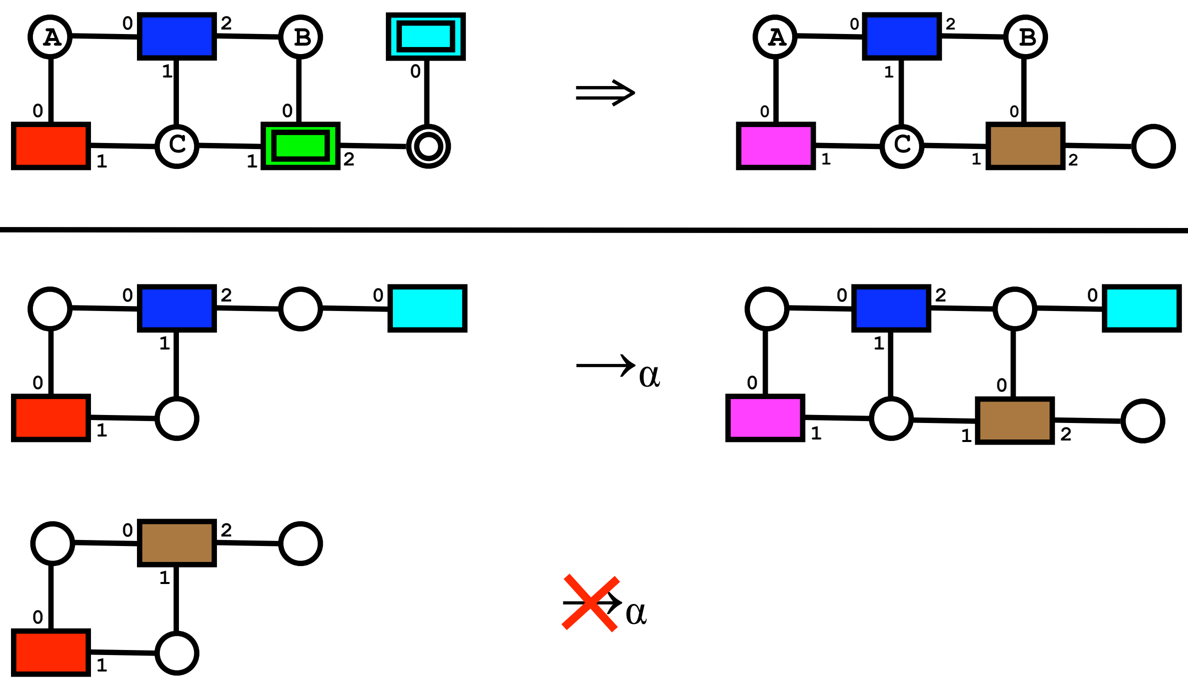
...

Hypergraph rewriting rules

- Left hand side: pattern
- Right hand side: hypergraph
- Semantics - replacing positive part by RHS

Rule α

Example rewritings

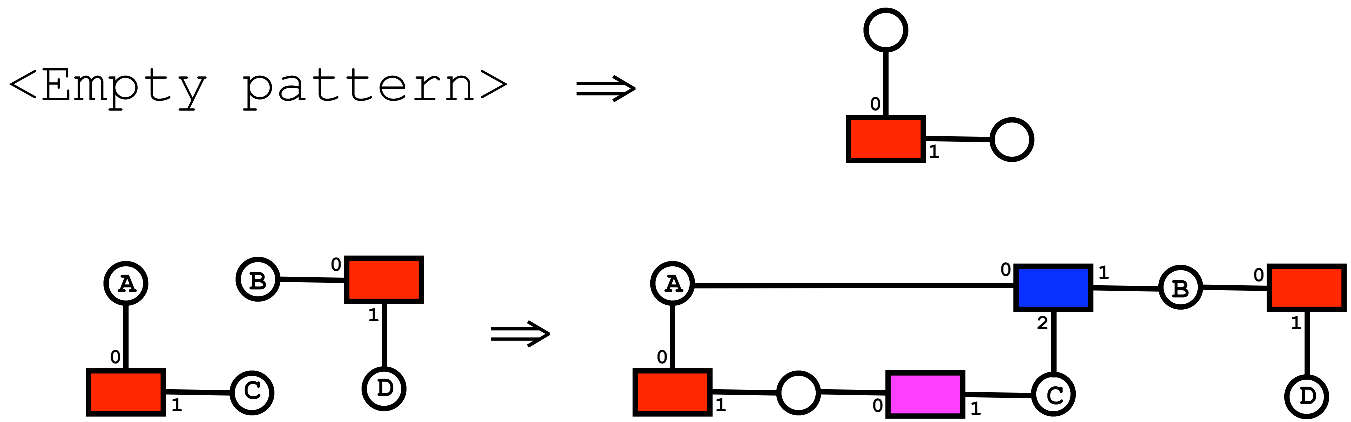


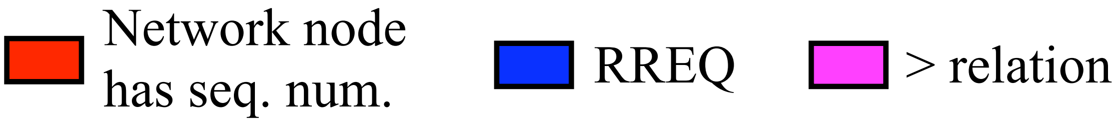
Modeling DYMO

- Initial graph: Empty
- Example rules: New network node and RREQ generation (simplified)

Rule NewNode

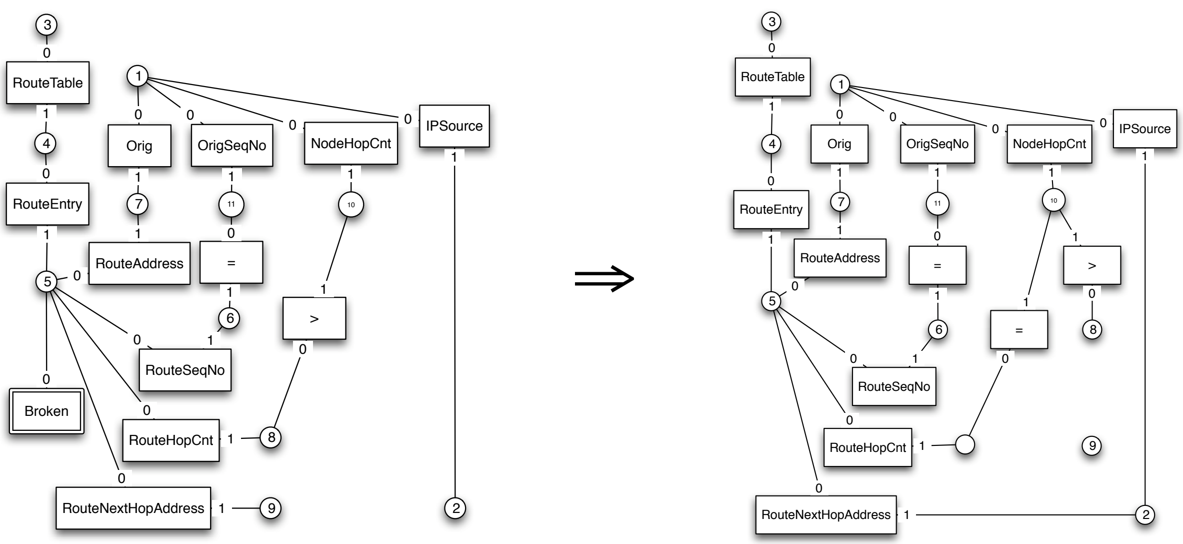
Rule GenRREQ





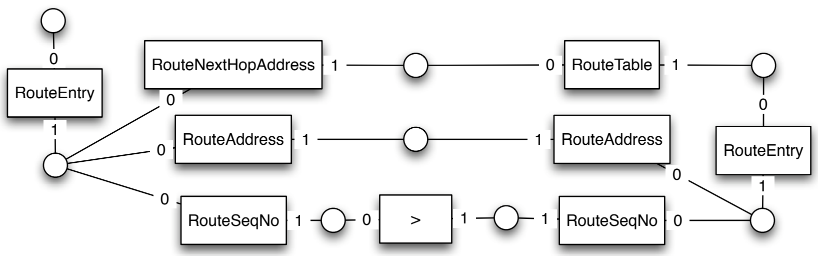
Modeling DYMO

- Followed latest version
  - DYMO Internet Draft v10
  - 77 rewriting rules
    - 38 of these are update rules



Verifying DYMO

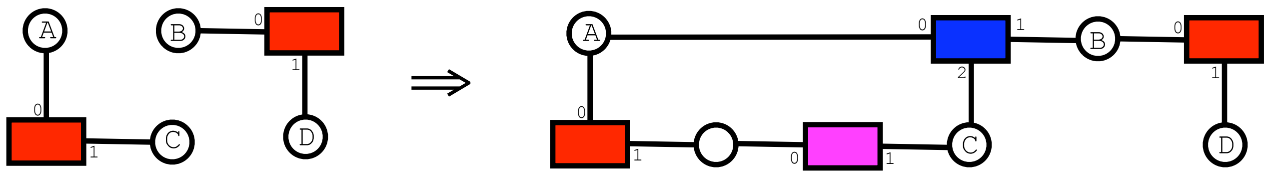
- Starting from undesirable graph patterns
  - Representing all bad system configurations
  - Negation of loop freedom property



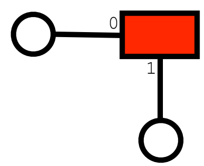
- Check if reachable from initial state
  - Backward reachability analysis

Backward reachability

Rule



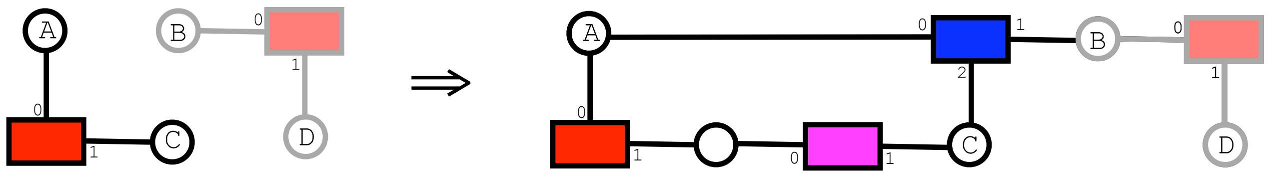
Pattern



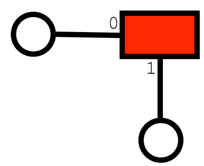
Pre set

Backward reachability

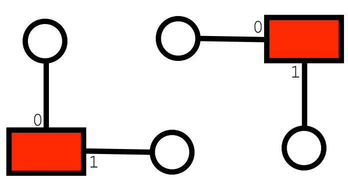
Rule



Pattern

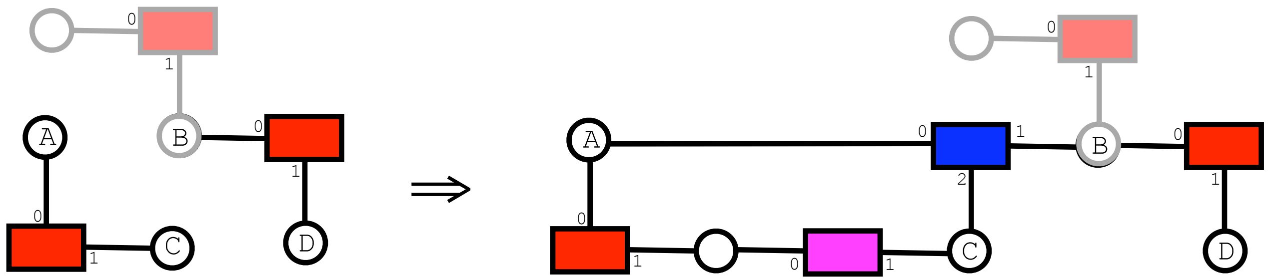


Pre set

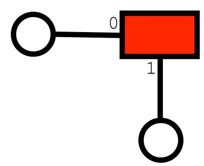


Backward reachability

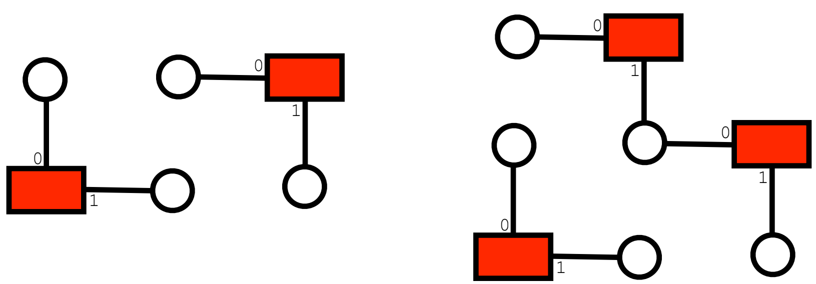
Rule



Pattern

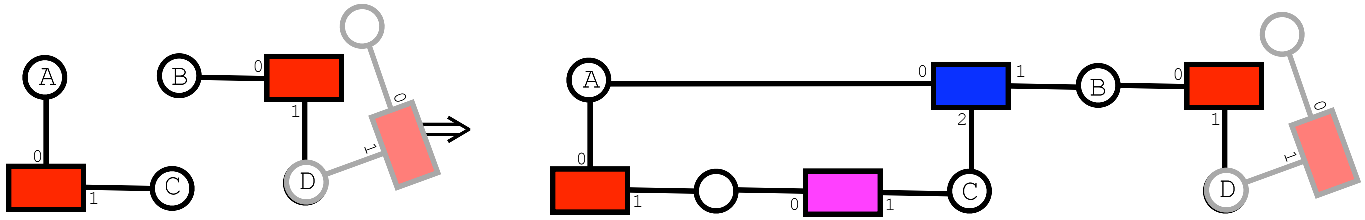


Pre set

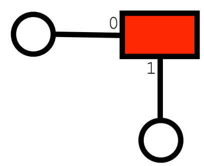


Backward reachability

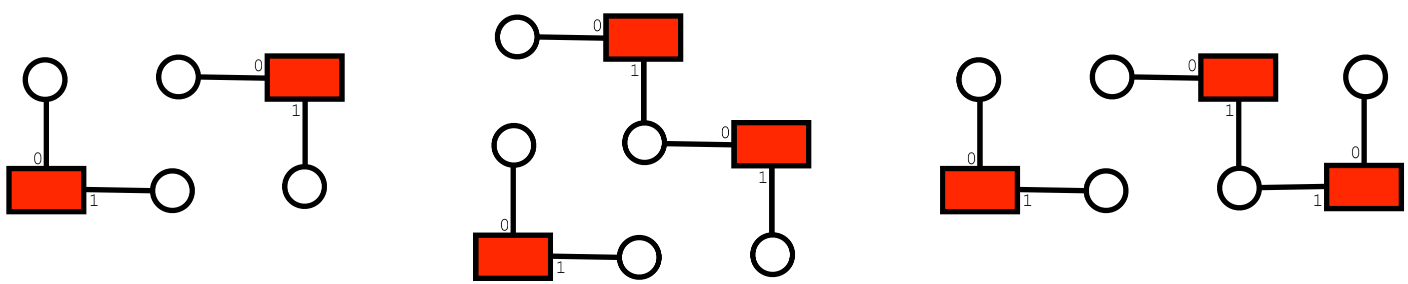
Rule



Pattern



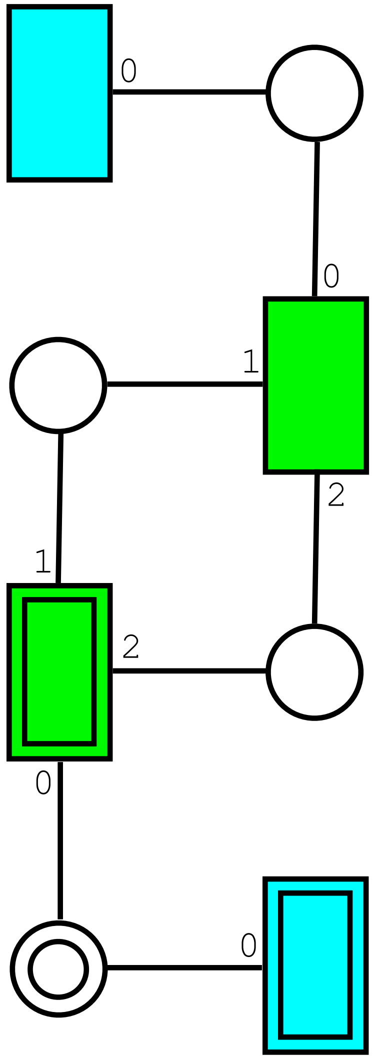
Pre set



...

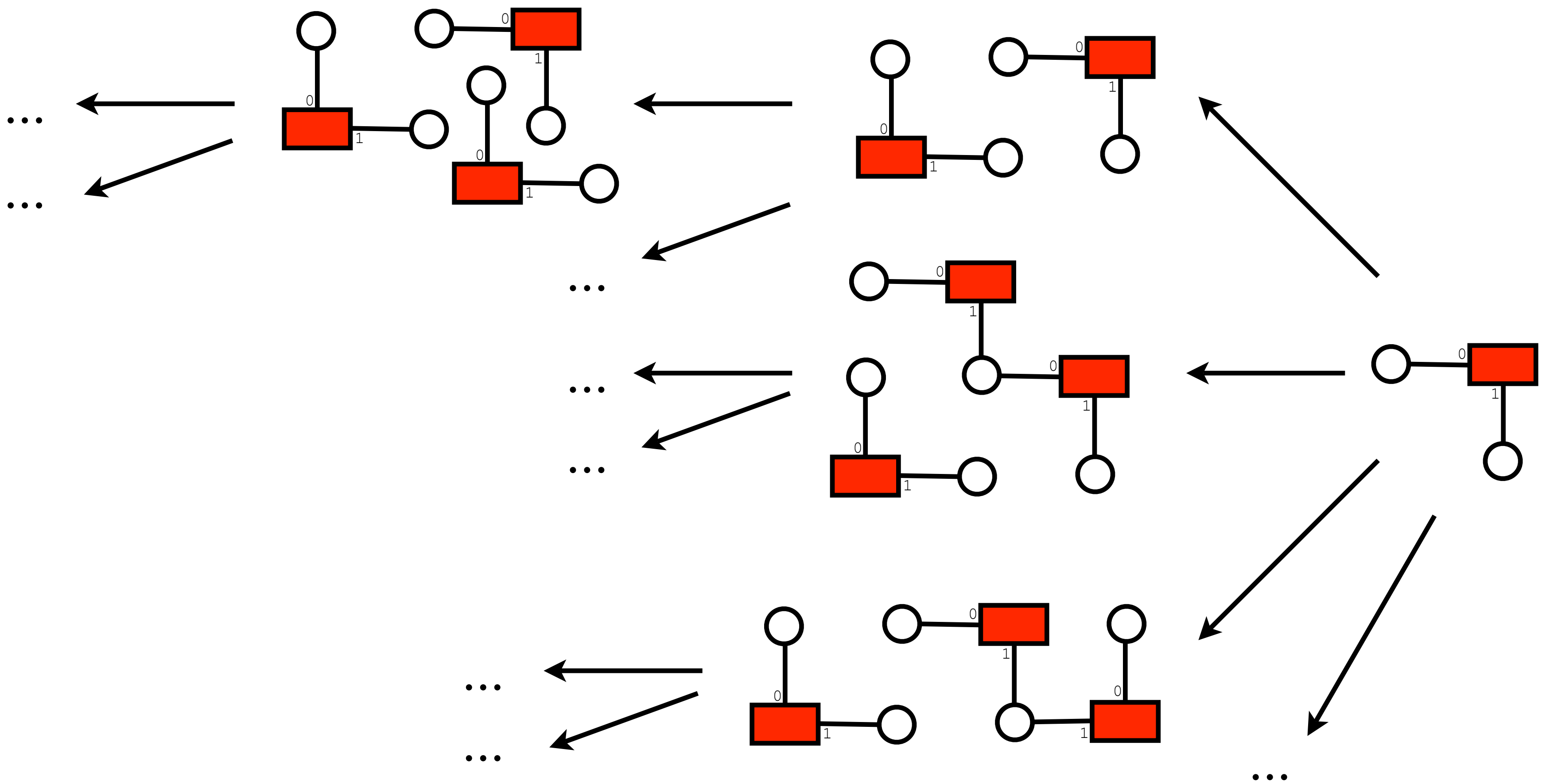
Backward reachability

- Some special situations
  - Graph segments created by rule
    - Not every overlap possible
  - Inconsistent patterns
    - Need to be detected and removed
    - Except when...
  - Inconsistency involves segment removed by rule
    - Inconsistency resolved, introduces abstraction



Many patterns generated

- May never terminate

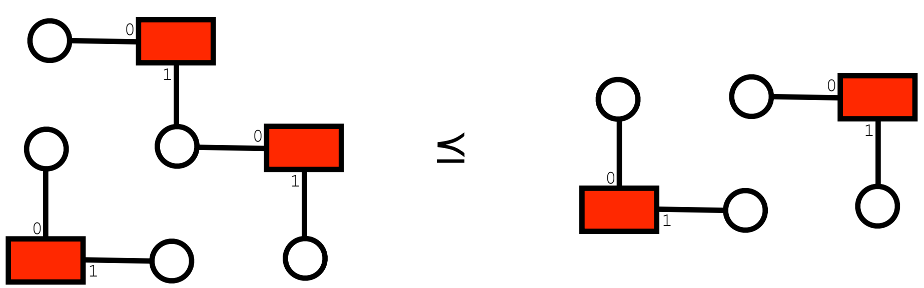


Many patterns generated

- Optimizations needed
- Want to safely discard patterns
  - Pattern subsumption
  - Simple type checking

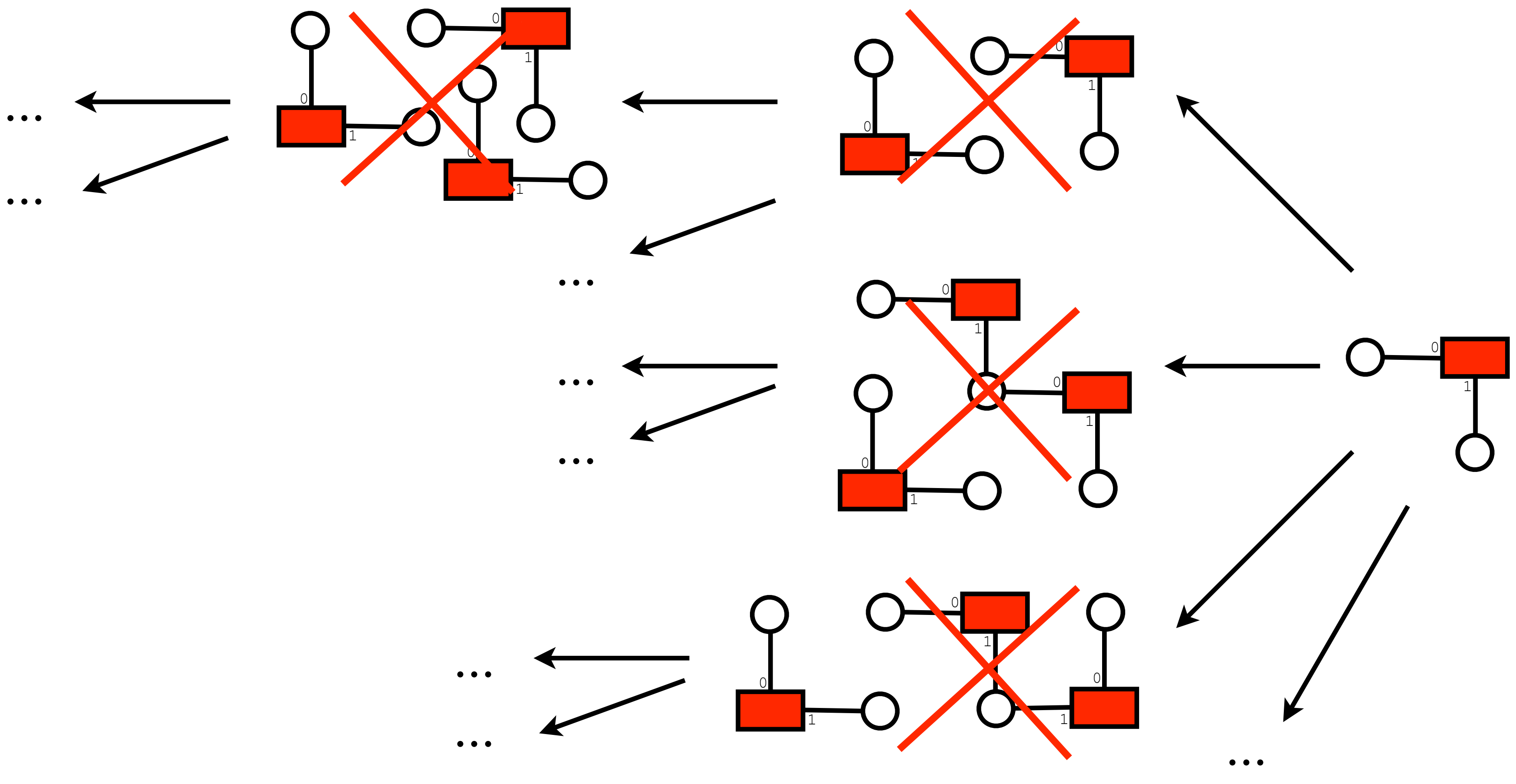
Pattern subsumption

- Patterns are generators for sets of graphs
- For a pattern φ
  - ⟦φ⟧ - set of graphs represented by φ
- For patterns φ and ψ
  - $\varphi \preceq \psi$ iff $[\![\varphi]\!] \subseteq [\![\psi]\!]$ (definition)
- Example



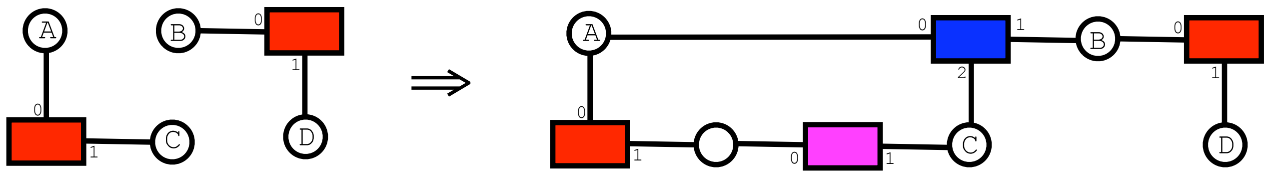
Pattern subsumption

- Discard covered predecessors

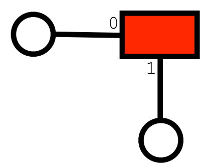


Simple type checking

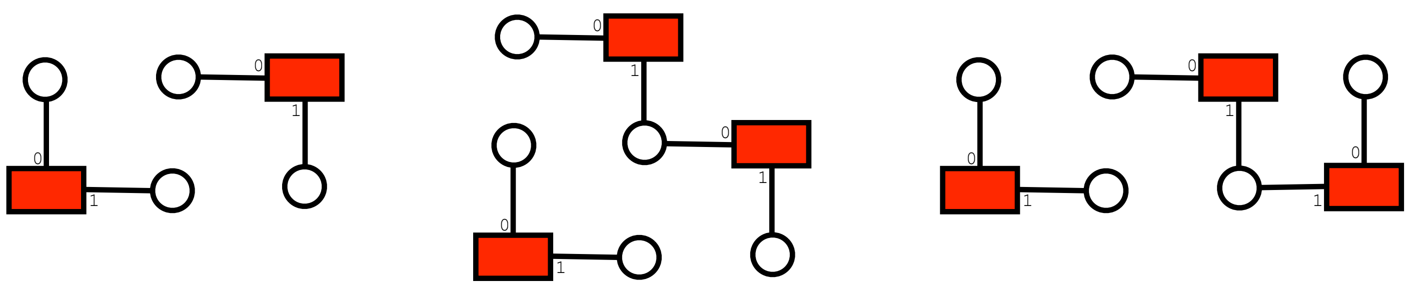
Rule



Pattern

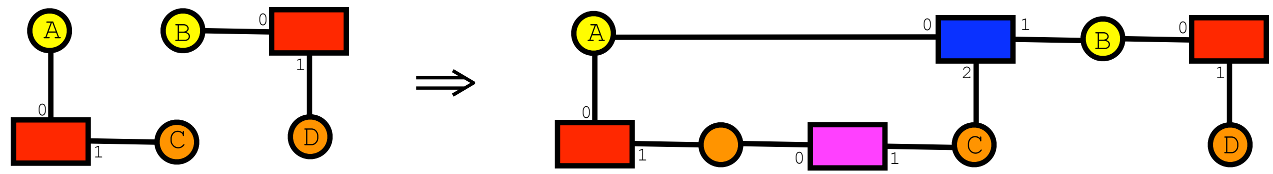


Pre set

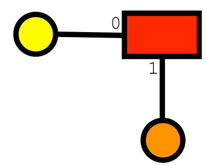


Simple type checking

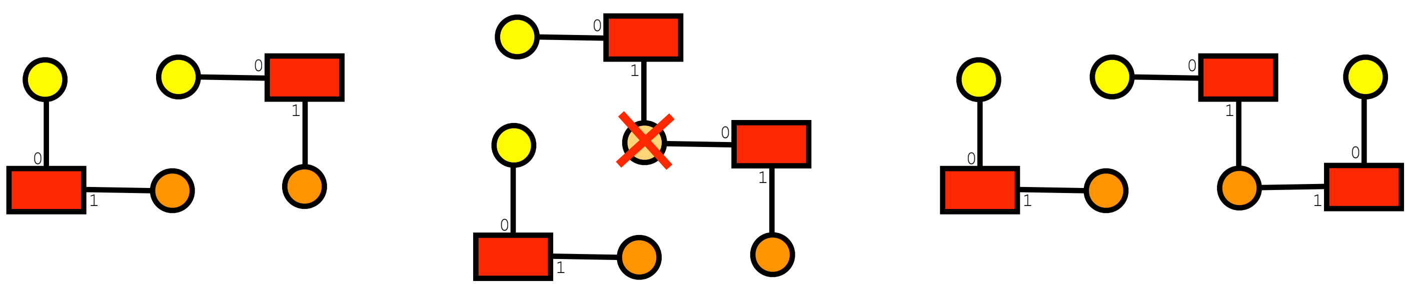
Rule



Pattern



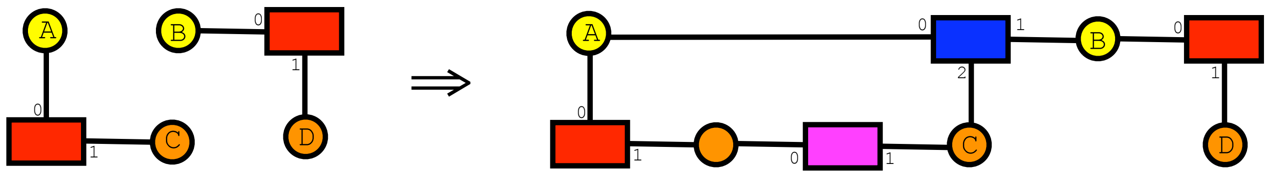
Pre set



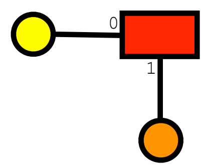
Network node   Sequence number

Simple type checking

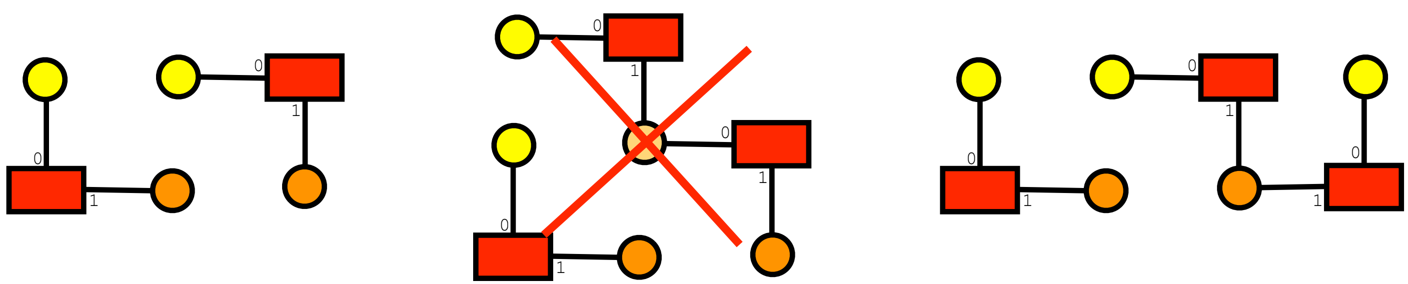
Rule



Pattern



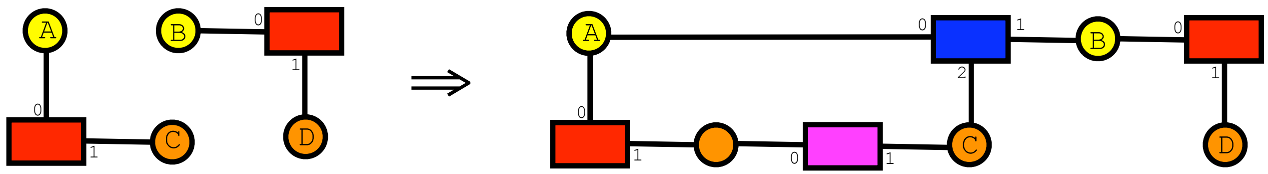
Pre set



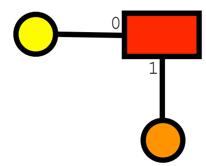
Network node Sequence number

Simple type checking

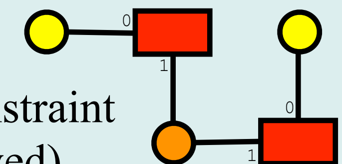
Rule



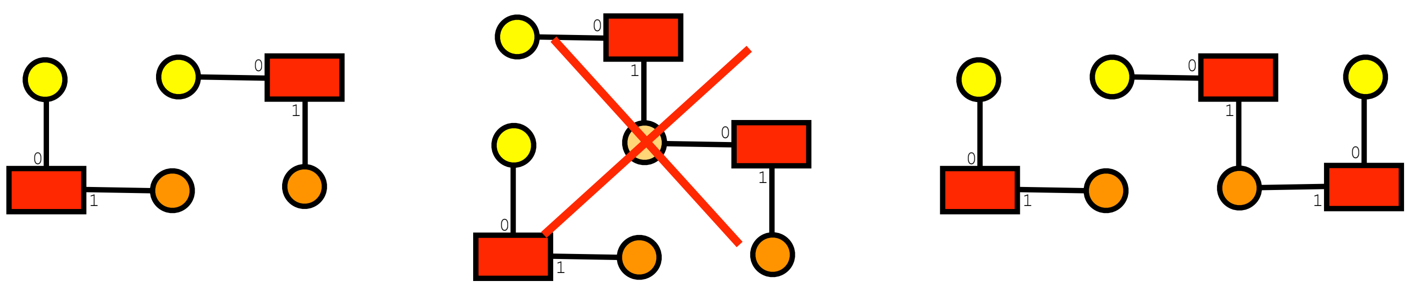
Pattern



Type constraint
(disallowed)



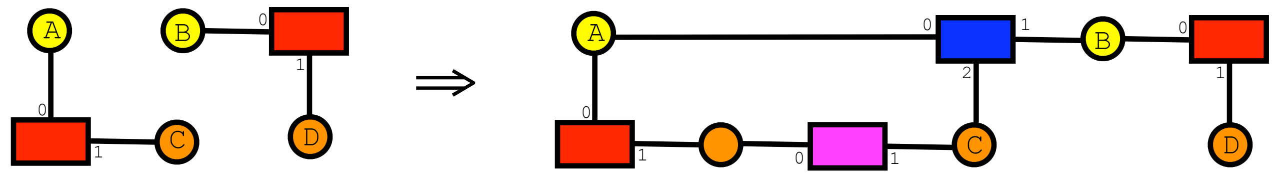
Pre set



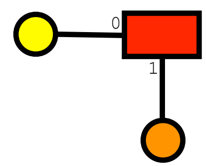
Network node   Sequence number

Simple type checking

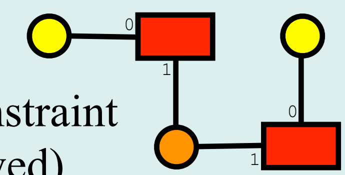
Rule



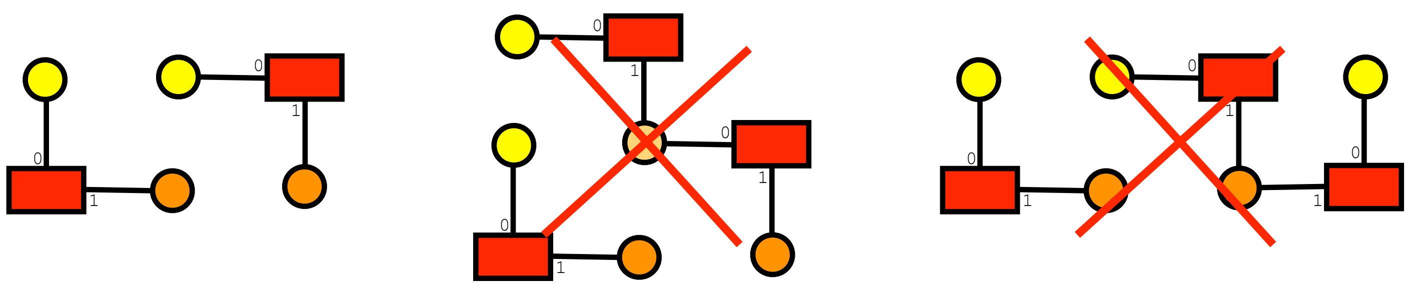
Pattern



Type constraint
(disallowed)



Pre set



Network node   Sequence number

Graph Backwards Tool (GBT)

- Takes `.grm` file as input
  - Lists `.dot` files describing:
    - initial graph, rewriting rules,
    - undesirable patterns, and
    - type constraints
- Outputs *"verification successful"* or error trace
  - Can be spurious (due to over-approximation)
- Tool and examples available for download
  - `http://www.it.uu.se/research/group/mobility/adhoc/gbt`

Verification results

- Two DYMO versions verified
  - Difference:
    - Intermediate nodes can reply in v10
    - Slight change in update rules
- Machine: 64-bit 2.8 GHz processor, 8 GB available memory

Protocol	**Actions**	**Checked**	**Covered**	**Left**	**Loop free**	**Time**
DYMO v10	77	295164	295108	56	Yes	4h 31 min
DYMO v05	50	118685	118637	48	Yes	1h 20 min

Updated verification results

- New optimizations implemented since paper
  - Example
    - Patterns often subsumed by immediate predecessor
    - Check this early
    - Reduces memory footprint, increases speed

Protocol	Actions	Checked	Covered	Left	Loop free	Time
DYMO v10	77	254620	254610	10	Yes	1h 59 min
DYMO v05	50	119506	119496	10	Yes	39 min 20 s

Related work

- König and Kozioura
  - Over-approximate graph grammars using Petri nets
  - No negative conditions
- Becker et al.
  - Graph grammars, verification of mechatronic systems
  - Only check given inductive invariant
- Abstract interpretation
  - Predicate abstraction
    - Need to find/devise relevant predicates
    - Abstractions may be too coarse

Conclusions and Future work

- Verified loop freedom of DYMO automatically
  - Optimize model - faster verification?
- A few other example systems
  - Working on more case studies
- Implemented some optimizations
  - More can be done
    - Early detection of unfruitful mappings
- We can get spurious counterexamples
  - CEGAR - ongoing
    - Introduce more abstraction, force termination