

Static Analysis
of
Dynamic Communication Systems

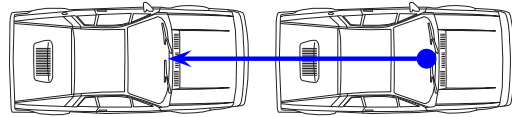
Jörg Bauer
Technical University of Denmark
February 21st, 2007

Motivation – Car Platooning

Coordinated groups of **fully automated** vehicles, called **platoons**, can double or triple **highway capacity** when operating in their own dedicated lanes.

DEMO

Motivation – Car Platooning



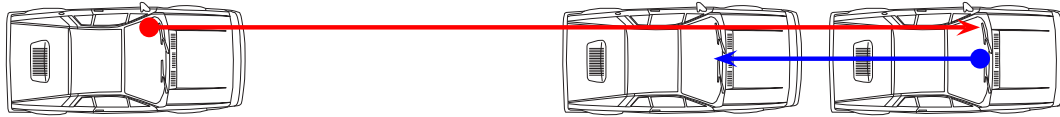
A car **platoon** of size two.

Motivation – Car Platooning



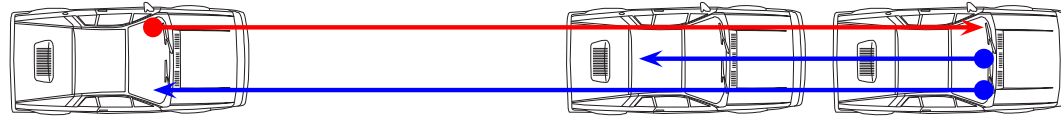
A **free agent** appears and approaches the platoon.

Motivation – Car Platooning



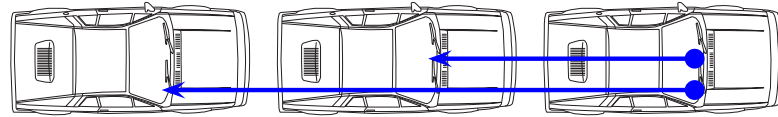
The free agent request a [merge](#) from the platoon [leader](#).

Motivation – Car Platooning



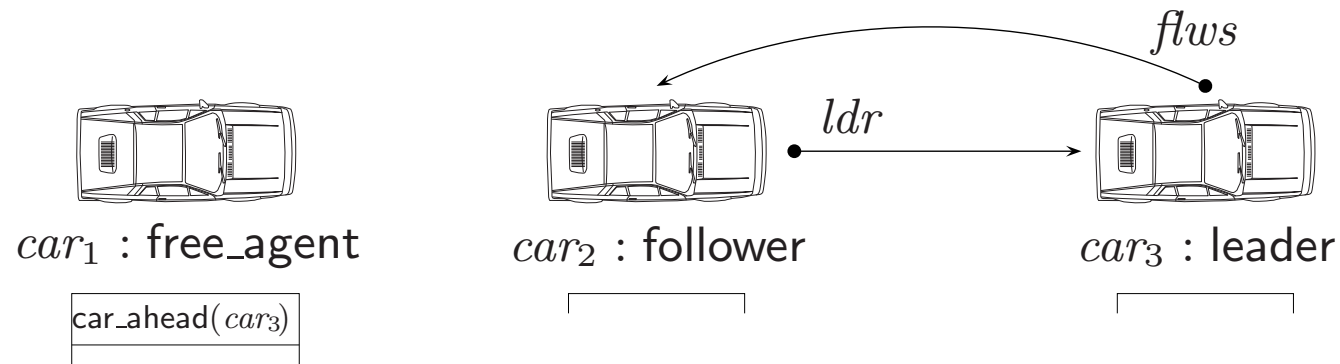
The leader accepts the free agent as a new **follower**.

Motivation – Car Platooning



A car platoon of size three is established.

Communication Topology



- Local **states**: free_agent, follower, leader.
- **Channels**: ldr, flws.
- **Message queues**.

Challenge

Specification and verification of systems with

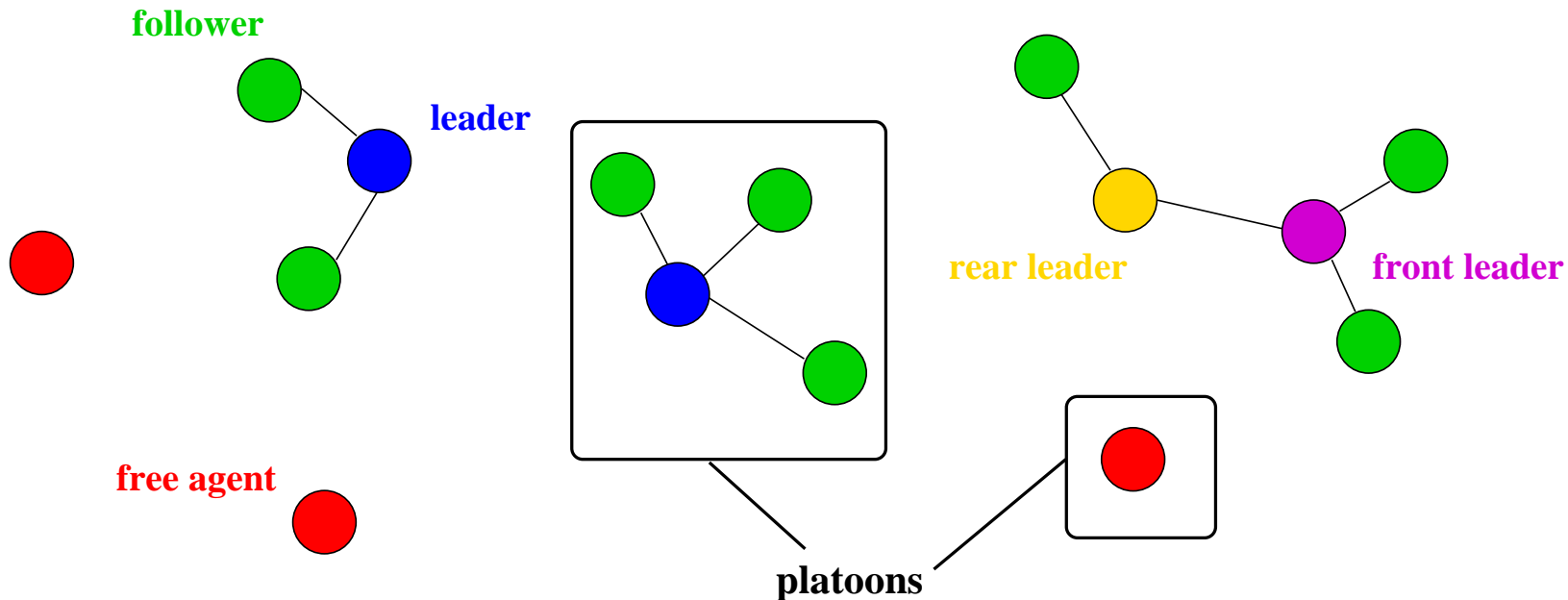
- Dynamically evolving communication topologies.
- Changing and unbounded number of objects.
- Properties of interest (platoons):
 - Topology Properties: unique leaders, no lonely followers, no leader cycles, ...
 - Temporal properties: Merge will finish (successfully), cars will always be able to leave the platoon.

Static Analysis of DCS

- Outline:
 - Partner graph grammars (PGG)
 - Abstract interpretation of PGG's
 - Platoon Case Study: Results
 - Extensions, conclusions, further reading

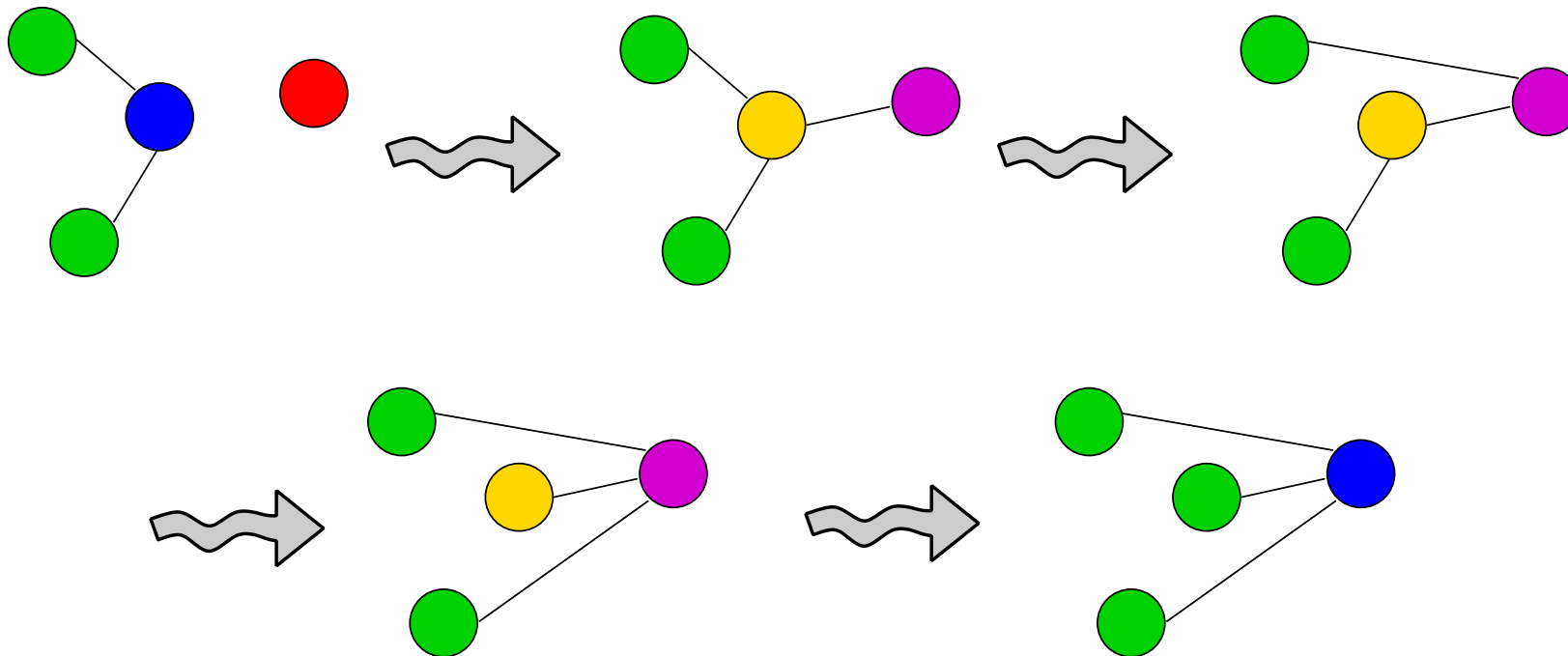
Platoons as Graphs – Idealized

- Abstract from **queues** and **explicit messages**.
- **Perfect** communication.
- Platoon terminology



Platoon Merge

- Non-deterministically start a merge.
- Rear leader hands over followers to front leaders.
- Finish merge after the hand-over.



Partner Graph Grammars

- Assume undirected, node-labeled graph $G = (V_G, E_G, \ell_G)$ over finite set \mathcal{N} of node labels
- A PGG is a pair $(\mathcal{R}, \mathcal{I})$ of rules and initial graph.
- A rule is a four-tuple (L, h, p, R) , where
 - L is the left graph
 - $h : V_L \rightarrow V_R$ is a (partial) mapping
 - $p : V_L \rightarrow \mathcal{P}(\mathcal{N})$ are partner constraints
 - R is the right graph
- SPO approach with injective matching and a mild form of negative application conditions.

Semantics

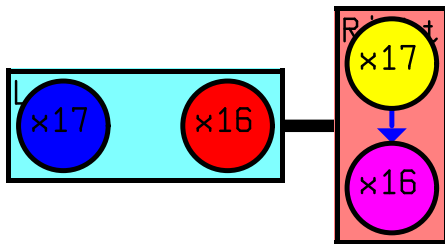
- **Application** of rule (L, h, p, R) to graph G :
 1. **Match**: an **injective** morphism $m : V_L \rightarrow V_G$
 2. Partner constraint **satisfaction**: For all $u \in \text{dom}(p)$ holds:

$$p(u) = \{\ell_G(v) \mid \{u, v\} \in E_G\}$$

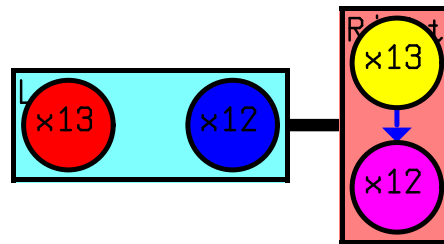
3. Replace image of m by R .
- **Graph semantics** $\llbracket (\mathcal{R}, \mathcal{I}) \rrbracket$: set of all graphs obtainable by iterated rule application from \mathcal{I} . (undecidable)

Platoon Partner Graph Grammar

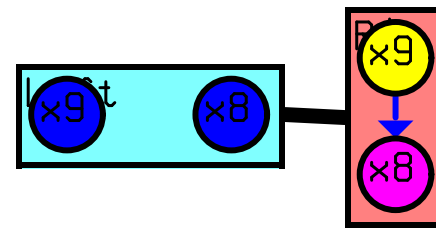
- Arbitrary creation and destruction of free agents



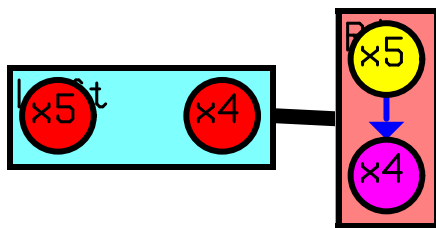
free agent and leader start to merge...



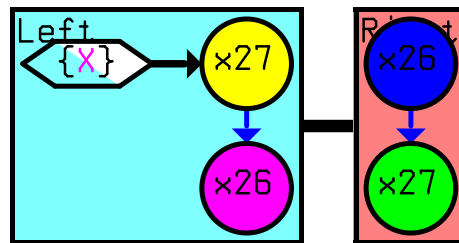
...to be rear leader and front leader



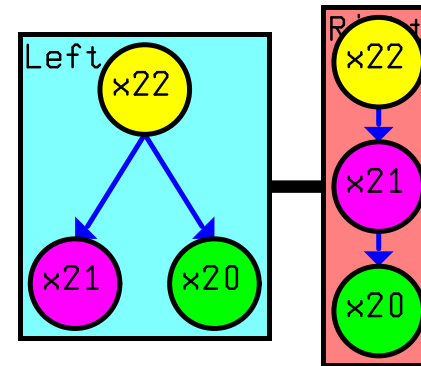
two leaders start to merge



two free agents start to merge



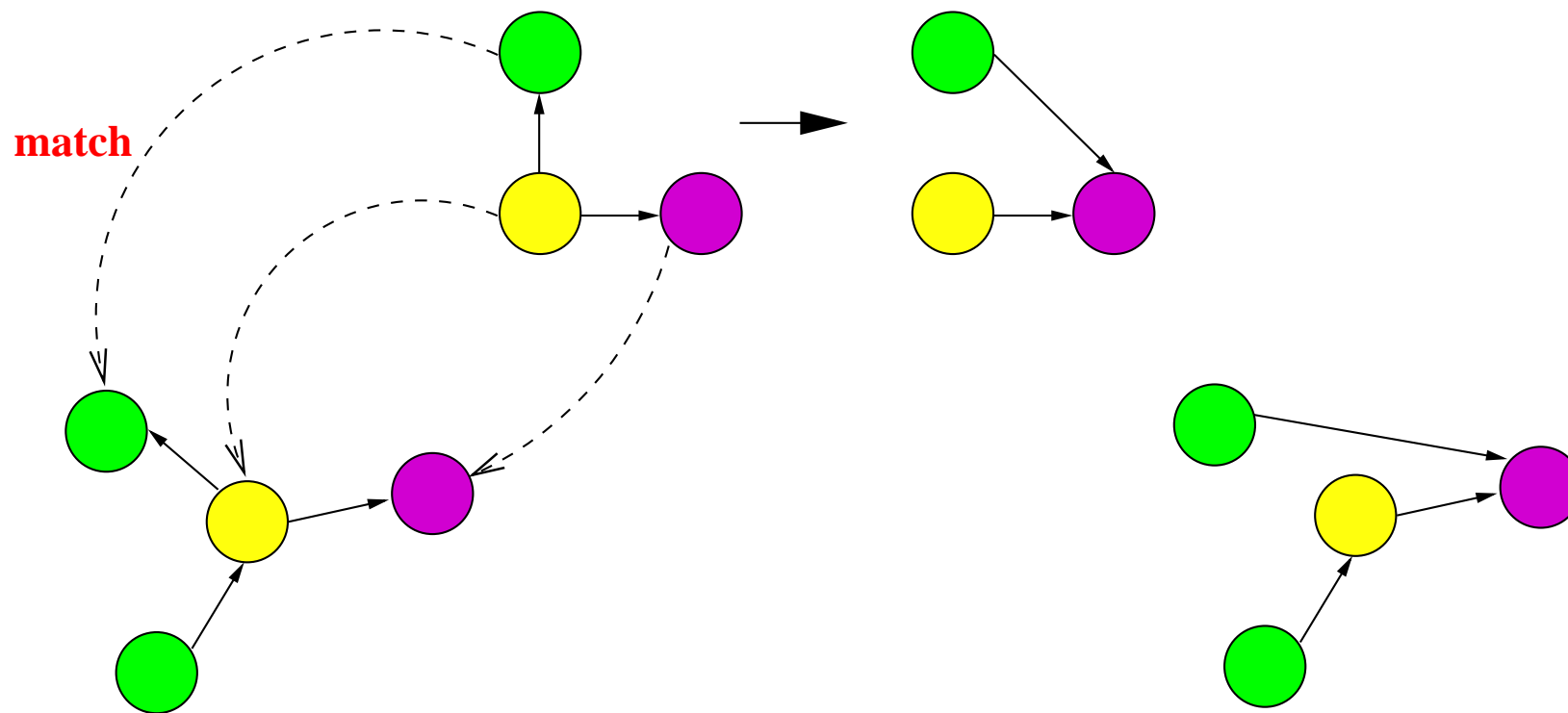
end of merge



hand over follower from rear to front

Sample Rule Application

Follower Hand-Over



Overview

- Partner graph grammars (PGG)
- Abstract interpretation of PGG's
 - Abstraction function
 - Abstract transformers
- Results
- Extensions, conclusions, further reading

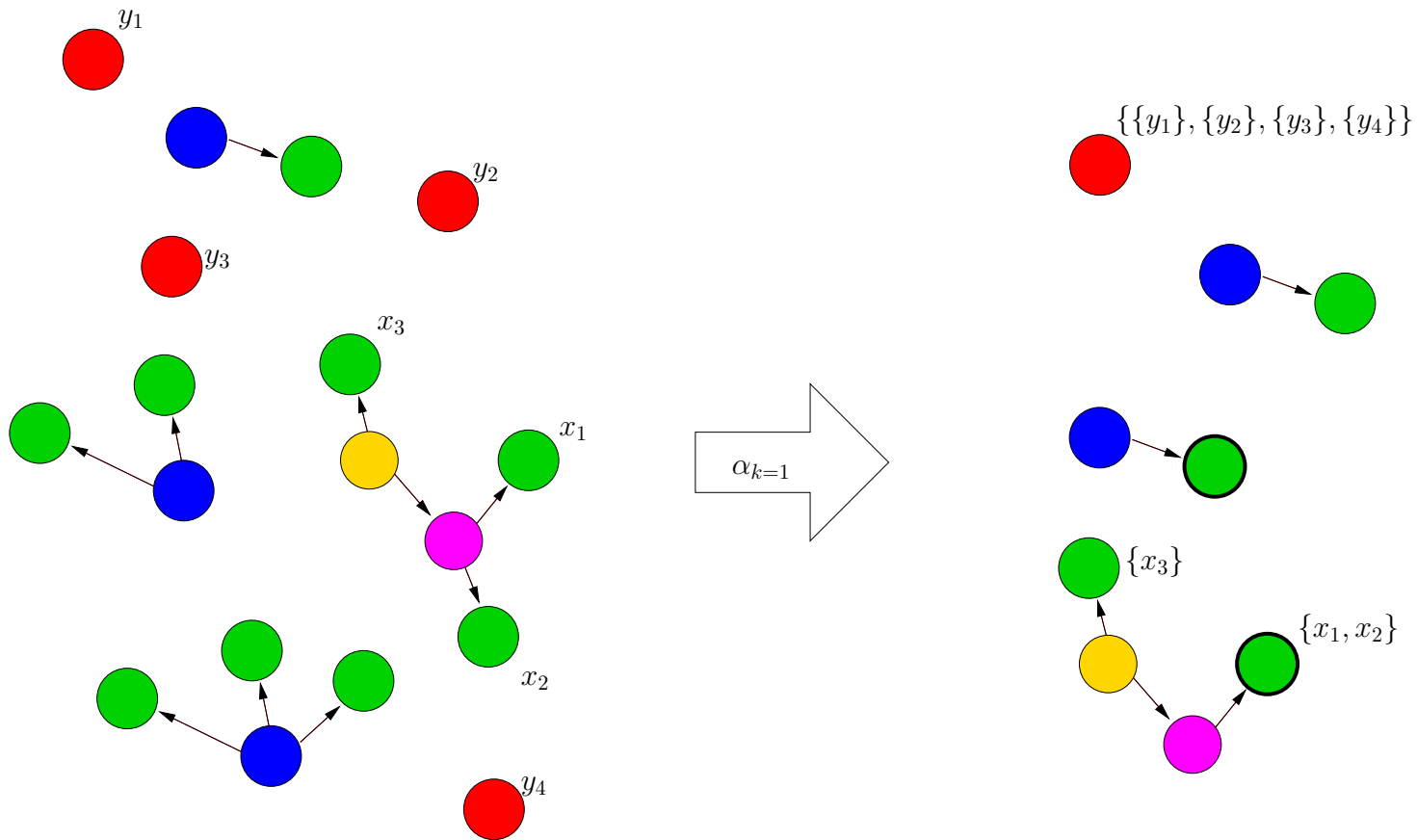
Abstraction of Graphs

- Two-layered abstraction.
- For each connected component (**cluster**) of graph (V, E, ℓ) , two nodes are **partner equivalent** iff

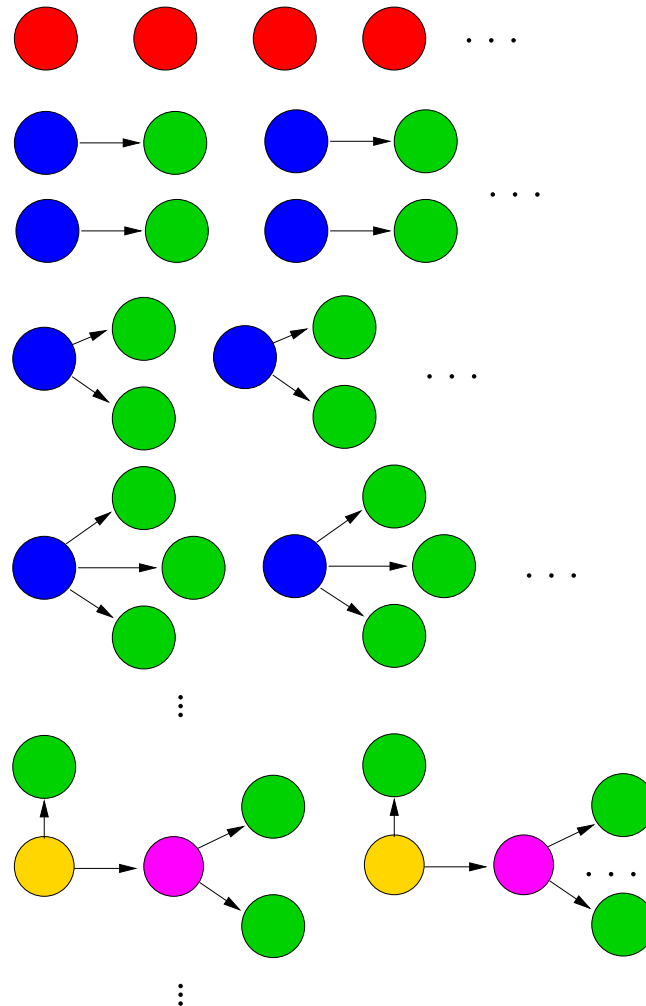
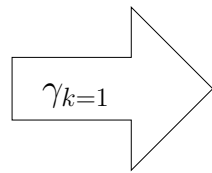
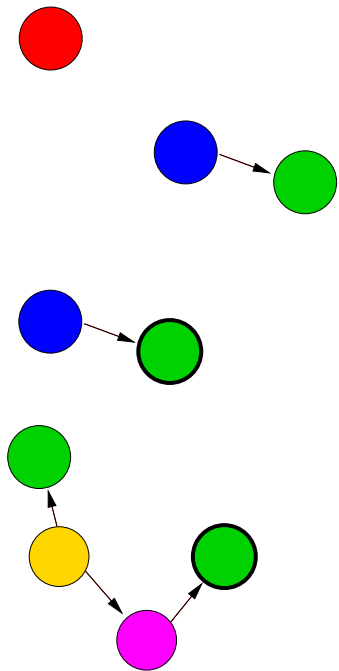
$$\ell(u) = \ell(v) \wedge \{\ell(u') \mid \{u', u\} \in E\} = \{\ell(v') \mid \{v', v\} \in E\}$$

1. Build quotient graph **cluster-wise** wrt. partner equivalence while **tracking multiplicity** up to some k .
2. Summarize **isomorphic** connected components: **abstract clusters**

Example Abstraction



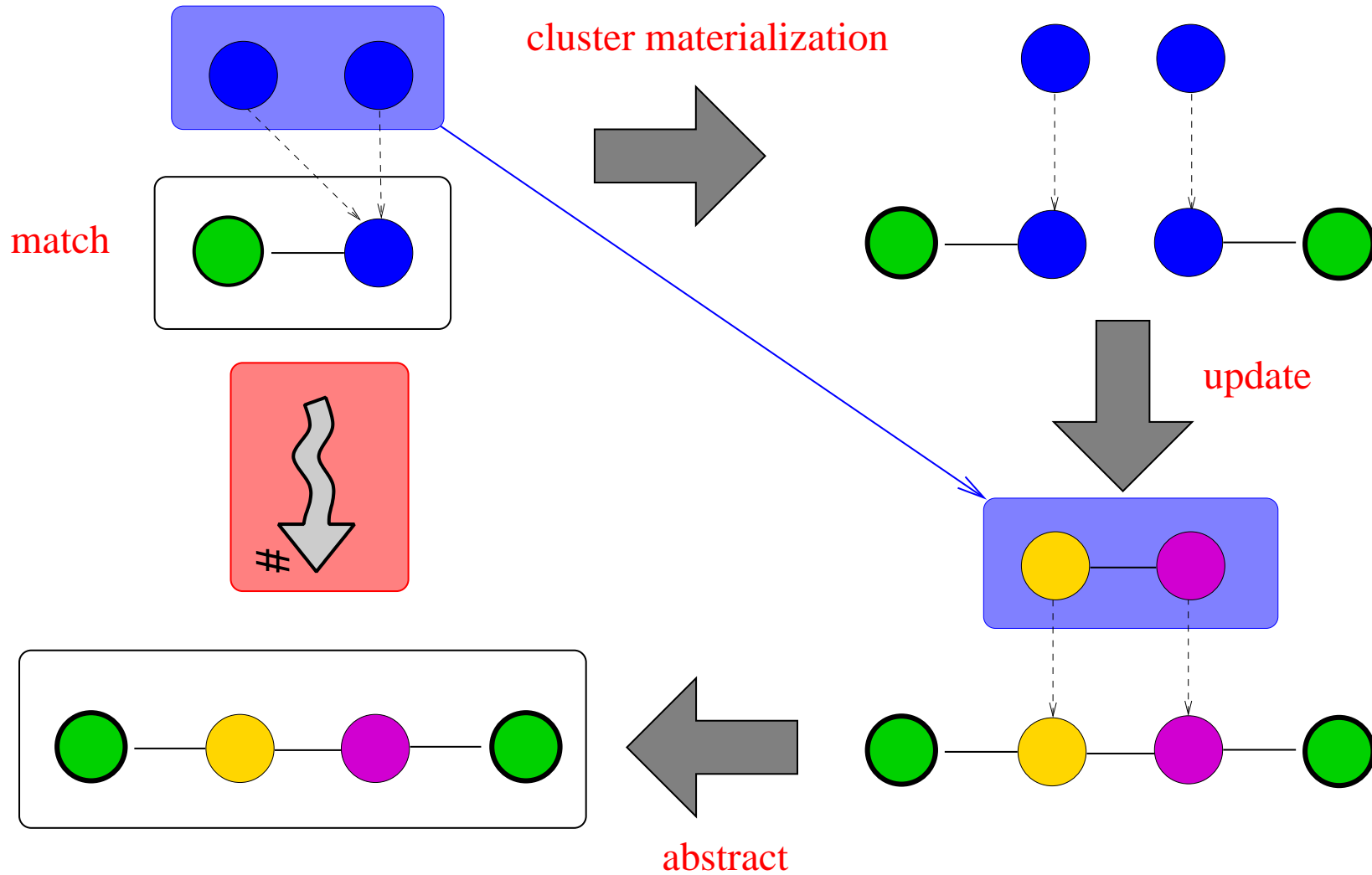
Concretization



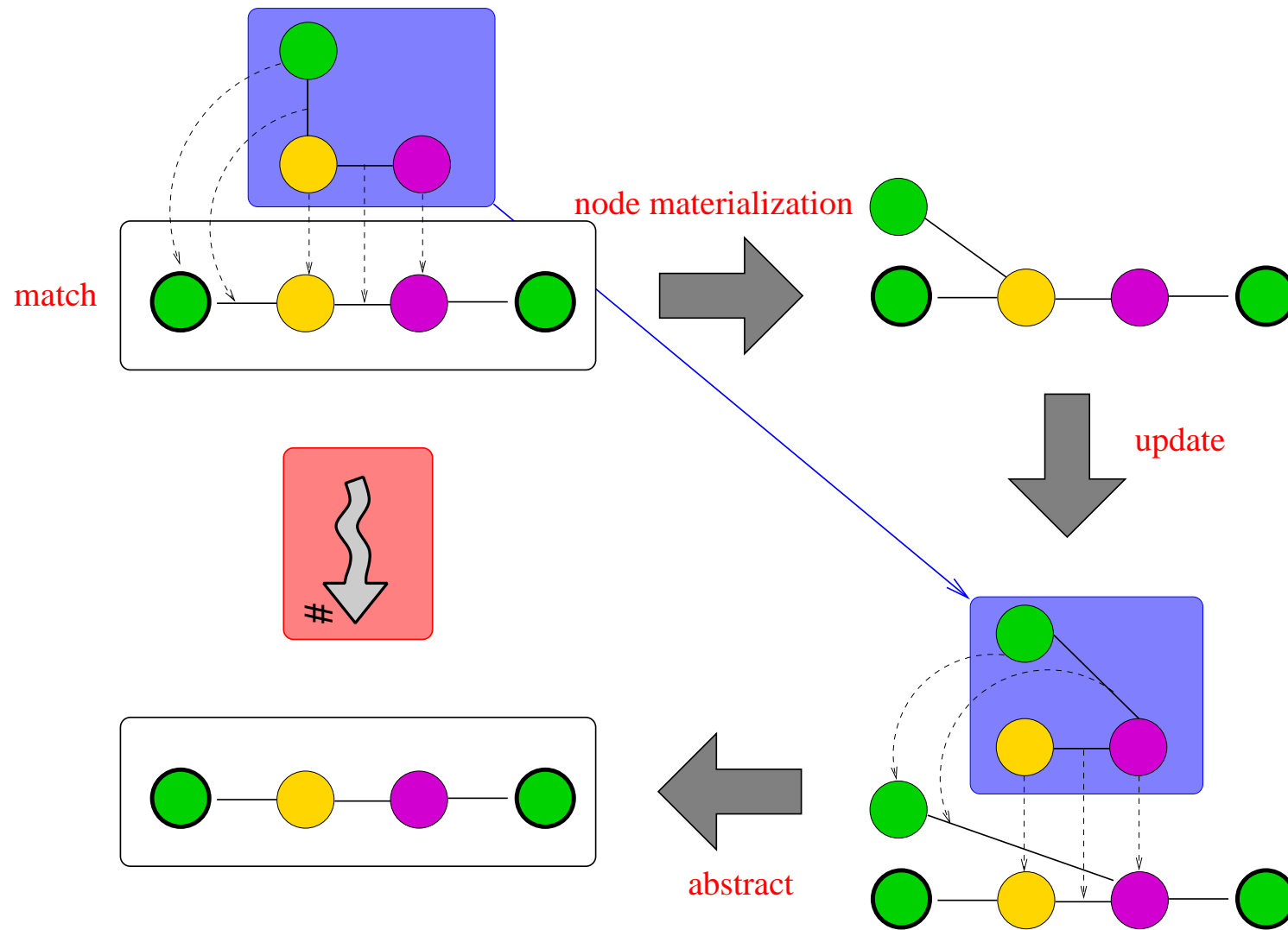
Abstract Updates—Best Abstract Transformers

1. **Non-injective** matches due to abstraction.
2. **Materialization** locally undoes abstraction: injective match
 - **Node** materialization
 - **Cluster** materialization
3. Update like in the concrete case.
4. **Abstract** to guarantee boundedness.
 - **Abstract graph semantics**: finite, bounded set $[[\mathcal{R}]]^k$ of abstract clusters

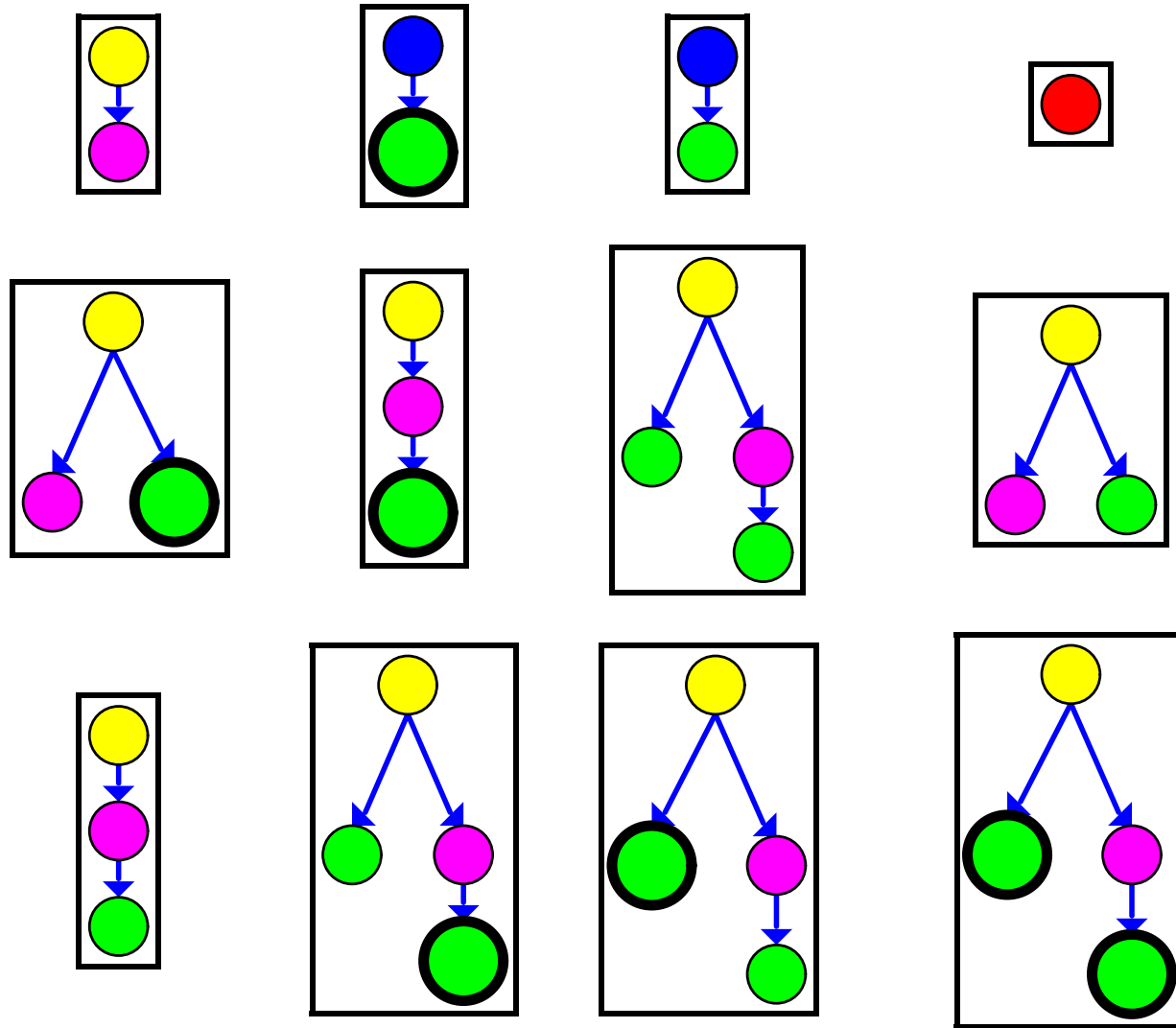
Sample Abstract Update



Sample Abstract Update



Platoon Abstract Graph Semantics



Overview

- Partner graph grammars (PGG)
- Abstract interpretation of PGG's
- Results
 - Soundness
 - Completeness
 - Experiments
- Extensions, conclusions, further reading

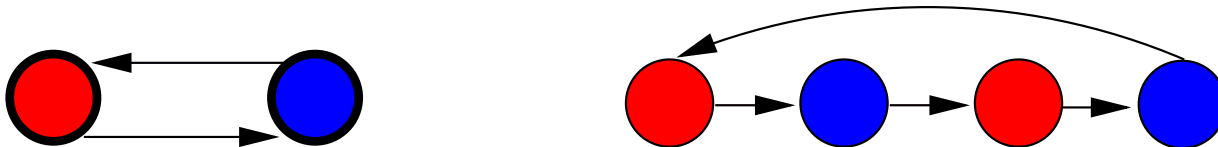
Soundness

- The abstract graph semantics is an **over-approximation** of the concrete graph semantics: $\bigcup_{G \in \llbracket \mathcal{R} \rrbracket} \alpha_k(G) \subseteq \llbracket \mathcal{R} \rrbracket^k$
- **Forbidden Subgraphs**
 - Given a forbidden subgraph F
 - $\mathcal{N} := \mathcal{N} \dot{\cup} \{evil\}$
 - $\mathcal{R} := \mathcal{R} \dot{\cup} \{F \rightarrow evil\}$
 - If $evil \notin \llbracket \mathcal{R} \rrbracket^k$, then F does not occur in $\llbracket \mathcal{R} \rrbracket$

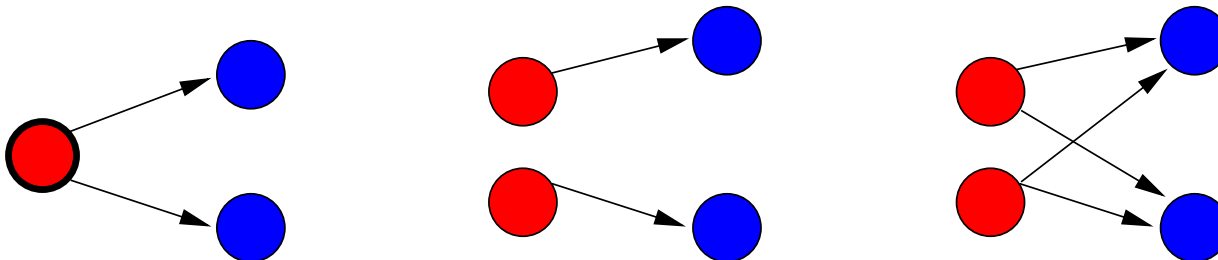
Towards Completeness

- **Friendly Grammars:** No initial graph implies **cluster multiplicity**.

- **Summary Cycles.**



- **Unique Partners.**



- **Lone Summaries:** No adjacent summary nodes.

- **Inductive Summaries:** all concretizations possible.

Completeness

- No summary cycles + unique partner imply **Matching**
Theorem: Abstract matches are **equivalent** to concrete matches.
- Lone summaries + unique partner + inductive summaries
imply **cluster completeness**: $\bigcup_{G \in \llbracket \mathcal{R} \rrbracket} \alpha_k(G) = \llbracket \mathcal{R} \rrbracket^k$.
- **Property preservation**: Lone summaries + unique partners
imply (for connected graphs C) $C \models \psi \iff \alpha_k(C) \models \psi$, for
first-order formulas ψ **without equality**.

Experimental Evaluation of Platoons

Example	N. Labels	E. Labels	Rules	Constraints	Clusters	Rule App.	Iter
Merge	5	1	8	1	12	19	5
Split	6	1	4	4	13	15	4
Combined	6	1	12	5	169	302	10
Combined+	6	1	12	9	22	34	7
Queues	18	4	30	34	159	163	40
Faulty1	5	1	32	1	20	53	6
Faulty3	5	2	32	1	450	1206	10

- Relevant topology properties proven, sometimes completeness.
- Useful for protocol design, too.
- Partner graph grammars are highly sensitive to changes, in particular to partner constraints.

Overview

- Partner graph grammars (PGG)
- Abstract interpretation of PGG's
- Results
- Extensions, conclusions, further reading

Existing Extensions

- Syntactic sugar for PGG's; directed, edge-labeled graphs.
- **Counting** clusters.
- **General** clusters.
- A **logic** for reasoning about PGG's.

Potential Extensions

- **Attributed graphs** for queue implementations.
- **Unified framework** with general, **property-driven** . notions of (partner) equivalence.
- More **hierarchy** than just 2 (ad hoc routing).
- **Beyond dynamic communication systems.**

What have you learnt?

- **Graph Grammars**: powerful specification formalism for highly dynamic systems
 - Motives: Reiko Heckel, Andy Schürr.
 - <http://gratra.org/>
 - Handbook of Graph Grammars and Computing by Graph Transformation. World Scientific 1997 - 99, published in three volumes.
- **Platoon** case study
 - <http://www.path.berkeley.edu/>

What have you learnt?

- **Abstract interpretation:** powerful verification formalism
 - Patrick & Radhia Cousot. **Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints.** ACM POPL 1977.
 - Patrick & Radhia Cousot. **Systematic design of program analysis frameworks.** ACM POPL, 1979.
 - Hanne Nielson, Flemming Nielson, and Chris Hankin. **Principles of Program Analysis.** Springer 1999.
 - Motives: Sylvie Putot, Reinhard Wilhelm, Hanne Riis Nielson.

What have you learnt?

- Graph grammar verification
 - Barbara König, Vitali Kozioura. *Counterexample-Guided Abstraction Refinement for the Analysis of Graph Transformation Systems.*, TACAS 2006.
 - Arend Rensink, Dino Distefano. *Abstract Graph Transformation.* *Electr. Notes Theor. Comput. Sci.* 157(1): 39-59 (2006).
 - Reiko Heckel. *Compositional Verification of Reactive Systems Specified by Graph Transformation.* FASE 1998.
 - <http://www2.imm.dtu.dk/~joba/phd.pdf>