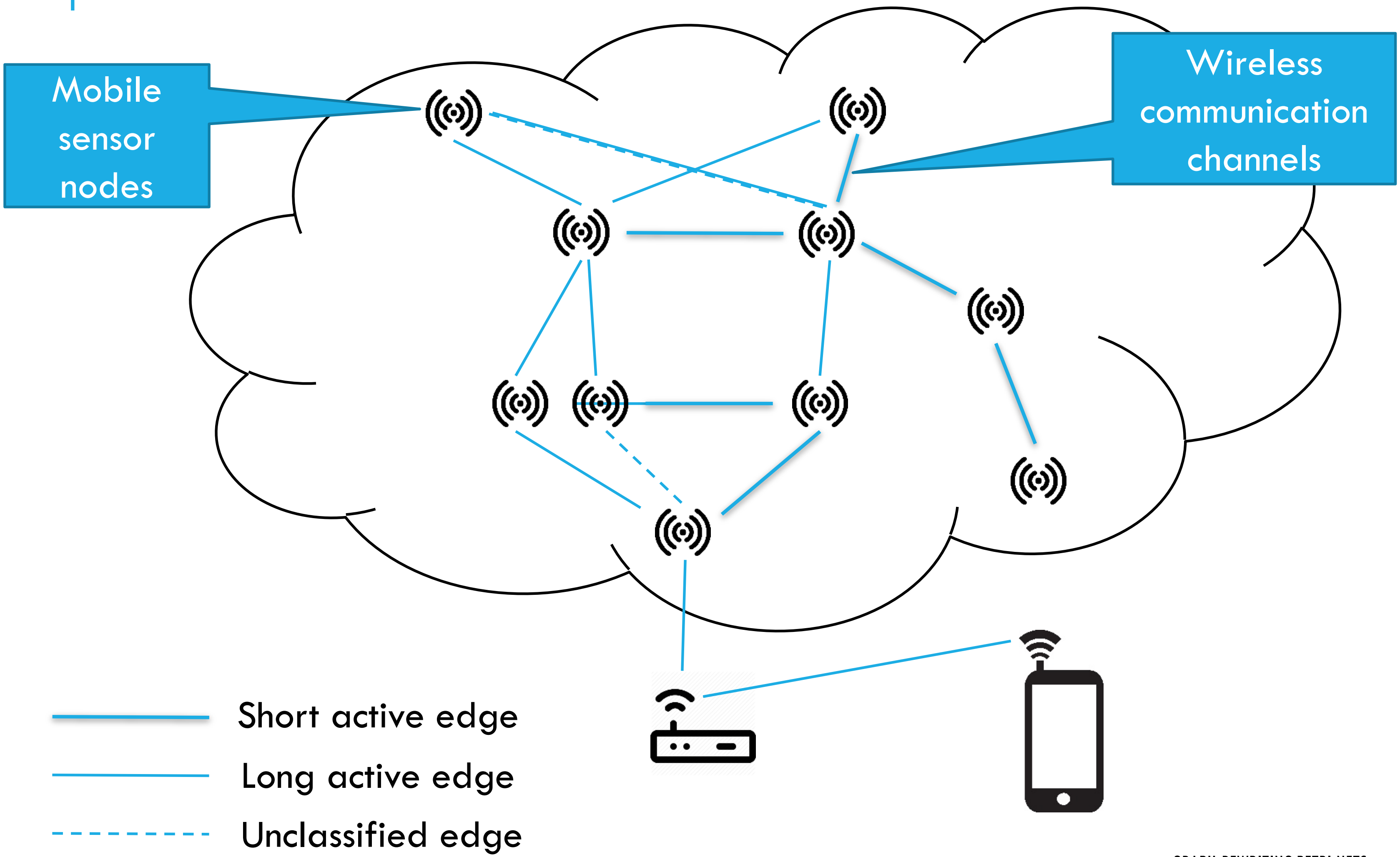
GRAPH-REWRITING PETRI NETS

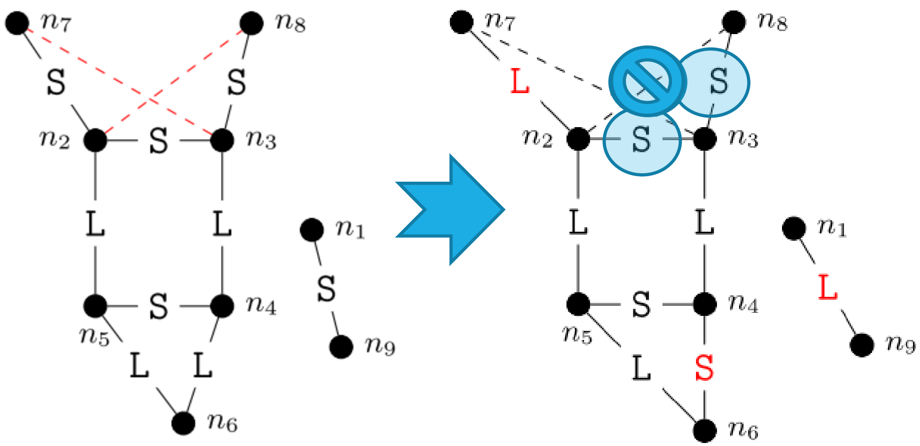
ICGT@STAF 2018, Toulouse

Géza Kulcsár, Malte Lochau,
Andy Schürr

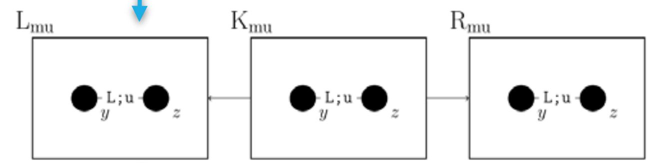
EXAMPLE: WSN TOPOLOGY CONTROL



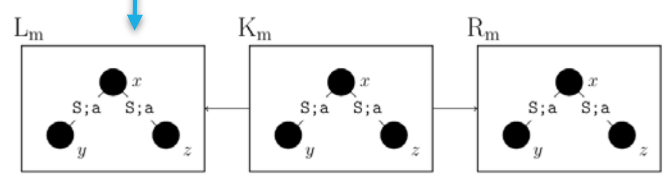
GRAPH-REWRITING PROCESSES



environmental event: short edge becomes long edge

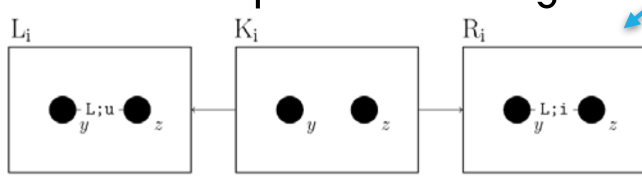


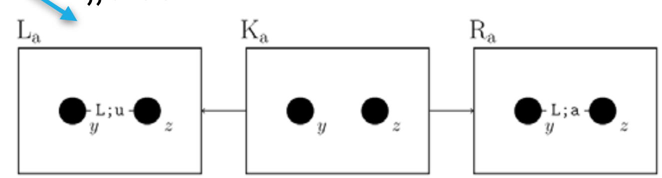
match unclassified long edge



match two adjacent short edges

„else“





inactivate long edge

activate long edge

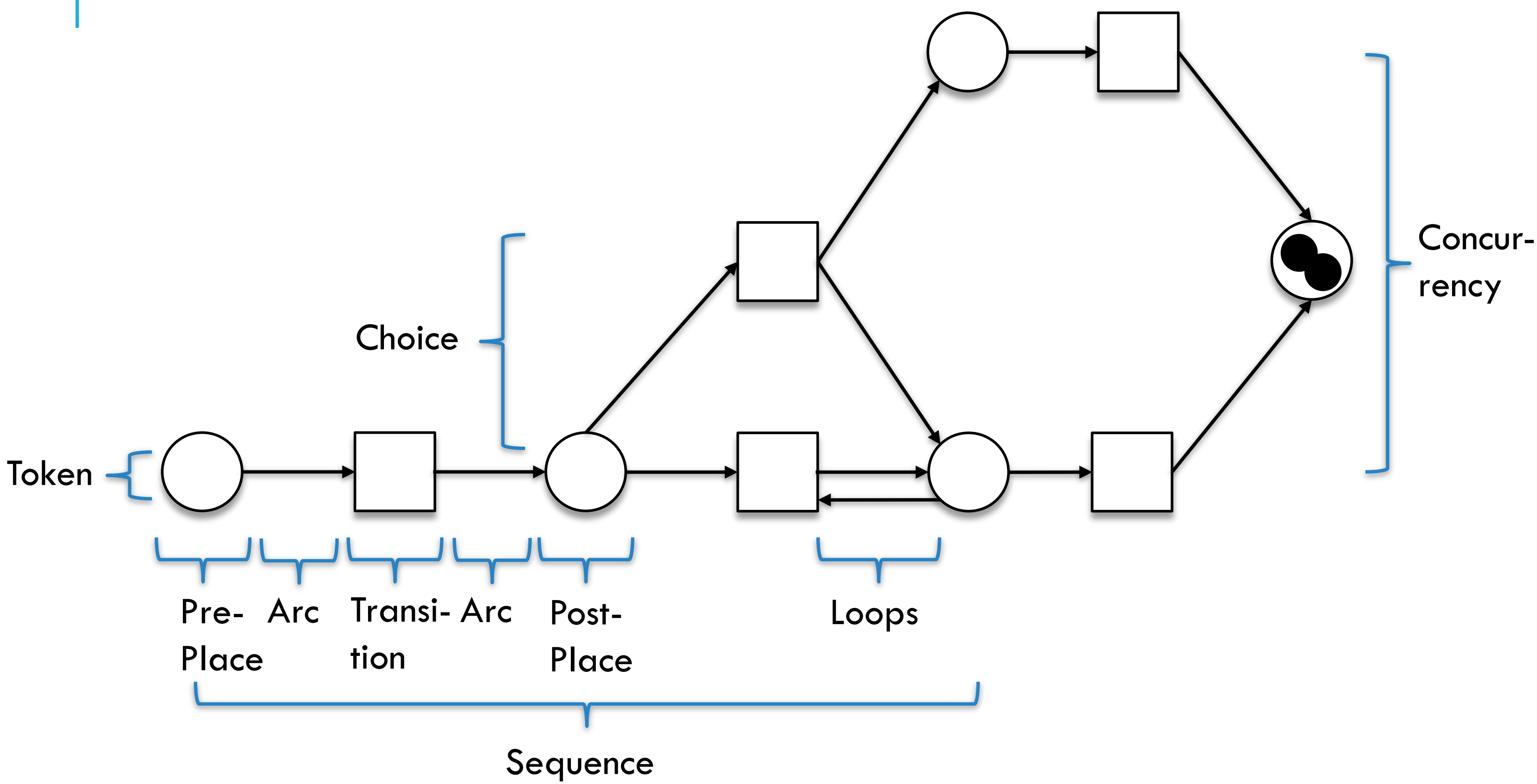
„loop forever“

OPEN CHALLENGES IN CONTROLLED GRAPH REWRITING

- Control-flow specification for graph-rewriting processes to restrict the **orderings** of graph-rewriting rule applications
- Data-flow specification for graph-rewriting processes to restrict the **matches** of graph-rewriting rule applications
- Control-flow and data-flow specification for graph-rewriting processes with **concurrent** rule composition and **synchronization** of rule applications
- Techniques for **automated analysis of correctness properties** for controlled graph-rewriting process specifications
    - „TC must preserve connectedness of input topology graphs“
    - „TC should eventually inactivate redundant long edges“
    - „TC must not deadlock due to concurrent interactions with environmental events“

➢ **Graph-Rewriting Petri Nets (GPN) = Coloured Petri Nets + DPO Graph Rewriting**

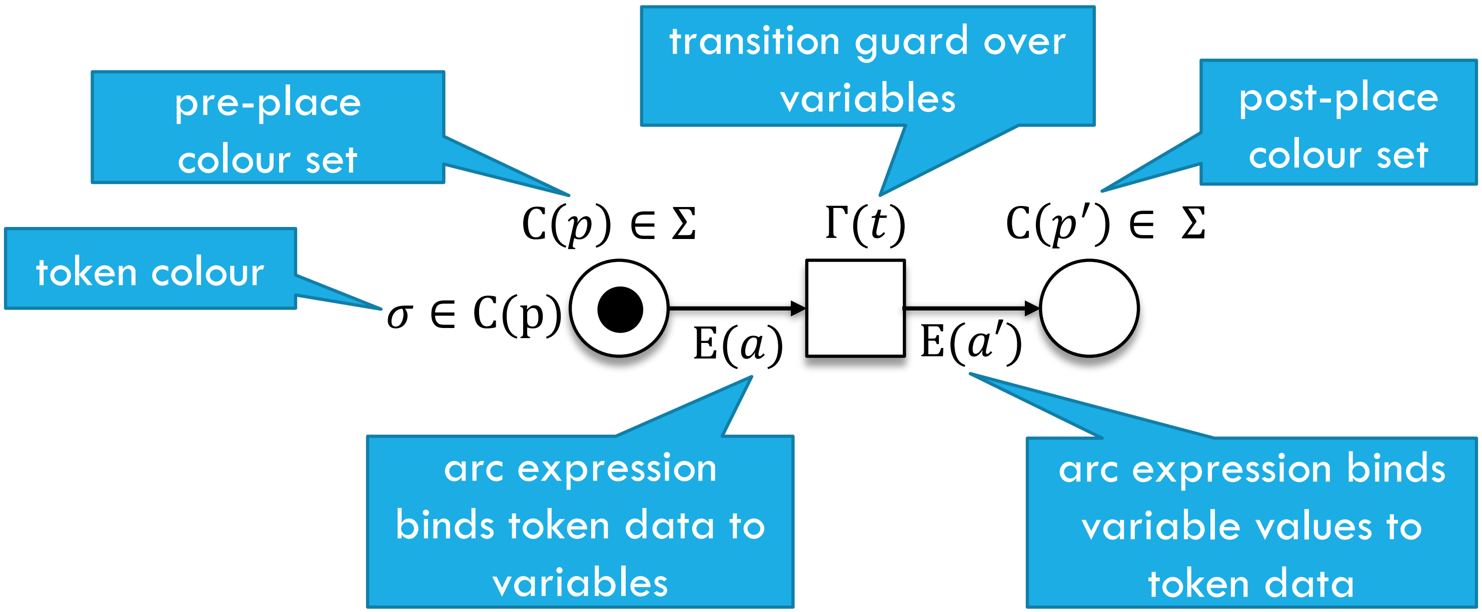
PETRI NETS, VISUALLY



PETRI NETS, FORMALLY

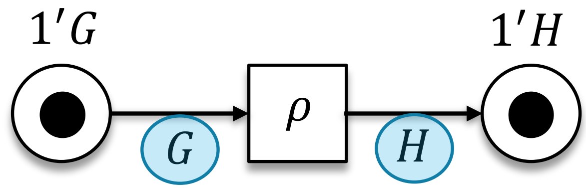
- A Petri Net N consists of a set of *places* P and a set of *transitions* T
- A *marking* $M \in \mathbb{N}^P$ of N is a multiset, where $M(p) \geq 0$ denotes the number of tokens on place p
- A *step* $M \overset{X}{\rightarrow} M'$ consists of a multiset of transitions $\mathrm{X} \in \mathbb{N}^T$
- A marking M is *reachable* from *initial marking* M_0 iff there exists a sequence of steps $M_0 \overset{X_0}{\rightarrow} \cdots \overset{X}{\rightarrow} M$
- Two transitions $t, u \in T$ are *concurrent* if there exists a reachable marking $M \xrightarrow{\{t,u\}} M'$
- A Petri Net N is *k-bounded* iff $M(p) \leq k$ for all reachable markings M and all places p
- A Petri Net N is *live* if for all reachable markings $M \overset{X}{\rightarrow} M'$

COLOURED PETRI NETS [JENSEN & CHRISTENSEN 2008]



- Places are **typed** over sets Σ of **colours**
- Tokens carry **data,** typed over sets of **colours**
- Transitions and arcs are augmented with **inscriptions** over typed **variables** $v \in V$

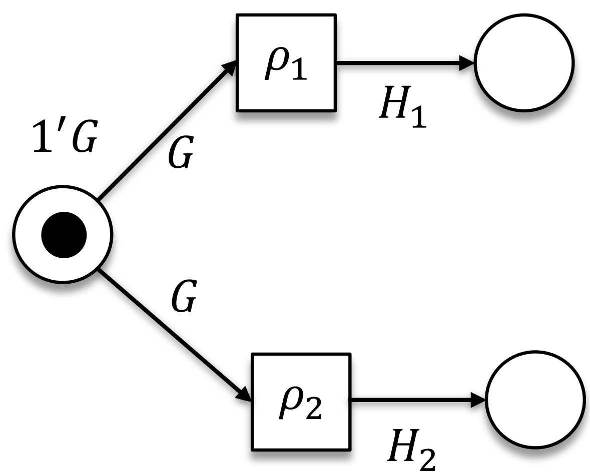
DPO RULE APPLICATION AS GPN TRANSITION



$$
\begin{tikzcd}
\mathbf{L} \arrow[d, "m"'] \arrow[dr, phantom, "(PO)"] & \mathbf{K} \arrow[l, "\mathbf{l}"'] \arrow[r, "\mathbf{r}"] \arrow[d, "k"] \arrow[dr, phantom, "(PO)"] & \mathbf{R} \arrow[d, "n"] \\
G & D \arrow[l, "f"] \arrow[r, "g"'] & H
\end{tikzcd}
$$

- GPN colour set $\Sigma_{TG} = \{Obj(\mathbf{Graph}_{TG}), Mor(\mathbf{Graph}_{TG})\}$
- Variable G is bound to graph $G \in Obj(\mathbf{Graph}_{TG})$ carried by input token $1'G$
- Transition guard ρ corresponds to the DPO diagram for the application of rule $\rho : (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ on match m in G
- Variable H is bound to output graph $H \in Obj(\mathbf{Graph}_{TG})$ of the rule application and assigned to output token $1'H$

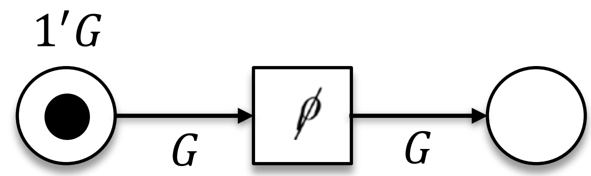
NON-DETERMINISTIC CHOICE



NEGATIVE RULE APPLICATIONS & DETERMINISTIC CHOICE

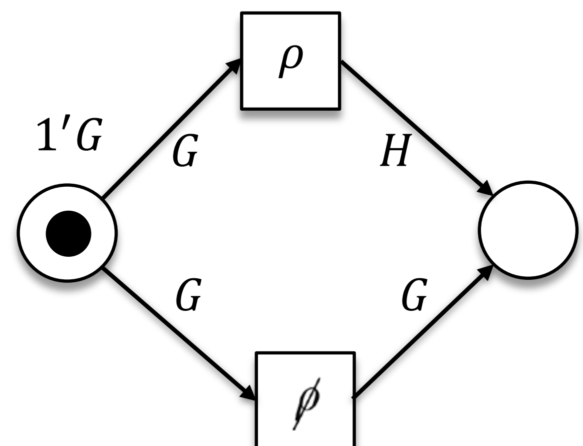
- Non-applicability of rule

$$\rho : (L \xleftarrow{l} K \xrightarrow{r} R)$$

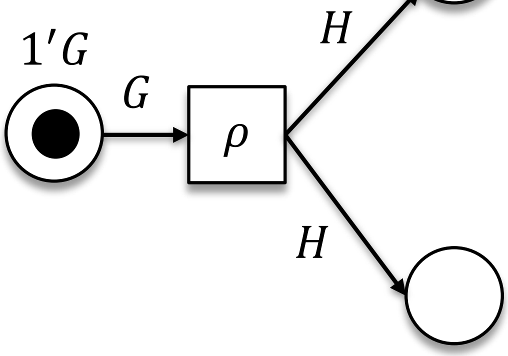


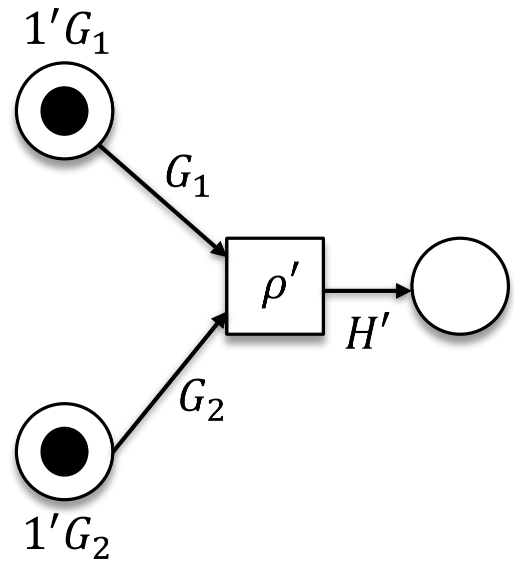
$$\mathbf{L} \not\longrightarrow \mathrm{G}$$

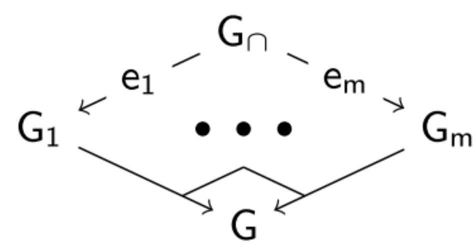
- If-then-else fragment



CONCURRENCY

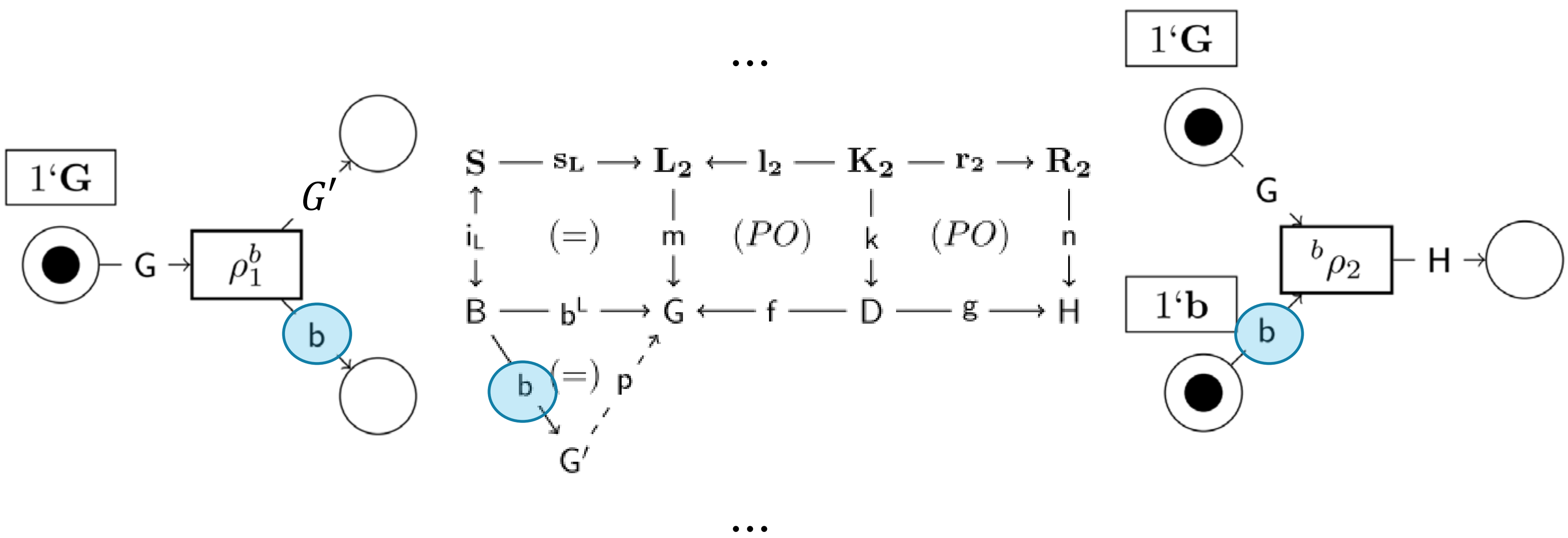






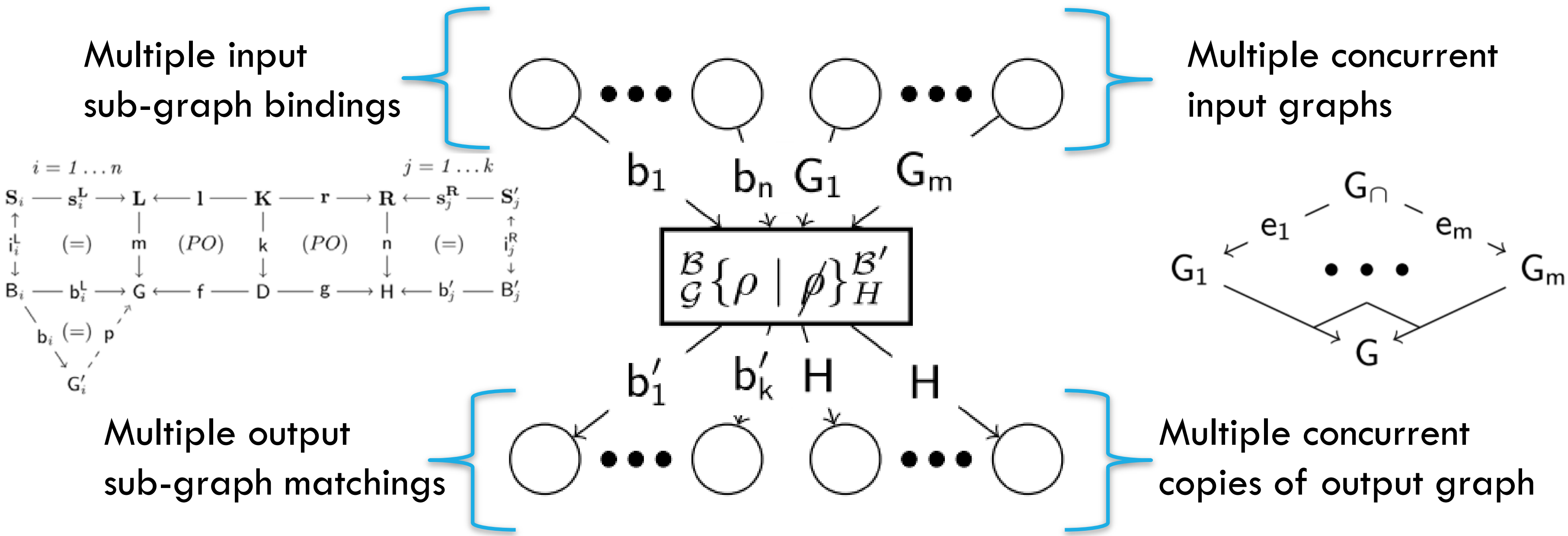
- Rule application produces **multiple concurrent copies of the output** graph
- Rule application requires **multiple concurrent input graphs**

SUB-GRAPH BINDING AND MATCHING



- Output token $b \in Mor(\mathbf{Graph}_{TG})$ denote sub-graph **bound** in output graph H
- Input token $b \in Mor(\mathbf{Graph}_{TG})$ denotes sub-graph **to be matched** in input graph G

GENERIC GPN TRANSITION TEMPLATE



GPN SEMANTICS: TWO PERSPECTIVES

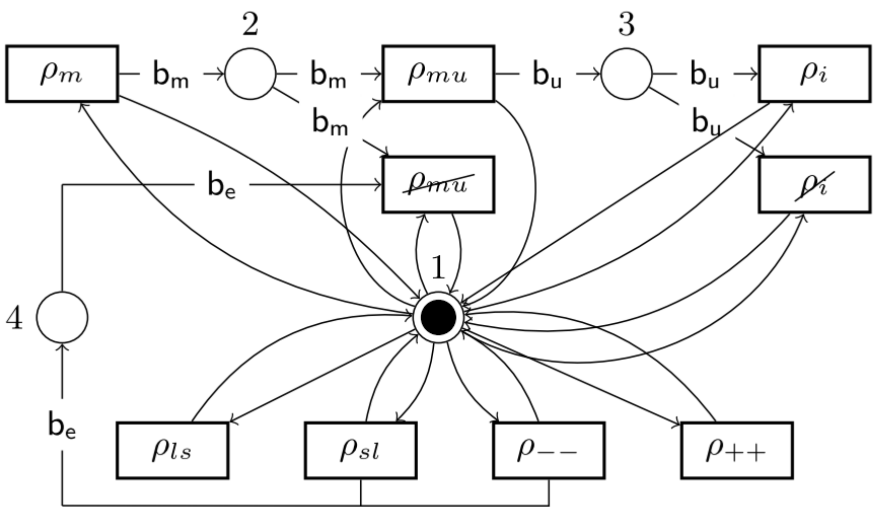
GPN language

- A GPN marking M is **completed** if all tokens are from $Obj(\mathbf{Graph}_{TG})$
- The **language** of a GPN is the set of all completed markings reachable from initial completed marking M_0

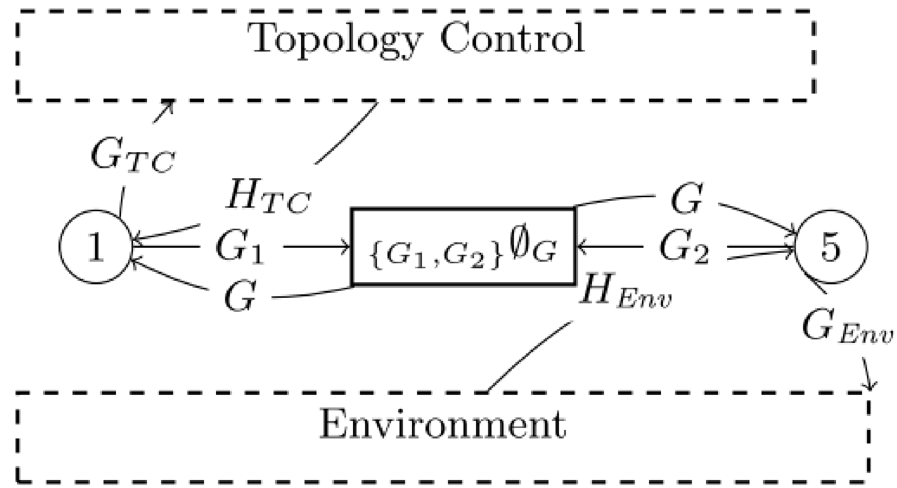
GPN processes

- Let $\mathcal{P} = \{pred \ | pred : Obj(\mathbf{Graph}_{TG}) \to Bool\}$ be a set of **graph predicates**
- The **processes** of a GPN are defined by the LTS $(S, s_0, \to, \mathcal{P})$ with set of states $S = \mathbb{N}^{\Sigma_{TG}}$, initial state $s_0 = M_0$, step relation $M \xrightarrow{X} M'$, and a set of state properties $\mathcal{P}$

WSN CASE STUDY REVISITED



- Centralized model



- Concurrent model

- „TC must preserve connectedness of input topology graphs“ → *Invariant property*
- „TC should eventually inactivate redundant long edges“ → *Fairness property*
- „TC must not deadlock …“ → *Liveness property*

SUMMARY & FUTURE WORK

- GPN provide a visual, very expressive and formally founded modeling language for controlled graph-rewriting processes
- Novel features: explicit notion of concurrency and sub-graph binding/matching

Future Work

- Tool support based on CPN tools
- Further (application-specific) merge-operators
- Notions of expressiveness for GPN languages
- Notions of parallel independence for GPN processes
- Notions of equivalence for GPN processes

RELATED WORK

- First proposal of programmed (a.k.a. controlled) graph grammers [Bunke 1978]
- Formal notion of concurrent graph processes [Corradini et al. 1996, Baldan et al. 1999]
- Denotational characterization of input/output semantics of graph-rewriting processes [Schürr 1996]
- Composable (graph) transformation units [Kreowski et al. 2008]
- Operational semantics of GP language [Plump & Steinert 2009]
- Tool support: PROGRES, GReAT, Fujaba, eMofIon, Henshin…
- Combination of graph rewriting and Petri net theory [Hoffmann & Mossakowski 2002, Wimmer et al. 2009, Guerra & de Lara 2014]

REFERENCES

1. Baldan, P., Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Löwe, M.: Concurrent Semantics of Algebraic Graph Transformation. In: Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 3. pp. 107–187. World Scientific (1999)
2. Bunke, H.: Programmed Graph Grammars. In: Workshop on Graph Grammars and Their Application to Computer Science. pp. 155–166. Springer (1978)
3. Corradini, A., Ehrig, H., Löwe, M., Montanari, U., Rossi, F.: Abstract Graph Derivations in the Double Pushout Approach. In: Dagstuhl Workshop on Graph Transformations in Computer Science, 1993. pp. 86–103. Springer (1994)
4. Corradini, A., Montanari, U., Rossi, F.: Graph Processes. Fundam. Inf. **26**(3-4), 241–265 (Jun 1996)
5. Corradini, A., Montanari, U., Rossi, F.: Graph processes. Fundamenta Informaticae **26**(3/4), 241–265 (1996)
6. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Springer (2006)
7. Esparza, J.: Decidability and complexity of petri net problems - an introduction. In: Lectures on Petri Nets I. pp. 374–428. Springer (1998)
8. Fischer, T., Niere, J., Torunski, L., Zündorf, A.: Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java. In: TAGT 98. LNCS, vol. 1764, pp. 157–167. Springer (2000)
9. Ghamarian, A.H., de Mol, M.J., Rensink, A., Zambon, E., Zimakova, M.V.: Modelling and Analysis using GROOVE. International Journal on Software Tools for Technology Transfer **14**, 15–40 (2012)
10. Guerra, E., de Lara, J.: Colouring: execution, debug and analysis of qvt-relations transformations through coloured petri nets. Software & Systems Modeling **13**(4), 1447–1472 (2014)
11. Hoffmann, K., Mossakowski, T.: Algebraic higher-order nets: Graphs and petri nets as tokens. In: WADT (2002)

REFERENCES

13. Kluge, R., Stein, M., Varró, G., Schürr, A., Hollick, M., Mühlhäuser, M.: A systematic approach to constructing families of incremental topology control algorithms using graph transformation. Software & Systems Modeling (2017)
14. Kreowski, H.J., Kuske, S., Rozenberg, G.: Graph Transformation Units - An Overview. In: Concurrency, Graphs and Models. pp. 57–75. Springer (2008). https://doi.org/10.1007/978-3-540-68679-8_5
15. Leblebici, E., Anjorin, A., Schürr, A.: Developing eMoflon with eMoflon. In: ICMT 2014. pp. 138–145. Springer International Publishing
16. Li, M., Li, Z., Vasilakos, A.V.: A survey on topology control in wireless sensor networks: Taxonomy, comparative study, and open issues. Proceedings of the IEEE **101**(12), 2538–2557 (2013)
17. Plump, D., Steinert, S.: The Semantics of Graph Programs. In: RULE 2009 (2009)
18. Schürr, A.: Logic-Based Programmed Structure Rewriting Systems. Fundam. Inf. **26**(3,4), 363–385 (1996)
19. Schürr, A., Winter, A.J., Zündorf, A.: The PROGRES-Approach: Language and Environment. In: Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2, pp. 487–550. World Scientific (1999). https://doi.org/10.1142/9789812815149_0013
20. Strüber, D., Born, K., Gill, K.D., Groner, R., Kehrer, T., Ohrndorf, M., Tichy, M.: Henshin: A usability-focused framework for emf model transformation development. In: ICGT 2017. Springerg (2017)
21. Wimmer, M., Kappel, G., Kusel, A., Retschitzegger, W., Schönböck, J., Schwinger, W.: Right or wrong?–verification of model transformations using colored petri nets. In: DSM 2009

GRAPH-REWRITING INSCRIPTIONS

- Colour set of GPN $\Sigma_{TG} = \{Obj(\mathbf{Graph}_{TG}), Mor(\mathbf{Graph}_{TG})\}$
- Graph-rewriting inscriptions $I \in INSCR_{TG,V}$
  - Finite category of bound variables $\mathbf{BV_I}$
  - Finite category of free variables $\mathbf{FV_I}$ s.t. $\mathbf{BV_I} \subseteq \mathbf{FV_I}$
  - Binding functor $B_I : \mathbf{BV_I} \rightarrow \mathbf{Graph}_{TG}$ s.t.
    $\forall v \in \mathbf{BV_I} : Type(v) = Type(B_I(v))$
  - A set of categorial properties Φ_I for the images of $\mathbf{FV_I}$
- An inscription binding is a functor $B : \mathbf{FV_I} \rightarrow \mathbf{Graph}_{TG}$ s.t.
  - $B\,|_{\mathbf{BV_I}} = B_I$ and $\forall v \in \mathbf{FV_I} : Type(v) = Type(B(v))$
  - $I\langle B\rangle$ is satisfied if $B(\mathbf{FV_I})$ satisfies Φ_I