A Temporal Graph Logic for Abstractions of Graph Rewriting Systems

Andrea Corradini
Dipartimento di Informatica, Pisa, Italy

Joint work with
Paolo Baldan
Università Ca’ Foscari di Venezia, Italy
Barbara König
Universität Duisburg-Essen, Germany
Alberto Lluch Lafuente
Dipartimento di Informatica, Pisa, Italy
Motivations

Graph Rewriting Systems (GRSs) are a very expressive formalism for modeling the evolution of concurrent and distributed systems:
- specification (visual) language;
- semantics/implementations for other formalisms (e.g., Ambient Calculus, Pi-Calculus, CommUnity).

Expressiveness is not enough: there is a strong need of analysis and verification techniques for GRSs.

The relationship between GRS and Petri nets was exploited to develop a rich concurrency theory for GRSs.

May we use it also to exploit verification techniques and tools developed for Petri nets?
In previous works...

- We introduced a logic to express properties relevant to Graph Rewriting Systems, $\mu L_2$.

- We designed a technique for the formal verification of such properties over finite representations of the GRS (which in general is infinite-state).

- We showed how to reduce the verification of such properties to the verification of corresponding properties over Petri nets, for which existing techniques and tools can be used.

- A tool providing a partial implementation of such verification technique, AUGUR, has been developed in Stuttgart by the group of Barbara König.
Limitations of the previous approach

- The logic $\mu L_2$ is a **propositional** $\mu$-calculus, where propositional variables are formulæ of a **second-order monadic logic** over graphs.

- Logic $\mu L_2$ is quite expressive, but since it is **propositional** in the temporal dimension, it does not allow to track the identity of items during rewriting. For instance, properties like

  a given edge is never deleted

cannot be expressed.
The new contribution

We introduce a more expressive temporal graph logic, called $\mu G^2$, where quantifications and temporal modalities can be interleaved.

Next we generalize our verification approach ⇒ not trivial...

1. For interpreting $\mu G^2$ formulæ, we introduce (unfolded) graph transition systems (gTS) and their morphism

2. We identify fragments of $\mu G^2$ which are preserved/ reflected by gTS-morphisms

3. We show how to get a gTS from a GRS and from a “Petri graph” approximating it, and how to encode part of $\mu G^2$ into a Petri net logic.

4. Summarizing, we show how to reduce the verification of a formula over a GRS to a formula over a suitable net.
Our Graph Rewriting Systems

Graphs are directed, edge-labeled graphs

\[
G_0 = \begin{bmatrix}
\text{\small \text{e}_1: \text{b}} \\
\text{\small \text{u}_0} \\
\text{\small \text{e}_2: \text{a}} \\
\text{\small \text{u}_1} \\
\text{\small \text{e}_3: \text{a}} \\
\text{\small \text{u}_2} \\
\text{\small \text{e}_4: \text{a}} \\
\text{\small \text{u}_3}
\end{bmatrix}
\]

Rules are triples \( r = \langle G_L, G_R, \alpha \rangle \), with \( \alpha : V_L \to V_R \) injective

Thus rules are DPO rules where nodes are neither deleted nor merged, and with discrete interface.
The Monadic Second Order logic $\mathcal{L}_2$

Graph formulae with quantification over first- and second-order variables ranging over (sets of) edges [Courcelle]

\[
F ::= x = y \mid s(x) = s(y) \mid s(x) = t(y) \mid t(x) = t(y) \mid \\
lab(x) = \ell \mid x \in X \mid F \lor F \mid \neg F \\
\exists x.F \mid \exists X.F
\]

Example of properties:

- \( NP(x, y) \): “No path connecting the edges \( x \) and \( y \)”
  \[
  NP(x, y) \equiv \neg \forall X. (\forall z.(t(x) = s(z) \lor \exists w.(w \in X \land t(w) = s(z))) \Rightarrow z \in X) \Rightarrow y \in X
  \]

- \( NC_\ell \) “No cycle including two distinct edges labelled \( \ell \)”
  \[
  NC_\ell \equiv \forall x.\forall y.(lab(x) = \ell \land lab(y) = \ell \land \neg(x = y) \Rightarrow NP(x, y) \lor NP(y, x))
  \]

Remark: Not expressible in first order logic!
Temporal extensions of $\mathcal{L}2$

- $\mu \mathcal{L}2$: an $\mathcal{L}2$-propositional $\mu$-calculus

\[
G ::= F \mid X \mid \Diamond G \mid \neg G \mid G_1 \lor G_2 \mid \mu X.f
\]

with atomic predicates $F$ taken from $\mathcal{L}2$.

- $\mu X. (F \lor \Diamond X)$ “eventually $F$” (liveness)
- $\nu X. (F \land \Box X)$ “always $F$” (safety) [$\nu X. (NC_a \land \Box X)$]

- $\mu \mathcal{G}2$: an MSO $\mu$-calculus over graphs

\[
F ::= \eta(x) = \eta'(y) \mid x = y \mid l(x) = a \mid \neg F \mid F \lor F \mid \\
\exists x.F \mid \exists X.F \mid x \in X \mid Z \mid \mu Z.F \mid \Diamond F
\]

where $\eta, \eta' \in \{s, t\}, \ x, y \in V_x, \ X \in V_X, \ a \in \Lambda \ \text{and} \ \ Z \in V_Z$. 

A Temporal Graph Logic for Abstractions of Graph Rewriting Systems – p.8/22
Some properties and their intuitive meaning:

- **Del**(b): “No b-labeled loop is preserved by a transition”
  \[
  \text{Del}(b) \equiv \neg \exists x. (s(x) = t(x) \land l(x) = b \land \diamond \exists y. x = y)
  \]

- **Moves**(b): “No next state has a b-loop on the same node as the current state”
  \[
  \text{Moves}(b) \equiv \neg \exists x. (s(x) = t(x) \land l(x) = b \land \\
  \diamond (\exists y. (s(y) = t(y) = s(x) \land l(y) = b)))
  \]

  **Remark:** Not expressible in \( \mu \mathcal{L}2 \! \)!

- \( \nu Z.(\text{Del}(b) \land \Box Z) \): “\text{Del}(b) holds in all reachable states”

These properties hold true for our toy example [show!]

But how can this be formalized? The same variable has to be interpreted on different graphs...

Solution: unfolded Graph Transition Systems
Graph Transition Systems

Logic $\mu L_2$ can be interpreted on transition systems where states are graphs. $\mu G_2$ needs more, to track edge identities. A graph transition system (gTS) $\mathcal{M}$ is a diagram in $\text{PGraph}$, i.e., a pair $\langle M, \langle g^S, g^T \rangle \rangle$, where

- $M$ is a transition system
- $g^S(s)$ is a graph for each state $s \in S_M$
- $g^T(t) : g^S(s) \to g^S(s')$ is an injective partial graph morphism for each transition $t : s \to s' \in T_M$.

$gTS$-morphisms are natural transformations.

Given a GRS $\mathcal{R} = \langle G_0, R \rangle$, a gTS representing its state space, denoted by $gTS(\mathcal{R})$, can be obtained easily.
A gTS of our toy example

A gTS $\mathcal{M} = \langle M, g \rangle$ is unfolded if

- $M$ is a tree,
- for each $t \in T_M$ the morphism $g^T(t)$ is a partial inclusion
- item names are not re-used.

Any gTS has an unfolded one which is behaviorally equivalent to it, called its unfolding.
Interpretation of $\mu G_2$ over a gTS

Given an unfolded gTS $\mathcal{M} = \langle M, g \rangle$, $[\cdot]_\sigma^{\mathcal{M}} : \mu G_2 \to 2^{S_M}$, is defined inductively as:

\[
\begin{align*}
[\eta(x) = \eta'(y)]_\sigma &= [\eta_{\mathcal{M}}(\sigma_x(x)) = \eta'_{\mathcal{M}}(\sigma_x(y))] \\
[l(x) = a]_\sigma &= [\operatorname{lab}_{\mathcal{M}}(\sigma_x(x)) = a] \\
[\neg F]_\sigma &= S_M \setminus [F]_\sigma \\
[Z]_\sigma &= \sigma_Z(Z) \\
[\diamond F]_\sigma &= \{ s \in S_M \mid s \xrightarrow{t} s' \land s' \in [F]_\sigma \} \\
[\exists x. F]_\sigma &= \{ s \in S_M \mid \exists e \in E_{g(s)} \cdot s \in [F]_\sigma[e/x] \} \\
[\exists X. F]_\sigma &= \{ s \in S_M \mid \exists E \subseteq E_{g(s)} \cdot s \in [F]_\sigma[E/X] \}
\end{align*}
\]

where $[\cdot]$ maps true and false to $S_M$ and $\emptyset$, respectively.

A GRS $\mathcal{R} = \langle G_0, R \rangle$ satisfies a closed formula $F$, written $\mathcal{R} \models F$, if the unfolding of $gTS(\mathcal{R})$ satisfies $F$. 

A Temporal Graph Logic for Abstractions of Graph Rewriting Systems – p.12/22
How can a $\mu G^2$ formula be verified?

- In previous works, we showed how to generate the covering of a GRS $\mathcal{R}$, a finite Petri graph $\mathcal{C}(\mathcal{R})$ over-approximating it.

- We show how to get a gTS from a Petri graph, such that there is a morphism $gTS(\mathcal{R}) \rightarrow gTS(\mathcal{C}(\mathcal{R}))$.

- If a formula $F$ is reflected by gTS-morphisms, we have
  
  \[ gTS(\mathcal{C}(\mathcal{R})) \models F \quad \implies \quad gTS(\mathcal{R}) \models F \]

- We provide an encoding $\mathcal{P}(\cdot)$ of the first-order fragment of $\mu L^2$ into a Petri net logic, such that
  
  \[ gTS(\mathcal{C}(\mathcal{R})) \models F \quad \iff \quad PN(\mathcal{C}(\mathcal{R})) \models \mathcal{P}(F) \]

In summary, for a suitable fragment of $\mu G^2$,

\[ PN(\mathcal{C}(\mathcal{R})) \models \mathcal{P}(F) \quad \implies \quad gTS(\mathcal{R}) \models F \]

thus reducing the verification to a finite Petri net.
Petri graphs

**Petri graph** for a given GRS: a graph with a P/T Petri net over it, where

- places are edges
- transitions are labelled by rules of the GRS
A marking $m$ of a Petri graph naturally correspond to a graph $\text{graph}(m)$ obtained by “duplicating” or “removing” edges according to their weight in the marking.

Examples
gTS generated by a Petri graph

The gTS generated by a Petri graph $P = \langle G, N, p \rangle$, denoted by $gTS(P)$, is $\langle M, g \rangle$ where

- $S_M$ is the set of markings reachable in $P$;
- $T_M = \{ \langle m, t, X, m' \rangle : m \rightarrow m' \mid m[t]m' \text{ and } X \in \text{SignificantPermutations}(m, t) \}$;
- $s_0^M = m_0$;
- $g^S(m) = \text{graph}_P(m)$;
- $g^T(\langle m, t, X, m' \rangle) = f_{m,t,X}$, where $f_{m,t,X} : \text{graph}_P(m) \rightarrow \text{graph}_P(m')$ is any injective partial graph morphism which is the identity over nodes, and whose domain over edges is exactly $X$. 
The following type system identifies classes of $\mu G^2$ graph formulæ preserved / reflected by gTS-morphisms such that the graph morphism components are edge-bijective.

$$\eta(x) = \eta'(y): \rightarrow x = y, \ l(x) = a, \ x \in X, \ Z: \leftrightarrow$$

$$\frac{F: d}{\neg F: d^{-1}} \quad \frac{F_1, F_2: d}{F_1 \lor F_2: d} \quad \frac{F: d}{\exists x. F: d} \quad \frac{F: d}{\exists X. F: d}$$

$$\frac{F: \rightarrow}{\Diamond F: \rightarrow} \quad \frac{F: \leftarrow}{\Box F: \leftarrow} \quad \frac{F: d}{\mu Z. F: d}$$
Coverings preserves and approximates

Let $\mathcal{M}$ and $\mathcal{M}'$ be two unfolded gTSs such that there is a morphism $\langle h_M, h_g \rangle : \mathcal{M} \to \mathcal{M}'$ having all $h_g$ components edge-bijective. Then for each closed formula $F \in \mu \mathcal{G}2$ we have

- if $F :\leftarrow$ then $\mathcal{M}' \models F$ implies $\mathcal{M} \models F$
- if $F :\rightarrow$ then $\mathcal{M} \models F$ implies $\mathcal{M}' \models F$.

Let $\mathcal{R}$ be a GRS and let $F \in \mu \mathcal{G}2$ be a closed formula. Then

- if $F :\leftarrow$ then $\mathcal{C}(\mathcal{R}) \models F$ implies $\mathcal{R} \models F$
- if $F :\rightarrow$ then $\mathcal{R} \models F$ implies $\mathcal{C}(\mathcal{R}) \models F$. 
Exploiting the underlying Petri net

**Goal:** Reuse existing verification tools for Petri nets

**Proposed solution:** Reduce the verification of $\mu G^2$ formulæ over the covering of a GRS to the verification of suitable *multiset formulæ*, expressing *marking properties* over the underlying Petri net. [This is possible because the Petri graph (and thus the net) is fixed and finite.]

The syntax of the Petri net logic $\mathcal{P}$ is given by the following grammar, where $p \in N_P$, $t \in N_T$, $c \in \mathbb{N}$ and $Z \in V_Z$:

$$\phi ::= \#p \leq c \mid \phi \lor \phi \mid \neg \phi \mid Z \mid \mu Z.\phi \mid \langle t \rangle \phi.$$  

A sound and complete encoding into $\mathcal{P}$ has been provided for the first-order fragment of $\mu G^2$. 

---

A Temporal Graph Logic for Abstractions of Graph Rewriting Systems – p.19/22
Let $P = \langle G, N, p \rangle$ be a Petri graph, $F$ be a fixed-point-free $\mu G1$ formula, $\rho : \text{free}(F') \rightarrow E_G$ and $Q \subseteq 2^{\text{free}(F)}$ be an equivalence relation, $R \subseteq Q$ and $xQy$ implies $\rho(x) = \rho(y)$ for all $x, y \in \text{free}(F')$. The encoding $[\cdot] : \mu G1 \rightarrow \mathcal{P}$ is defined as follows:

$[\neg F, \rho, Q, R] = \neg [F, \rho, Q, R]$  
$[F_1 \lor F_2, \rho, Q, R] = [F_1, \rho, Q, R] \lor [F_2, \rho, Q, R]$  
$[x = y, \rho, Q, R] = \text{true if } xQy, \text{false otherwise}$  
$[l(x) = a, \rho, Q, R] = \text{true if } \text{lab}_G(\rho(x)) = a, \text{false otherwise}$  
$[s(x) = s(y), \rho, Q, R] = \text{true if } s_G(\rho(x)) = s_G(\rho(y)), \text{false otherwise}$  

analogously for $t(x) = t(y)$ and $s(x) = t(y)$

$[\exists x. F, \rho, Q, R] = \bigvee_{k \in Q \setminus R} [F, \rho \cup \{\rho(k)/x\}, Q \setminus \{k\} \cup \{k \cup \{x\}\}, R] \lor$  
$\bigvee_{e \in E_G} ([F, \rho \cup \{e/x\}, Q \cup \{\{x\}\}, R] \land (#e \geq n_{Q\setminus R}, \rho(e) + 1))$  

$[\diamond F, \rho, Q, R] = \bigvee_{t \in T_N} \bigvee_{R' \in S} (\bigwedge_{e \in \cdot t} (#e \geq n_{Q \setminus (R \cup R')}, \rho(e) + \cdot t(e)) \land$  
$\langle t\rangle [F, \rho, Q, R \cup R'])$  

$[Z, \rho, Q, R] = Z$

where $S$ abbreviates $\{Q' \in 2^{(Q \setminus R) \cap \text{rep}^{-1}(\rho^{-1}(\cdot t))} \mid \bigwedge_{e \in \cdot t} n_{Q'}, \rho(e) \leq \cdot t(e)\}$.
On-going and future Work

- Related work by Arend Rensink
- Identification of decidable fragments of the logic
- Extension of the encoding into Petri net logic to the second-order fragment
- Extension of the approach to hypergraphs (for the logical part), and to more general rules (non-discrete interfaces).
- Implement the approach by extending the existing tool AUGUR