On Quantified Modal Logics

Fabio Gadducci (UNIPI)

Joint work with:
Alberto Lluch Lafuente (DTU)
Andrea Vandin (DTU)

Software Validation and Verification
Structure of the presentation

(1) Motivation and case study
(2) Kripke models and their limits
(3) Counterpart semantics for Quantified Modal Logics
(4) State-space reductions and approximated model checking
Adaptive Systems

Roberto Bruni, Andrea Corradini, Fabio Gadducci, Alberto Lluch Lafuente, Andrea Vandin, A Conceptual Framework for Adaptation, International Conference on Fundamental Approaches to Software Engineering (FASE’12), LNCS.

Roberto Bruni, Andrea Corradini, Fabio Gadducci, Alberto Lluch-Lafuente, Andrea Vandin, Modeling and analyzing adaptive self-assembling strategies with Maude, International Workshop on Rewriting Logic and its Applications (WRLA’12), LNCS.

Specification and Analysis of Systems with Dynamic Structure

AFLine, EFLine

$$AG \forall x, y \left( \begin{array}{c} \circ \circ \rightarrow \circ \circ \end{array} \right) \Rightarrow AG \begin{array}{c} \circ \circ \end{array}$$

$$\forall x, y AG \left( \begin{array}{c} \circ \circ \rightarrow \circ \circ \end{array} \right)$$

Fabio Gadducci, Alberto Lluch Lafuente, Andrea Vandin, Counterpart Semantics for a Second-Order mu-Calculus, International Conference on Graph Transformation (ICGT’10), LNCS;

Fabio Gadducci, Alberto Lluch Lafuente, Andrea Vandin, Counterpart Semantics for a Second-Order mu-Calculus, Journal Fundamenta Informaticae 118(1-2) (2012);

Fabio Gadducci, Alberto Lluch Lafuente, Andrea Vandin, Exploiting Over- and Under-Approximations for Infinite-State Counterpart Models, International Conference on Graph Transformation (ICGT’12), LNCS;

Alberto Lluch Lafuente, José Meseguer, Andrea Vandin, State Space c-Reductions of Concurrent Systems in Rewriting Logic, International Conference on Formal Engineering Methods (ICFEM’12), LNCS.
Structure of the presentation

(1) Motivation and case study
(2) Kripke models and their limits
(3) Counterpart semantics for Quantified Modal Logics
(4) State-space reductions and approximated model checking
Motivation

Reasoning on software systems focuses on global behaviour

- global correctness, liveness and safety.

For structured systems we can focus on components evolution

- life of components (alive, dead, old, new)
- relations between components (there exist two disconnected components that will become connected?)
Circle elimination games

Processes connected via left/right ports in a ring topology
Circle elimination games

Processes connected via left/right ports in a ring topology
The game evolves through elimination rounds
Circle elimination games

Processes connected via left/right ports in a ring topology
The game evolves through elimination rounds
The game ends with a leader, having same left/right port
Considering graph rewriting as specification mechanism

- We can specify the evolution of the topology with a simple rewriting rule;
- We compute the state space applying the rule starting from an initial state.

Circle elimination games
Graph Transition System
Graph Transition System
Graph Transition System
Graph Transition System
Graph Transition System

\[ n_0 \rightarrow e_0 \rightarrow n_1 \rightarrow e_1 \rightarrow n_2 \rightarrow e_2 \rightarrow n_0 \]

\[ e_2 \rightarrow n_1 \]
Morphisms and local names
Morphisms and local names
Morphisms and local names
Morphisms and local names
Interesting properties

1. In which states do we have a leader?
Interesting properties

In which states do we have a leader, and which processes?
Interesting properties

1. In which states do we have a leader?
2. Can two distinct leaders be elected in the same execution?
Interesting properties

1. In which states do we have a leader?
2. Can two distinct leaders be elected in the same execution?
3. Will a leader be elected in all possible executions?
Interesting properties

1. In which states do we have a leader?
2. Can two distinct leaders be elected in the same execution?
3. Will a leader be elected in all possible executions?
4. Is there a process that eventually becomes the leader?

Fixed an i:
Interesting properties

1. In which states do we have a leader?
2. Can two distinct leaders be elected in the same execution?
3. Will a leader be elected in all possible executions?
4. Is there a process that eventually becomes the leader?
5. Which processes eventually become the leader?
Interesting properties

1. In which states do we have a leader?
2. Can two distinct leaders be elected in the same execution?
3. Will a leader be elected in all possible executions?
4. Is there a process that eventually becomes the leader?
5. Which processes eventually become the leader?
6. Are process connections correctly updated after each round?
In order to express such properties we need languages able to reason about state components.

\[ \exists_{\text{Edge}} x. \psi \]
In order to express such properties we need languages able to reason about state components.

1. In which states do we have a leader?

$$\exists_{Edge} \exists_e x \cdot s(x) = t(x)$$
In order to express such properties we need languages able to reason about state components.

1. In which states do we have a leader?

\[ \exists_{Edge} \exists_{x} x.\text{leader}(x) \]
In order to express such properties we need languages able to reason about state components.

2. Can two distinct leaders be elected in the same execution?

$$\exists_{Ed_i \in E} \neg[\text{leader}(x) \land \text{leader}(x_2) \rightarrow (x = x_2)]$$
In order to express such properties we need languages able to reason about state components and system evolution.

\[ \exists_{Edge} x. \psi \]

\[ \Diamond \psi \]

\[ AG \psi \]
In order to express such properties we need languages able to reason about state components and system evolution.

3. Will a leader be elected in all possible executions?

\[ \exists_{Edge} \forall x. leader(x) \]
In order to express such properties we need languages able to reason about

- state components
- system evolution
- component evolution

$$\exists_{Edge} x. \psi$$

$$\Diamond \psi$$

$$AG\psi$$

$$\exists x. \Diamond \psi$$

$$\forall x. AG(\psi_1 \rightarrow AF\psi_2)$$
In order to express such properties we need languages able to reason about

state components  system evolution  component evolution

4. Is there a process that eventually becomes the leader?

\[
\exists_{Edge} \exists_x [EF(\text{leader}(x))]
\]
In order to express such properties we need languages able to reason about state components, system evolution, and component evolution.

5. Which processes eventually become the leader?

\[ \exists_{Edges} \text{present}(x) \land EF(\text{leader}(x)) \]
In order to express such properties we need languages able to reason about state components, system evolution, and component evolution.

6. Are process connections always properly updated?

\[ \exists_{\text{Edges}} \text{present}(x_E) \land (x_N = s(x_E)) \land (y_N = t(x_E)) \land \Diamond [\neg \text{present}(x_E) \land (x_N \neq y_N)] \]
In the context of software analysis, such logics have been proposed:

1) to reason about the life of components: has a component been allocated or deallocated, has it been created in the current state?

2) Another example are safety and liveness properties like asking whether there is a pair of components that is disconnected in the current state but become connected in some future state.
• Description Logics
• Spatial and spatio-temporal Logics
• Predicative extensions of CTL and LTL
• Graph Transformation Logics
• ...
A temporal graph logic for verification of graph transformation systems
Baldan, P., Corradini, A., König, B., Lluch Lafuente, A. [WADT'06]

- Mixes $\mu$-calculus with MSO Logic for graphs
- Tool support for propositional fragment (AUGUR)
The syntax

Fixed a signature $\Sigma$, the formulae are given by:

$$\psi ::= \; tt \; | \; \neg \psi \; | \; \psi \lor \psi \; | \; \epsilon \in_{\tau} \chi$$
$$\; | \; \exists_{\tau} x. \psi \; | \; \exists_{\tau} \chi. \psi \; | \; \Diamond \psi \; | \; Z \; | \; \mu Z. \psi$$

Equivalence operator $\epsilon_1 =_{\tau} \epsilon_2$ definable as $\forall_{\tau} x. (\epsilon_1 \in_{\tau} \chi \leftrightarrow \epsilon_2 \in_{\tau} \chi)$

Other operators can be derived: $\Box$, $AG$, $AF$, $\nu$
The syntax

Fixed a signature $\Sigma$, the formulae are given by:

$$
\psi ::= \text{tt} \mid \neg \psi \mid \psi \lor \psi \mid \epsilon \in_{\tau} \chi \\
\mid \exists_{\tau} x.\psi \mid \exists_{\tau} \chi.\psi \mid \Diamond \psi \mid Z \mid \mu Z.\psi
$$

Equivalence operator $\epsilon_1 =_{\tau} \epsilon_2$ definable as $\forall_{\tau} x. (\epsilon_1 \in_{\tau} \chi \leftrightarrow \epsilon_2 \in_{\tau} \chi)$

Other operators can be derived: $\Box, AG, AF, \nu$

Some derived predicates used in this presentation:

$\text{leader}(x) \equiv s(x) = t(x)$  \hspace{1cm} $\text{present}(x) \equiv \exists y. x = y$
Structure of the presentation

(1) Motivation and case study
(2) Kripke models and their limits
(3) Counterpart semantics for Quantified Modal Logics
(4) State-space reductions and approximated model checking
UNIQUE DOMAIN
UNIQUE DOMAIN

DELETION

FRESH CREATION
In which elements of the domain should we map $e_1$ and $e_2$?
In which elements of the domain should we map $e_1$ and $e_2$?
NAME REUSE

$\times$ Is $e_2$ of $w_3$ the same $e_2$ of $w_0$ even if previously deleted?
We can restrict the admitted models:

- NO merging
- NO rename
- NO reuse
- NO cycles in the evolution relation

... but we ignore interesting cases.

Or we can reformulate the models

- e.g. state space unraveling/unfolding

... but we hamper the intuitive meaning of the logic.
A temporal graph logic for verification of graph transformation systems
Baldan, P., Corradini, A., König, B., Lluch Lafuente, A. [WADT'06]

✔ Mixes $\mu$-calculus with MSO Logic for graphs
✔ Tool support for propositional fragment (AUGUR)

✗ Semantics given for “unravelled” GTrSs:
  • a tree representing the unfolded state space;
  • partial morphisms between worlds are partial inclusions;
    • no renaming of graph items;
    • no merging of graph items.
  • no reuse of item names.
**Model checking Birth and Death**  
Distefano, D., Rensink, A., Katoen, J.P. [IFIP'02]

**A/TL (Allocational Temporal Logic)**
- Extends propositional LTL with
  - First order variables
  - Partial assignments (death)
  - First Order quantification over living entities
- Interpreted over an extension of History Dependent Automata
- Properties on dynamic de/re/allocation of entities

✗ Injective renaming (no merging)

False predicates for deallocated components
(x=x may be false)
QCTL (Quantified Computation Tree Logic)

- Extends propositional CTL with
  - First and Second order variables
  - Partial assignments (death)
  - First and Second Order quantification over living entities

- Interpreted over Algebra Automata
  - Minimizable up to bisimilarity

- Properties on dynamic de/re/allocation of entities

✗ Injective renaming (no merging)

False predicates for deallocated components
(x=x may be false)
Structure of the presentation

1. Motivation and case study
2. Kripke models and their limits
3. Counterpart semantics for Quantified Modal Logics
4. State-space reductions and approximated model checking
Inspired by Lewis's Counterpart Theory
Counterpart model
Semantics

Formulae are evaluated \( ([\psi[\Gamma; \Delta]]^M) \) as set of pairs

\[
(w, \sigma)
\]

\[
(w_0, (x_{Node} \rightarrow n(1), x_{Edge} \rightarrow e(2))
\]

A world and a set of (partial) assignments over the first- and second-order variables occurring in the context
\[
\begin{align*}
\mathbb{[} & \rho : F^{[\Gamma; \Delta]} \to \Omega^{[\Gamma; \Delta]} \\
\mathbb{[} tt^{[\Gamma; \Delta]} \mathbb{]}_\rho &= \Omega^{[\Gamma; \Delta]} \\
\mathbb{[} \in_\tau \chi^{[\Gamma; \Delta]} \mathbb{]}_\rho &= \left\{ (w, \sigma) \in \Omega^{[\Gamma; \Delta]} \mid \sigma(\epsilon) \text{ is defined and } \sigma(\epsilon) \in \sigma(\chi) \right\} \\
\mathbb{[} \neg \psi^{[\Gamma; \Delta]} \mathbb{]}_\rho &= \Omega^{[\Gamma; \Delta]} \setminus \mathbb{[} \psi^{[\Gamma; \Delta]} \mathbb{]}_\rho \\
\mathbb{[} \psi_1 \lor \psi_2^{[\Gamma; \Delta]} \mathbb{]}_\rho &= \mathbb{[} \psi_1^{[\Gamma; \Delta]} \mathbb{]}_\rho \cup \mathbb{[} \psi_2^{[\Gamma; \Delta]} \mathbb{]}_\rho \\
\mathbb{[} \exists_\tau x. \psi^{[\Gamma; \Delta]} \mathbb{]}_\rho &= 2^{\perp_x} \left( \left\{ (w, \sigma) \in \mathbb{[} \psi^{[\Gamma, x; \Delta]} \mathbb{]}_{(2^{\perp_x})} \mid \sigma(x) \text{ is defined} \right\} \right) \\
\mathbb{[} \exists_\tau \chi. \psi^{[\Gamma; \Delta]} \mathbb{]}_\rho &= 2^{\perp_x} \left( \mathbb{[} \psi^{[\Gamma; \Delta, \chi]} \mathbb{]}_{(2^{\perp_x})} \right) \\
\mathbb{[} \diamond \psi^{[\Gamma; \Delta]} \mathbb{]}_\rho &= \left\{ (w, \sigma) \in \Omega^{[\Gamma; \Delta]} \mid \exists w \xrightarrow{cr} w'. (w', cr \circ \sigma) \in \mathbb{[} \psi^{[\Gamma; \Delta]} \mathbb{]}_\rho \right\} \\
\mathbb{[} Z^{[\Gamma; \Delta]} \mathbb{]}_\rho &= \rho(Z) \\
\mathbb{[} \mu Z. \psi^{[\Gamma; \Delta]} \mathbb{]}_\rho &= \text{lfp}(\lambda Y. \mathbb{[} \psi^{[\Gamma; \Delta]} \mathbb{]}_\rho[Y/Z])
\end{align*}
\]
$$[[\Diamond \psi[\Gamma;\Delta]]_{\rho} = \{(w, \sigma) \in \Omega^{[\Gamma;\Delta]} |$$
Examples of evaluations

In which states do we have a leader?

$$\exists_{Edge x.\text{leader}(x)}$$
Examples of evaluations

In which states do we have a leader?

$$\exists E_{\text{edge}} x. \text{leader}(x)$$

$$\{(w_4, \lambda), (w_5, \lambda), (w_6, \lambda)\}$$
Examples of evaluations

In which states do we have a leader, and which processes?

\[ \text{present}(x) \land \text{leader}(x) \]
Examples of evaluations

In which states do we have a leader, and which processes?

\( \text{present}(x) \land \text{leader}(x) \)
Examples of evaluations

Can two distinct leaders be elected in the same execution?

\[ \neg[\text{leader}(x) \land \text{leader}(x_2) \rightarrow (x = x_2)] \]
Examples of evaluations

Can two distinct leaders be elected in the same execution?

\[ \neg[\text{leader}(x) \land \text{leader}(x_2) \rightarrow (x = x_2)] \]
Examples of evaluations

3. Will a leader be elected in all possible executions?

\[ AGAF [\exists_{Edge} x. leader(x)] \]
Examples of evaluations

3. Will a leader be elected in all possible executions?

\[ \mathcal{A} \mathcal{G} \mathcal{A} \mathcal{F} \left[ \exists_{\text{Edge}} x. \text{leader}(x) \right] \]

\[ \{(w_0, \lambda), \ldots, (w_6, \lambda)\} \]
Examples of evaluations

Is there a process that eventually becomes the leader?

$$\exists_{Edge \ x} [EF(\text{leader}(x))]$$

Fixed an i:
Examples of evaluations

Is there a process that eventually becomes the leader?

\[ \exists_{Edge \, x}. [EF(\text{leader}(x))] \]

Fixed an i:

\[ \{(w_0, \lambda), \ldots, (w_6, \lambda)\} \]
Examples of evaluations

Which process eventually becomes the leader?

\[ \text{present}(x) \land EF(\text{leader}(x)) \]

Fixed an i:
Examples of evaluations

Which process eventually becomes the leader?

\[ \text{present}(x) \land EF(\text{leader}(x)) \]

Fixed an \( i \):

\[ \{ (w_0, x \mapsto e_0), (w_0, x \mapsto e_1) \ldots \} \]
An overview of our proposal

Second-Order $\mu$-calculus [ICGT'10], [Fund. Inf.'12]

- Simple but reasonably expressive syntax
- Counterpart-like semantics [Lewis'11]
- Formulae evaluated against counterpart models (similar to GTrS)
- Model Checking decidable for finite counterpart models

Prototypal model checker [GT-VMT'11]

Exploitation of counterpart model approximations [ICGT'12]

State space reduction technique [ICFEM'12]
Structure of the presentation

(1) Motivation and case study
(2) Kripke models and their limits
(3) Counterpart semantics for Quantified Modal Logics
(4) State-space reductions and approximated model checking
State-space reductions and approximated model checking

✔ Approach and semantics seem convincing;
✗ Prototype tool is a good proof of concept, but inefficient
✗ Our model checking problem is
  • Indecidable in general (decidable for finite models);
  • Complex in the rest of the cases.

➔ Directions:
  • Less expressive logics & better MC algorithms [Rensink et al.];
  • State space reductions and approximated model checking.
State-space reductions and approximated model checking

✔ Approach and semantics seem convincing;

✗ Prototype tool is a good proof of concept, but inefficient

✗ Our model checking problem is
  • Indecidable in general (decidable for finite models);
  • Complex in the rest of the cases.

➔ Directions:
  • Less expressive logics & better MC algorithms [Rensink et al.];
  • State space reductions and approximated model checking. [ICGT'12] + [ICFEM'12]
Particular case: bisimilar models
Particular case: bisimilar models
Particular case: bisimilar models
Particular case: bisimilar models
Particular case: bisimilar models
• Behavioural preorders and equivalences for counterpart models

\[
\begin{align*}
M & \sqsubseteq_R M \\
M & \sqsubseteq_R \overline{M}
\end{align*}
\]

Under-approximation \hspace{1cm} Original model \hspace{1cm} Over-approximation

\[\text{e.g. truncations...} \hspace{1cm} \text{simulates} \hspace{1cm} \text{simulates} \hspace{1cm} \text{e.g. merging of components...}\]

• Approximated formulae evaluation in sets of under- and over-approximations

\[
\begin{align*}
M_0 & \sqsubseteq_{R_0} \overline{M_0} \\
M & \sqsubseteq_{R_0} \overline{M} \\
M_n & \sqsubseteq_{R_n} \overline{M_n} \\
M_m & \sqsubseteq_{R_m} \overline{M_m}
\end{align*}
\]
• Behavioural preorders and equivalences for counterpart models

\[ \overline{M} \preceq_R M \preceq_R \overline{M} \]

Under-approximation \hspace{1cm} Original model \hspace{1cm} Over-approximation

\text{e.g. truncations...} \hspace{1cm} \text{simulates} \hspace{1cm} \text{simulates} \hspace{1cm} \text{e.g. merging of components...}

• Approximated formulae evaluation in sets of under- and over-approximations

\[(\sigma, w) \models \left\{ \overline{R_0} \ldots \overline{R_m} \right\} \left[ \psi[\Gamma; \Delta] \right]^M \]

\[ \implies T \]

\[ \implies F \]
• Behavioural preorders and equivalences for counterpart models

\[ M \sqsubseteq_R M \sqsubseteq_R \overline{M} \]

Under-approximation \quad \text{simulates} \quad \text{Original model} \quad \text{simulates} \quad \text{Over-approximation}

e.g. truncations...

\text{e.g. merging of components...}

• Approximated formulae evaluation in sets of under- and over-approximations

\[
(\sigma, w) \models \{ \overline{R_0} \ldots \overline{R_m} \} \left[ \psi[\Gamma; \Delta] \right]^M \quad \text{For bisimilar approximations}
\]
Approximation techniques in literature

1) Verification up-to-iso [Rensink, …]
Approximation techniques in literature

1) Verification up-to-iso [Rensink, ...]

Bisimilar models. R-morphisms are actually isomorphisms
Approximation techniques in literature

2) Neighbourhood abstraction [Rensink]
Approximation techniques in literature

2) Neighbourhood abstraction [Rensink]

Over-approximations. R-morphisms are total, node-surjective, edge-surjective
Approximation techniques in literature

3) Unfoldings [Baldan, Koenig]
Approximation techniques in literature

3) Unfoldings [Baldan, Koenig]

Over-approximations. R-morphisms are total, node-bijective, edge-surjective
Canonizer-based reduction

C(n_2, n_1, e_1, e_2)
Canonizer-based reduction
Canonizer-based reduction
Canonizer-based reduction
Building (reduced) models

Number of components

- Not reduced
- Name compact
- Up to iso

Number of states vs. Number of components

Time necessary (secs) vs. Number of components
Building (reduced) models

Number of components

- Name compact
- Up to iso

Graphs showing the relationship between the number of components and the number of states, as well as the time necessary, for both reduced and non-reduced models.
Building (reduced) models

Number of components vs. number of states

- Name compact
- Up to iso

Number of components vs. time necessary (secs)
Conclusions

Framework to reason on concurrent/distributed systems:

- Convincing approach and semantics,
- Widely applicable (to any system modelled as enriched transition system),
- Powerful abstraction techniques to mitigate state-space explosion,
- Prototype implementation,
- Good dissemination results.

Future works:

- Fully implement our approximation framework,
- Further exploit previous works on GTrSs approximations,
- Develop better MC algorithms and less expressive logics,
- Validate the approach against interesting case studies.