An heuristic for verifying safety properties of infinite-state systems

Gerardo Schneider

Uppsala University
Department of Information Technology
Uppsala, Sweden

Joint work with Michael Baldamus and Richard Mayr
Motivation

• How to build **correct** complex systems?
Motivation

- How to build correct complex systems?
- Synthesis (from the specification)
Motivation

- How to build **correct** complex systems?
- Synthesis (from the specification)
- Build them and then
  - Test
  - Simulate
Motivation

- How to build correct complex systems?
- Synthesis (from the specification)
- Build them and then
  - Test
  - Simulate
- Alternative: Formal verification
What is Verification?

- **Instance:**
  - $P$: Program (Hw circuit, communication protocol, distributed system, C program, Real-time system, etc)
  - $\phi$: Specification

- **Question:**
  - Does $P$ satisfies $\phi$?
Formal Verification

- It is a very active field for theoretical research and practical development
- Deductive vs Algorithmic approach
Formal Verification

- Model Checking (Algorithmic)
  - By now, a quite well-established theory (80’s)
  - Exhaustive exploration of the state-space
  - Fully automatic
  - Practical applications:
    - Hardware controllers
    - Circuit design
    - Many communication protocols
Formal Verification

- Limitations of Model Checking:
  - Finite-state systems
  - State explosion problem
Formal Verification

- Limitations of Model Checking:
  - Finite-state systems
  - State explosion problem
- Infinite-state systems: More general but more difficult to analyse!
Verification of Infinite-State Systems

- Key aspects to take into account
  - Non-bounded variables and/or data structures (e.g. counters, clocks, queues)
  - Parameterised systems (e.g. nets of unbounded number of id. processes)
  - Mobility
  - Security
Verification of Infinite-State Systems

- Examples of infinite-state systems
  - Timed and hybrid automata
  - Process rewrite systems
  - Push-down automata
  - Communicating FSA (e.g. Lossy channel systems)
  - Petri nets
  - Parameterised systems (mutual exclusion protocols, broadcast protocols, etc)
Verification of Infinite-State Systems

- Techniques:
  - Abstraction
  - Symbolic analysis
  - Well-quasi-ordering (WQO)
The Problem

- Our Dream: Verify the $\pi$-calculus!
The Problem

- Our Dream: Verify the $\pi$-calculus!
- Not yet there! We start with something simpler: CCS-like Calculus
Our Dream: Verify the $\pi$-calculus!

Not yet there! We start with something simpler: CCS-like Calculus

Which kind of properties?
- Safety properties (Reachability)
The Problem

- Our Dream: Verify the $\pi$-calculus!
- Not yet there! We start with something simpler: CCS-like Calculus
- Which kind of properties?
  - Safety properties (Reachability)
- Problems?
  - Verifying safety properties is undecidable in CCS
  - Termination
Our Solution

Algorithm:

- Give a Petri net semantics to CCS-like Agents
  Agent: $A$, Petri net: $N_A$
- Obtain an over-approximation Petri net $W(N_A)$
- Prove that $W(N_A)$ is a Well-Structured System
- Reachability is decidable in $W(N_A)$
Our Solution

- Our algorithm is partial:
  - If it says (NO) YES: the property is (not) satisfied
  - Sometimes it says UNKNOWN
Agenda

- Preliminaries
  - Well-Structured Systems
  - An Agent Language (CCS-like)
  - Petri Nets
- Petri Nets Semantics of the Agent Lang.
- Safety Properties Verification
- Concluding Remarks
Well-Structured Systems: Preliminaries

Let $<S, \rightarrow>$ (where $S = Q \times D$ is a set of states) be a labelled transition system (LTS) and $\preceq$ a preorder (reflexive and transitive)
Well-Structured Systems: Preliminaries

Let \( < S, \rightarrow > \) (where \( S = Q \times D \) is a set of states) be a labelled transition system (LTS) and \( \preceq \) a preorder (reflexive and transitive)

- \( \preceq \) is a WQO if there is no infinite sequence \( a_0, a_1, \ldots \), so that \( a_i \not\preceq a_j \) for any \( i \leq j \)
Well-Structured Systems: Preliminaries

Let $< S, \rightarrow>$ (where $S = Q \times D$ is a set of states) be a labelled transition system (LTS) and $\preceq$ a preorder (reflexive and transitive)

- Let $D$ be a set. A subset $U \subseteq D$ is upward closed if whenever $a \in U, b \in D$ and $a \preceq b$, then $b \in U$. The upward closure of a set $A \subseteq D$ is

$$
\mathcal{C}(A) := \{b \in D \mid \exists a \in A. a \preceq b\}
$$
Well-Structured Systems: Preliminaries

Let \( < S, \rightarrow > \) (where \( S = Q \times D \) is a set of states) be a labelled transition system (LTS) and \( \preceq \) a preorder (reflexive and transitive)

- A LTS \( < S, \rightarrow > \) is monotonic if, whenever \( s \preceq t \) and \( s \xrightarrow{\alpha} s' \), then \( t \xrightarrow{\alpha} t' \) for some \( t' \) so that \( s' \preceq t' \)
Well-Structured Systems: Definition

A trans. system $\mathcal{L} = \langle S, \rightarrow \rangle$ (with $\leq$ on data values) is well-structured if

- $\leq$ is a well–quasi–ordering, and
- $\langle S, \rightarrow \rangle$ is monotonic with respect to $\leq$, and
- for all $s \in S$ and $\alpha \in L$, the set $\min(\text{pre}_\alpha(C(\{s\})))$ is computable.
Theorem:

- Let \(<S, \rightarrow>\) be a WSS, \(<q, d>\) a state and \(U\) an upward-closed subset of the set of data values
- Then it is decidable whether it is possible to reach, from \(<q, d>\), any state \(<q', d'>\) with \(d' \in U\)
An Agent Language (CCS-like)

- Given:
  - A set of *names*, \( \mathcal{N} (a, b, x, y \ldots) \)
  - A set of *co-names*, \( \overline{\mathcal{N}} = \{ \overline{a} \mid a \in \mathcal{N} \} \). The set of *visible actions*: \( \text{Act} = \mathcal{N} \cup \overline{\mathcal{N}} \)
  - We denote by \( \text{Act}_\tau = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{ \tau \} (\alpha) \)
An Agent Language (CCS-like)

- Given:
  - A set of *names*, \( \mathcal{N} (a, b, x, y \ldots) \)
  - A set of *co-names*, \( \overline{\mathcal{N}} = \{ \overline{a} \mid a \in \mathcal{N} \} \). The set of *visible actions*: \( \text{Act} = \mathcal{N} \cup \overline{\mathcal{N}} \)
  - We denote by \( \text{Act}_\tau = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{ \tau \} (\alpha) \)
  - The syntax is given by:

\[
P ::= 0 \mid \alpha.P \mid P + Q \mid P\backslash c \mid P \parallel P \mid A
\]

Where \( A \overset{\text{def}}{=} P \)
Petri Nets

• A Petri net is a tuple $N = (P, A, T, M_0)$:
  • $P$ is a finite set of places
  • $A$ is a finite set of actions (or labels)
  • $T \subseteq \mathcal{M}(P) \times A \times \mathcal{M}(P)$ is a finite set of transitions
  • $M_0$ is the initial marking

where $\mathcal{M}(P)$ is a collection of multisets (bags) over $P$
Petri Nets: Graphical representation

Marking

$M :$ is a mapping from places to the set of natural numbers

$M(P_1) = 3 \quad M(P_2) = 1$

$M(P_3) = 0 \quad M(P_4) = 2$
Agenda

- Preliminaries
  - Well-Structured Systems
  - An Agent Language (CCS-like)
  - Petri Nets
- Petri Nets Semantics of the Agent Lang.
- Safety Properties Verification
- Concluding Remarks
Petri Nets Semantics of the Agent Lang.

- We will use Coloured Petri Nets
Petri Nets Semantics of the Agent Lang.

- We will use Coloured Petri Nets
Petri Nets Semantics of the Agent Lang.

- We will use Coloured Petri Nets

- In particular, we will use \textit{strings} as colours
Petri Nets Semantics of the Agent Lang.: Formal Definition

- Places: all agent constants together with all agents and sub-agents that occur on the right-hand side of any defining equation within the environment
Petri Nets Semantics of the Agent Lang.: Formal Definition

- **Places**
- **Transitions** :

\[
\begin{align*}
\text{Trans}(\alpha.P) &= \left\{ \langle \{\alpha.P\}, \{P\} \rangle \mapsto \alpha \right\} \\
\text{Trans}(P + Q) &= \left\{ \langle \{P + Q\}, \{P\} \rangle, \langle \{P + Q\}, \{Q\} \rangle \right\} \\
\text{Trans}(P|Q) &= \left\{ \langle \{P\}, \{P \rightarrow l, Q \rightarrow r\} \rangle \right\} \\
\text{Trans}(P\setminus c) &= \left\{ \langle \{P\setminus c\}, \{P\} \rangle \mapsto \setminus c \right\} \\
\text{Trans}(A) &= \left\{ \langle \{A\}, \{P\} \rangle \right\}, \text{ given that } A \triangleq P
\end{align*}
\]
Petri Nets Semantics of the Agent
Lang.: Example

\[ A \overset{\text{def}}{=} ((((a \cdot 0 + b \cdot 0) \parallel (\overline{a} \cdot 0 + c \cdot 0)) \parallel A) \setminus a \]
Petri Nets Semantics of the Agent
Lang.: Example

\[(a \cdot 0 + b \cdot 0) \parallel (\bar{a} \cdot 0 + c \cdot 0) \parallel A\]
Petri Nets Semantics of the Agent
Lang.: Example
Petri Nets Semantics of the Agent Lang.: Example

\[(\{(a \cdot 0 + b \cdot 0) \parallel (\overline{a} \cdot 0 + c \cdot 0)\} \parallel A)\backslash a\]

\[((a \cdot 0 + b \cdot 0) \parallel (\overline{a} \cdot 0 + c \cdot 0)) \parallel A\]

\[(a \cdot 0 + b \cdot 0) \parallel (\overline{a} \cdot 0 + c \cdot 0)\]
Petri Nets Semantics of the Agent Lang.: Example

An heuristic for verifying safety properties of infinite-state systems – p.18/31
Petri Nets Semantics of the Agent Lang.: Example

\[ (((a \cdot 0 + b \cdot 0) \parallel (\bar{a} \cdot 0 + c \cdot 0)) \parallel A) \backslash a \]

\[ ((a \cdot 0 + b \cdot 0) \parallel (\bar{a} \cdot 0 + c \cdot 0)) \parallel A \]

\[ (a \cdot 0 + b \cdot 0) \parallel (\bar{a} \cdot 0 + c \cdot 0) \]

\[ a \cdot 0 + b \cdot 0 \]

\[ b \cdot 0 \parallel a \cdot 0 \]

\[ b \parallel a \]

\[ 0 \parallel 0 \]

\[ \tau \]

An heuristic for verifying safety properties of infinite-state systems – p.18/31
Petri Nets Semantics of the Agent Lang.: Example

(((a . 0 + b . 0) || (\overline{a} . 0 + c . 0)) || A) \setminus a

(((a . 0 + b . 0) || (\overline{a} . 0 + c . 0)) || A

(a . 0 + b . 0) || (\overline{a} . 0 + c . 0)

a . 0 + b . 0

b . 0

b
Petri Nets Semantics of the Agent
Lang.: Formal Definition

- **Tokens**: \((Act \cup \{1, r\})^*\); Empty token: \(\epsilon\).
  They carry history information about:
  - Concurrent threads, and
  - In which scope w.r.t. restriction they are
Petri Nets Semantics of the Agent Lang.: Formal Definition

- Tokens
- Firing (Enabling of Transitions):
  - For transition $t$ with one input place and a token $\theta$, $t$ is enabled if some of the following hold
    - $t$ is not labelled with a visible action
    - $t$ is labelled with a visible action $a$ and $\theta$ doesn’t contain $a$
Petri Nets Semantics of the Agent Lang.: Formal Definition

- Tokens

- Firing (Enabling of Transitions):
  - For transition $t$ with two input places $p_1$ and $p_2$ and tokens $\theta_1$ and $\theta_2$, $t$ is enabled if both of the following hold
    - $\text{pc}(\text{pre}_i(t)) \setminus \text{Act} \neq \emptyset$, $i = 1, 2$, while $\text{pc}(\text{pre}_1(t)) \setminus \text{Act} \neq \text{pc}(\text{pre}_2(t)) \setminus \text{Act}$
    - $\maxpref_a(\text{pc}(\text{pre}_1(t))) = \maxpref_a(\text{pc}(\text{pre}_2(t)))$
Petri Nets Semantics of the Agent
Lang.: Example

\[ A \overset{\text{def}}{=} (((a.0 + b.0) \parallel (\overline{a}.0 + c.0)) \parallel A) \setminus a \]
Petri Nets Semantics of the Agent Lang.: Example
Petri Nets Semantics of the Agent Lang.: Example

An heuristic for verifying safety properties of infinite-state systems – p.20/31
Petri Nets Semantics of the Agent Lang.: Example
Petri Nets Semantics of the Agent

Lang.: Example

An heuristic for verifying safety properties of infinite-state systems – p.20/31
Petri Nets Semantics of the Agent
Lang.: Example

An heuristic for verifying safety properties of infinite-state systems – p.20/31
Petri Nets Semantics of the Agent Lang.: Example
Petri Nets Semantics of the Agent Lang.: Example

An heuristic for verifying safety properties of infinite-state systems – p.20/31
Petri Nets Semantics of the Agent
Lang.: Example

An heuristic for verifying safety properties of infinite-state systems – p.20/31
Petri Nets Semantics of the Agent

Lang.: Example

An heuristic for verifying safety properties of infinite-state systems – p.20/31
Petri Nets Semantics of the Agent
Lang.: Extra Structure

• We define a preorder between tokens:

\[ \eta \preceq \theta \text{ if } \eta \text{ is a (not necessarily contiguously) substring of } \theta \]

Example:

\[ all \preceq ararall \]
Petri Nets Semantics of the Agent  
Lang.: Extra Structure

- We define an ordering between markings: 
  \( m_1 \sqsubseteq m_2 \)  

**Example:** \( m_1 \)

\[\begin{align*}
P_1 &\quad P_2 &\quad P_3 &\quad P_4 \\
P_5 &\quad P_6 &\quad P_7 &\quad P_8 \\
& & & & \\
\text{arall} & b & & c \\
& a &\quad \tau & \overline{a} \\
& & & & \\
\text{arall} & & & & \\
\end{align*}\]
Petri Nets Semantics of the Agent Lang.: Extra Structure

- We define an ordering between markings:

  \[ m_1 \sqsubseteq m_2 \]

**Example:**

\[
\begin{align*}
  m_1 &= \{ \ldots, (P_1, \{ arall \}), (P_2, \{ all \}), (P_3, \{ alr, aralr \}), \\
          (P_4, \{ \}), (P_5, \{ \}), (P_6, \{ \}), (P_7, \{ \}), (P_8, \{ \}) \} \\
  m_2 &= \{ \ldots, (P_1, \{ arall \}), (P_2, \{ ararall \}), (P_3, \{ alr, aralr \}), \\
          (P_4, \{ araralr \}), (P_5, \{ all \}), (P_6, \{ \}), (P_7, \{ \}), (P_8, \{ \}) \}
\end{align*}
\]
Petri Nets Semantics of the Agent
Lang.: Extra Structure

- We define an ordering between markings:
  \[ m_1 \sqsubseteq m_2 \]

**Intuition:** \( m \sqsubseteq m' \) if \( m' \) represents a (not necessarily strictly) longer firing history than \( m \)
Petri Nets Semantics of the Agent Lang.: Extra Structure

- We define an ordering between markings:
  \[ m_1 \sqsubseteq m_2 \]

- Markings represent upward closed sets

Example:

\[
m_1 = \{ \ldots, (P_1, \{ar\ all\}), (P_2, \{all\}), (P_3, \{alr, ar\ alr\}), (P_4, \{}), (P_5, \{}), (P_6, \{}), (P_7, \{}), (P_8, \{}), \ldots \} \]

An heuristic for verifying safety properties of infinite-state systems – p.21/31
Our Petri Nets are not WSS

Very nice, but...
Our Petri Nets are not WSS

Very nice, but...

- Our Petri nets are not monotonic!
Our Petri Nets are not WSS

- Counter-example: Let

\[ m_1 = \{ \ldots, (P_1, \{ arall \}), (P_2, \{ all \}), (P_3, \{ alr, aralr \}), (P_4, \{ \}), (P_5, \{ \}), (P_6, \{ \}), (P_7, \{ \}), (P_8, \{ \}) \} \]
Our Petri Nets are not WSS

- **Counter-example:** Let

  \[
  m_1 = \{ \ldots, (P_1, \text{arall}), (P_2, \text{all}), (P_3, \text{alr, aralr}), (P_4, \{}), (P_5, \{}), (P_6, \{}), (P_7, \{}), (P_8, \{})) \}
  \]

  \[
  m_2 = \{ \ldots, (P_1, \text{arall}), (P_2, \text{ararall}), (P_3, \text{alr, aralr}), (P_4, \text{araralr}), (P_5, \text{all}), (P_6, \{}), (P_7, \{}), (P_8, \{})) \}
  \]

- **Notice that** \( m_1 \sqsubseteq m_2 \)
Our Petri Nets are not WSS

- **Counter-example:** Let

\[ m_1 = \{ \ldots, (P_1, \{ \text{arall} \}), (P_2, \{ \text{all} \}), (P_3, \{ \text{alr}, \text{aralr} \}), (P_4, \{}), (P_5, \{}), (P_6, \{}), (P_7, \{}), (P_8, \{} \} \} \]

- **Moreover,** \( m_1 \rightarrow m_1' \), where

\[ m_1' = \{ \ldots, (P_1, \{ \text{arall} \}), (P_2, \{}), (P_3, \{ \text{aralr} \}) (P_4, \{}), (P_5, \{}), (P_6, \{ \text{all} \}), (P_7, \{ \text{alr} \}), (P_8, \{} \} \} \]
Our Petri Nets are not WSS

- Counter-example: Let

\[ m_1 = \{ \ldots, (P_1, \{ \text{arall} \}), (P_2, \{ \text{all} \}), (P_3, \{ \text{alr, aralr} \}), (P_4, \{ \}), (P_5, \{ \}), (P_6, \{ \}), (P_7, \{ \}), (P_8, \{ \}) \} \]

But, there is no \( m'_2 \) such that \( m'_1 \subseteq m'_2 \) and \( m_2 \rightarrow m'_2 \)

\[ \rightarrow \text{It is not monotonic!} \]
Petri Nets as WSS

We make an over-approximation of the Petri net

- We change the synchronisation policy: A transition may be fired even if the tokens don’t synchronise (Weak Firings)
Petri Nets as WSS

We make an over-approximation of the Petri net

- We change the synchronisation policy: A transition may be fired even if the tokens don’t synchronise (Weak Firings)

Lemma: P–nets with weak firings are well-structured systems
Petri Nets as WSS

We make an over-approximation of the Petri net

- We change the synchronisation policy: A transition may be fired even if the tokens don’t synchronise (Weak Firings)

**Lemma:** P–nets with weak firings are well-structured systems

**Corollary:** The control state reachability problem is decidable for p–nets with weak firings
Agenda

- Preliminaries
  - Well-Structured Systems
  - An Agent Language (CCS-like)
  - Petri Nets
- Petri Nets Semantics of the Agent Lang.
- Safety Properties Verification
- Concluding Remarks
Verification of Safety Properties: The Problem

Instance: An agent $A$ with initial state $ini$ and an atomic action $a$

Question: Can agent $A$ ever execute action $a$?
Verification of Safety Properties: The Algorithm

Preparatory phase:

[1] Build the p–net $N$ associated with $A$

[2] For every transition $t$ labelled with $a$ there is a minimal marking $m_t$ that enables $t$. It is given by an $\epsilon$–token on all places in $\text{pre}(t)$. Then $M^a = \{ m_t \mid t \text{ labelled by } a \}$. 
Verification of Safety Properties: The Algorithm

Preparatory phase:


[2] For every transition $t$ labelled with $a$ there is a minimal marking $m_t$ that enables $t$. It is given by an $\epsilon$–token on all places in $\text{pre}(t)$. Then $M^a = \{m_t \mid t \text{ labelled by } a\}$.

Remark: $m_t$ is an upward closed set: “At least one token in $\text{pre}(t)$”
Verification of Safety Properties: The Algorithm

Algorithm:

\[
\text{function } \text{Reachability}(N, M^a, ini) : \\
\quad (OB, s) := \text{Search}_{\text{backward}}(M^a, ini) \\
\quad \text{if } ini \notin OB \\
\quad \text{then } \leftarrow \text{NO} \\
\quad \text{else } \leftarrow \text{Search}_{\text{forward}}(ini, M^a, OB, b(s))
\]
Agenda

- Preliminaries
  - Well-Structured Systems
  - An Agent Language (CCS-like)
  - Petri Nets
- Petri Nets Semantics of the Agent Lang.
- Safety Properties Verification
- Concluding Remarks
Concluding Remarks

- We have given a (finite-control) Petri net semantics to a CCS-like calculus
- We have presented a general technique for reachability analysis of non-WSS
  - It combines backward and forward reachability analysis
  - It produces answers: YES, NO, UNKNOWN (YES and NO always correct)
- We have applied it to partially decide the reachability problem for a CCS-like calculus
Future Work (Research Topics)

- Use this methodology for verifying safety properties of
  - $\pi$-calculus
  - Concurrent Constraint Programming
  - Others?
- Implementation of the Algorithm
MUITO OBRIGADO!