A New Approach for the Symbolic Model Checking of Petri Nets

Nicolas AMAT

Supervisors: Silvano DAL ZILIO, Hubert GARAVEL, Didier LE BOTLAN

LAAS-CNRS x Univ. Grenoble Alpes

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d’avenir

June 23, 2020
“It is fair to state, than in the digital era correct systems for information processing are more valuable than gold.”

— H. Barendregt, The quest for correctness.

seL4, CompCert, Protocole de cohérence de cache “Futurebus+”, Algorithmes distribués randomisés.

— H. Garavel, Three Decades of Success Stories in Formal Methods.
A Petri net example; Christian Stahl.

Motivations

Introduction

**Context:** Model Checking of “General” Petri nets
- Not only 1-safe nets
- Inhibitor and Read arcs

**Goal:** Use of net reductions to overcome *state-space explosion*
- Great results for model counting [Berthomieu, 2019]
- SMT-based methods
A property $P$ is correct if for all reachable marking $m$ in $R_N(m_0)$, $m$ satisfies $P$, denoted $m \models P$.

- Proving $P$ correct is equivalent to checking $\Box P$ in LTL or $AG P$ in CTL.

Formula with variables in $\vec{x}$ that is only “satisfiable at marking $m$”:

$$m(\vec{x}) \equiv \bigwedge_{i \in 1..n}(x_i = m(p_i))$$

Check satisfiability of $\neg P(\vec{x}) \land m(\vec{x})$.
Some Examples of Interesting Properties

Introduction

- **PlaceReach**: $\text{REACH}(p) \equiv m(p) \geq 1$

- **QuasiLiveness**: $\text{LIVE}(t) \equiv \bigwedge_{p \in \bullet t} \text{COVER}(p, \text{pre}(t, p))$

- **ReachabilityDeadlock**: $\text{DEAD} \equiv \bigwedge_{t \in T} \neg \text{LIVE}(t)$

- **ConcurrentPlaces**: $p_1 \parallel p_2 \equiv \text{REACH}(p_1) \land \text{REACH}(p_2)$

- **OneSafe, StableMarking, ...**
Net Reductions
Introduction

Net reduction example, with equation $E : a = x + y$

Relation between state-spaces
State-space abstraction by a “polyhedral approach”
1. Formalization of Net Reductions

2. Model Checking Algorithms

3. SMPT: Another Model-Checker

4. Application: Concurrent Places Problem
Formalization of Net Reductions
Reduction Rule Example: Concatenate (CONCAT)

Formalization of Net Reductions

Equation: \( x = y_1 + y_2 \)
A marking $m$ can be associated to \textit{system of equations} $m(\vec{x})$ defined as, $x_1 = m(p_1), \ldots, x_n = m(p_n)$ where $P = \{p_1, \ldots, p_n\}$

$E$ is \textit{satisfiable} for $m$ if the system $E, m$ has solutions

Given two markings $m_1, m_2$ from two nets $N_1, N_2$, we say that $m_1$ and $m_2$ are \textit{compatible}, denoted $(m_1 \uplus m_2)$, when $m_1(p) = m_2(p)$ for all $p$ in $P_1 \cap P_2$ (or equivalently $m_1, m_2$ is satisfiable)
\( E \)-abstraction: \((N_1, m_1) \sqsupseteq_E (N_2, m_2)\)

(A1) system \( E \) is solvable for \( N_1, N_2 \) and the initial markings are compatible with \( E \), meaning \( m_1 \uplus m_2 \models E \)

(A2) for all firing sequence \( \sigma_1 \) such that \((N_1, m_1) \xrightarrow{\sigma_1} (N_1, m'_1)\) then for all marking \( m'_2 \) over \( P_2 \) such that \( m'_1 \uplus m'_2 \models E \) we must have a firing sequence \( \sigma_2 \) in \( N_2 \) with the same observables, meaning: that \((N_2, m_2) \xrightarrow{\sigma_2} (N_2, m'_2)\) and \( l_1(\sigma_1) = l_2(\sigma_2)\).

\( E \)-abstraction equivalence: \((N_1, m_1) \triangleright_E (N_2, m_2)\)

- Iff \((N_1, m_1) \sqsupseteq_E (N_2, m_2)\) and \((N_2, m_2) \sqsupseteq_E (N_1, m_1)\)
Basic Property of $E$-Equivalence

Formalization of Net Reductions

- **Bounded Model-Checking**: If $(N_1, m_1) \triangleright_E (N_2, m_2)$, then for all marking $m'_1$ in $R_{N_1}(m_1)$ there exists $m'_2$ in $R_{N_2}(m_2)$ such that $m'_1 \uplus m'_2 \models E$.

- **Invariance Checking**: If $(N_1, m_2) \triangleright_E (N_2, m_2)$, then for all pair of markings $m'_1, m'_2$ over $N_1, N_2$ such that $m'_1 \uplus m'_2 \models E$ and $m'_2 \in R_{N_2}(m_2)$ it is the case that $m'_1 \in R_{N_1}(m_1)$. 
Composition Laws
Formalization of Net Reductions

**Axioms**: Reduction Rules (CONCAT, etc.)

**(COMP) Composability**
- If \((N_1, m_1) \triangleright_E (N_2, m_2)\), then
  \((N_1, m_1) \parallel (N_3, m_3) \triangleright_E (N_2, m_2) \parallel (N_3, m_3)\)

**(TRANS) Transitivity**
- If \((N_1, m_1) \triangleright_E (N_2, m_2)\) and \((N_2, m_2) \triangleright_{E'} (N_3, m_3)\), then
  \((N_1, m_1) \triangleright_{E, E'} (N_3, m_3)\).

**(RENAME) Relabeling**
- If \((N_1, m_1) \triangleright_E (N_2, m_2)\), then \((N_1[a/b], m_1) \triangleright_E (N_2[a/b], m_2)\)
Net Reduction Example Step by Step

Formalization of Net Reductions

Initial net, $S_1$, with a pattern for rule (CONCAT) emphasized in blue.

$$E_0 = \emptyset$$
Net $S_2$, with the result of applying rule (CONCAT) emphasized in blue.

$$E_1 = \{ a_1 = p_1 + p_2 \} \quad (2)$$

We have: $S_1 \triangleright_{E_1} S_2$. 
Net Reduction Example (Step 2)

Formalization of Net Reductions

Net $S_3$, with the result of applying rule (CONCAT) emphasized in blue.

$$E_2 = \begin{cases} 
  a_2 &= p_3 + p_4, \\
  a_3 &= p_5 + p_6, \\
  a_4 &= p_7 + p_8 
\end{cases}$$

(3)

We have: $S_2 \supseteq E_2 S_3$. 

Net Reduction Example
Formalization of Net Reductions

By transitivity, $S_1 \triangleright_{E_1,E_2} S_3$
Model Checking Algorithms
Bounded Model Checking \((BMC)\): counter-examples
Property Directed Reachability \((IC3)\): invariant proof
Bounded Model Checking (BMC)

Model Checking Algorithms

[Biere et al., 1999]
- Find counter-example violating a property
- Unroll Transitions
- SAT based

BMC method representation

$P$, $\neg P \equiv R$

k-path?
Algorithm adaptation (SMT-based)

- ENBLD$_t$(\(\vec{x}\)) \equiv \bigwedge \{ (x_i \geq k) \mid k = \text{pre}(t, p_i) > 0 \}

- \Delta_t(\vec{x}, \vec{x}') \equiv \bigwedge \{ (x'_i = x_i + \delta_i) \mid \delta_i = \text{post}(t, p_i) - \text{pre}(t, p_i), 1 \leq i \leq n \}

- T(\vec{x}, \vec{x}') \equiv \text{ALLEQ}(\vec{x}, \vec{x}') \lor \bigvee_{t \in T} (\text{ENBLD}_t(\vec{x}) \land \Delta_t(\vec{x}, \vec{x}'))

**Lemma:** \( m(\vec{x}) \land T(\vec{x}, \vec{x}') \land m'(\vec{x}') \): \( m' \) is at most one-step from \( m \)
Bounded Model Checking (BMC)
Model Checking Algorithms

\[
\begin{aligned}
\phi_0(N, m_0)(\vec{x}_0) & \equiv m_0(\vec{x}_0) \\
\phi_{i+1}(N, m_0)(\vec{x}_{i+1}) & \equiv \phi_i(N, m_0)(\vec{x}_i) \land T(\vec{x}_i, \vec{x}_{i+1})
\end{aligned}
\]

For \( k \geq 0 \), check \( \phi_k(\vec{x}_k) \land R(\vec{x}_k) \) until SAT

BMC Algorithm
We can find counter-examples to $R$ on $N_1$ by finding counter-examples to $E \land R$ on $N_2$.
(usually $k$ and $|T|$ are much smaller).

$$\phi^R_i(N_1, m_1)(\vec{x}) \equiv \phi_i(N_2, m_2)(\vec{y}_i) \land E(\vec{x}, \vec{y}_i) \land R(\vec{x})$$
[Bradley, 2011]

- Induction, Over-approximation & SAT Solving
- Unroll at most one transition
- Generate clauses that are inductive

IC3 method representation
SMPT: Another Model-Checker
Available on GitHub under GPLv3 license

github.com/nicolasAmat/SMPT

Python language (≈ 3,000 LoC)

Z3 (SMT-LIB v2)

Input Petri nets at the .net format

Run the tool: ./smpt.py --deadlock <.net>

Take advantage of net reductions

./smpt.py --deadlock <.net> --reduced <.net>
Features
SMPT: Another Model-Checker

Property verification
- Deadlock --deadlock
- Quasi-liveness --liveness <t>
- (Place) Reachability --reachability <p>
- Concurrent Places: --concurrent-places <p1>,...,<pk>

Debug
- Verbose: --verbose
- Print SMT-LIB input/output --debug
We check if a particular place can be marked in the model.

<table>
<thead>
<tr>
<th>MODEL</th>
<th># STATES</th>
<th>RESULT</th>
<th>TIME</th>
<th>T_{Reduced}</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirplaneLD–10</td>
<td>$4.3 \times 10^4$</td>
<td>CEX</td>
<td>9.17s</td>
<td>0.16s</td>
</tr>
<tr>
<td>AirplaneLD–20</td>
<td>$3.1 \times 10^5$</td>
<td>CEX</td>
<td>50.26s</td>
<td>0.16s</td>
</tr>
<tr>
<td>AirplaneLD–$\infty$</td>
<td>$\infty$</td>
<td>CEX</td>
<td>n.a.</td>
<td>0.16s</td>
</tr>
<tr>
<td>IBM319 (merge...)</td>
<td>$2.4 \times 10^3$</td>
<td>CEX</td>
<td>$&gt; 200s$</td>
<td>0.14s</td>
</tr>
<tr>
<td>IBM319 (callTo...)</td>
<td>$2.4 \times 10^3$</td>
<td>PROOF</td>
<td>$&gt; 200s$</td>
<td>12.02s</td>
</tr>
</tbody>
</table>
We check if places $P1$ and $P2$ can be marked together in model AirplaneLD (we know it is not possible).}

<table>
<thead>
<tr>
<th>Model</th>
<th># States</th>
<th>Result</th>
<th>Time</th>
<th>$T_{Reduced}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirplaneLD–10</td>
<td>$4.3 \times 10^4$</td>
<td>PROOF</td>
<td>1.50s</td>
<td>0.26s</td>
</tr>
<tr>
<td>AirplaneLD–20</td>
<td>$3.1 \times 10^5$</td>
<td>PROOF</td>
<td>2.51s</td>
<td>0.26s</td>
</tr>
<tr>
<td>AirplaneLD–4000</td>
<td>$2.1 \times 10^{12}$</td>
<td>PROOF</td>
<td>1680s</td>
<td>0.26s</td>
</tr>
</tbody>
</table>

\(^{1}\text{time to generate the state space of AirplaneLD-4000 with ITS is } > 2500s.\)

\(^{2}\text{time to reduce: 67.79s}\)
Application: Concurrent Places Problem
Useful for the decomposition into Nested-Unit Petri Nets (NUPNs)

Two places $p_1$ and $p_2$ are concurrent, denotes as $p_1 \parallel p_2$ iff there exists a reachable marking $m$ in $R_N(m_0)$ such that $m(p_1) > 0$ and $m(p_2) > 0$. 
A new method that take advantage of net reductions:

**Step 1)** Compute the concurrency relation of the reduced net $N_2$

**Step 2)** *Change of Basis*, compute the concurrency relation of the initial net $N_1$ from the system of equations $E$ and the concurrency relation of the reduced net $N_2$
Concurrent Relation Construction
Application: Concurrent Places Problem

Concurrency relation: undirected graph \((P, R)\), where vertices are places and there is an edge \((p, q) \in R\) when \(p \parallel q\)

Output: Concurrency relation \(C\)

\[
C \leftarrow \{\};
\]

\[
m \leftarrow \text{initial marking } m_0;
\]

\[
\text{while } C \leftarrow C \cup \text{stepper}(m, C);
\]

\[
\text{do}
\]

\[
\text{parallel}
\]

\[
\text{begin}
\]

\[
\text{if } IC3 \text{ proves that we found all concurrent places}
\]

\[
\text{then return } C;
\]

\[
\text{begin}
\]

\[
\text{if } BMC \text{ finds a counter-example } m' \text{ with new concurrent places then}
\]

\[
m \leftarrow m';
\]

\[
\text{continue;}
\]

\[
\text{end}
\]

\[
\text{end}
\]

\[
\text{end}
\]
Change of Basis using Reduction Equations

Application: Concurrent Places Problem

# R |- P₃ = P₂
# A |- a₁ = Pout₁ + Pm₁
# A |- a₂ = Pback₁ + a₁
# A |- a₃ = Pout₂ + Pm₂
# A |- a₄ = Pback₂ + a₃
# A |- a₅ = Pout₃ + Pm₃
# A |- a₆ = Pback₃ + a₅
# A |- a₇ = Pout₄ + Pm₄
# A |- a₈ = Pback₄ + a₇
# A |- a₉ = a₈ + P₄
# R |- a₉ = 5
# R |- a₆ = a₄
# A |- a₁₀ = a₄ + P₂
# R |- a₁₀ = 5
# A |- a₁₁ = a₂ + P₁
# R |- a₁₁ = 5

Output of tool reduce on the Kanban instance for \( N = 5 \)
(\#states: 2 546 400 – 16 places, 16 transitions, 40 arcs)
Change of Basis using Reduction Equations
Application: Concurrent Places Problem

# R |- P3 = P2
# A |- a1 = Pout1 + Pm1
# A |- a2 = Pback1 + a1
# A |- a3 = Pout2 + Pm2
# A |- a4 = Pback2 + a3
# A |- a5 = Pout3 + Pm3
# A |- a6 = Pback3 + a5
# A |- a7 = Pout4 + Pm4
# A |- a8 = Pback4 + a7
# A |- a9 = a8 + P4
# R |- a9 = 5
# R |- a6 = a4
# A |- a10 = a4 + P2
# R |- a10 = 5
# A |- a11 = a2 + P1
# R |- a11 = 5
# R |- a11 = 5
# A |- a11 = a2 + P1
# A |- a2 = Pback1 + a1
# A |- a1 = Pout1 + Pm1

Output of tool reduce on the Kanban instance for $N = 5$
(#states: 2,546,400 – 16 places, 16 transitions, 40 arcs)
The approach used in SMPT is promising

Contributions for SMT-based model-checking algorithms

New equivalence relation: *E*-abstraction equivalence

New method for the *Concurrent Places Problem*
Thank you for your attention!
Any questions?
### Bounded Model Checking (BMC)

#### Model Checking Algorithms

**Assertion stack**

<table>
<thead>
<tr>
<th>Init</th>
<th>Step 1</th>
<th>Step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{m_0(x_0)}$</td>
<td>$\overline{m_0(x_0)}$</td>
<td>$\overline{m_0(x_0)}$</td>
</tr>
<tr>
<td>$\overline{R(x_0)}$</td>
<td>$T(x_0, x_1)$</td>
<td>$T(x_0, x_1)$</td>
</tr>
<tr>
<td>$\overline{R(x_1)}$</td>
<td>$T(x_1, x_2)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\overline{R(x_2)}$</td>
</tr>
</tbody>
</table>
### Bounded Model Checking (BMC) + Reductions

#### Model Checking Algorithms

Assertion stack with reductions

<table>
<thead>
<tr>
<th></th>
<th>Init</th>
<th>Step 1</th>
<th>Step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R(\vec{x})$</td>
<td>$R(\vec{x})$</td>
<td>$R(\vec{x})$</td>
<td></td>
</tr>
<tr>
<td>$m_0(\vec{y}_0)$</td>
<td>$m_0(\vec{y}_0)$</td>
<td>$m_0(\vec{y}_0)$</td>
<td></td>
</tr>
<tr>
<td>$E(\vec{x}, \vec{y}_0)$</td>
<td>$T(\vec{y}_0, \vec{y}_1)$</td>
<td>$T(\vec{y}_0, \vec{y}_1)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E(\vec{x}, \vec{y}_1)$</td>
<td></td>
<td>$E(\vec{x}, \vec{y}_2)$</td>
</tr>
</tbody>
</table>
Over-Approximated Reachability Sequence (OARS) of formulas $F_0, \ldots, F_{k+1}$ such that:

- $(F_0 = I \subseteq F_1 \subseteq \cdots \subseteq F_{k+1} = P)$
- For all $i \in 0 \ldots k + 1$. $F_i(\vec{x}) \land T(\vec{x}, \vec{x}') \Rightarrow F_{i+1}(\vec{x}')$

Each $F_i$ describes a set of states that:

1. Includes the states $s$ less than $i$ steps from $I$,
2. Contains only states $s$ which are more than $k - i + 1$ steps from $R$.

Proved when $F_i = F_{i+1}$.
Perspectives

- Continue to work on SMT-based algorithms
  - Add states equations
  - Add invariants
  - Add BDDs

- Explore new reduction rules
  - Theorem Prover
  - Specific rules

- Model Counting
  - Convex analysis [Barvinok]
  - Combinatorial approach

Participation in *Reachability* category of the Model Checking Contest.
Prevalence of Reductions over the MCC Instances

Nicolas AMAT
Master Project Defense