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

Ariane 5, 1996

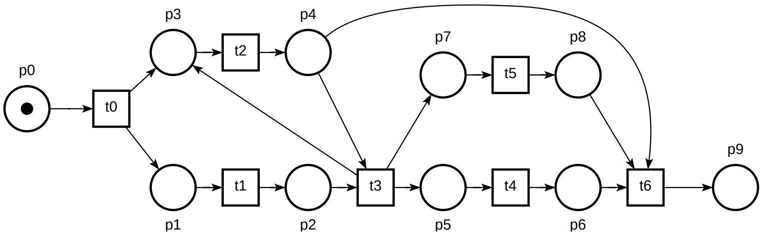
"It is fair to state, than in the digital era correct systems for information processing are more valuable than gold."

— 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.

Mathematical model: Petri net
Introduction



A Petri net example; Christian Stahl.

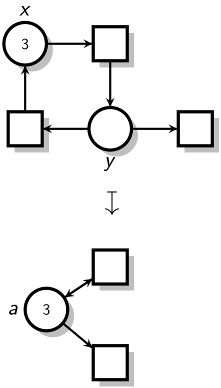
"Decomposing Petri net state spaces." In 18th German Workshop on Algorithms and Tools for Petri Nets. 2011.

- **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

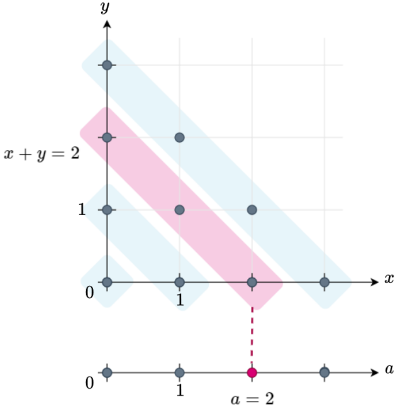
- A property P is correct if for all reachable marking m in $R_N(m_0)$, m satisfies P, denoted $m \models P$
  - proving P correct is equivalent to checking $\Box P$ in LTL or $\mathrm{AG}\, P$ in CTL
- Formula with variables in $\vec{x}$ that is only "satisfiable at marking m": $\underline{m}(\vec{x}) \equiv \bigwedge_{i \in 1..n}(x_i = m(p_i))$
- Check satisfiability of $\neg P(\vec{x}) \wedge \underline{m}(\vec{x})$

Some Examples of Interesting Properties

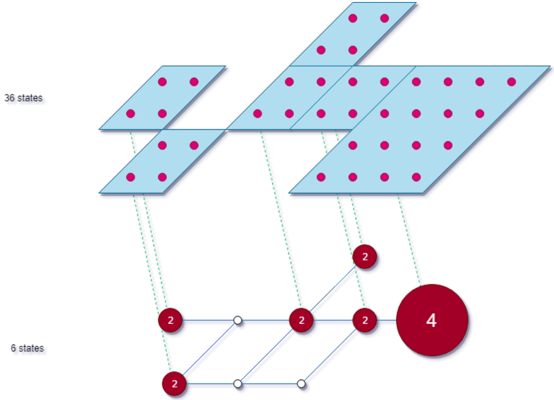
- **PlaceReach**: $\mathrm{REACH}(p) \;\equiv\; m(p) \geq 1$
- **QuasiLiveness**: $\mathrm{LIVE}(t) \;\equiv\; \bigwedge_{p \in {}^\bullet t} \mathrm{COVER}(p, \mathbf{pre}(t, p))$
- **ReachabilityDeadlock**: $\mathrm{DEAD} \;\equiv\; \bigwedge_{t \in T} \neg \mathrm{LIVE}(t)$
- **ConcurrentPlaces**: $p_1 \| p_2 \;\equiv\; \mathrm{REACH}(p_1) \wedge \mathrm{REACH}(p_2)$
- **OneSafe, StableMarking,** ...



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



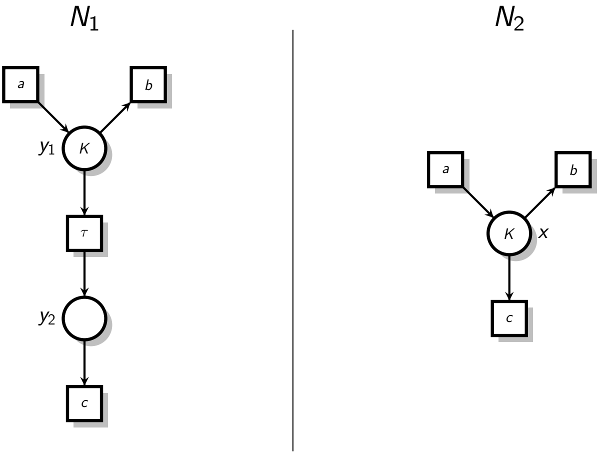
Relation between state-spaces



State-space abstraction by a "polyhedral approach"

Formalization of Net Reductions

Reduction Rule Example: Concatenate (CONCAT)



Equation: $x = y_1 + y_2$

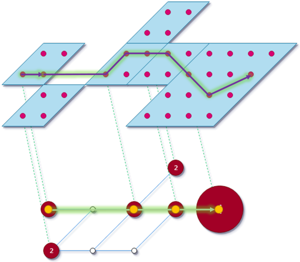
- A marking m can be associated to *system of equations* $\underline{m}(\vec{x})$ defined as, $x_1 = m(p_1), \ldots, x_n = m(p_n)$ where $P = \{p_1, \ldots, p_n\}$
- E is *satisfiable* for m if the system $E, \underline{m}$ has solutions
- Given two markings m_1, m_2 from two nets N_1, N_2, we say that m_1 and m_2 are *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 $\underline{m_1}, \underline{m_2}$ 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 σ_1 such that $(N_1, m_1) \stackrel{\sigma_1}{\Longrightarrow} (N_1, m'_1)$ then for all marking m'_2 over P_2 such that $m'_1 \uplus m'_2 \models E$ we must have a firing sequence σ_2 in N_2 with the same observables, meaning: that $(N_2, m_2) \stackrel{\sigma_2}{\Longrightarrow} (N_2, m'_2)$ and $l_1(\sigma_1) = l_2(\sigma_2)$.

- E-abstraction equivalence: $(N_1, m_1) \rhd_E (N_2, m_2)$
  - Iff $(N_1, m_1) \sqsupseteq_E (N_2, m_2)$ and $(N_2, m_2) \sqsupseteq_E (N_1, m_1)$

- **Bounded Model-Checking**: If $(N_1, m_1) \rhd_E (N_2, m_2)$, 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 $(N_1, m_2) \rhd_E (N_2, m_2)$, 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 $m'_1 \in R_{N_1}(m_1)$.



Axioms: Reduction Rules (CONCAT, etc.)

(COMP) **Composability**

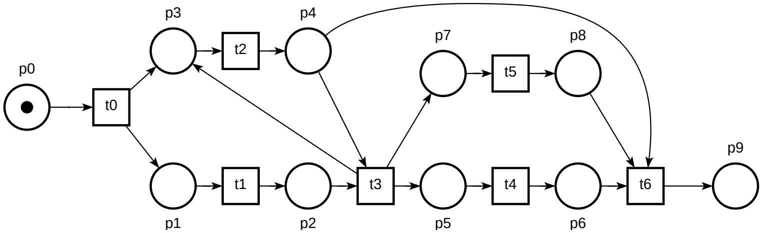
- If $(N_1, m_1) \rhd_E (N_2, m_2)$, then $(N_1, m_1) \| (N_3, m_{,3}) \rhd_E (N_2, m_2) \| (N_3, m_3)$

(TRANS) **Transitivity**

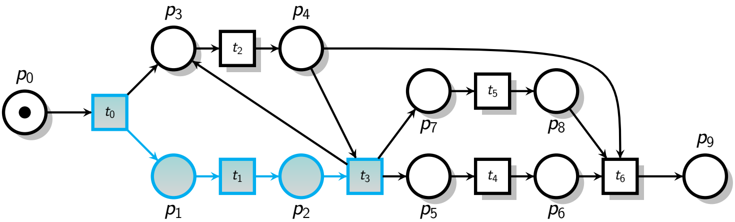
- If $(N_1, m_1) \rhd_E (N_2, m_2)$ and $(N_2, m_2) \rhd_{E'} (N_3, m_3)$, then $(N_1, m_1) \rhd_{E,E'} (N_3, m_3)$.

(RENAME) **Relabeling**

- If $(N_1, m_1) \rhd_E (N_2, m_2)$, then $(N_1[a/b], m_1) \rhd_E (N_2[a/b], m_2)$



Christian Stahl. "*Decomposing Petri net state spaces.*" In 18th German Workshop on Algorithms and Tools for Petri Nets. 2011.

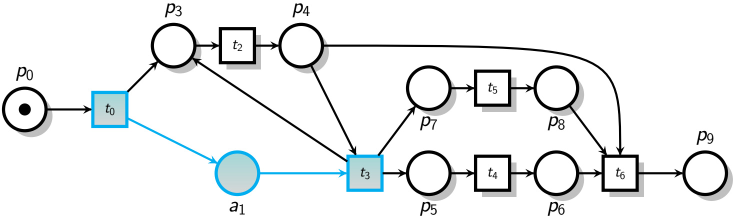


Initial net, S_1, with a pattern for rule (CONCAT) emphasized in blue.

$$E_0 = \emptyset \qquad (1)$$

Net Reduction Example (Step 1)

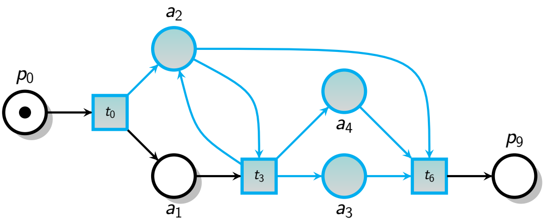
Formalization of Net Reductions



Net S_2, with the result of applying rule (CONCAT) emphasized in blue.

$$E_1 = \left\{ \; a_1 \quad = p_1 + p_2 \right. \qquad (2)$$

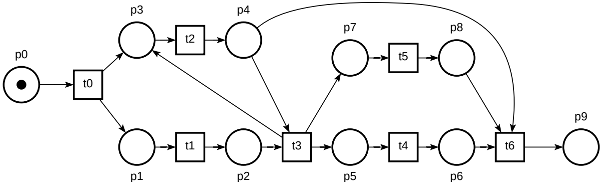
We have: $S_1 \rhd_{E_1} S_2$.



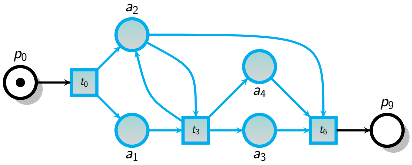
Net S_3, with the result of applying rule (CONCAT) emphasized in blue.

$$E_2 = \begin{cases} a_2 & = p_3 + p_4, \\ a_3 & = p_5 + p_6, \\ a_4 & = p_7 + p_8 \end{cases} \tag{3}$$

We have: $S_2 \rhd_{E_2} S_3$.



Net S_1



Net S_3

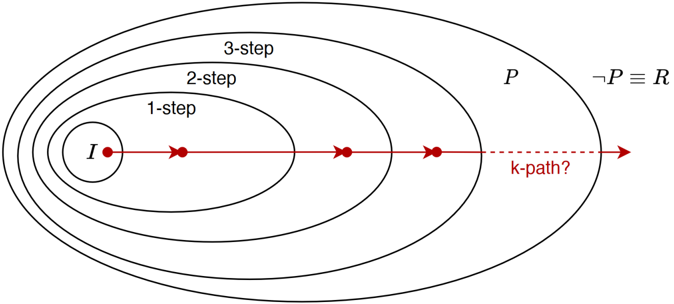
By transitivity, $S_1 \rhd_{E_1,E_2} S_3$

Model Checking Algorithms

- **Bounded Model Checking** (*BMC*): counter-examples
- **Property Directed Reachability** (*IC3*): invariant proof

[Biere et al., 1999]

- Find counter-example violating a property
- Unroll Transitions
- SAT based



BMC method representation

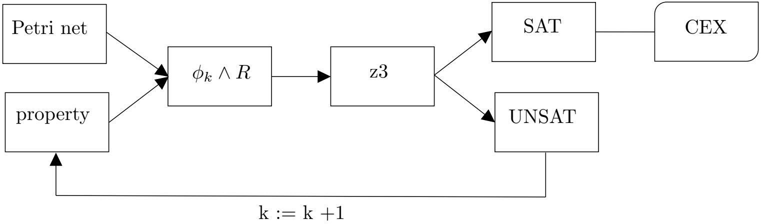
Algorithm adaptation (SMT-based)

- $\mathrm{ENBLD}_t(\vec{x}) \equiv \bigwedge\{(x_i \geq k) \mid k = \mathbf{pre}(t, p_i) > 0\}$
- $\Delta_t(\vec{x}, \vec{x}') \equiv \bigwedge\{(x_i' = x_i + \delta_i) \mid \delta_i = \mathbf{post}(t, p_i) - \mathbf{pre}(t, p_i), 1 \leq i \leq n\}$
- $T(\vec{x}, \vec{x}') \equiv \mathrm{ALLEQ}(\vec{x}, \vec{x}') \vee \bigvee_{t \in T} (\mathrm{ENBLD}_t(\vec{x}) \wedge \Delta_t(\vec{x}, \vec{x}'))$

Lemma: $\underline{m}(\vec{x}) \wedge T(\vec{x}, \vec{x}') \wedge \underline{m}'(\vec{x}')$: m' is at most one-step from m

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

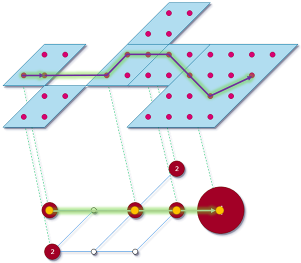
For $k \geq 0$, check $\phi_k(\vec{x}_k) \land \underline{R}(\vec{x}_k)$ until SAT



BMC Algorithm

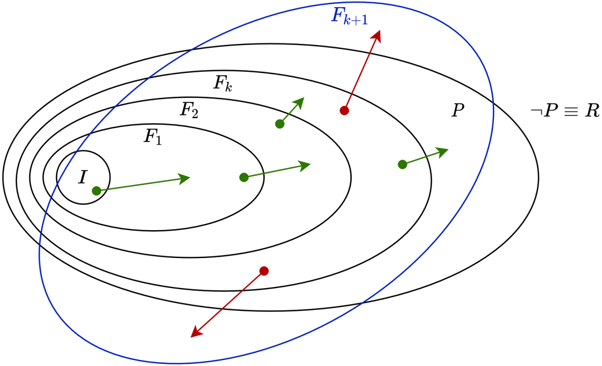
We can find counter-examples to R on N_1 by finding counter-examples to $E \wedge R$ on N_2.
(usually k and $|T|$ are much smaller).

$$\phi_i^r(N_1, m_1)(\vec{x}) \;\equiv\; \phi_i(N_2, m_2)(\vec{y}_i) \wedge E(\vec{x}, \vec{y}_i) \wedge R(\vec{x})$$



[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 ($\approx$ 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>`

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.

Model	# states	Result	Time	T_{Reduced}
AirplaneLD–10	$4.3\,10^4$	CEX	9.17s	0.16s
AirplaneLD–20	$3.1\,10^5$	CEX	50.26s	0.16s
AirplaneLD–∞	∞	CEX	*n.a.*	0.16s
IBM319 (merge...)	$2.4\,10^3$	CEX	$> 200s$	0.14s
IBM319 (callTo...)	$2.4\,10^3$	PROOF	$> 200s$	12.02s

We check if places $P1$ and $P2$ can be marked together in model AirplaneLD (we know it is not possible)[1].

MODEL	# STATES	RESULT	TIME	T_{Reduced}
AirplaneLD–10	$4.3\,10^4$	PROOF	1.50s	0.26s
AirplaneLD–20	$3.1\,10^5$	PROOF	2.51s	0.26s
AirplaneLD–4000	$2.1\,10^{12}$	PROOF	1 680s	0.26s[2]

[1] time to generate the state space of AirplaneLD-4000 with ITS is $> 2\,500s$.
[2] 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 $p_1 \| p_2$ 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

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

Output: Concurrency relation $\mathcal{C}$
$\mathcal{C} \longleftarrow \{\}$;
$m \longleftarrow$ initial marking m_0;
while $\mathcal{C} \longleftarrow \mathcal{C} \cup$ *stepper*$(m, \mathcal{C})$;
do
  parallel
    begin
      if *IC3 proves that we found all concurrent places*
        then return $\mathcal{C}$;
    begin
      if *BMC finds a counter-example* m' *with new concurrent places* **then**
        $m \longleftarrow m'$;
        continue;

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)

Conclusion

- 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)

Init	Step 1	Step 2
$\underline{m_0}(\overrightarrow{x_0})$	$\underline{m_0}(\overrightarrow{x_0})$	$\underline{m_0}(\overrightarrow{x_0})$
$\underline{R}(\overrightarrow{x_0})$	$T(\overrightarrow{x_0}, \overrightarrow{x_1})$	$T(\overrightarrow{x_0}, \overrightarrow{x_1})$
	$\underline{R}(\overrightarrow{x_1})$	$T(\overrightarrow{x_1}, \overrightarrow{x_2})$
		$\underline{R}(\overrightarrow{x_2})$

Assertion stack

Init	Step 1	Step 2
$\underline{R}(\vec{x})$	$\underline{R}(\vec{x})$	$\underline{R}(\vec{x})$
$\underline{m_0}(\overrightarrow{y_0})$	$\underline{m_0}(\overrightarrow{y_0})$	$\underline{m_0}(\overrightarrow{y_0})$
$E(\vec{x}, \overrightarrow{y_0})$	$T(\overrightarrow{y_0}, \overrightarrow{y_1})$	$T(\overrightarrow{y_0}, \overrightarrow{y_1})$
	$E(\vec{x}, \overrightarrow{y_1})$	$T(\overrightarrow{y_1}, \overrightarrow{y_2})$
		$E(\vec{x}, \overrightarrow{y_2})$

Assertion stack with reductions

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 $i \in 0 \ldots k+1$. $\underline{F_i}(\vec{x}) \wedge T(\vec{x}, \vec{x}') \Rightarrow \underline{F_{i+1}}(\vec{x}')$

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