LTL Model Checking for Modular Petri Nets

T. Latvala and M. Mäkelä
Presented by Deian Tabakov

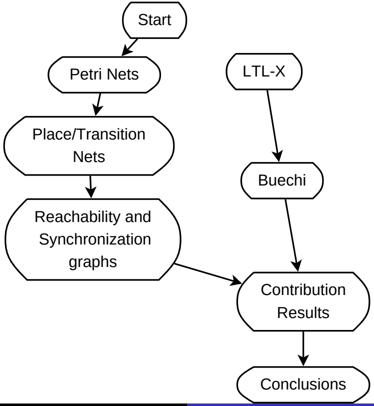
September 25, 2006

Executive Summary

- This paper presents...
  - A method for LTL-X model checking of modular Petri Nets
  - An implementation and experimental comparison

Executive Summary

- This paper presents...
  - A method for LTL-X model checking of modular Petri Nets
  - An implementation and experimental comparison
- Roadmap



Petri Nets

Figure: C. A. Petri

- *Kommunikation mit Automaten*, 1962
  - "Description, in a uniform and exact manner, of as great as possible a number of phenomena related to **information transmission and information transformation**..."
  - Regulated flows (information, products...)
- P.N.H.N.C.W.P.D.

Petri Nets

- Mathematical representation of discrete distributed systems
- Causal dependencies and independencies represented explicitly
- Partial order relation of concurrency, unsynchronized functions...
- Different levels of abstraction
- Formal system $\longrightarrow$ verify properties

Petri Nets

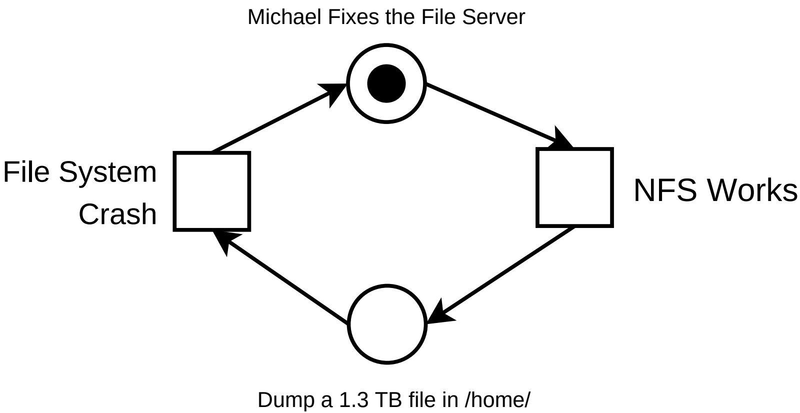


Figure: Petri Nets: Events, Conditions, Tokens

Petri Nets

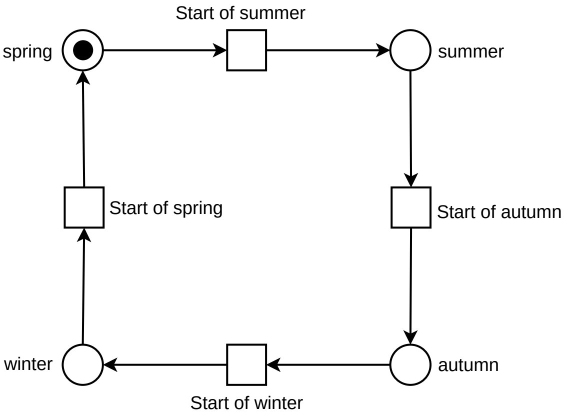


Figure: Petri Nets: Pre- and Post-conditions, Case

Petri Nets

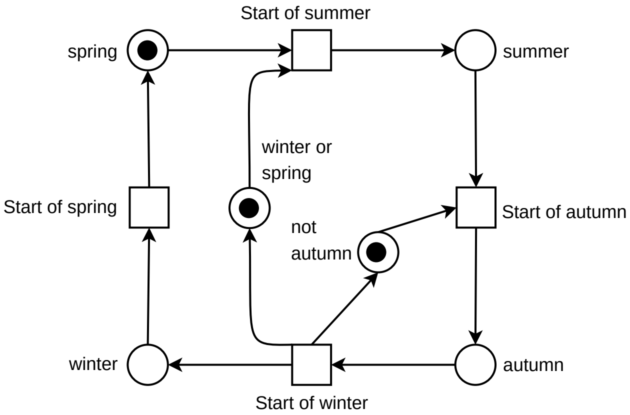


Figure: Petri Nets: Multiple Preconditions and Postconditions

Petri Nets

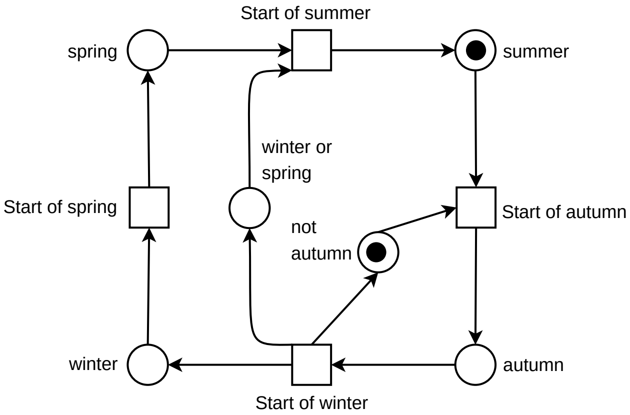


Figure: Petri Nets: Multiple Preconditions and Postconditions

Petri Nets

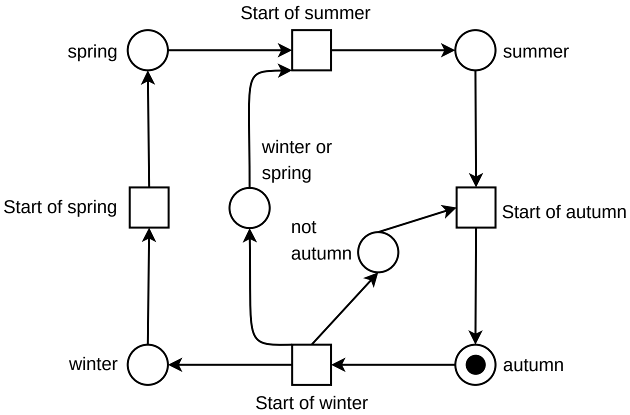


Figure: Petri Nets: Multiple Preconditions and Postconditions

Petri Nets

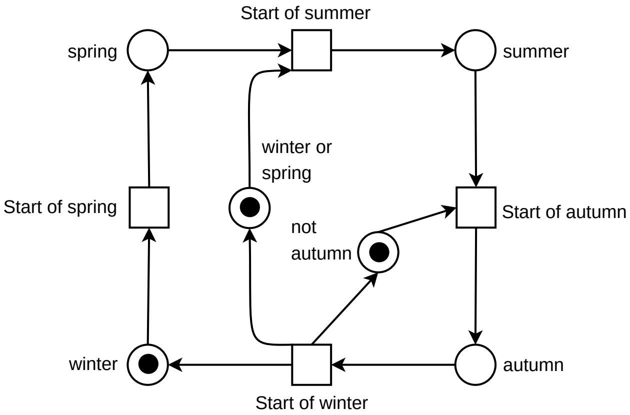


Figure: Petri Nets: Multiple Preconditions and Postconditions

Petri Nets

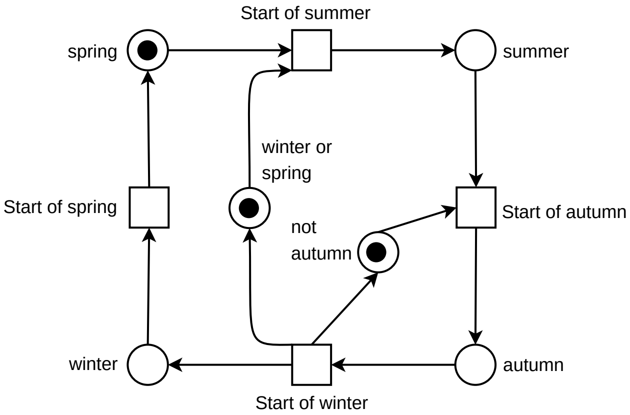


Figure: Petri Nets: Multiple Preconditions and Postconditions

Petri Nets

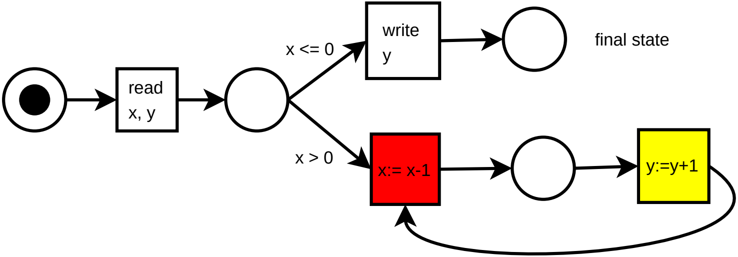


Figure: Petri Nets: Concurrency

Petri Nets

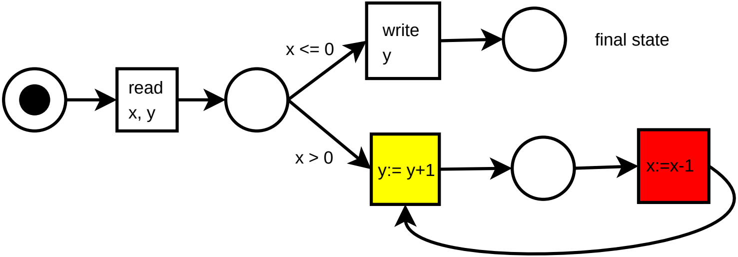


Figure: Petri Nets: Concurrency

Petri Nets

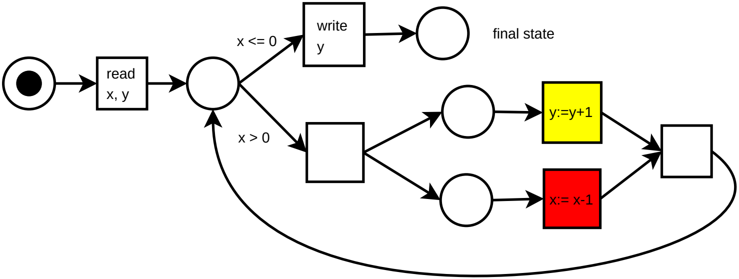


Figure: Petri Nets: Concurrency

Petri Nets

Definition (Net)

A triple $N = (P, T; F)$ is called a *net* iff

- P and T are disjoint sets
- $F \subseteq (P \times T) \cup (T \times P)$ is a binary relation (the *flow relation* of N).

Definition (Pre- and Post-sets)

Let N be a net. For $x \in N$

- $\cdot x = \{y | yFx\}$ is called the *preset* of x
- $x \cdot = \{y | xFy\}$ is called the *postset* of x

Producer-Consumer Systems

- Modelling manufacturing processes
- Combine a set of conditions "(places $p_1, p_2, \ldots, p_n$ are used)" into one object.
- Conditions $\longrightarrow$ *Places*
- Events $\longrightarrow$ *Transitions*
- Capacities, weights

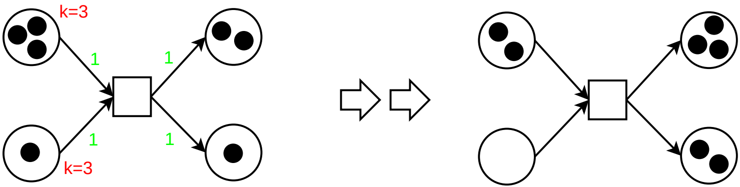


Figure: Producer-Consumer Systems: Firing Rules

Producer-Consumer Systems

A model of a bakery with two consumers:

- The baker generates 3 cookies per time step
- The display case can hold at most 5 cookies
- At most one consumer can access cooling rack
- Each consumer buys cookies in batches of 2
- The production steps of the baker are counted

Producer-Consumer Systems

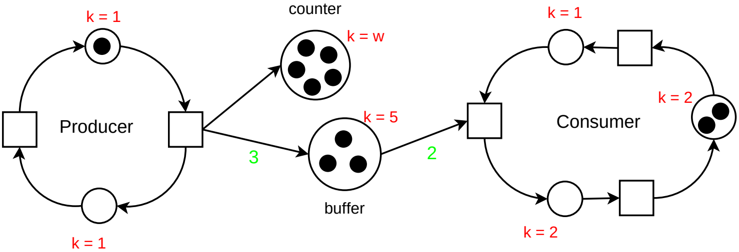


Figure: The Bakery Model

Formal Definition

Definition (Place/Transition Nets)

A tuple $N = (P, T; F, K, M, W)$ is a *place/transition net* iff

- (P, T; F) is a finite net
- $K : P \rightarrow \mathbb{N} \cup \{\omega\}$ gives a *capacity* for each place.
- $W : F \rightarrow \mathbb{N} \backslash \{0\}$ attaches a *weight* to each arc.
- $M : P \rightarrow \mathbb{N} \cup \{\omega\}$ is *initial marking* (must respect capacities).

Definition (Latvala and Mäkelä)

A PT-net is a tuple $N = (P, T, W, M_0)$ where

- P and T are finitie, $P \cap T = \emptyset$
- $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is the arc weight function
- $M_0 : P \rightarrow \mathbb{N}$ is the initial marking

The Life of a PT-net

An *execution* of a net is an infinite sequence of markings $M_0 M_1 \ldots M_n$:

- Start at M_0
- Transition to some marking M_1 via *enabled* transition
- Repeat until no transitions are enabled, or forever
- Stutter (if necessary)

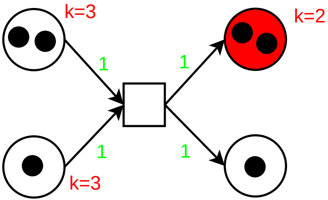


Figure: Destination Full

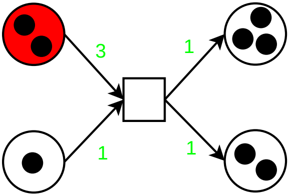
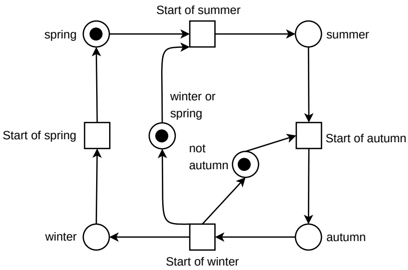


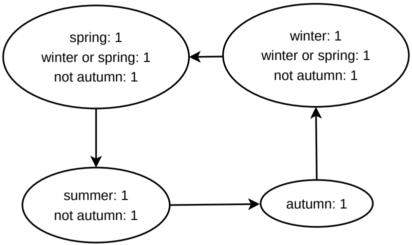
Figure: Source not sufficient

The Life of a PT-net

Definition (Reachability Graph)

- Vertices correspond to all reachable markings
- An edge connects M_i and M_j if there is an enabled transition between M_i and M_j.



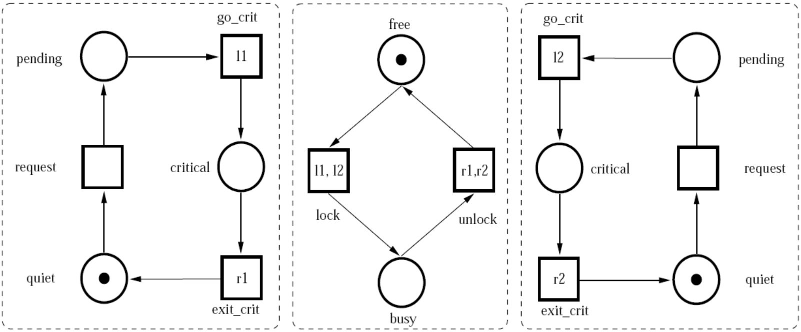


Modular PT-nets

- Use structure to reduce complexity
- Analyze modules separately
- Synchronize when needed via shared transitions

Modular PT-nets

- Use structure to reduce complexity
- Analyze modules separately
- Synchronize when needed via shared transitions

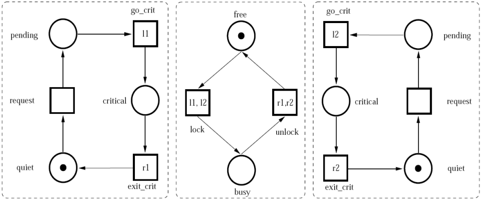


Modular PT-nets

Definition (Modular PT-net)

A *Modular PT-Net* is a tuple $\Sigma = (S, TF)$ where

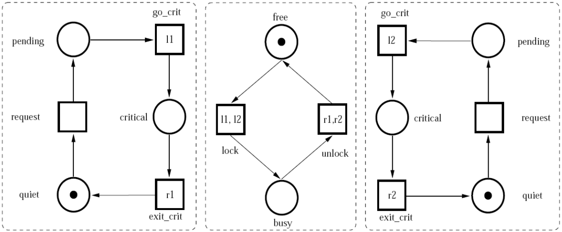
- S is a finite set of modules
  - Each module is a $PT - net$
  - The nodes are pairwise disjoint
- $TF \subseteq 2^{\text{All Trans}}$ is a finite set of *transition fusion sets* (Each module contributes up to 1 transition to a fusion transition.)

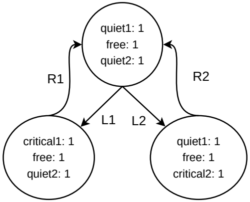


Modular Analysis

- Christensen & Petrucci '00, *"Modular Analysis of Petri Nets"*
- Key idea: "reachability graph for transition fusions".
  - The *Synchronisation graph* (V, E)
  - Vertices are markings
  - Start with the initial marking M_0
  - If $M_i \in V$ and M_j can be reached by firing a transition fusion tf, then $M_j \in V$ and $(M_i, tf, M_j) \in E$

Modular Analysis





LTL-X

Recall from Sumit's talk:

Definition (LTL)

- $\varphi \in AP$ is an LTL formula
- If φ and ψ are LTL formulas, so are $\neg\varphi, \mathbf{X}\varphi, \psi\mathbf{U}\varphi$, and $\psi \vee \varphi$.
- Shorthand notation: $\mathbf{F}\varphi \equiv \top\mathbf{U}\varphi$ and $\mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$.

LTL-X

Recall from Sumit's talk:

Definition (LTL)

- $\varphi \in AP$ is an LTL formula
- If φ and ψ are LTL formulas, so are $\neg\varphi, \mathbf{X}\varphi, \psi\mathbf{U}\varphi$, and $\psi \vee \varphi$.
- Shorthand notation: $\mathbf{F}\varphi \equiv \top\mathbf{U}\varphi$ and $\mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$.

Today:

Definition (LTL-X)

LTL-X is LTL without the **X** operator.

Intuition: Cannot distinguish between modules that only differ in their "internal" transitions.

Büchi Automata

Recall: every LTL formula has a corresponding Büchi automaton that accepts the same language.

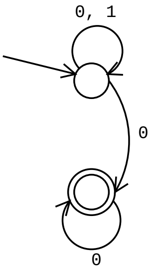


Figure: Finitely many 1's

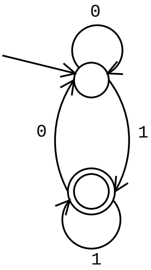


Figure: Infinitely many 1's

Standard Model Checking Approach

- Standard program verification: does program P satisfy property φ?
  - Abstract P and φ to Büchi automata
  - $P \models \varphi \Leftrightarrow L(P) \subseteq L(A_\varphi)$
  - Equivalently: $A_P \cap \overline{A_\varphi} = \emptyset$
- If P is a Petri Net: Synchronize A_φ with every move of P.
- *Key idea*: Synchronize with the visible transitions.

Main Theoretical Contribution

Definition (Product Automaton)

- Two machines in parallel (The net N and $\overline{A_\varphi}$)
- Both move on visible transitions
- Only the net moves on invisible transitions

Theorem

Any execution that violates the LTL-X specification will induce either an illegal livelock or an illegal ω-execution, which will also show up in the product automaton.

Implementation

- Experimental Setup
  - Implemented on top of reachability analyzer *Maria*
  - Compare with Maria and *PROD* (has P.O. reductions)
- Results
  - *Faster* for loosely coupled models
  - *Faster* for models with lots of synchronisation (Leader Election)
  - *Questionable* for some models (Sliding Window protocol)
  - *Faster* than PROD (but not too much)

Experiemental Results

Sys-	Flat state space $G = (V, E, v_0)$				Modular $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{v}_0)$				
tem	$\lvert V\rvert$	$\lvert E\rvert$	product	time/s	$\lvert\mathbf{V}\rvert$	$\lvert\mathbf{E}\rvert$	product	time/s	ψ
AGV	30,965,760	216,489,984	N/A	N/A	87,480	464,616	87,492	27.3	$\mathbf{GF}\varphi$
SW_4	6,360	16,608	14,857	1.3	4,456	16,016	8,889	2.6	$\mathbf{GF}\varphi$
SW_5	24,270	68,760	52,891	5.8	16,930	72,660	31,991	13.1	$\mathbf{GF}\varphi$
SW_6	82,884	248,400	169,645	20.6	57,564	286,488	103,477	118	$\mathbf{GF}\varphi$
LE_3	159	303	314	0.0	35	65	68	0.0	$\mathbf{FG}\varphi$
LE_4	716	1,851	1,428	0.2	92	229	182	0.1	$\mathbf{FG}\varphi$
LE_5	3,432	11,198	6,860	1.3	253	802	504	0.2	$\mathbf{FG}\varphi$
LE_6	16,792	66,043	33,580	8.0	715	2,748	1,428	0.8	$\mathbf{FG}\varphi$
LE_7	82,667	380,267	165,330	49.3	2,043	9,212	4,084	2.9	$\mathbf{FG}\varphi$
LE_8	407,699	2,146,965	815,394	295	5,865	30,308	11,728	10.2	$\mathbf{FG}\varphi$

Figure: Against Maria: Number of states in product

Experiemental Results

System	Flat state space $G = (V, E, v_0)$				Modular $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{v}_0)$				ψ
	$\lvert V\rvert$	$\lvert E\rvert$	product	time/s	$\lvert\mathbf{V}\rvert$	$\lvert\mathbf{E}\rvert$	product	time/s	
AGV	30,965,760	216,489,984	N/A	N/A	87,480	464,616	87,492	27.3	$\mathbf{GF}\varphi$
SW_4	6,360	16,608	14,857	1.3	4,456	16,016	8,889	2.6	$\mathbf{GF}\varphi$
SW_5	24,270	68,760	52,891	5.8	16,930	72,660	31,991	13.1	$\mathbf{GF}\varphi$
SW_6	82,884	248,400	169,645	20.6	57,564	286,488	103,477	118	$\mathbf{GF}\varphi$
LE_3	159	303	314	0.0	35	65	68	0.0	$\mathbf{FG}\varphi$
LE_4	716	1,851	1,428	0.2	92	229	182	0.1	$\mathbf{FG}\varphi$
LE_5	3,432	11,198	6,860	1.3	253	802	504	0.2	$\mathbf{FG}\varphi$
LE_6	16,792	66,043	33,580	8.0	715	2,748	1,428	0.8	$\mathbf{FG}\varphi$
LE_7	82,667	380,267	165,330	49.3	2,043	9,212	4,084	2.9	$\mathbf{FG}\varphi$
LE_8	407,699	2,146,965	815,394	295	5,865	30,308	11,728	10.2	$\mathbf{FG}\varphi$

Figure: Against Maria: Time used

Experiemental Results

System	PROD $G = (V, E, v_0)$ $\lvert V\rvert$	$\lvert E\rvert$	product	time/s	Modular (Maria) $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{v}_0)$ $\lvert\mathbf{V}\rvert$	$\lvert\mathbf{E}\rvert$	product	time/s	ψ
$SW_{2,2}$	8,384	13,388	17,622	8.0	7,376	48,860	9,709	8.0	$\mathbf{GF}\varphi$
$SW_{3,2}$	131,555	198,466	270,142	245	86,995	802,650	101,551	148	$\mathbf{GF}\varphi$
$SW_{3,3}$	422,484	590,298	859,724	969	267,192	2,885,022	302,551	757	$\mathbf{GF}\varphi$
$SW_{4,2}$	1,434,750	2,056,176	2,914,484	7556	762,870	9,379,788	836,275	3031	$\mathbf{GF}\varphi$

Figure: Against PROD: Number of states in product

Experiemental Results

System	PROD $G = (V, E, v_0)$ $\|V\|$	$\|E\|$	product	time/s	Modular (Maria) $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{v}_0)$ $\|\mathbf{V}\|$	$\|\mathbf{E}\|$	product	time/s	ψ
$SW_{2,2}$	8,384	13,388	17,622	8.0	7,376	48,860	9,709	8.0	$\mathbf{GF}\varphi$
$SW_{3,2}$	131,555	198,466	270,142	245	86,995	802,650	101,551	148	$\mathbf{GF}\varphi$
$SW_{3,3}$	422,484	590,298	859,724	969	267,192	2,885,022	302,551	757	$\mathbf{GF}\varphi$
$SW_{4,2}$	1,434,750	2,056,176	2,914,484	7556	762,870	9,379,788	836,275	3031	$\mathbf{GF}\varphi$

Figure: Against PROD: Time used

Summary and Conclusions

- LTL-X model checking on the synchronization graph of modular Petri Nets
- Implementation available
- Method relies on efficient modular analysis