LTL Model Checking for Modular Petri Nets

Timo Latvala and Marko Mäkelä

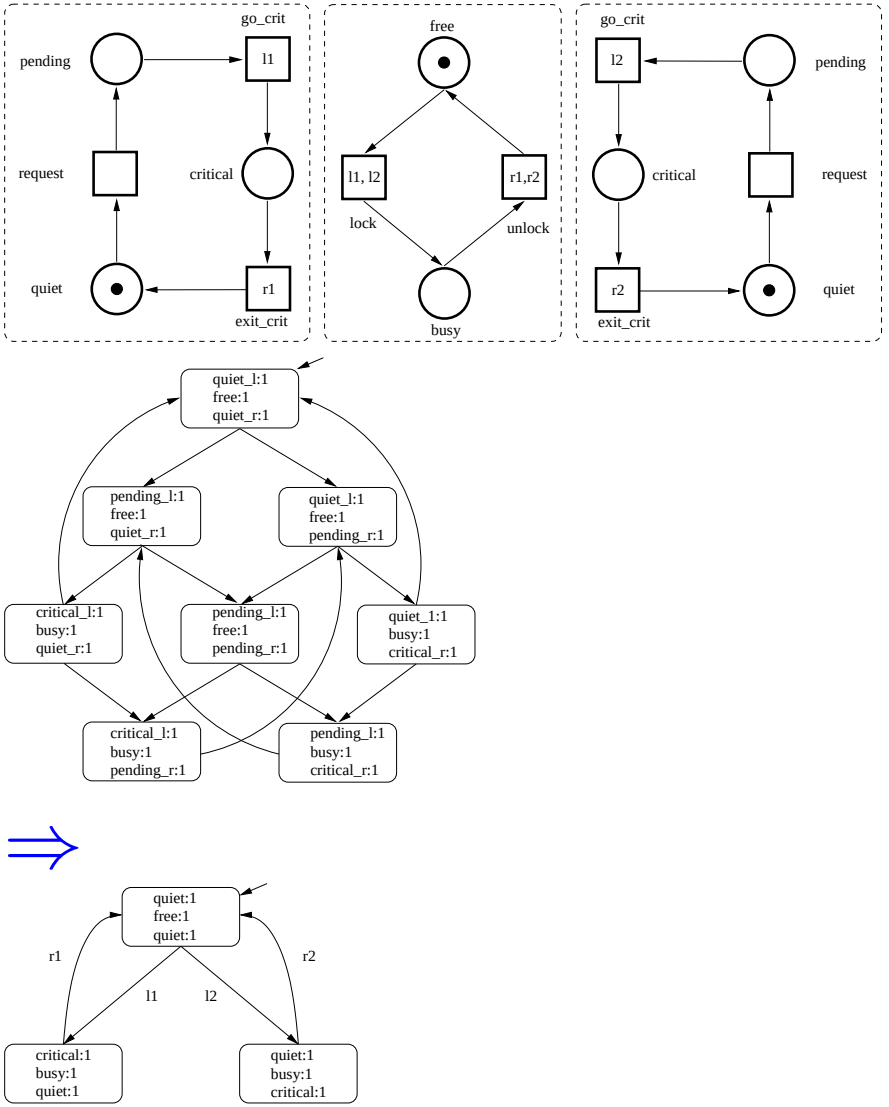
Timo.Latvala@hut.fi

Laboratory for Theoretical Computer Science

Helsinki University of Technology

Introduction

- Modular Petri Nets [Christensen & Petrucci] add structure
- Communication through shared transitions
- The structure can be utilised in reachability analysis $\Rightarrow$ modular analysis
- What about model checking?

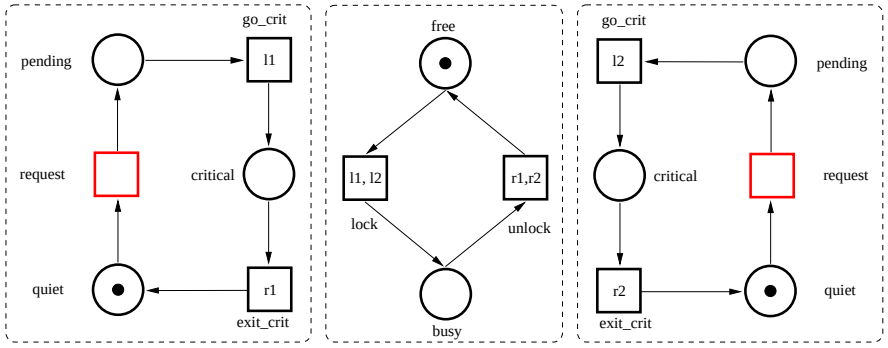


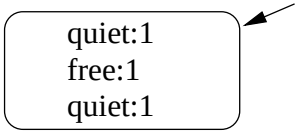
Modular Analysis

Modular Analysis

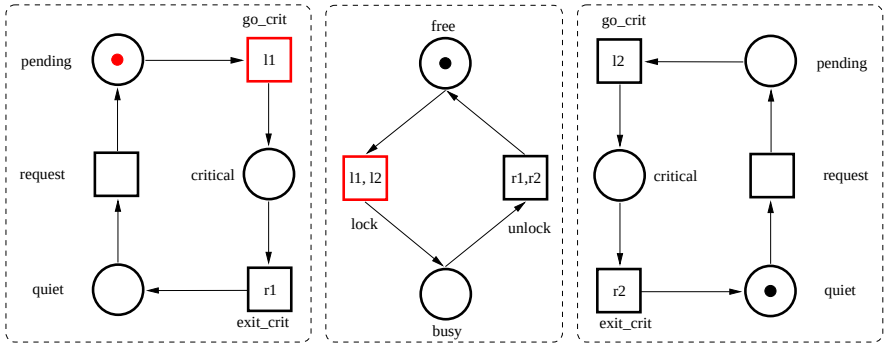
- A technique for producing a reduced reachability graph
- Idea: only transitions which synchronise are important, internal transition are hidden
- Only states reached after a synchronising transition are stored
- Result: synchronisation graph

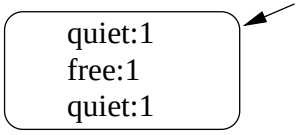
Example


go_crit
pending
l1
request
critical
quiet
r1
exit_crit
free
l1, l2
r1,r2
lock
unlock
busy
go_crit
l2
pending
critical
request
r2
quiet
exit_crit

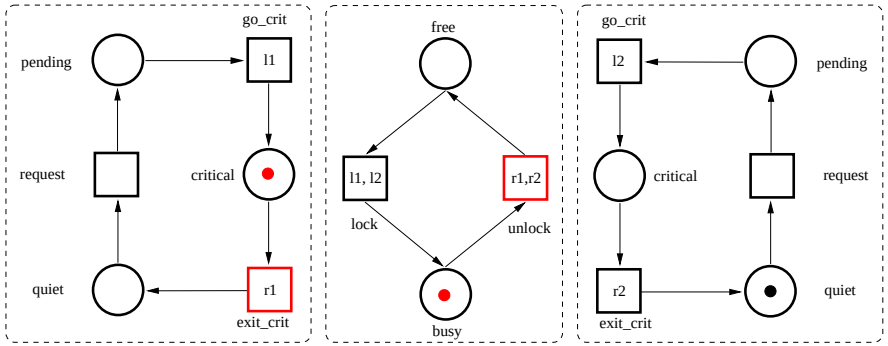

quiet:1
free:1
quiet:1

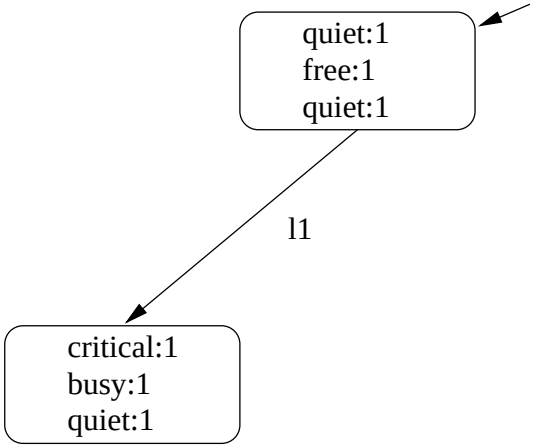
Example


go_crit
pending
l1
request
critical
quiet
r1
exit_crit
free
l1, l2
r1,r2
lock
unlock
busy
go_crit
l2
pending
critical
request
r2
quiet
exit_crit

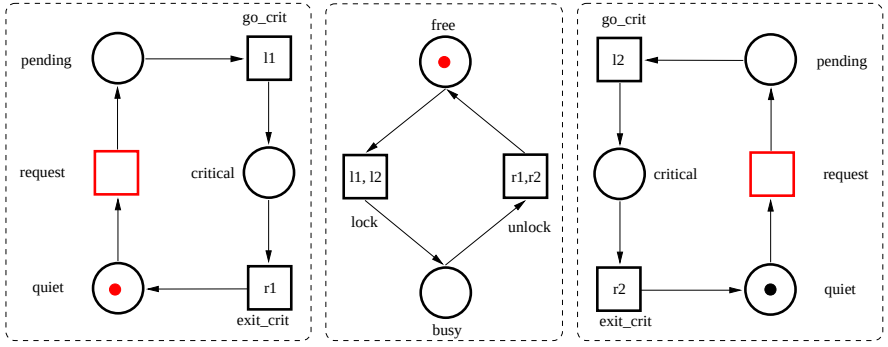

quiet:1
free:1
quiet:1

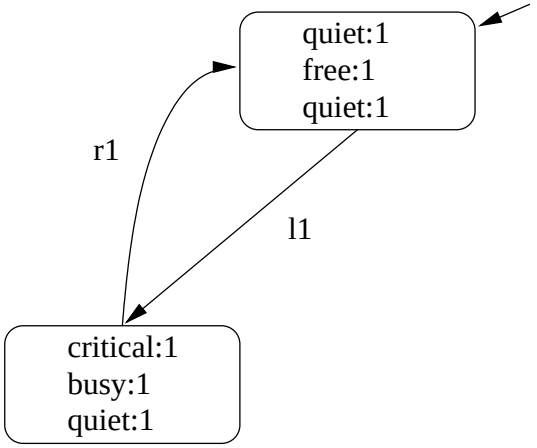
Example


go_crit
pending
l1
request
critical
quiet
r1
exit_crit
free
l1, l2
r1,r2
lock
unlock
busy
go_crit
l2
pending
critical
request
r2
quiet
exit_crit

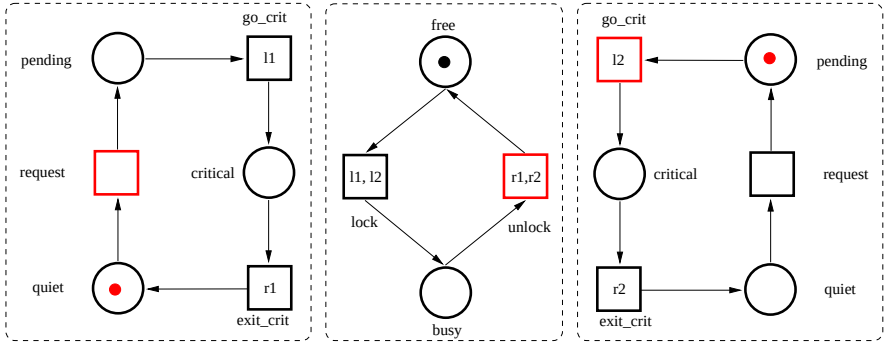

quiet:1
free:1
quiet:1
l1
critical:1
busy:1
quiet:1

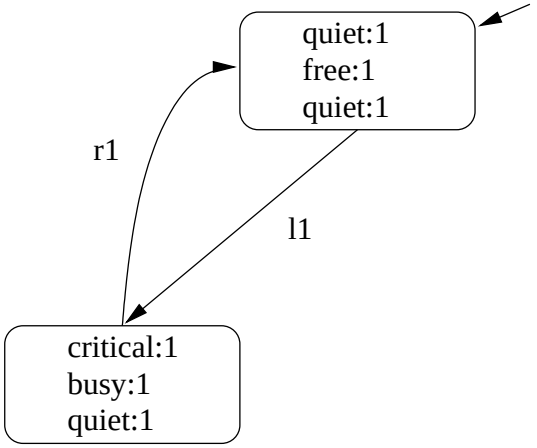
Example


go_crit
pending
l1
request
critical
quiet
r1
exit_crit
free
l1, l2
r1,r2
lock
unlock
busy
go_crit
l2
pending
critical
request
r2
quiet
exit_crit

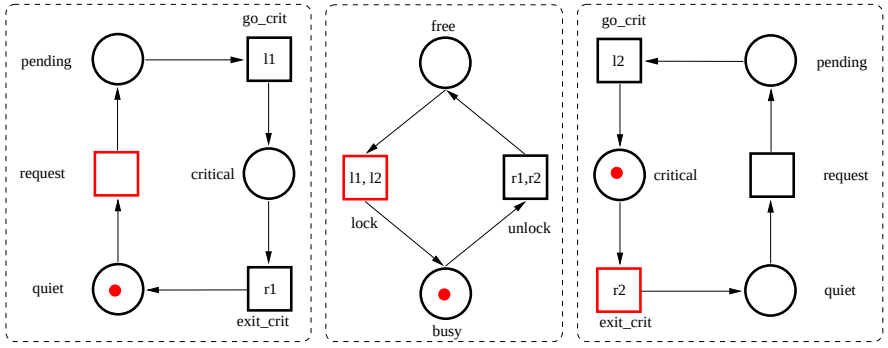

quiet:1
free:1
quiet:1
r1
l1
critical:1
busy:1
quiet:1

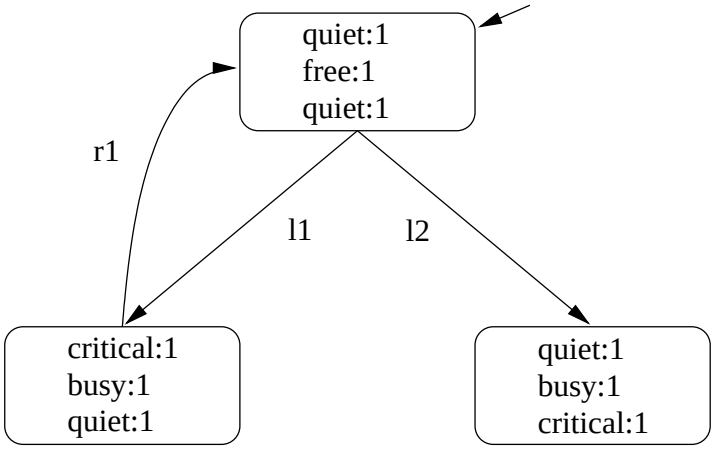
Example


go_crit
pending
l1
request
critical
quiet
r1
exit_crit
free
l1, l2
r1,r2
lock
unlock
busy
go_crit
l2
pending
critical
request
r2
quiet
exit_crit

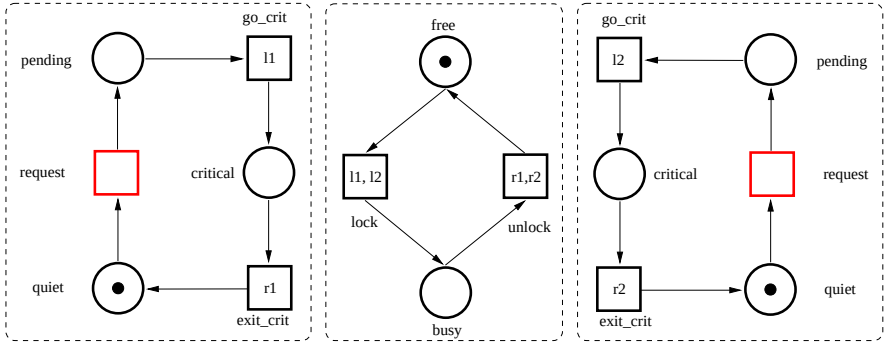

quiet:1
free:1
quiet:1
r1
l1
critical:1
busy:1
quiet:1

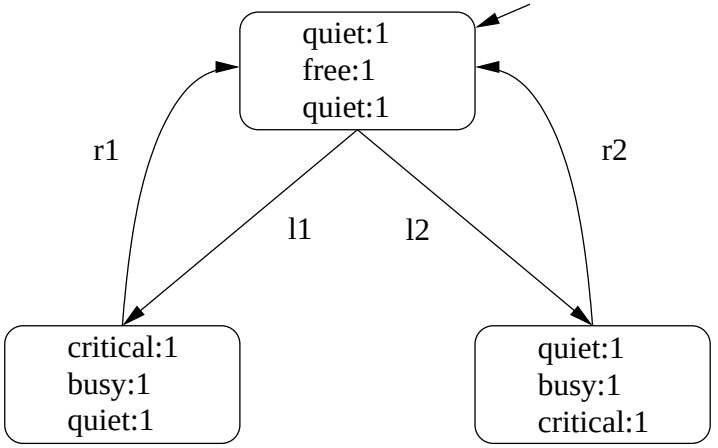
Example


go_crit
pending
l1
request
critical
quiet
r1
exit_crit
free
l1, l2
r1,r2
lock
unlock
busy
go_crit
l2
pending
critical
request
r2
quiet
exit_crit

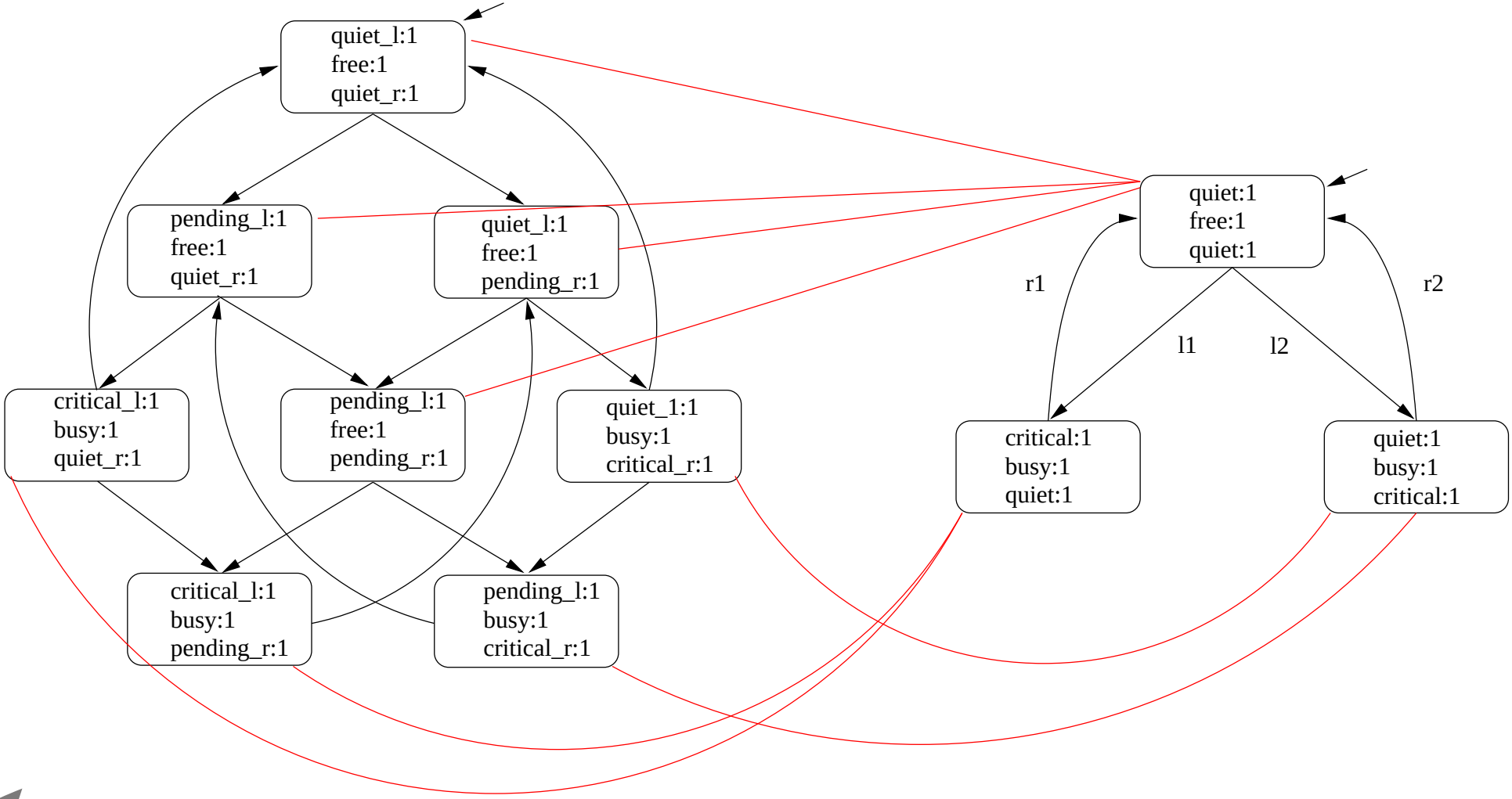

quiet:1
free:1
quiet:1
r1
l1
l2
critical:1
busy:1
quiet:1
quiet:1
busy:1
critical:1

Example


go_crit
pending
l1
request
critical
quiet
r1
exit_crit
free
l1, l2
r1,r2
lock
unlock
busy
go_crit
l2
pending
critical
request
r2
quiet
exit_crit


quiet:1
free:1
quiet:1
r1
r2
l1
l2
critical:1
busy:1
quiet:1
quiet:1
busy:1
critical:1

RG and Synchronisation Graph



Model Checking

Model Checking

- Technique for proving properties specified in temporal logic
- The automata theoretic approach to model checking LTL:
  1. Translate negation of the LTL formula ψ into a Büchi automaton $\mathcal{A}_{\neg\psi}$.
  2. Compute synchronisation with reachability graph $RG \times \mathcal{A}_{\neg\psi}$
  3. Check emptiness $\mathcal{L}\left(RG \times \mathcal{A}_{\neg\psi}\right) \neq \emptyset$?

MC and Modular Analysis

- Standard MC synchronises $\mathcal{A}_{\neg\psi}$ with every move of the net
- Synchronisation graph hides information
- Making all moves visible $\Rightarrow$ full reachability graph

MC and Modular Analysis

- Standard MC synchronises $\mathcal{A}_{\neg\psi}$ with every move of the net
- Synchronisation graph hides information
- Making all moves visible $\Rightarrow$ full reachability graph
- How do we adapt model checking?

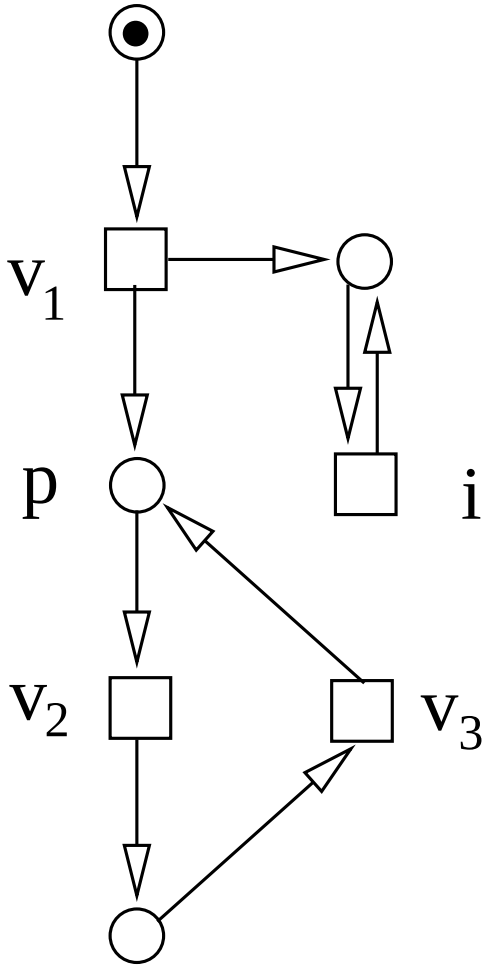
Solution: Preliminaries

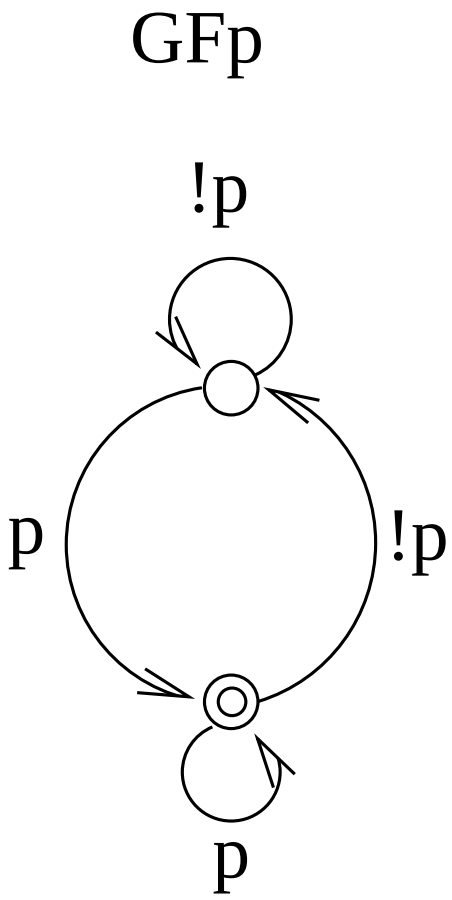
- We want standard interleaving semantics
- Synchronisation graph hides stuttering $\Rightarrow$ We use LTL-X
- A place p of the net is visible if it is mentioned in ψ
- The visible transitions are transitions connected to a visible place
- We require that all visible transitions are synchronising transitions

Solution: Idea

- Since synchronisation graph hides internal moves, synchronise Büchi automaton only with visible moves
- [Esparza & Heljanko, SPIN 2001]: The net Σ is correct w.r.t. ψ iff the visible synchronisation has no illegal ω-traces and no illegal livelocks
- Illegal ω-trace: an accepting infinite sequence with infinitely many visible moves
- Illegal livelocks: an accepting infinite sequence with finitely many visible moves

Livelocks and ω-traces





Solution: New Synchronisation

Büchi automaton: $\mathcal{A}_{\neg\psi} = (Q, A, \rho, q_0, Q_F)$

Synchronisation graph: $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{v}_0)$

- Given a state (M, q) in the product, with t as a possible move in the net, and a in the automaton:
  - If t visible and $eval(M) \in a$, both the net and the automaton synchronise
  - If t is visible only the system moves
- $F = \{(M, q) \in V_p \| \; q \in Q_F\}$
- $I = \{(M, q) \in V_p \| \; \mathcal{A}^{q}_{\neg\psi}$ accepts $eval(M)^{\omega}\}$

Main Result

Theorem 1.
Given a modular net Σ and a Büchi automaton $\mathcal{A}_{\neg\psi}$, $\Sigma \not\models \psi$ iff the product of the automaton and the synchronisation graph of the net has an illegal ω-trace or an illegal livelock.

Algorithm

proc model check(V_p, E_p, p_0, F, I) **begin**
  for all states $(M, q) \in V_p$ **do**
    if $(M, q) \in I$ **then**
      if $\exists$ loop of invisible transitions from M **then**
        return “illegal livelock found”
    fi
    if $(M, q) \in F$ **then**
      if (M, q) is reachable from (M, q) **then**
        return “illegal ω-execution found”
    fi
  od
end

Benchmarks

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

Benchmarks II

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

Future Work

- Refine concept of visibility
- Combine with partial order reductions (stubborn sets)
- Generalise to nested modular Petri nets