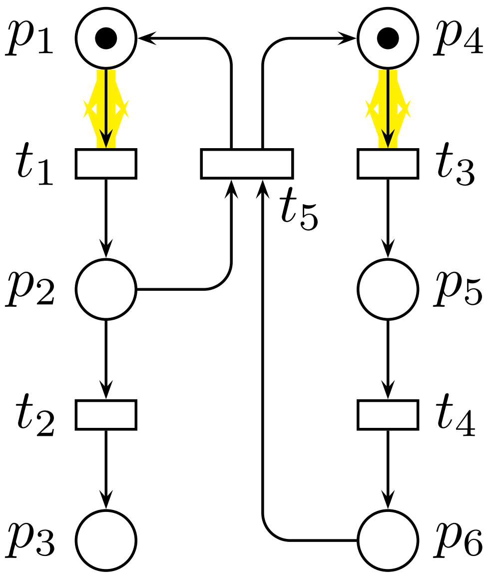
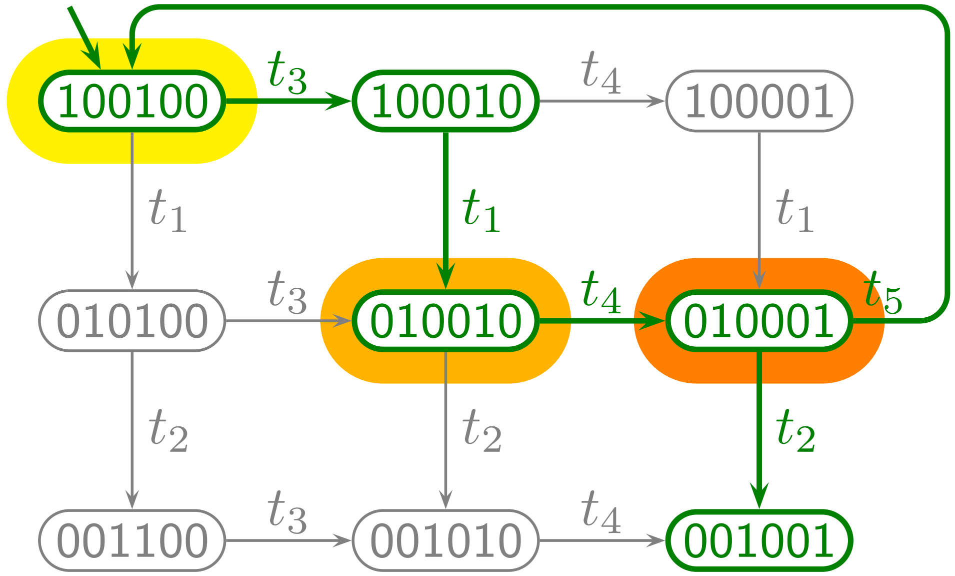
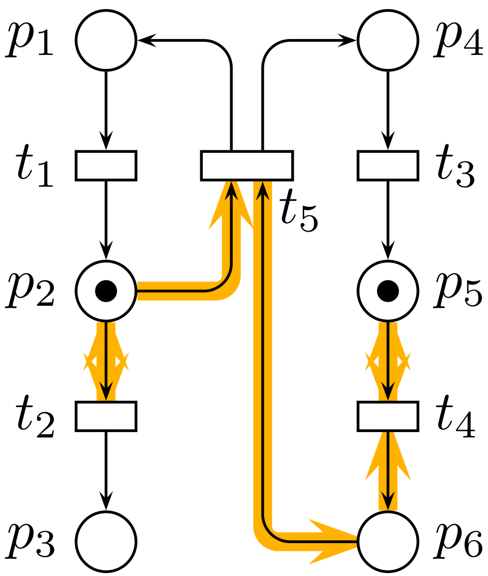
Stop It, and Be Stubborn!

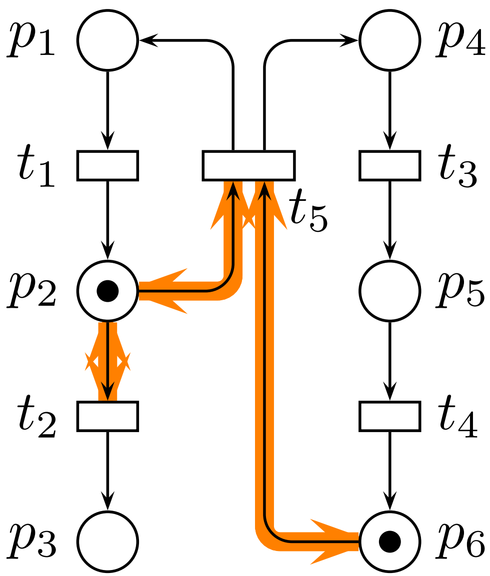
Antti Valmari
Tampere University of Technology
Department of Mathematics

1 Stubborn Sets









2 Static and Dynamic

Algorithm deletion → Static def.

Algorithm closure → Static def.

$\exists t : M[t\rangle \Rightarrow \exists t \in \mathit{Stubb}(M) : M[t\rangle$
$M[t\rangle \Rightarrow (\bullet t)\bullet \subseteq \mathit{Stubb}(M)$
$\neg M[t\rangle \Rightarrow \exists p \in \bullet t : M(p) < W(p,t) \wedge \bullet p \subseteq \mathit{Stubb}(M)$

Static def. → Dynamic def. → Property

Dynamic def.: D0, D1, D2

Property: deadlocks

Static def. → Dynamic def.

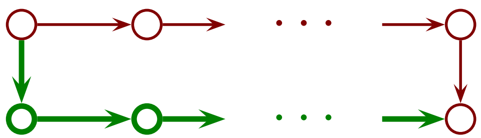
$\ldots$
$\ldots \forall p \in \bullet t : \forall t' \notin \mathit{Stubb}(M) : \min(W(t,p), W(t',p)) \geq \ldots$
$\ldots W(p,t') \geq \min(W(t',p), W(p,t))$

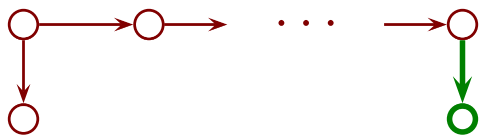
Algorithm strong components → Static def.

better static def.s,
better algorithms
$\Rightarrow$ better reduction

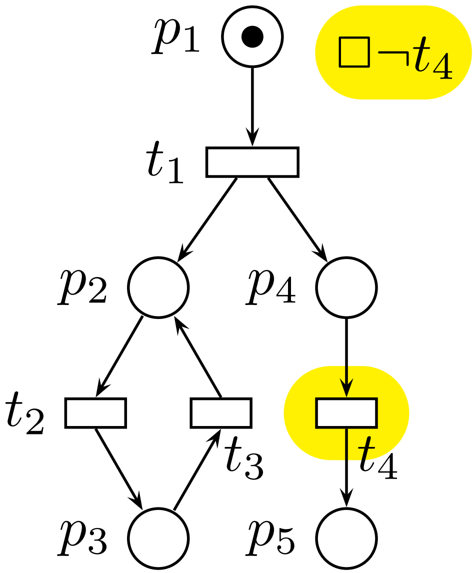
D0: if $\exists t : s_0 -t\rightarrow$, then $\exists t \in \mathit{Stubb}(s_0) : s_0 -t\rightarrow$

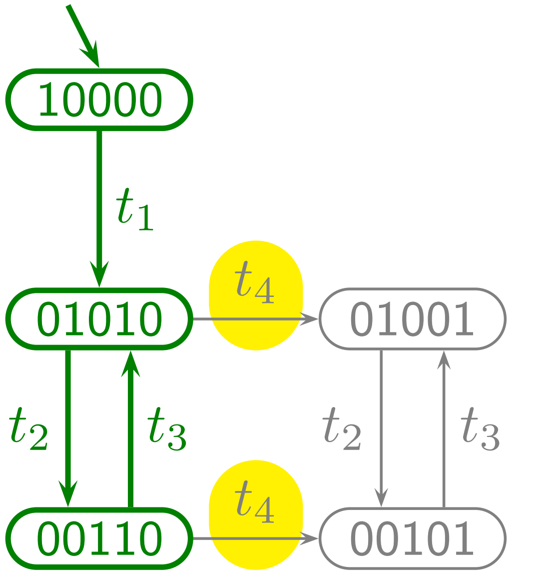
If red, then green $\quad \rightarrow \notin \mathit{Stubb}(s_0) \quad \downarrow \in \mathit{Stubb}(s_0)$

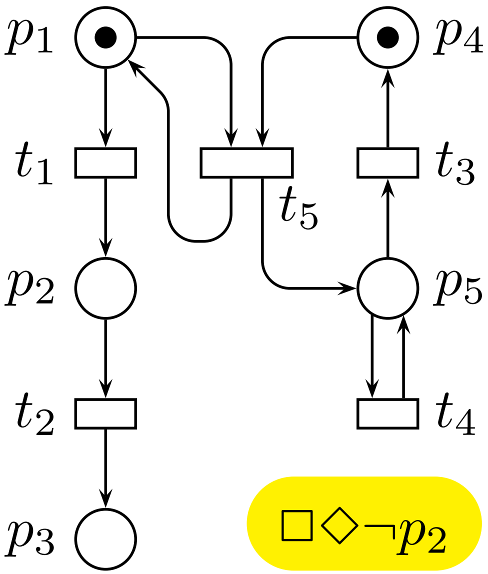
D1: 

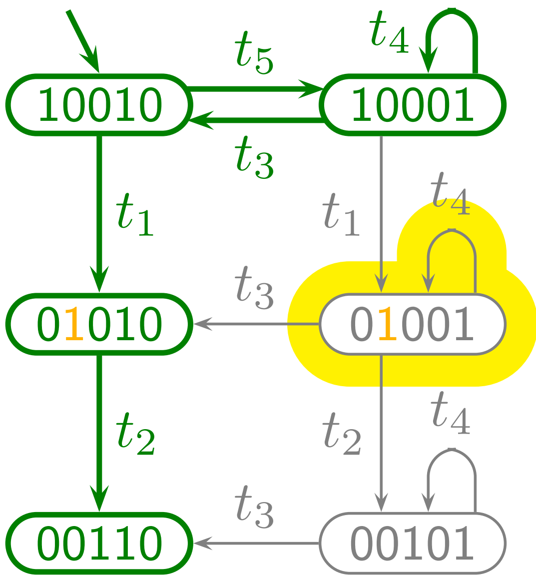
D2: 

3 The Ignoring Problem









4 Method Explosion

	dead-locks	∃ inf. execu.	EF t, AG EF t	traces	*Sfail*	CSP	$\text{LTL}_{-\times}$, CFFD	determ. $\text{CTL}^*_{-\times}$	$\text{CTL}^*_{-\times}$ obs.eq.
D1	•	•	•	•	•	•	•	•	○
D2’	•	•	•	•	•	•	•	•	○
D3		•				•	•	(○)	○
V				•	•	•	•	○	○
L1					•	•	•	○	○
S			•	•				•	•
L2							•	○	○
B								•	○
NB									•

○ = condition follows from others

D2’ is a variant of D2 that implies D0:

Stubb(s) contains at least one enabled t that satisfies D2.

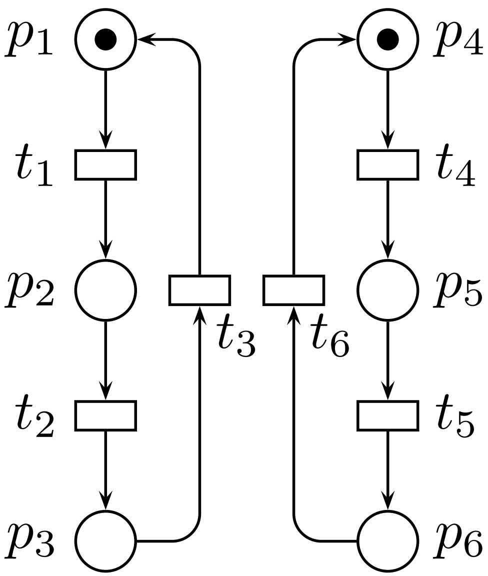
D3 is only needed when transitions are not deterministic.

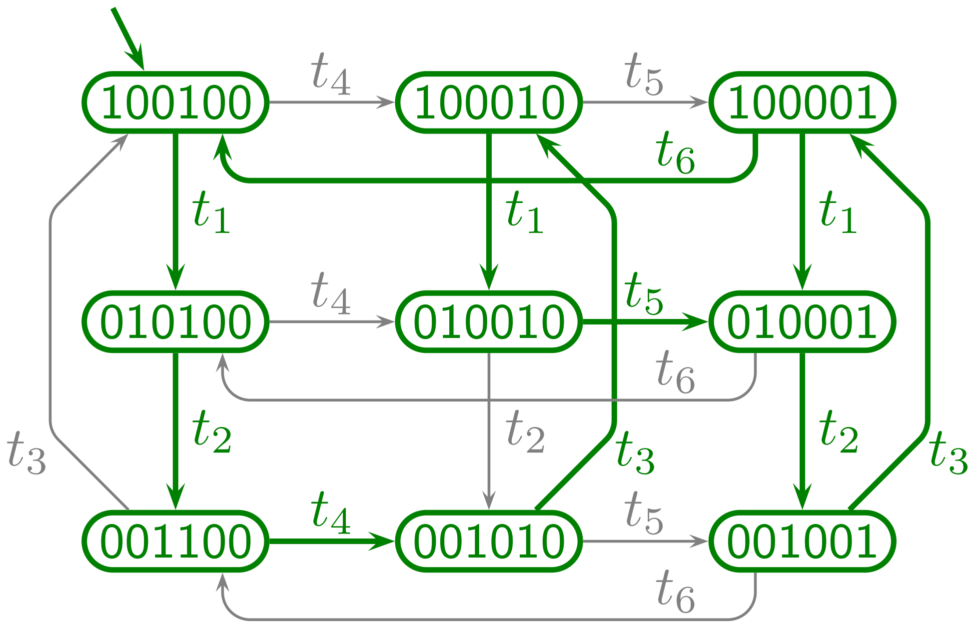
More properties ⇒ more and stronger conditions ⇒ worse reduction results

5 The Common Cycle Condition

If $t \in \mathit{Stubb}(s)$ closes a cycle in the reduced state space, then make $\mathit{Stubb}(s) = T$

Clarke, Grumberg, Peled 1999 Model Checking book and elsewhere





We do not know how common this problem is.

Something is known about how to avoid this, but not much. [Evangelista, Pajault 2010]

⇒ There is still room for better solutions to the ignoring problem.

6 Always May-Terminating Models

Typical requirements of a mutual exclusion system:

$\Box\neg(\text{in-cs}_1 \wedge \text{in-cs}_2)$

$\Box(\text{requesting}_1 \Rightarrow \Diamond\text{in-cs}_1)$ and $\Box(\text{requesting}_2 \Rightarrow \Diamond\text{in-cs}_2)$

$\Box(\text{in-cs}_1 \Rightarrow \Diamond\neg\text{in-cs}_1)$ and $\Box(\text{in-cs}_2 \Rightarrow \Diamond\neg\text{in-cs}_2)$

How about the following “solution”?

1: /* not requesting 1 */	1: /* not requesting 2 */
2: **wait until** *turn* = 1	2: **wait until** *turn* = 2
3: /* critical section 1 */	3: /* critical section 2 */
4: *turn* := 2; **goto** 1	4: *turn* := 1; **goto** 1

⇒ Must say that moving from 1 to 2 is not obligatory — while other moves are!

LTL solution: idling transitions and weak fairness

Process algebra solution: stable failures

1: **goto** 2 $\Box$ **goto** 5
5: **stop**

Always may-terminating :⇔ ∀ reachable state: a terminal state is reachable

Making models am-t (or something else) is necessary to catch certain liveness errors.

7 New Results

With always may-terminating models and many properties, no condition for the ignoring problem is needed!

Consider also not being am-t as an error that the tool should catch.

Theorem With D0, D1, D2, the model is am-t if and only if the reduced state space is.

Fast algorithms for checking the above condition

- on-the-fly: construct rss depth-first, recognize strong components
- afterwards: reverse the edges, perform any good graph-search

$\Rightarrow$ If the model has errors, at least one is caught. (It may be the not am-t error.)

Theorem With D0, D1, D2, and am-t models, the following are preserved:

- for each transition, the possibility of it occurring
  – safety properties
- existence of reachable states with no reachable progress states
  – "fairness-insensitive progress" (often used in process algebras)
- t_ω may occur ∞ times without any of T_* occurring ∞ times
  – e.g., if the channel is strongly fair to success, then the protocol succeeds

Counterexamples are valid even if the model is not am-t.

8 Measurements

Demand-driven token-ring (times in seconds)

	plain		stubborn sets		symmetries		both	
n	states	time	states	time	states	time	states	time
5	17 280	0.1	3 505	0.0	3 456	0.0	701	0.0
6	98 064	0.2	12 540	0.1	16 344	0.1	2 090	0.0
7	541 296	0.8	43 015	0.2	77 328	0.4	6 145	0.1
8	2 927 232	4.5	143 408	0.4	365 904	1.6	17 926	0.2
9	15 583 104	30.0	469 053	1.4	1 731 456	10.0	52 117	0.3
10	81 933 120	262	1 514 900	4.6	8 193 312	59.5	151 490	0.9
11	–	341	4 852 771	16.3	38 771 136	339	441 161	2.6
12			15 464 040	60.1	–	1039	1 288 670	9.1
13			..	65.0			3 777 949	30.0
14							11 116 762	96.1
15							32 826 001	353
16							..	131

Symmetric Peterson-n: exponential $\leadsto$ quadratic

More realistic Peterson-n: less spectacular, see paper

9 Discussion

Too much has been taken for granted in partial order reduction research.

- heuristics preserving various properties were developed
- little has been done to study and improve their reduction power
  - e.g., the common cycle condition
- possibilities of widening static definitions are largely unexploited
  - e.g., the controlling of currently disabled transitions
  - some tricks to that direction were used in my measurements

There has never been a well-working way of dealing with weak fairness.

- it seems that all other essential aspects of linear temporal logic are solved $\approx$ ok
- with loosely enough coupled systems, weak fairness becomes necessary
- this publication developed further methods that do not need weak fairness

I believe that for new good results, the static–dynamic dichotomy is very useful.

Thank you for attention!
Questions?