Verification with unfoldings

Javier Esparza

Fakultät für Informatik
Technische Universität München

Joint work with Keijo Heljanko, Stephan Melzer, Stefan Römer, Claus Schröter, Stefan Schwoon, Walter Vogler, and Frank Wallner.

The state explosion problem

System $\longrightarrow$ **State Space**

State-of-the-art techniques

- Compact representations: BDDs
- Partial-order methods
  - Reduce interleavings: Stubborn sets, sleep sets, ample sets
  - “True” partial-order methods: Unfoldings

A bit of personal history

1991 Partial-order semantics (“true concurrency”) well studied, but no impact on automatic verification. Work on this problem with Eike Best (CSL ’91). I get stuck.

1992 Ken McMillan (CAV ’92) gets me out of the hole. Paper (TAPSOFT ’93) on the unfolding technique.

1993 Move to Edinburgh, and to infinite state systems.

1995 Move to Munich. Back to work on unfoldings, together with Ph. D. students.

Plan of the talk

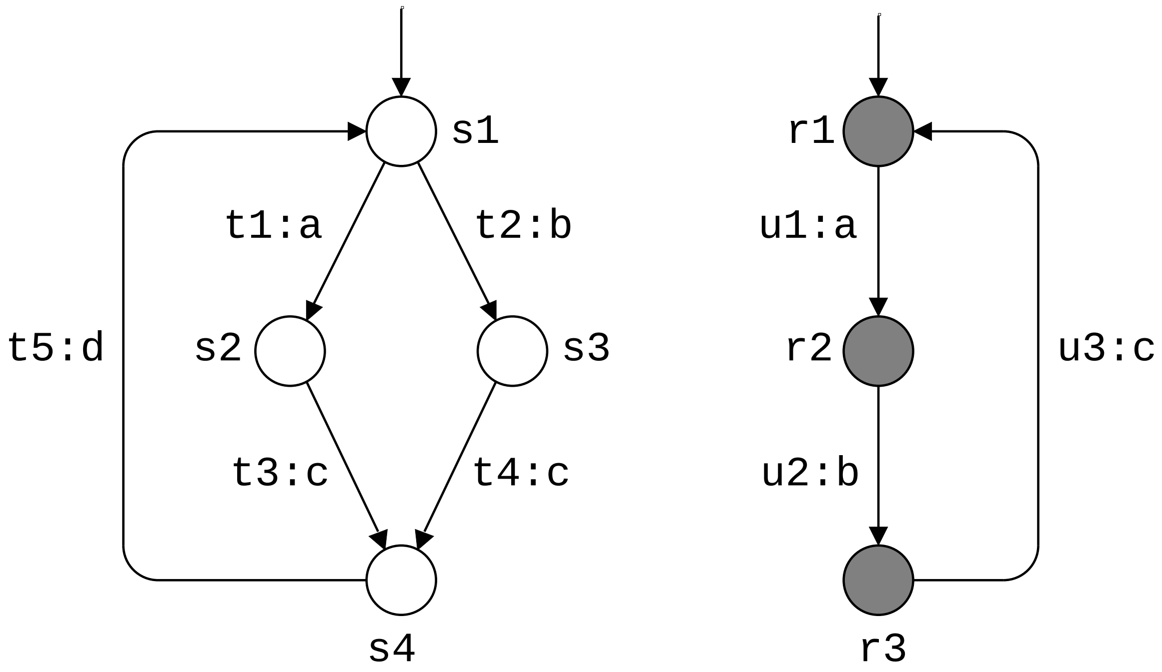
- State-of-the-art of the unfolding technique, with more emphasis on own work
- Comparison with BDDs and stubborn sets

System model

Any model with a natural definition of independent events can be used:

- Petri nets (McMillan, CAV '92 and many others)
- Contextual nets (Vogler and Yakovlev, CONCUR '98)
- Process algebras (Langerak and Brinksma, CAV '99)
- Synchronous products of transition systems
  (E. and Römer, CONCUR '99)

Synchronous product of transition systems

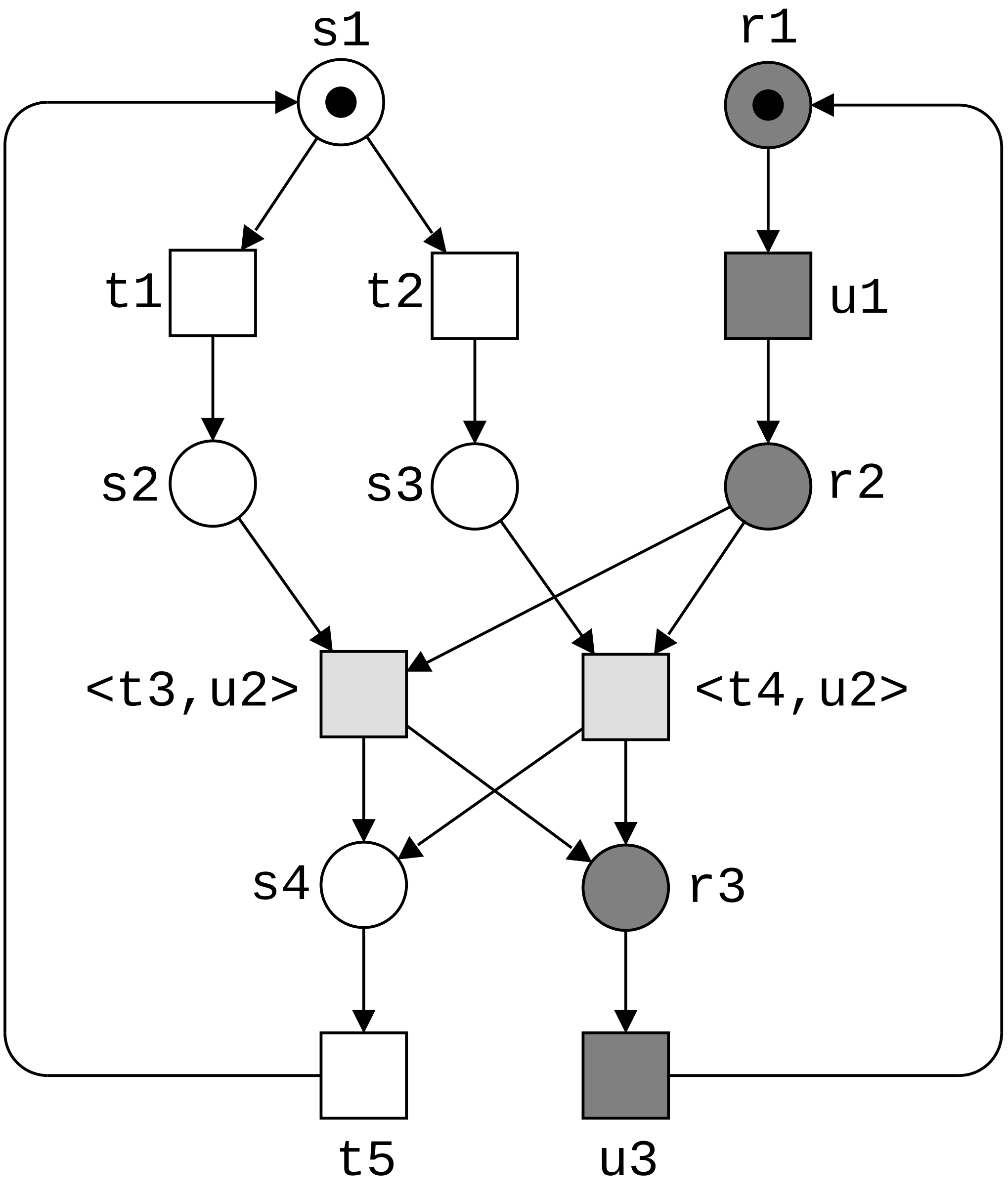


Synchronization vectors:

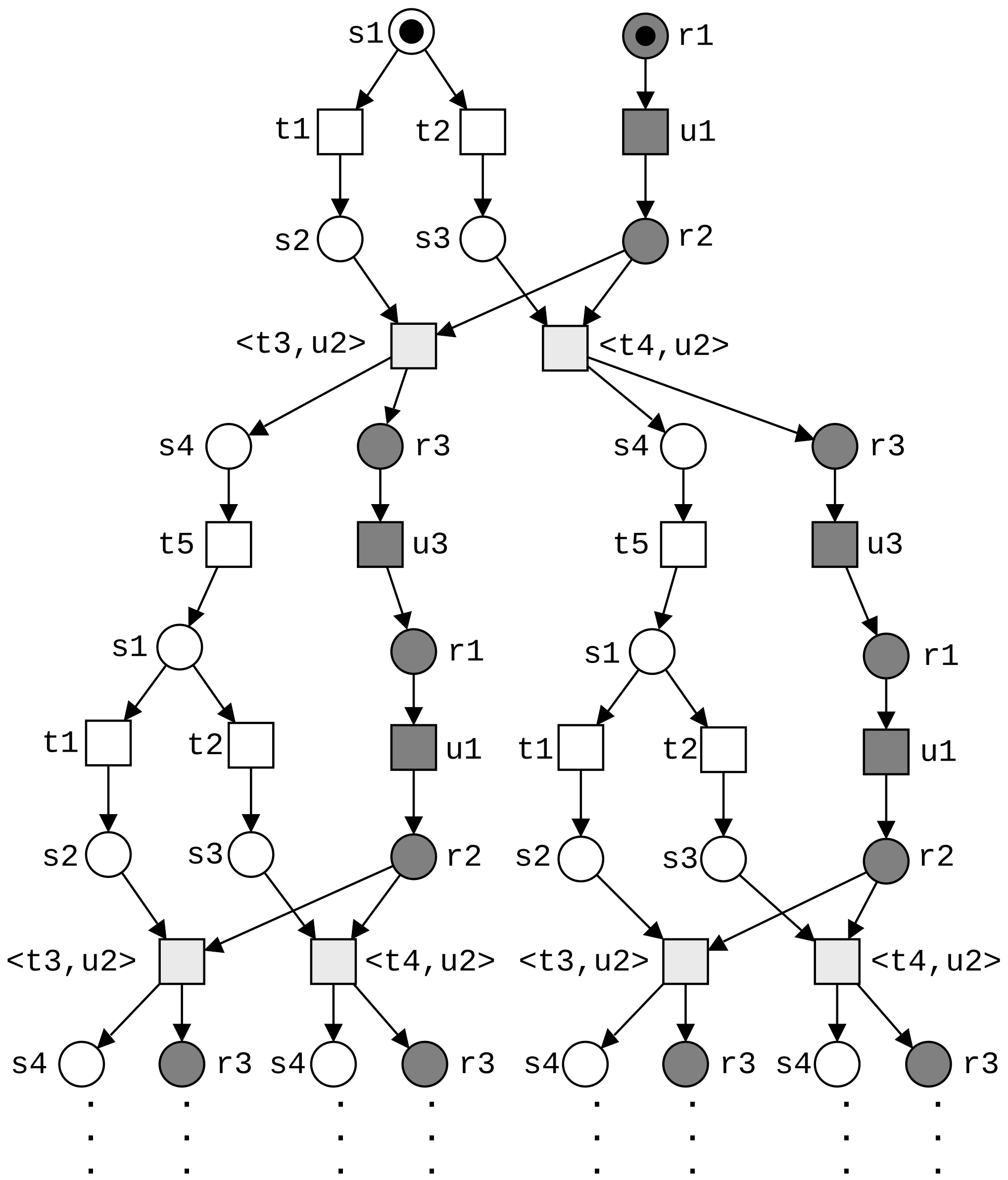
$\langle a, \varepsilon \rangle$ $\langle b, \varepsilon \rangle$ $\langle d, \varepsilon \rangle$ $\langle \varepsilon, a \rangle$ $\langle \varepsilon, c \rangle$

$\langle c, b \rangle$

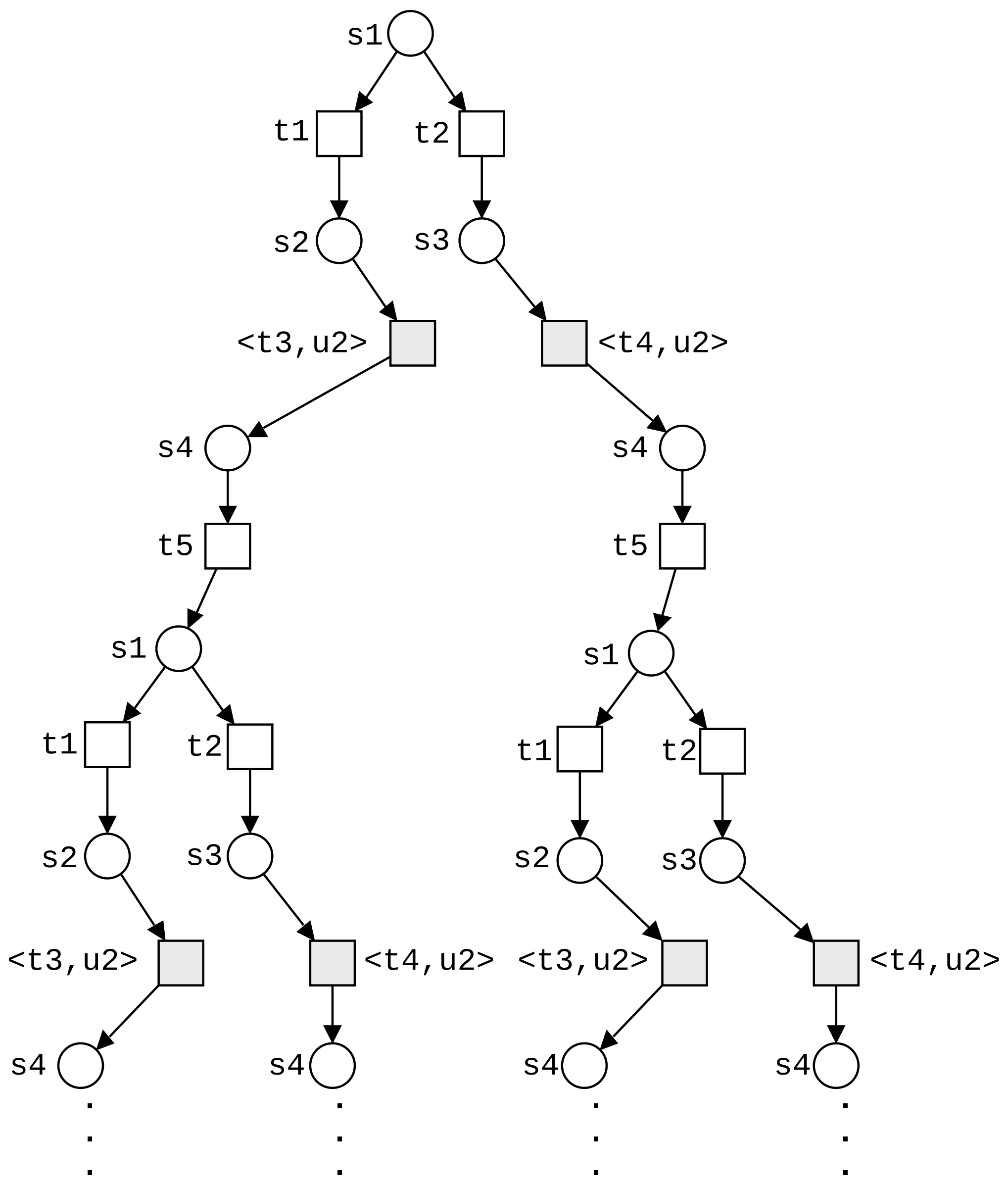
Synchronous product $\longrightarrow$ structured Petri net



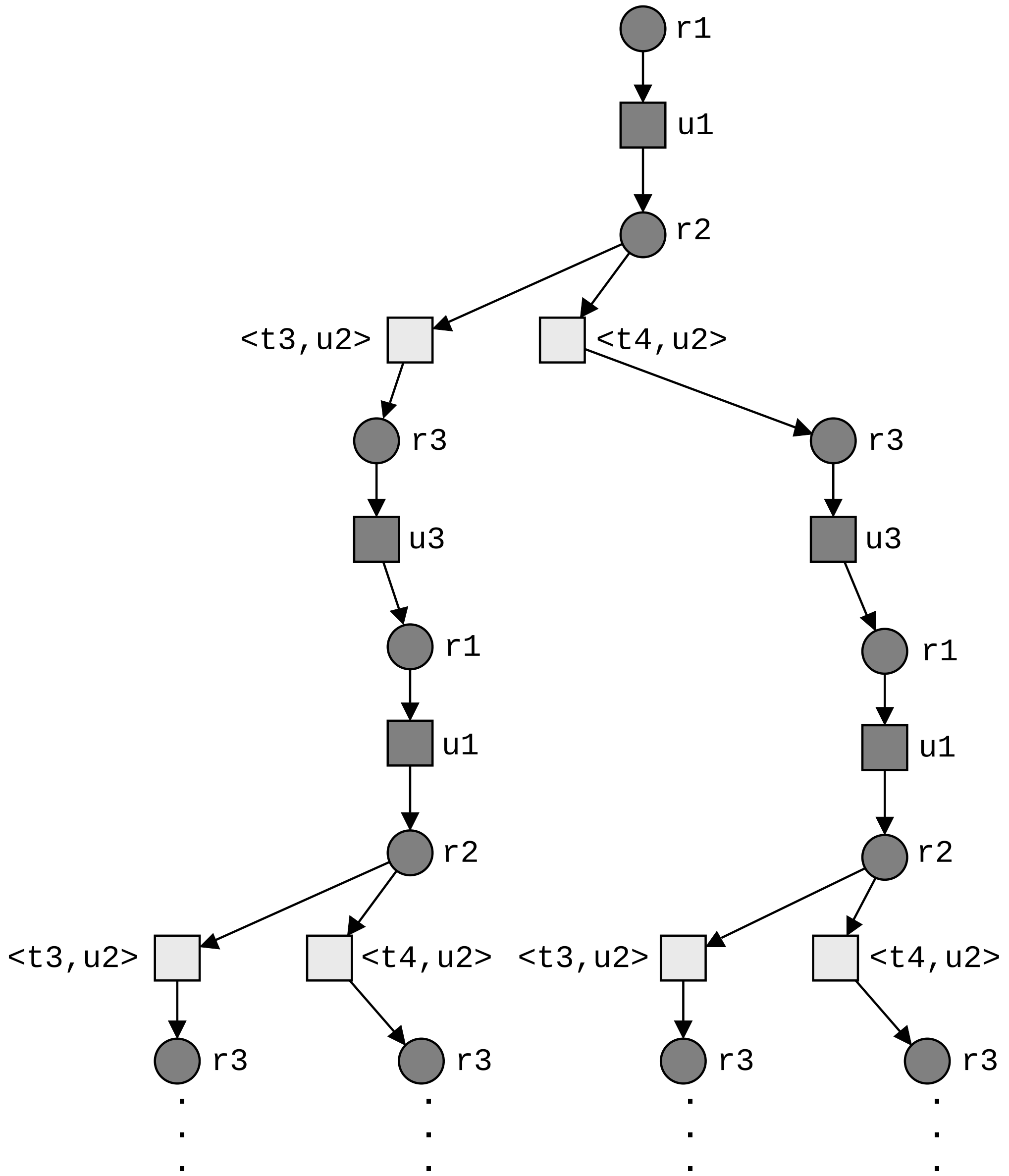
The unfolding



Decomposition into trees: first tree

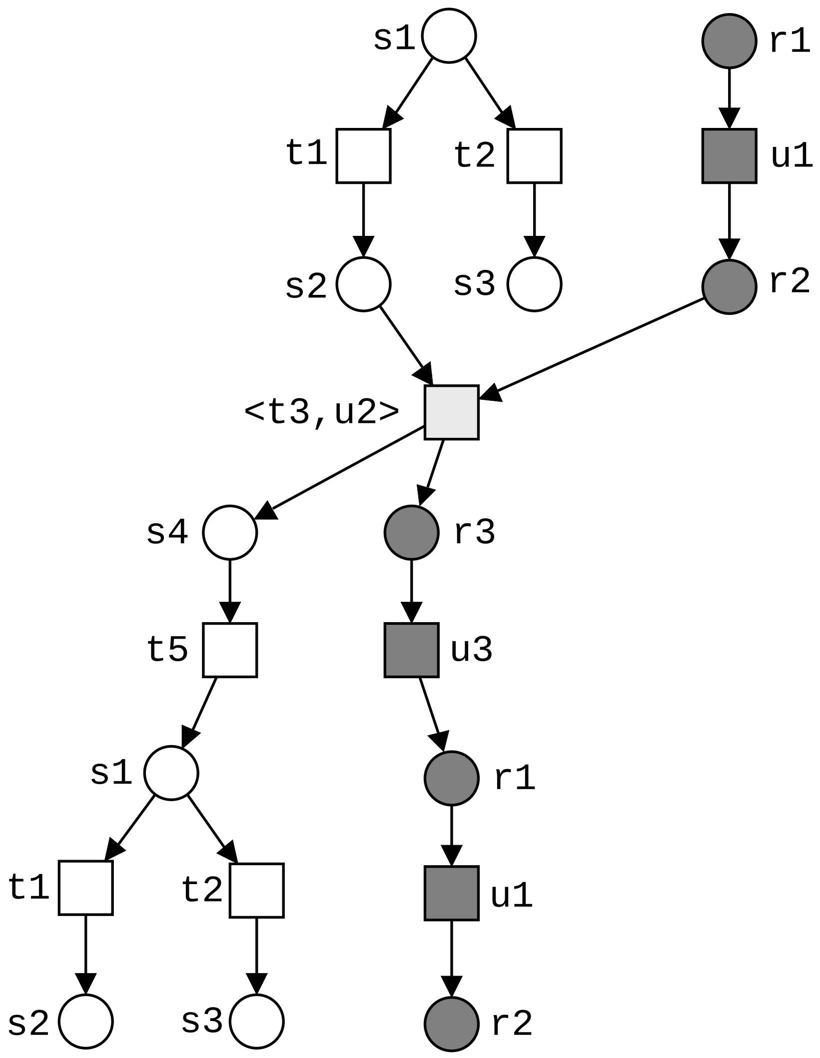


Decomposition into trees: second tree



Complete prefix

- Prefix of the unfolding "containing" all reachable states



Advantages of complete prefixes

- Same information as the reachability graph (even more)
- Usually much smaller size for concurrent systems

  Philosophers: state space grows exponentially

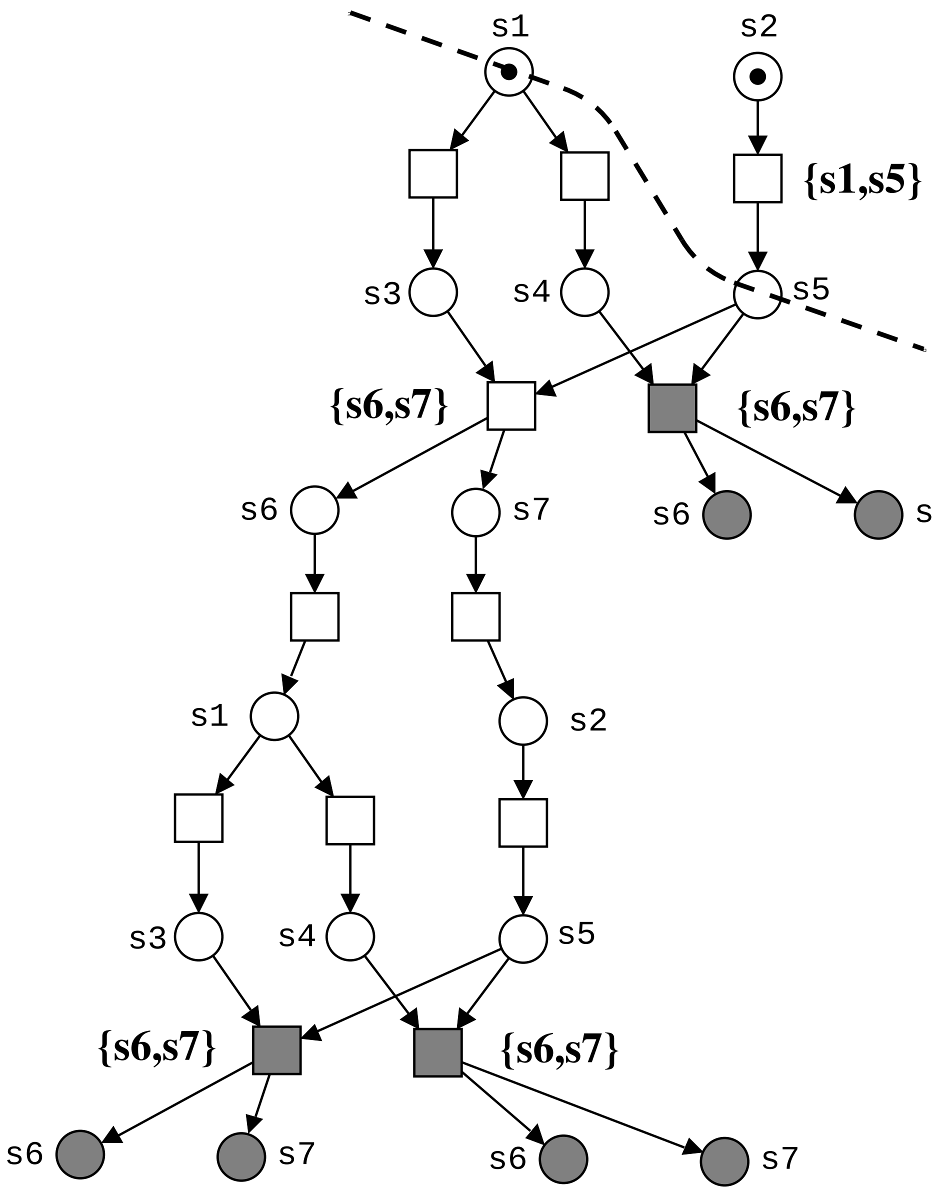
  complete prefixes grow linearly
- Structural characterisation of the reachable states
  - Graph-theoretical: as cuts of the prefix
  - Algebraic: as the integer solutions of a set of linear equations.
  - Logical: as the truth assigments satisfying a boolean formula

How to compute small complete prefixes?

McMillan (CAV ’92 and FMSD, 1995):
“Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits”.

Simple and elegant algorithm for the computation of a complete prefix using the notion of cut-off event:

> Starting from the initial global state, add events until all possible extensions have been cut-off.

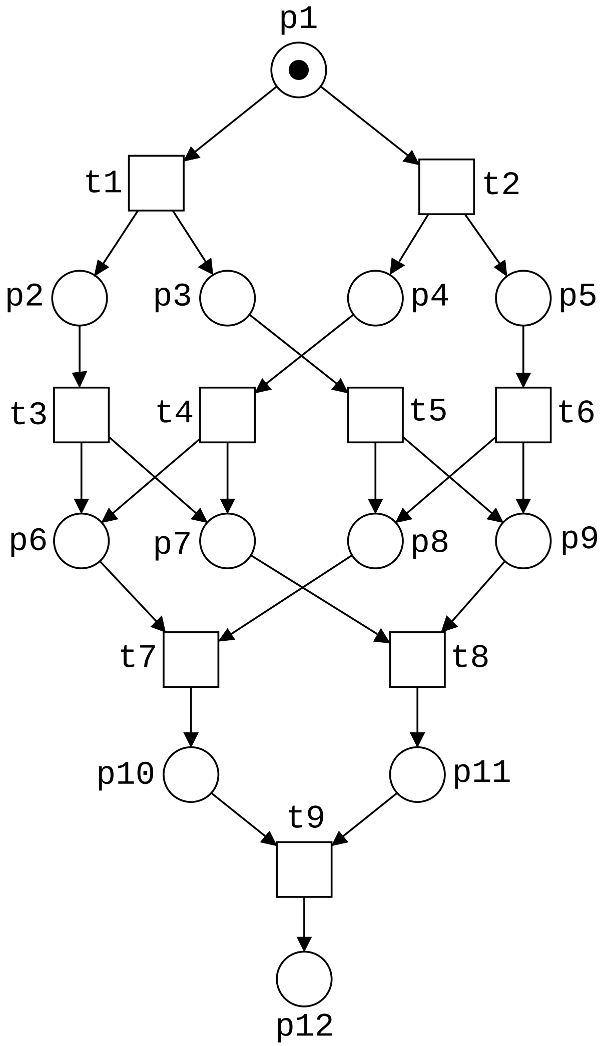


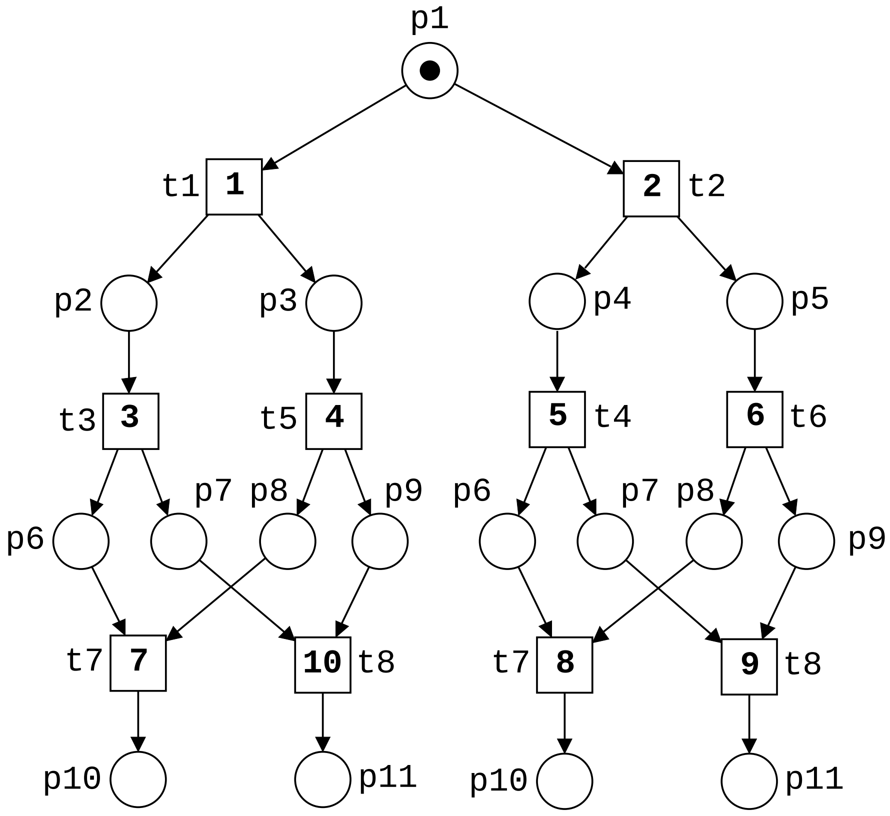
An event e is a cut-off event if there is an already generated event e' such that

(a) $\mathit{GlobalState}(e) = \mathit{GlobalState}(e')$, and

(b) e' is smaller than e with respect to an adequate order.

The construction without (b) is incorrect!

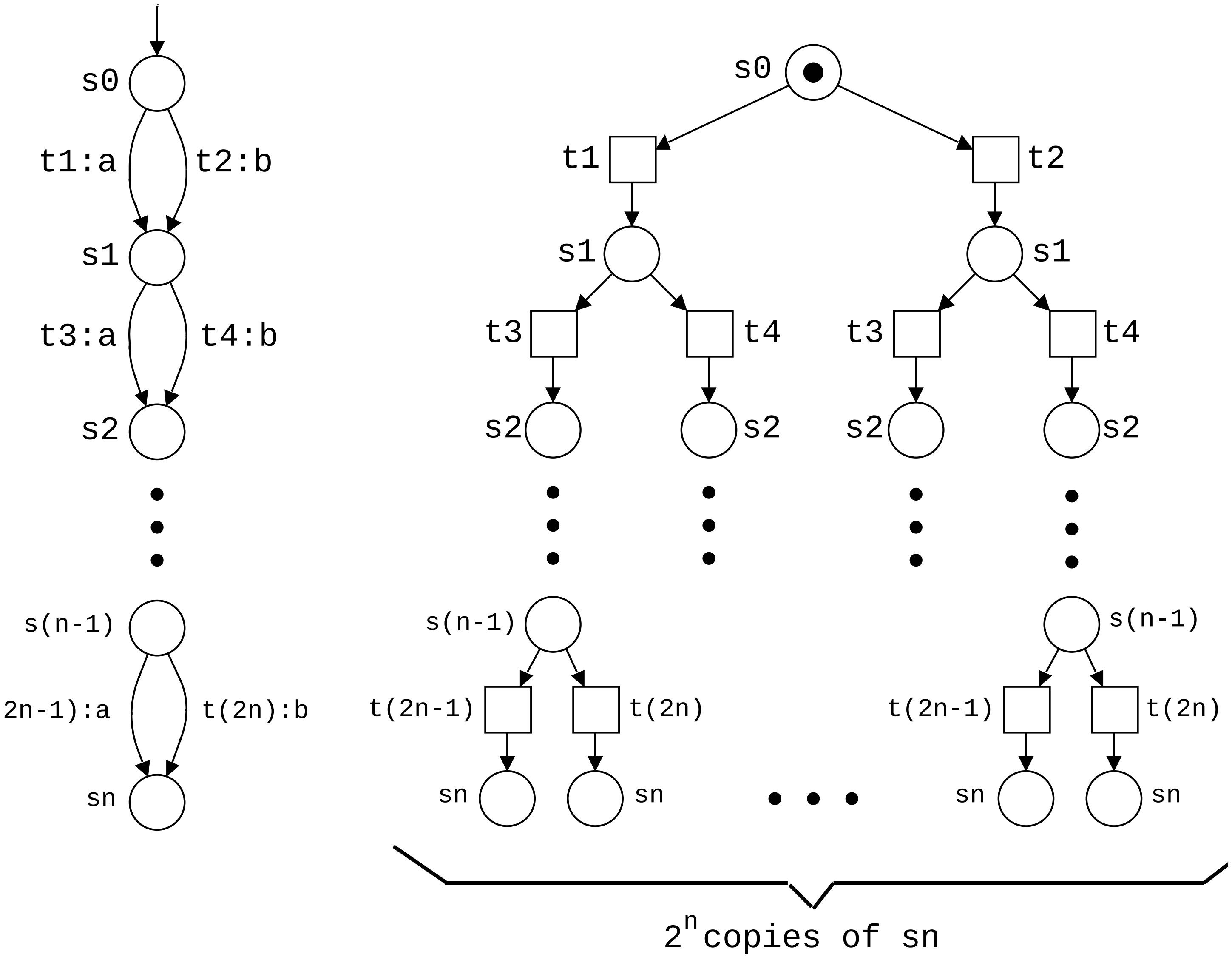




McMillan's order (CAV '92)

- $e \prec_m e'$ if the past of e contains less events than the past of e'
- PROBLEM: McMillan's complete prefix can be exponentially smaller than the state space ...

. . . but also exponentially larger!


s0
t1:a
t2:b
s1
t3:a
t4:b
s2
s(n-1)
2n-1):a
t(2n):b
sn
s0
t1
t2
s1
s1
t3
t4
t3
t4
s2
s2
s2
s2
s(n-1)
s(n-1)
t(2n-1)
t(2n)
t(2n-1)
t(2n)
sn
sn
sn
sn
2^n copies of sn

A solution for Petri net models

E., Römer and Vogler (TACAS '96):
"An improvement of McMillan's unfolding algorithm"

Total adequate order for unfoldings of 1-safe Petri nets.
(Later extended to Place/Transition nets.)
The number of non-cut-off events is never larger than the number of reachable states.
On average 20% smaller than McMillan's prefix, no time overload.

A solution for synchronous products

E. and Römer (CONCUR '99):
"An Unfolding Algorithm for Synchronous Products of Transition Systems"

Total adequate order for unfoldings of synchronous products of transition systems.
The number of non-cut-off events is never larger than the number of reachable states.
Improvements with respect to TACAS '96:

- Simpler order.
- Faster computation of the unfolding. Speed-up factors: 3 to 200.

Let A be a finite alphabet, and let $\sigma_1, \sigma_2 \in A^*$.
We say that σ_1 is smaller than σ_2 if

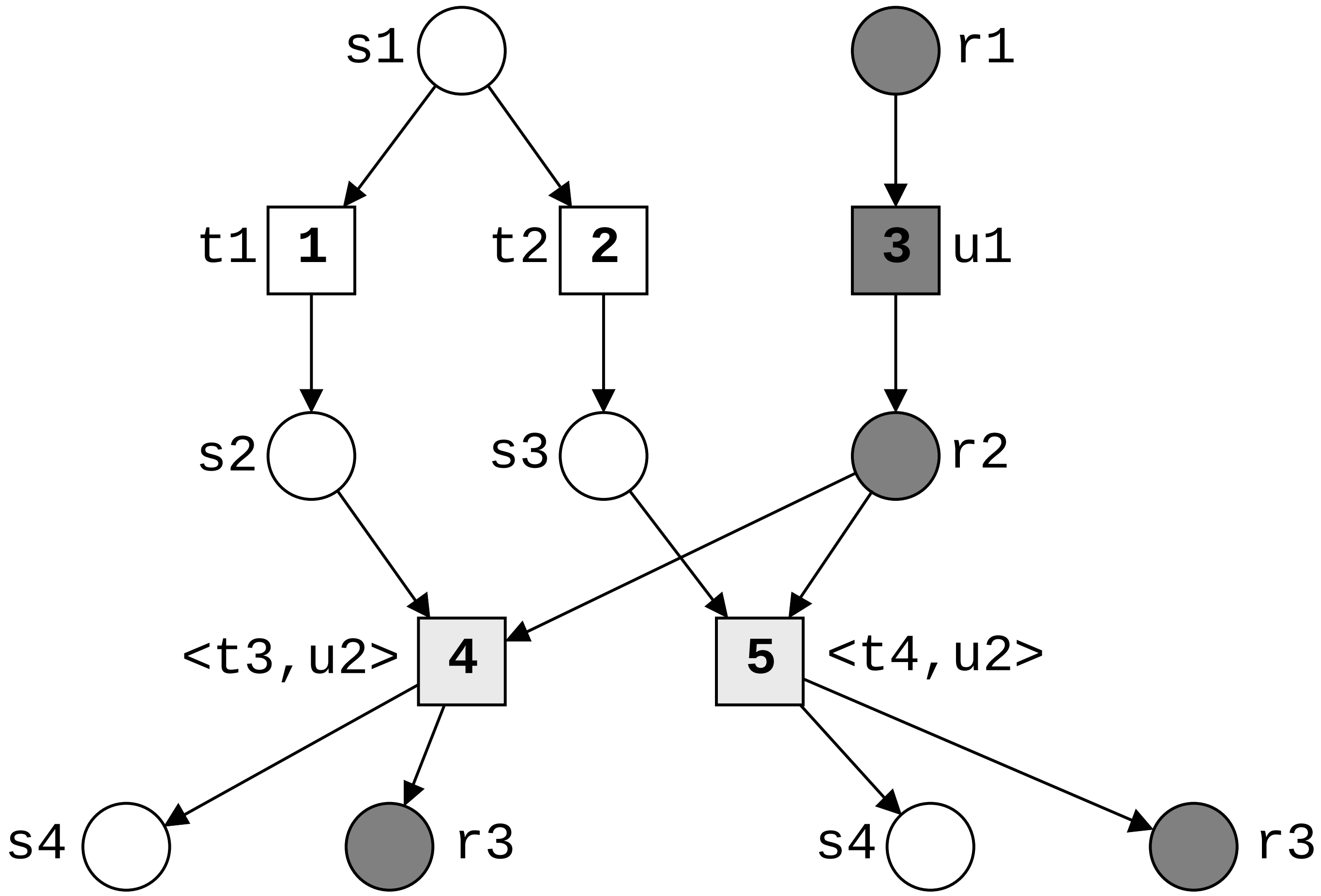
- $|\sigma_1| < |\sigma_2|$, or
- $|\sigma_1| = |\sigma_2|$ and σ_1 is lexicographically smaller than σ_2.

We compare the events 4 and 5.

- Compute the histories of the events.

Event	Component	History
4	1	$H(4,1) = t_1 \langle t_3, u_2 \rangle$
4	2	$H(4,2) = u_1 \langle t_4, u_2 \rangle$
5	1	$H(5,1) = t_2 \langle t_4, u_2 \rangle$
5	2	$H(5,2) = u_1 \langle t_4, u_2 \rangle$

- $4 \prec_{er} 5$ if
  - $H(4,1)$ is smaller than $H(5,1)$, or
  - $H(4,1) = H(5,1)$ and $H(4,2)$ is smaller than $H(5,2)$.



s1
r1
t1
1
t2
2
3
u1
s2
s3
r2
<t3,u2>
4
5
<t4,u2>
s4
r3
s4
r3

System (scale)	Structured Petri net			Prefix		BDD size (Petrify)
	Places	Trans.	Global States	Places	Trans.	
CY(12)	95	71	74264	232	104	
DPD(7)	63	63	109965	86310	4314	
SR(10)	100	100	8.1×10^{12}	119450	86180	
EL(4)	736	1939	43440	32354	16935	
PC	231	202	$> 3.1 \times 10^{6}$	2164	1035	40188
CP	150	140	2.8×10^{7}	1671	780	210249
DME(64)	257	256	1.8×10^{62}	385	256	45460
RW(10)	86	66	1.6×10^{6}	29132	15974	7576

CY : Cycler (Milner)

DPD: Philosophers with deadlock avoidance

SR: Slotted ring protocol

EL: Elevator

PC: Production cell

CP: Concurrent Pushers (Heimer)

DME: STG specification of a circuit for distributed mutual exclusion

RW: Readers and writers (Hellwagner)

Retrieving information from complete prefixes

- Deadlock detection and (arbitrary) reachability properties are NP-complete in the size of the prefix.

- Trade-off time-space

	—— More space ⟶			
NET	—	PREFIX	—	REACH. GRAPH
PSPACE-comp.	—	NP-comp.	—	Polynomial
	⟵ More time ——			

- Local reachability properties are polynomial in the size of the prefix.

Deadlock freedom and reachability properties

- McMillan (Formal Methods in System Design, 1995):
  "A Technique of State Space Search based on Unfolding"

  Branch-and-bound algorithm for deadlock detection.

- Melzer, Römer (CAV '97):
  "Deadlock Analysis Using Net Unfoldings"

  Prefix + Property $\longrightarrow$ set of linear equations
  Property holds iff equations are feasible.
  Algorithms: mixed-integer linear programming .
  Tools: CPLEX (commercial).

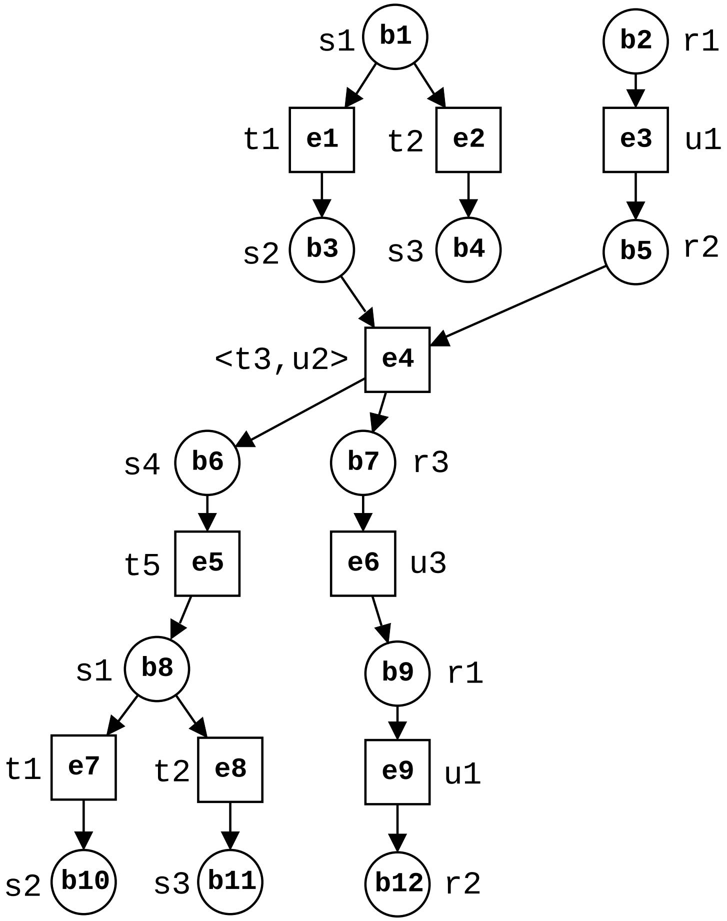
- Heljanko (TACAS '99):
  "Using Logic Programs with Stable Model Semantics to Solve Deadlock and Reachability Problems for 1-Safe Petri Nets"

  Prefix + Property $\longrightarrow$ boolean formula
  Property holds iff formula is satisfiable.
  Algorithms: constraint-programming.
  Tools: smodels (research, TU Helsinki).

- Schröter (ongoing):
  Graph-theoretic algorithm optimised for local reachability properties.

State-of-the-art: $10^4 - 10^6$ events in $1 - 10^3$ seconds.

Is the state $\{s_2, r_2\}$ reachable?



Variables: b_i, e_i : **boolean**

$$
\begin{array}{rcl}
b_1 & \leftarrow & \neg e_1 \wedge \neg e_2 \\
b_2 & \leftarrow & \neg e_3 \\
b_3 & \leftarrow & e_1 \wedge \neg e_4 \\
b_4 & \leftarrow & e_2 \\
b_5 & \leftarrow & e_3 \wedge \neg e_4 \\
 & \ldots & \\
b_{12} & \leftarrow & e_9 \\
 & \leftarrow & b_3 \vee b_{10} \\
 & \leftarrow & b_5 \vee b_{12}
\end{array}
$$

Model-checking I: safety properties

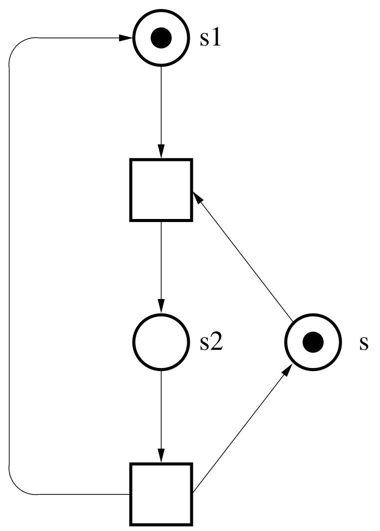
- E. (Science of Computer Programming, 1994):
  “Model-checking using net unfoldings”
  Graves (FCT&TCS ’97):
  “Computing reachability properties hidden in finite net unfoldings”
- Fragment of CTL: EF, AG
- Very fast for nearly deterministic systems, performance decreases rapidly with increasing nondeterminism.

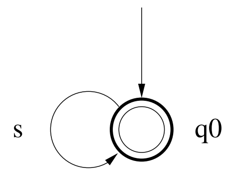
Model-checking II: liveness properties

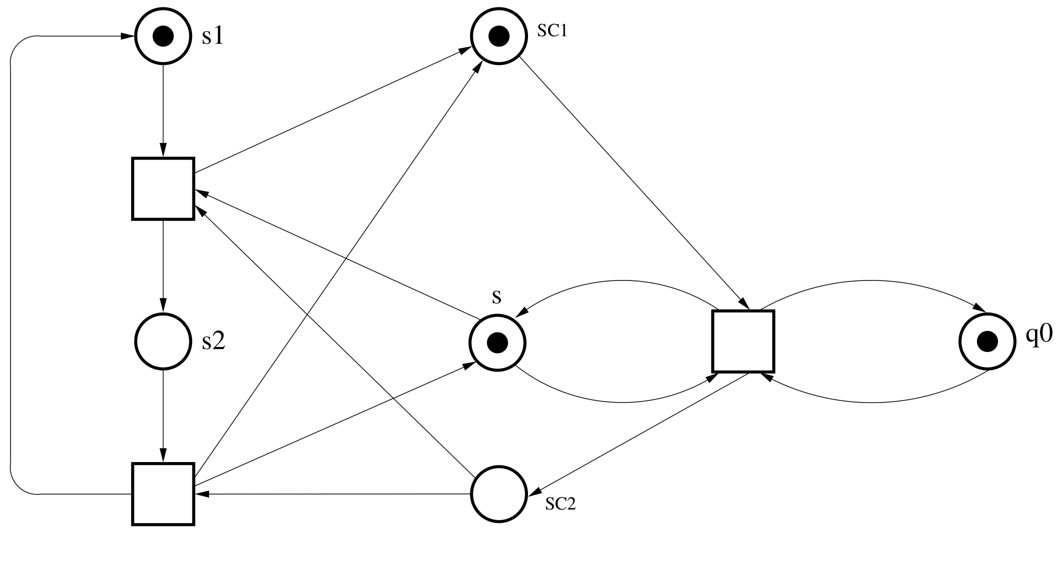
- Wallner (Ph. D. Thesis, forthcoming):
- Automata-theoretic approach to model-checking

$$\left.\begin{array}{r} N \\ \\ \phi \;\;\Rightarrow\;\; N_{\neg\phi} \end{array}\right\} \Longrightarrow \text{ Product net } N \times N_{\neg\phi}$$

- N satisfies ϕ iff the language of $N \times N_{\neg\phi}$ is empty
- Emptiness is checked on the prefix


s1
s2
s


s
q0


s1
sc1
s
s2
q0
sc2

Model-checking III: local temporal logics

- Interpreted on local states
- Related to knowledge logics
- Typical specification: if agent A satisfies P, then agent A knows that all other agents will eventually satisfy P
- Huhn, Niebert, and Wallner (FOSSACS '99):
  "Model-checking Logics for Communicating Sequential Agents"

Application areas

- Speed-independent circuits
  - McMillan (CAV '95): "Trace theoretic verification of asynchronous circuits using unfoldings"
  - Kondratyev et al. (CSD '98): "A Structural Approach for the Analysis of Petri Nets by Reduced Unfoldings"
- Telecommunication networks
  - Boubour and Jard (ATPN '97): "Fault Detection in Telecommunication Networks Based on a Petri Net Representation of Alarm Propagation"

- Communication protocols
  - Fleischhack and Grahlmann (ATPN ’98): “A compositional Petri Net Semantics for SDL”
  - Grahlman (RELECTRONIC ’95): “Verifying Telecommunication Protocols with PEP”
- Distributed Algorithms
  - Merkel (Tech. report, TU Munich, 1997): “Verification of Fault Tolerant Algorithms Using PEP”

Performance

Number of local states	Verified	Time
< 100	all	seconds/minutes
between 100 and 1000	many	minutes/hours
between 1000 and 10000	some	hours/days
> 10000	very few	days

Unfoldings vs. BDDs

Conceptual similarities and differences:

- Both techniques compress the state space
- BDDs exploit regularity
  Unfoldings exploit concurrency

Consequences:

$+$ Robustness: Unfoldings less sensitive to changes in the system

$+-$ Compression: Prefix smaller for loosely coupled systems, BDDs smaller for tightly coupled systems

- 100 random tables with right-handed, left-handed, and ambidextrous philosophers
- BDD for the set of reachable states (Petrify)

Nr. of phil.	BDD size				
	Average	Min.	Max.	St.Dev.	Aver./St.Dev.
4	178	94	355	52	0.30
6	583	248	1716	305	0.52
8	1553	390	8678	1437	0.92
10	3140	510	27516	4637	1.48
12	4855	632	47039	8538	1.76
14	33742	797	429903	85798	2.54

- 100 random tables with right-handed, left-handed, and ambidextrous philosophers
- Nodes of the complete prefix (PEP)

Nr. of phil.	Prefix size				
	Average	Min.	Max.	St.Dev.	Aver./St.Dev.
4	46	40	60	5.13	0.10
6	70	60	85	5.99	0.09
8	95	80	110	6.92	0.07
10	117	100	135	7.78	0.07
12	141	120	160	7.40	0.05
14	161	140	185	9.25	0.06

Checking deadlock-freedom with BDDs

- 100 random tables with right-handed, left-handed, and ambidextrous philosophers
- SMV on a SUN Ultra 60, 2 processors, 640 MB

Nr of phil.	Time in seconds				
	Average	Min.	Max.	St.Dev.	Aver./St.Dev.
4	0.08	0.05	0.13	0.02	0.29
6	0.36	0.20	1.18	0.16	0.46
8	4.14	1.25	14.60	2.45	0.59
10	56.60	15.80	388.00	46.90	0.83
12	1595.00	228.00	10616.00	1615.00	1.01

Checking deadlock-freedom with unfoldings

- 100 random tables with right-handed, left-handed, and ambidextrous-philosophers
- PEP + stable models on a SUN Ultra 60, 2 processors, 640 MB

Nr. of phil.	Time in seconds				
	Average	Min	Max	St. Dev	Aver./St. Dev
8	0.01	0.04	0.03	0.007	0.24
10	0.01	0.06	0.03	0.009	0.27
12	0.02	0.07	0.04	0.012	0.28
14	0.02	0.05	0.04	0.007	0.20
16	0.02	0.05	0.04	0.007	0.17
18	0.03	0.05	0.04	0.007	0.17

Unfoldings vs. stubborn sets

Conceptual similarities and differences:

- Both techniques exploit concurrency
- Stubborn sets discard information
  Unfoldings compress information
- Stubborn sets are conservative: small overhead is guaranteed, at the price of a suboptimal reduction
  Unfoldings "take risks": large overhead is possible, but optimal compression

Consequences:

\+ No loss of information $\rightarrow$ All reachability properties checkable on the same prefix

\+ Causality information available (ex. alarm patterns)

− Larger overheads for tightly coupled systems

Checking deadlock-fredom with stubborn sets and unfoldings

- 1 left-handed, 1 right-handed, and $(n-2)$ ambidextrous philosophers
- SUN Ultra 60, 2 processors, 640 MB

Nr. of phil.	Time in seconds	
	PROD	PEP + smodels
10	18	0.04
12	69	0.04
14	834	0.04
16	5003	0.04
18	29257	0.06

Tentative rules of thumb

- BDDs more suitable for very regular systems
- Stubborn sets and unfoldings more suitable for irregular but concurrent systems
  - Stubborn sets more suitable for systems with little concurrency
  - Unfoldings more suitable for highly concurrent systems

Fakultät für Informatik der Technischen Universität München,
Lehrstuhl für Theoretische Informatik und Grundlagen der KI

Partial-order verification techniques

Photo by G. Brad Lewis

Cooperations, **Financial Support**

Publications, **Tools**

Current Members:

- Javier Esparza
- Stefan Römer
- Claus Schröter
- Stefan Schwoon
- Frank Wallner

Former Members:

- Angelika Mader
- Stephan Melzer

Links

Online information on the unfolding technique is available at

http://www7.in.tum.de/gruppen/theorie/pom/

including a rather complete bibliography.
Copies of these slides avilable from my home page

http://www7.in.tum.de/~esparza/

Chair - Theory Group - Persons - Publications - How to find us

Partial Order | Constraint Programming | Infinite States | Mobile Systems | Algorithmic Learning

The Model-Checking Kit

- **Overview**
- **How to install the Kit**
- **A sample session**
- **Modelling a system**
- **Describing properties**
- **Checking properties**
- **Making additions to the sources**

Current Members:

- Javier Esparza
- Claus Schröter
- Stefan Schwoon

- Collection of input languages and checkers
- All checkers applicable to the same model
- Primarily designed for lab courses
- "Classical", partial-order-based, and BDD-based checkers
- Freely available at

  http://www7.in.tum.de/gruppen/theorie/KIT/