An $O(m \log n)$ algorithm for stuttering equivalence and branching bisimulation.

Jan Friso Groote, David Jansen,
Jeroen Keiren, Anton Wijs

TU/e Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

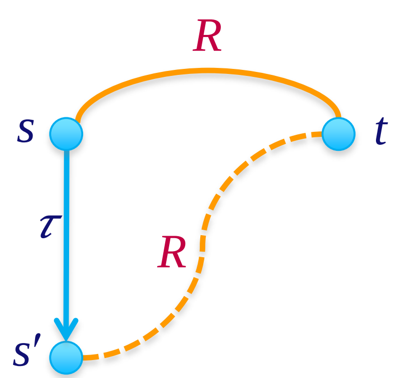
Milner introduced weak bisimulation.

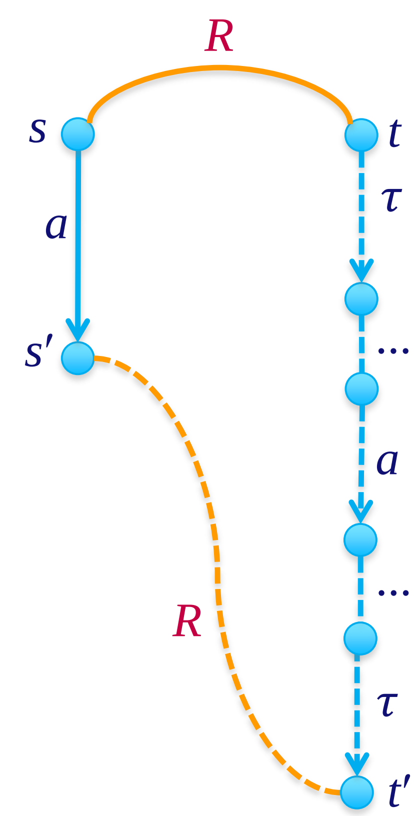
Weak bisimulation: 1980, Milner. Internal action τ.

Branching bisimulation: 1989, van Glabbeek and Weijland.

Weak bisimulation.

A relation R is a *weak bisimulation* relation iff

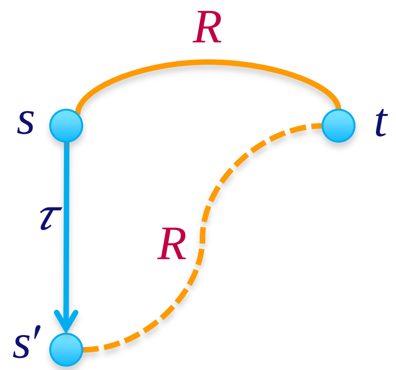


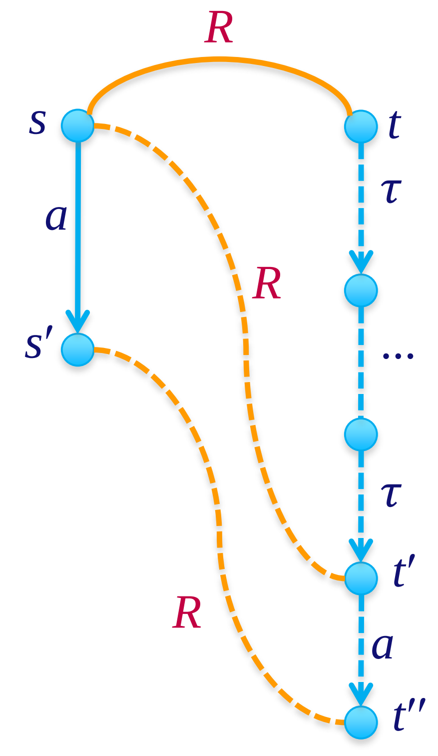


Symmetric case.

Branching bisimulation.

A relation R is a *branching bisimulation* relation iff





Why is branching bisimulation interesting?

It is conceptually the best equivalence.... $\square$

For all practical purposes branching bisimulation and weak bisimulation work equally well.

The algorithm for branching bisimulation outperforms the algorithms for all other equivalences!!

As branching bisimulation is finer than all other ‘weak’ equivalences, you want to reduce a transition system modulo branching bisimulation first.

Some history of bisimulation and their algorithms

Strong bisimulation	Milner, 1980
- Algorithm O(mn)	Kanellakis and Smolka, 1983
- Algorithm O($m \log n$)	Paige and Tarjan, 1986

Algorithm for weak bisimulation:

use strong bisimulation + transitive closure O(n^3).

Branching bisimulation	van Glabbeek and Weijland, 1989
Stuttering equivalence	Browne, Clarke and Grumberg, 1988
-Algorithm O(mn)	Groote and Vaandrager, 1990
-Algorithm ≧O(mn)	Blom and Orzan 2003
-Algorithm O($m \log n$)	????????

Transition systems/Kripke structures
m number of transitions
n number of states

Some history of bisimulation and their algorithms

Strong bisimulation — Milner, 1980
- Algorithm O(mn) — Kanellakis and Smolka, 1983
- Algorithm O($m \log n$) — Paige and Tarjan, 1986

Algorithm for weak bisimulation:

use strong bisimulation + transitive closure O(n^3).

Branching bisimulation — van Glabbeek and Weijland, 1989

Stuttering equivalence — Browne, Clarke and Grumberg, 1988

-Algorithm O(mn) — Groote and Vaandrager, 1990

-Algorithm ≧O(mn) Blom and Orzan 2003

??????

-Algorithm O($m \log n$)

Transition systems/Kripke structures

m number of transitions

n number of states

Some history of bisimulation and their algorithms

Strong bisimulation — Milner 1980
- Algorithm O(mn) — Kanellakis and Smolka, 1983
- Algorithm O($m \log n$) — Paige Tarjan, 1986

Algorithm for weak bisimulation:

use strong bisimulation + transitive closure O(n^3).

Branching bisimulation — van Glabbeek and Weijland, 1989

Stuttering equivalence — Browne, Clarke and Grumberg, 1988

-

-

-

Solved

Transition systems/Kripke structures

m number of transitions

n number of states

Benchmarks.

Our new algorithm:

Faster on large transition systems.

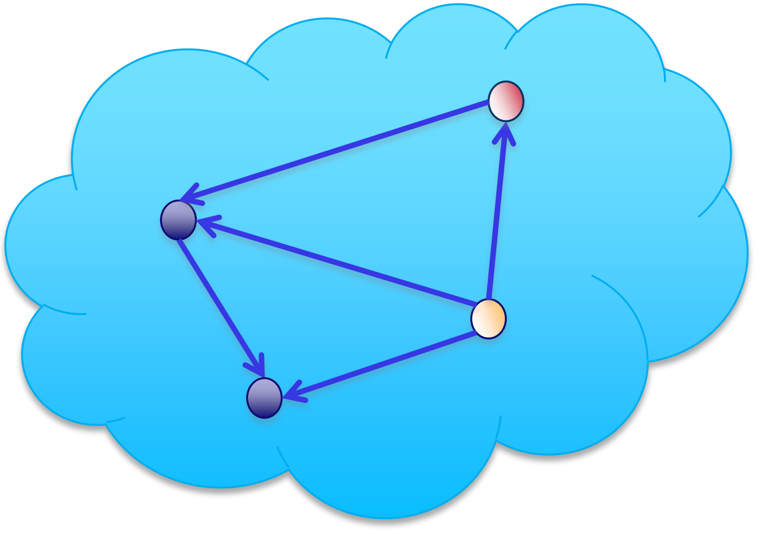
Memory usage: Comparable.

Model	*n*	*m*	min: *n*	min: *m*	Groote/ Vaandrager	Blom/ Orzan	Groote/ Jansen/ Keiren/ Wijs
Vasy40	40k	60k	20k	40k	24s	196s	0s
Vasy66	66k	1M	51k	1M	2s	9s	3s
Vasy116	116k	369k	22k	88k	1s	6s	1s
Vasy166	166k	651k	42k	197k	5s	3s	1s
CWI214	214k	684k	478	2k	1s	13s	1s
CWI2416	2M	18M	730	3k	30s	26s	19s
Vasy2581	3M	11M	704k	4M	700s	230s	31s
Vasy4220	4M	14M	1M	7M	1ks	460s	38s
Vasy4338	4M	16M	705k	4M	2ks	300s	41s
Vasy6020	6M	19M	256	510	40s	41s	20s
Vasy6120	6M	11M	3k	5k	130s	160s	24s
CWI7838	7M	59M	62k	470k	260s	7ks	160s
Vasy8082	8M	43M	290	680	100s	450s	57s
Vasy11026	11M	25M	776k	3M	2ks	1ks	68s
Vasy12323	12M	28M	876k	3M	3ks	1ks	77s
1394-fin3	127M	276M	160k	539k	68ks	10ks	1ks

Kripke structure.

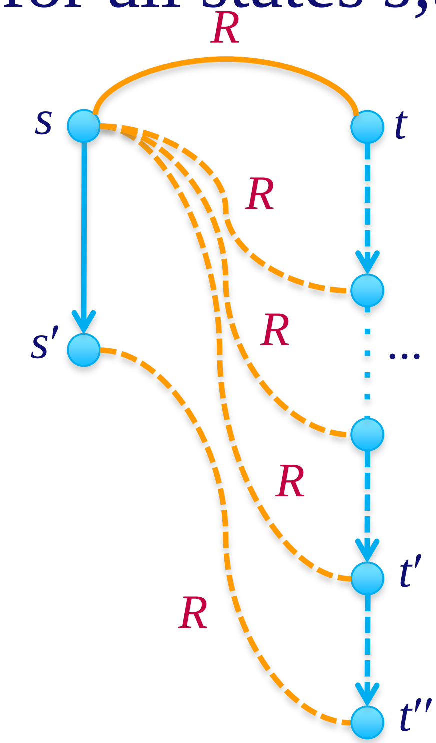
Definition. A *Kripke structure* is a four tuple $K=(S, AP, \rightarrow, L)$ where

- S is a finite set of states
- AP is a finite set of atomic propositions
- $\rightarrow \subseteq S \times S$ is a total transition relation.
- $L: S \rightarrow 2^{AP}$



Divergence blind stuttering equivalence.

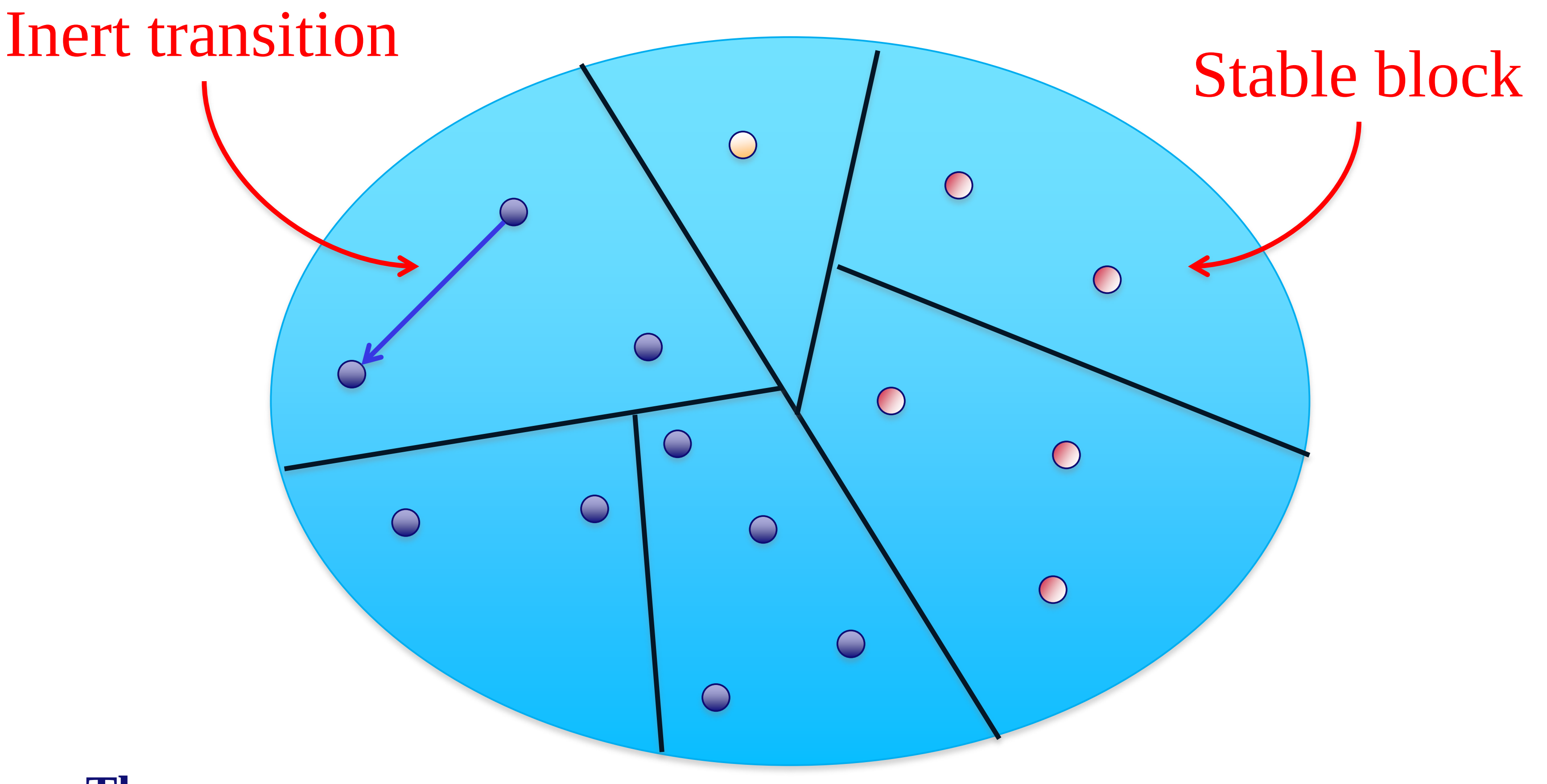
A relation R is a *db-stuttering equivalence* relation iff R is symmetric for all states $s,t \in S$, $L(s)=L(t)$ and



Two states s,t are *db-stuttering equivalent* iff there is a db-stuttering equivalence relation R such that sRt.

Partitioning algorithms.

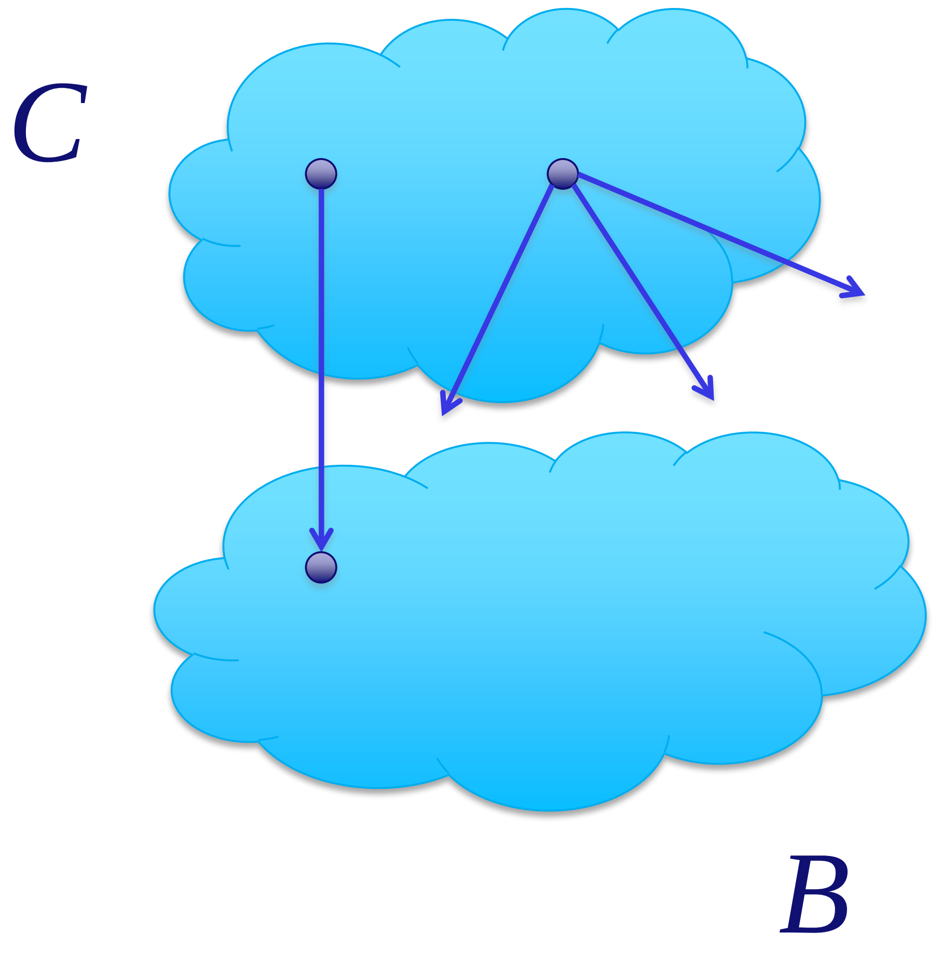
Initially, s, t in the same block iff $L(s)=L(t)$



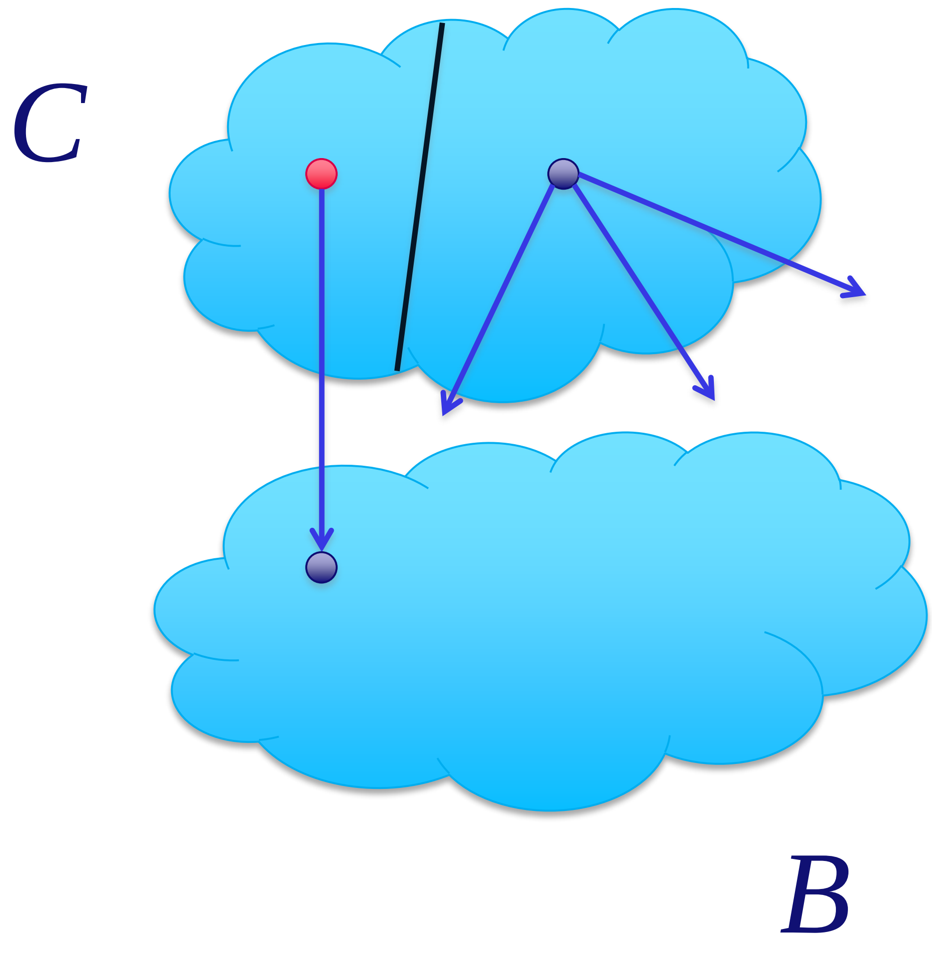
Theorem.

If stable, states are in the same block iff they are db-stuttering equivalent.

Partitioning algorithms.

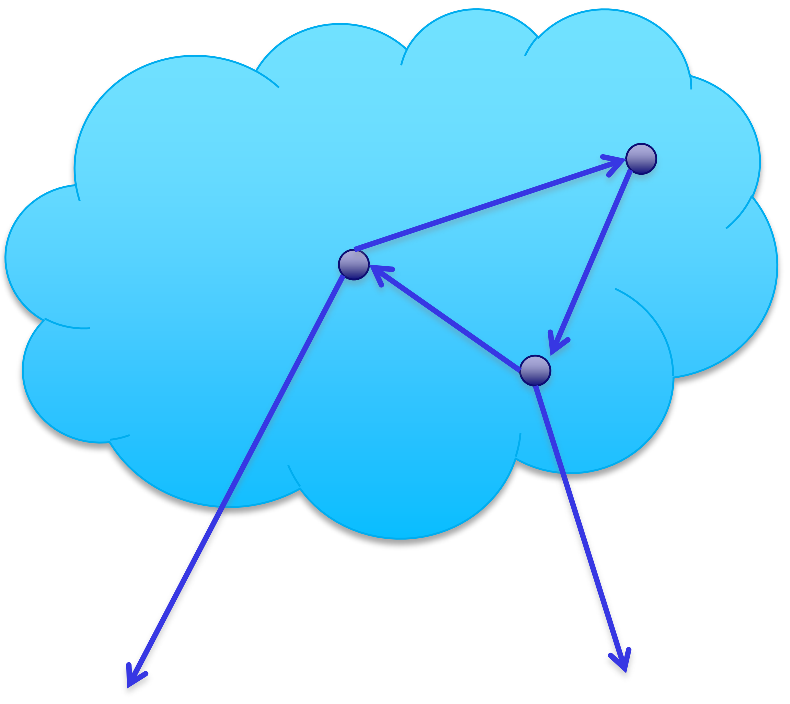


Partitioning algorithms.



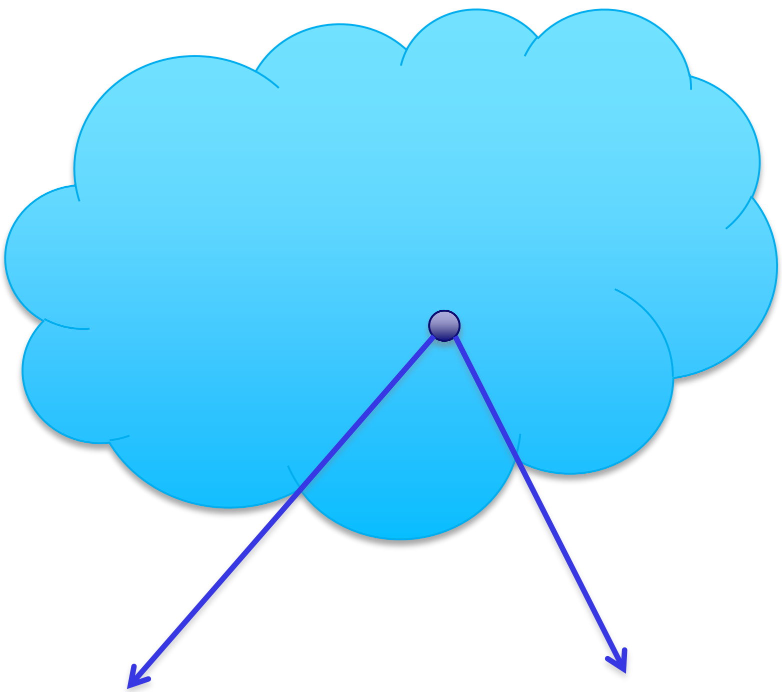
Technische Universiteit Eindhoven University of Technology

DB stuttering equivalence.



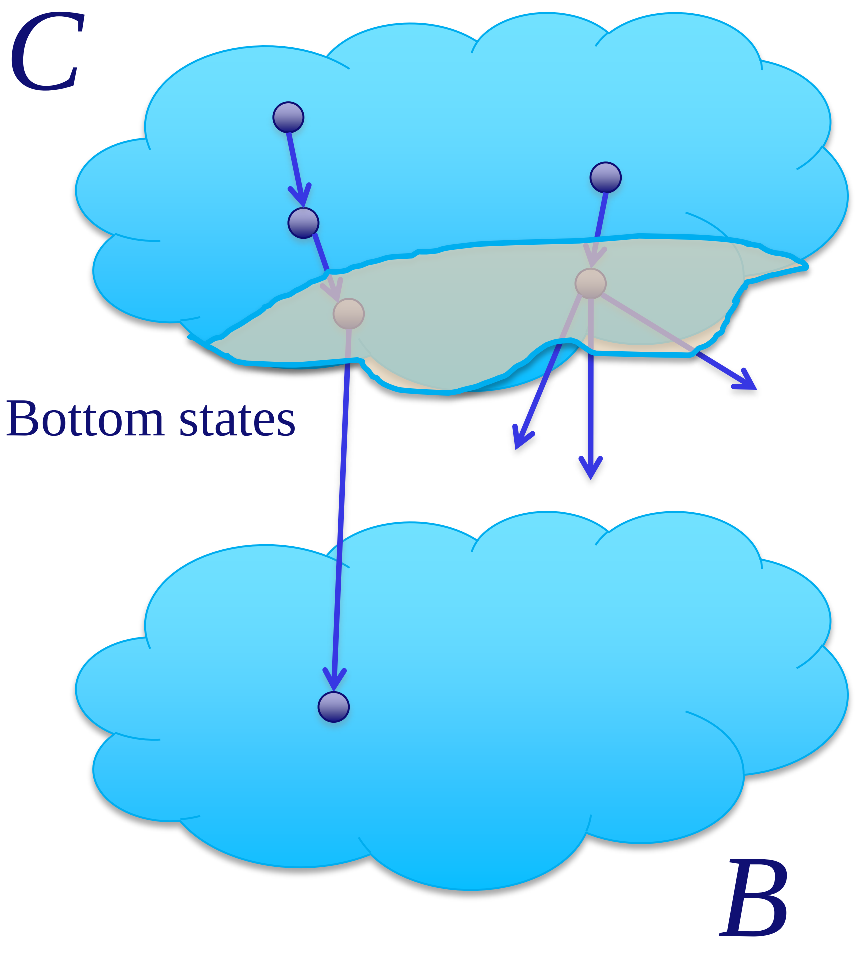
Remove loops in a block.

DB stuttering equivalence.

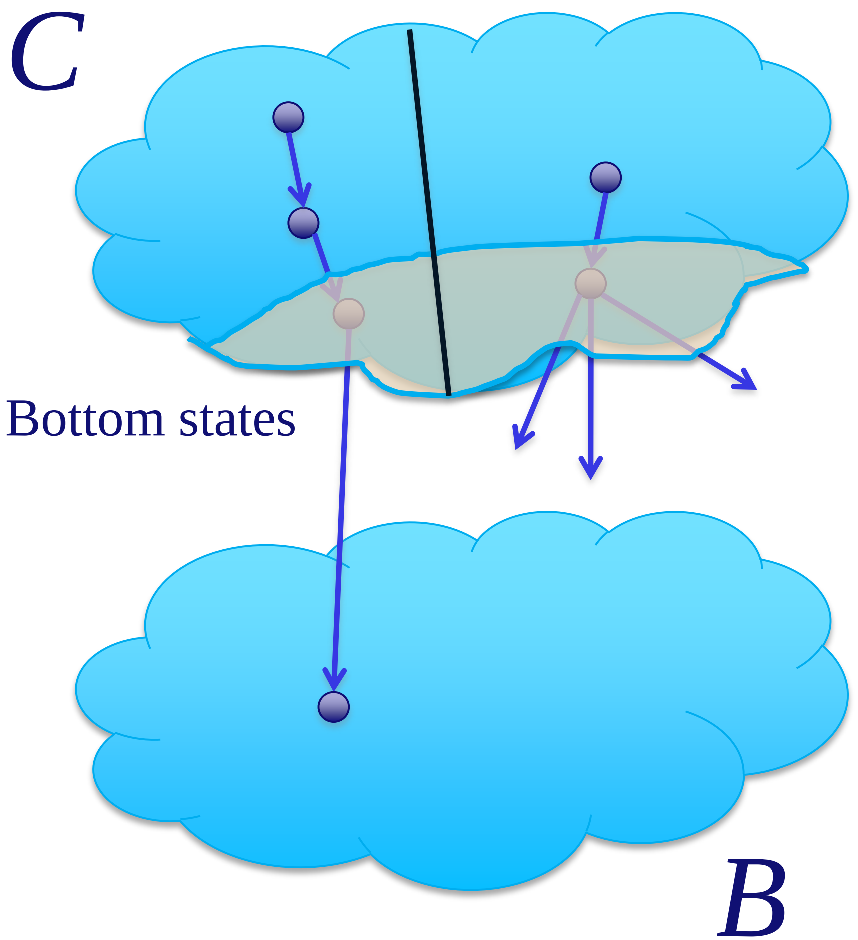


Remove loops in a block.

DB stuttering equivalence.



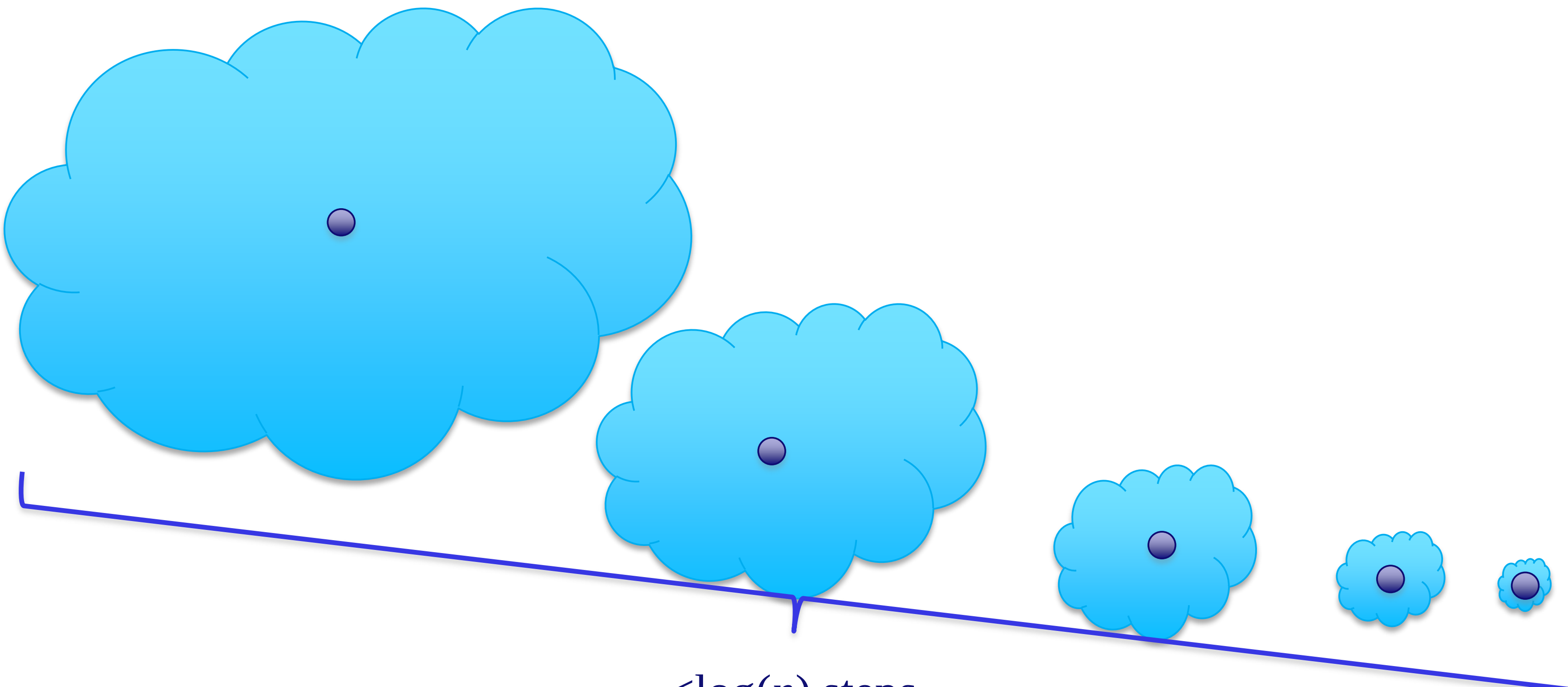
DB stuttering equivalence.



Theorem (Groote/Vaandrager 1990).
B splits *C* iff there is a transition from *C* to *B* and not all bottom states in *C* have a transition to *B*.

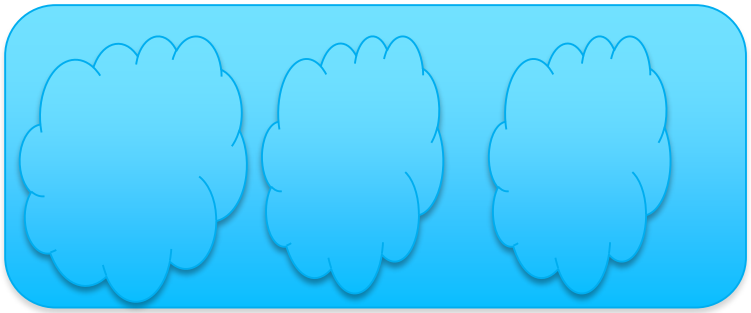
Paige-Tarjan $O(m\log n)$ algorithm

Whenever a state is involved in detecting a splitter/splitting
it does that as a member of a block half the size of the previous block.



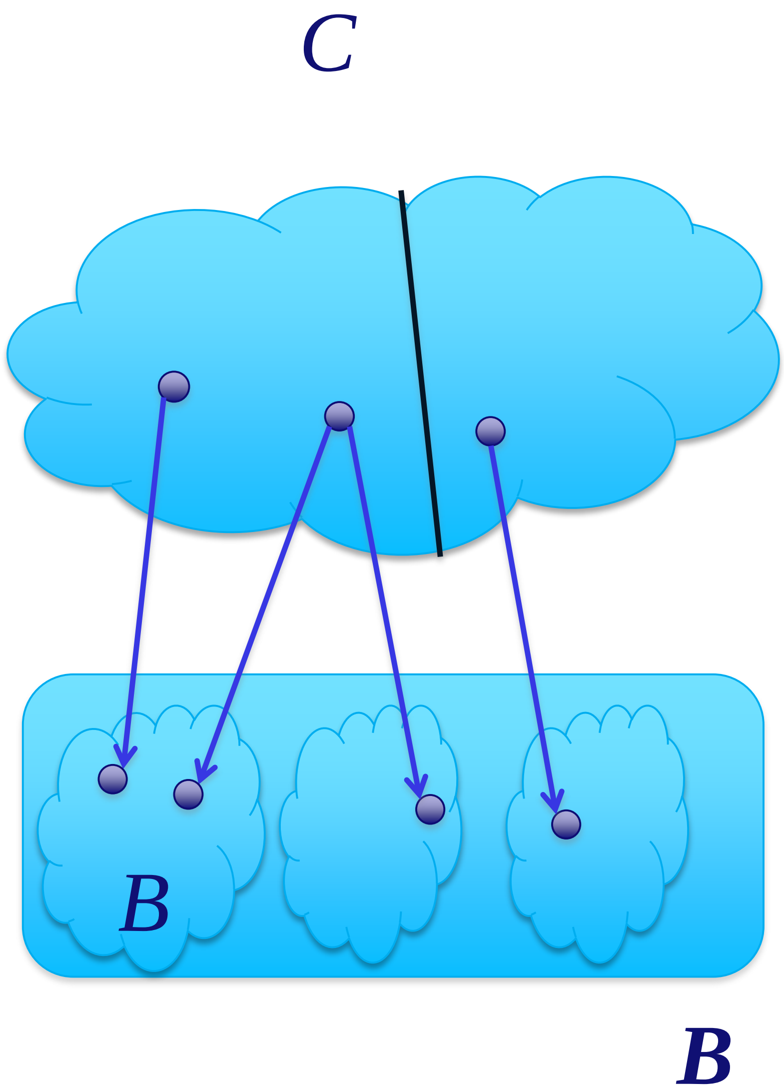
When visiting a state we visit the in and outgoing transitions a constant number of times. Complexity is O(*m* log *n*).

Constellation.



Constellation $\boldsymbol{B}$ is a set of blocks such that each block is stable for $\cup\boldsymbol{B}$.

Constellation.

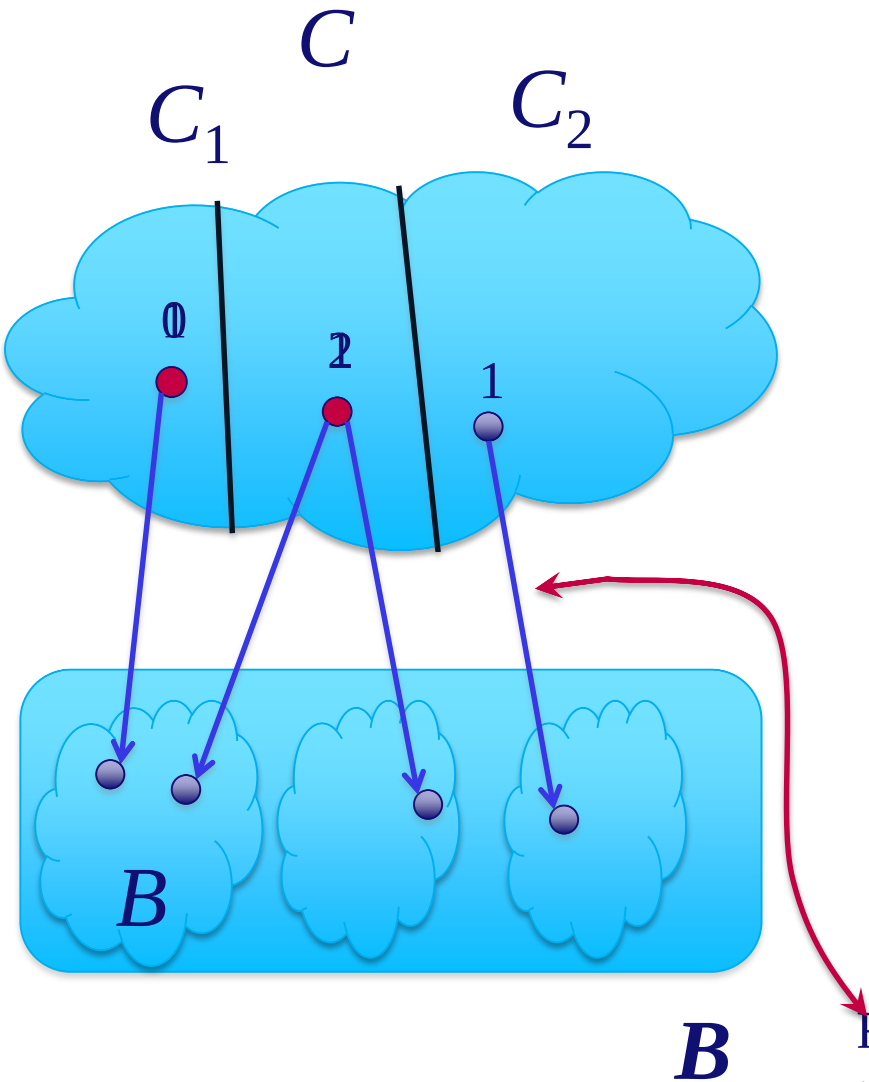


Splitting wrt. a block B in constellation $\boldsymbol{B}$.

Select a B such that $|B|<1/2|\cup\boldsymbol{B}|$.

Check whether C is splittable wrt. B.

Constellation.



Splitting wrt. a block B in constellation ***B*.**

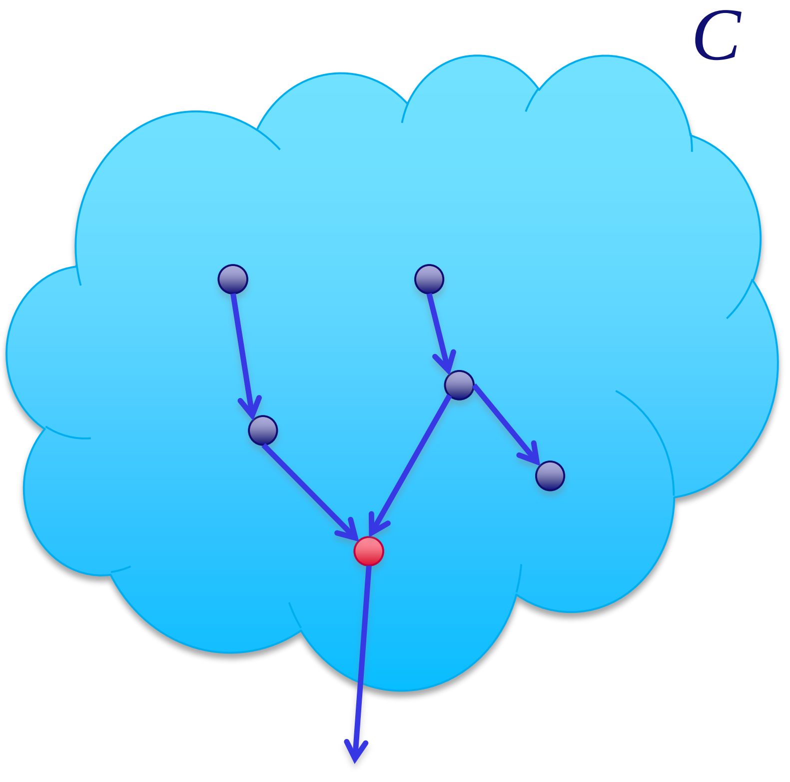
Select a B such that $|B|<1/2|\cup \boldsymbol{B}|$.

Check whether C is splittable wrt. B.

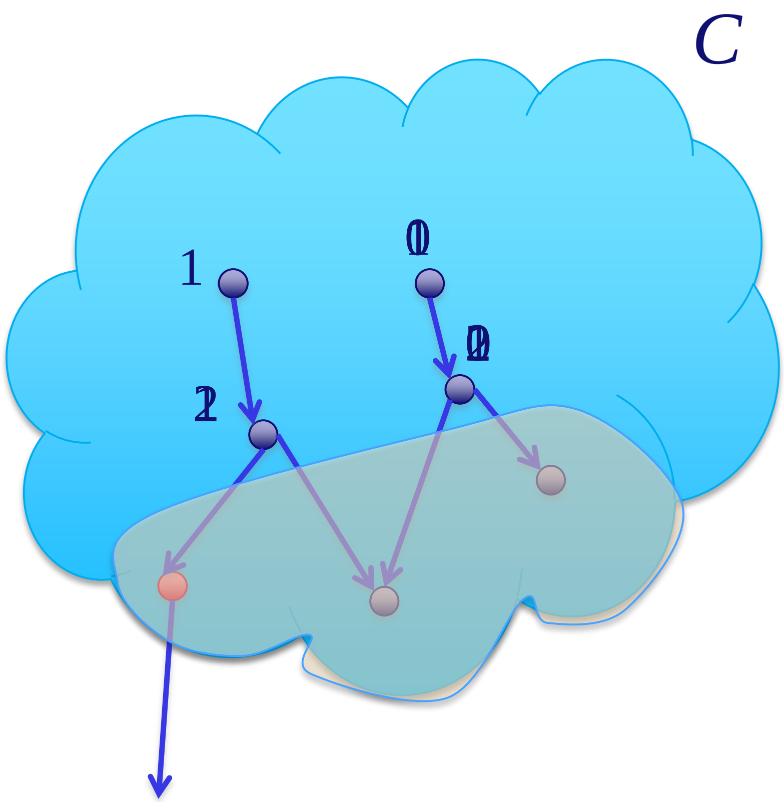
Check whether C_1 is splittable wrt. $\cup \boldsymbol{B} \setminus B$.

Recall in every state how many transitions there are that state to constellation ***B***.

Extend markings backward.



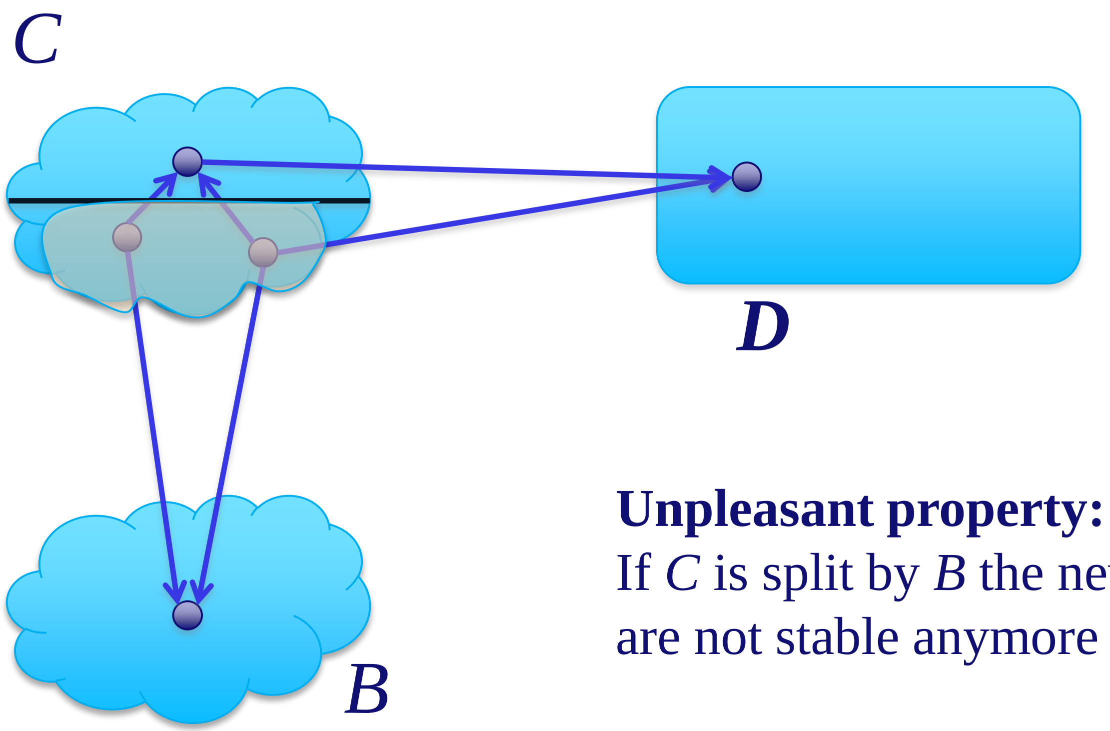
Extend nonmarkings backward.



Recall in each state
the number of outgoing
inert transitions.

Do both backward markings
in parallel until one terminates
or the marked block becomes
larger than 1/2|*C*|.

Complication.



Unpleasant property:
If C is split by B the new blocks of C are not stable anymore for constellation $\boldsymbol{D}$.

An unfortunately complex algorithm.

Algorithm is complex.

- 2 pages of data structures.
- 3 pages concise but precise description of the algorithm.

Two implementations, that took almost half a year to finish.

Implementation follows description precisely.

- Correctness and time complexity is proven for the core algorithm.
- Correctness of the implementation shown by extensive random testing against earlier implementations.
- Time complexity is verified by assigning time budgets in the algorithm.

And now people apply it to systems so big that memory is a problem.
Hence, we are working on a variant that does not translate to Kripke structures.

Tim Willems/Bas Luttik □ □ □ □ □

Conclusion.

1. The efficient algorithms for stuttering equivalence/branching bisimulation became even more efficient.

1. This is by far the most complex algorithm we ever encountered.

2. If you thank about implementing it, consider *stealing* the implementation from the mCRL2 toolset (www.mcrl2.org).

Technische Universiteit
Eindhoven
University of Technology