On-the-fly Stuttering in the Construction of Deterministic ω-Automata

Joachim Klein, Christel Baier

Institute of Theoretical Computer Science
Dresden University of Technology, Germany

http://www.ltl2dstar.de/

CIAA 2007

Context

Goal

- Deterministic ω-Automata (automata on infinite words) from LTL formulas

Problem

- LTL to nondeterministic Büchi automata:
  Exponential blow-up
- Determinization of nondet. Büchi automata: $2^{\mathcal{O}(n \log n)}$

In practice

- Implementation *ltl2dstar*, several heuristics
- Used e.g. by *LiQuor*, LTL model checker for Markov Decision Processes

Overview

Idea

- Use knowledge about *insensitiveness to stuttering* to avoid intermediate automata states during construction of deterministic automaton

Algorithm

- On-the-fly
- Can be used for deterministic automata with Rabin, Streett, Parity, Büchi acceptance
- During determinization, but also union, etc.
- Heuristic

Outline

Outline

Linear Temporal Logic (LTL)

Linear Temporal Logic (LTL)

- over set of Atomic Propositions AP ($\Sigma = 2^{AP}$)
- Propositional Logic: $\neg$, $\vee$, $\wedge$, $\rightarrow$, ...
- Temporal Operators:

U	Until	$try\ U\ success$
F	Finally, Eventually	$F\ success$
G	Globally, Always	$G \neg error$
X	Nextstep	$try \wedge X\ success$

Deterministic ω-Automaton

Deterministic ω-Automaton

$\mathcal{A} = (Q, \Sigma, \delta, q_0, \Omega)$, with

- Q: Set of states
- Σ: Alphabet
- δ: Transition function (deterministic, $Q \times \Sigma \rightarrow Q$)
- q_0: Starting state
- Ω: Acceptance condition

Acceptance/Language

- Accepts/rejects infinite words $\alpha \in \Sigma^\omega$
- $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$

Rabin acceptance (k acceptance pairs)

- $Acc = \{$, , $\}$
- Every state has acceptance signature: k-tuple $acc \in Acc^k$

- Run π over α is accepting iff $\exists i$:
  $acc[i] =$  infinitely often and
  $acc[i] =$  not infinitely often

Rabin acceptance (k acceptance pairs)

- $Acc = \{$⚪, 🟢, 🔴$\}$
- Every state has acceptance signature: k-tuple $acc \in Acc^k$

- Run π over α is accepting iff $\exists i$:
  $acc[i] =$ 🟢 infinitely often and
  $acc[i] =$ 🔴 not infinitely often

- Ordering: ⚪ $<$ 🟢 $<$ 🔴
- max$\{$acceptance signatures$\}$: maximum element-wise
  max$\{$⚪🟢, 🟢🔴$\}$ $=$ 🟢🔴
- Calculate $acc_{inf} = \max\{acc(\inf(\pi))\}$
- Run π is accepting iff $\exists$🟢 in acc_{inf}

Outline

1. Introduction
   - LTL, ω-Automata
2. Stuttering deterministic ω-automata
   - Stuttering
   - Stuttered Construction
   - Partial Stuttering
   - Experimental Results
3. Summary

Stuttering

Stutter Equivalence

Stutter equivalent words differ only by finite repetition (stuttering) of symbols.

$$a\ a\ a \mid b\ b \mid a\ a \mid b \ldots$$
$$a\ a \mid b \mid a\ a\ a \mid b\ b \ldots$$
$$a \mid b\ b\ b\ b \mid a\ a \mid b \ldots$$

Closure under stuttering

Language $\mathcal{L}$ is closed under stuttering iff any two stutter equivalent words are both accepted or both rejected.

Stutter Insensitiveness

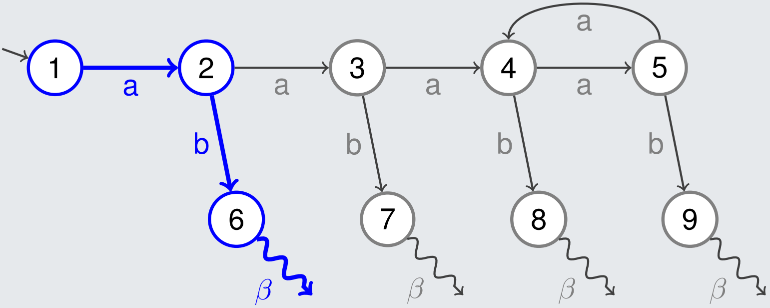
Stuttering Insensitiveness

- LTL formula, ω-automaton insensitive iff language is closed under stuttering
- Important property for e.g. partial-order reduction in model checking

How can we known?

- In general: PSPACE-complete in formula size
- All formulas without NextStep (X) are insensitive

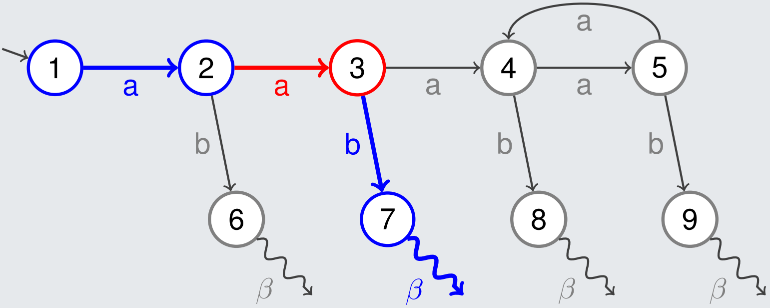
Run in stutter insensitive DRA



Input word α = abβ

Automaton is stutter insensitive $\Rightarrow$
Stuttering does not change acceptance of α

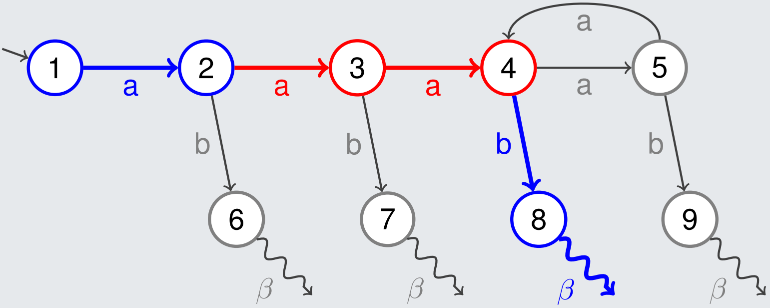
Run in stutter insensitive DRA



Input word α = aabβ

Automaton is stutter insensitive $\Rightarrow$
Stuttering does not change acceptance of α

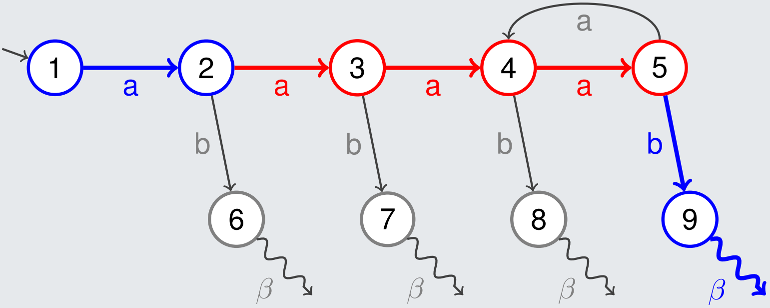
Run in stutter insensitive DRA



Input word α = aaabβ

Automaton is stutter insensitive $\Rightarrow$
Stuttering does not change acceptance of α

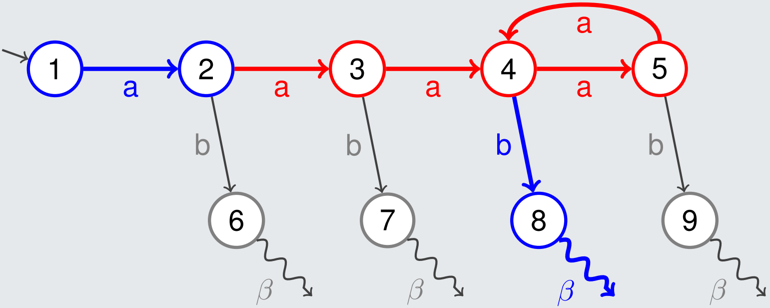
Run in stutter insensitive DRA



Input word α = aaaabβ

Automaton is stutter insensitive $\Rightarrow$
Stuttering does not change acceptance of α

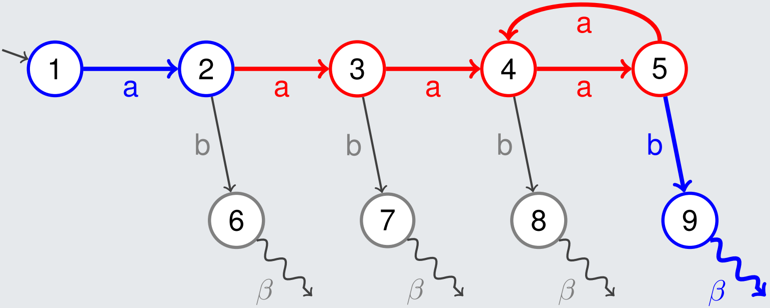
Run in stutter insensitive DRA



Input word α = aaaaabβ

Automaton is stutter insensitive $\Rightarrow$
Stuttering does not change acceptance of α

Run in stutter insensitive DRA



Input word α = aaaaaabβ

Automaton is stutter insensitive $\Rightarrow$
Stuttering does not change acceptance of α

Outline

1. Introduction
   - LTL, ω-Automata
2. Stuttering deterministic ω-automata
   - Stuttering
   - Stuttered Construction
   - Partial Stuttering
   - Experimental Results
3. Summary

Stuttered automaton

Stuttered Construction

Original DRA $\mathcal{A} = (Q, \Sigma, \delta, q_0, \Omega)$

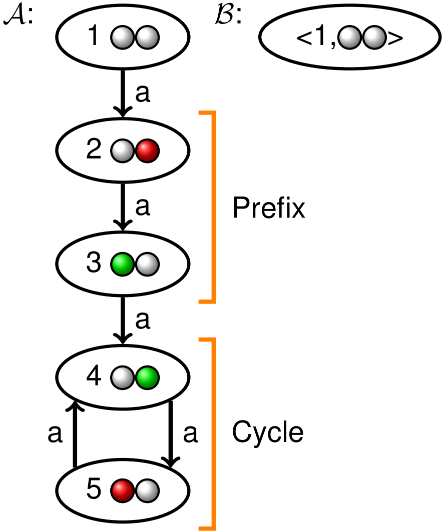
Stuttered DRA $\mathcal{B} = (Q', \Sigma, \delta', q_0', \Omega')$

- $Q' = Q \times Acc^k$
- $q_0' = < q_0, acc(q_0) >$
- Ω': according to the acceptance signature stored in the state

Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

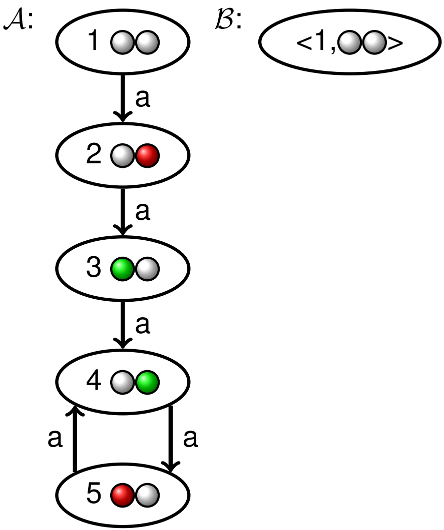
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

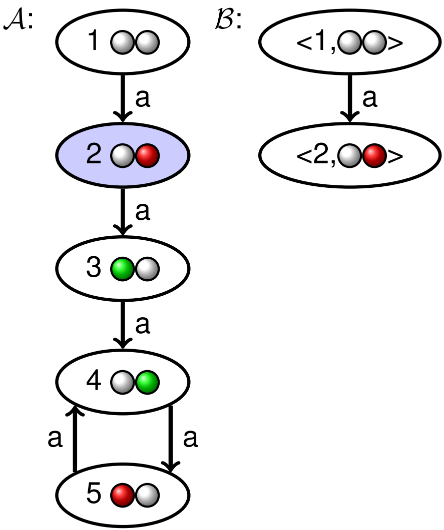
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

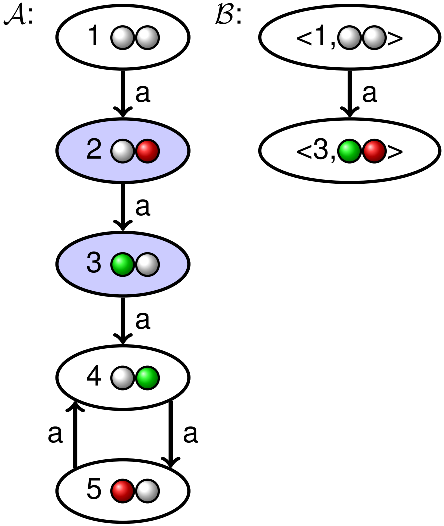
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

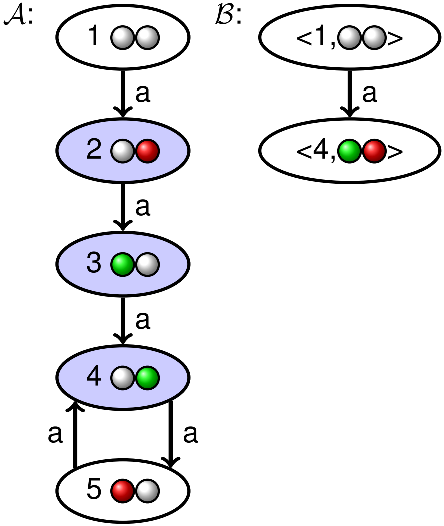
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

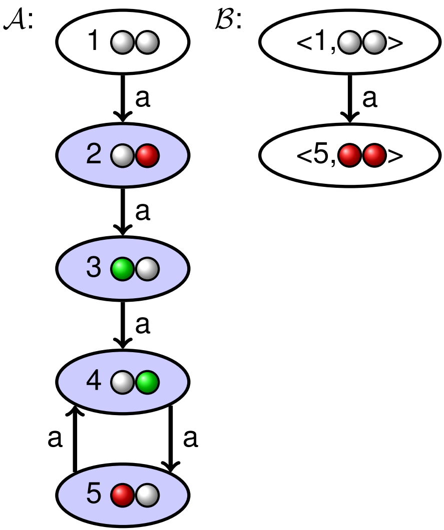
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(<q, acc>, a) = ?$

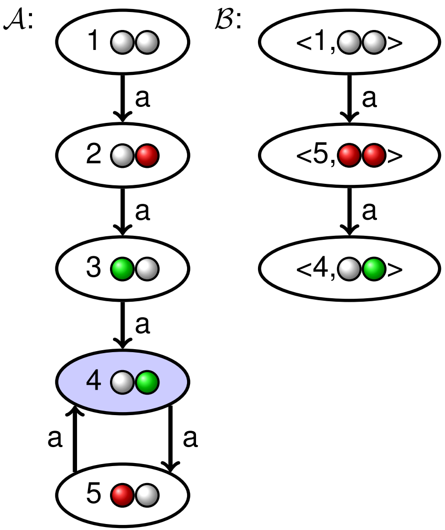
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

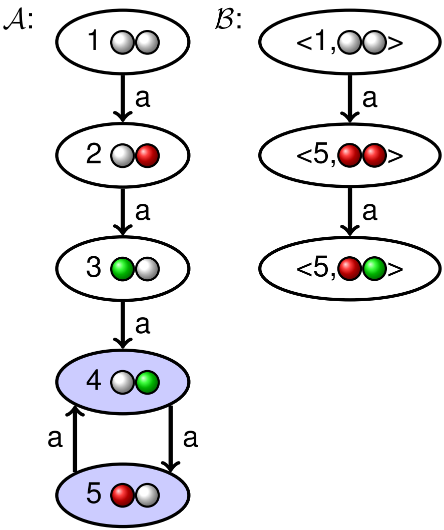
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

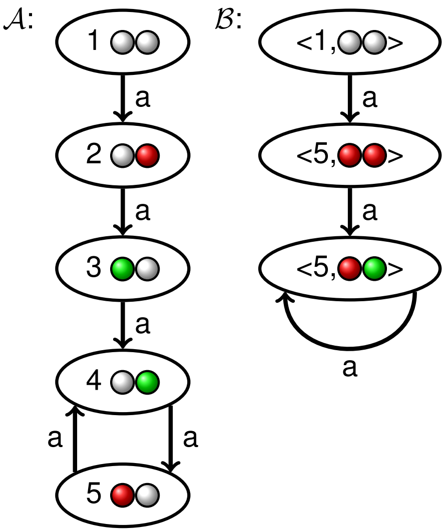
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

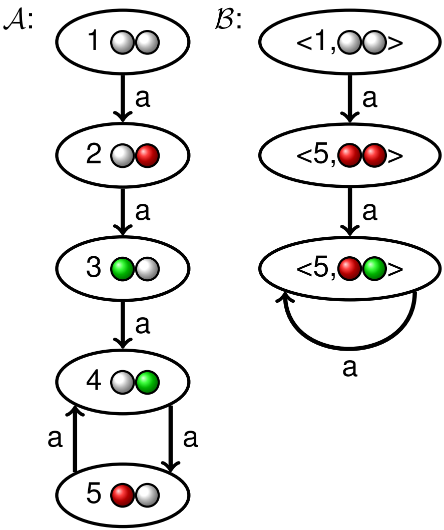
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \dots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

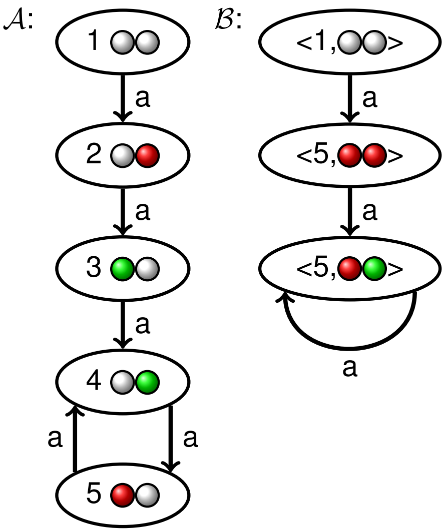
- $\delta(q, a^i)$ in $\mathcal{A}$, $i = 1, 2, \ldots$
- Prefix and cycle
- Pick a canonical cycle state
- Stutter as far as possible and then to canonical state
- Merge acc. signatures with max operator



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

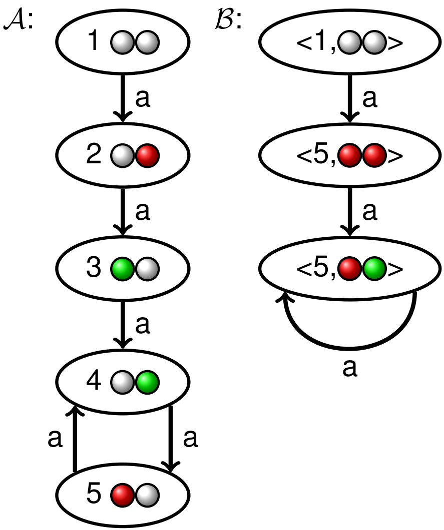
- If acc. sig. of cycle “dominates” the prefix acc. sig., prefix and cycle states collapse
- Prefix state can not always be avoided



Stuttered transition function δ'

- $\delta'(< q, acc >, a) = ?$

- If acc. sig. of cycle "dominates" the prefix acc. sig., prefix and cycle states collapse
- Prefix state can not always be avoided
- $\mathcal{B}$ can be larger than $\mathcal{A}$



Outline

Using Partial Stuttering

Partial Stuttering

- Only stuttering of certain symbols from Σ is allowed.
- For formula, determine stutter insensitive symbols $S \subseteq \Sigma$
- Use stutter construction only on transitions with $\sigma \in S$

Determining Stutter Insensitiveness

- Prototypical implementation
- Checking each $\sigma \in \Sigma$
- NBA emptiness check of product automaton of stutter closure NBA and complement formula NBA

Outline

1. Introduction
   - LTL, ω-Automata
2. Stuttering deterministic ω-automata
   - Stuttering
   - Stuttered Construction
   - Partial Stuttering
   - Experimental Results
3. Summary

Benchmark

Formulas

Literature 39 Formulas (25 without X)

- Etessami, Holzmann:
  Optimizing Büchi Automata
- Somenzi, Bloem:
  Efficient Büchi automata from LTL formulae

Pattern 55 Formulas (30 without X)
`http://patterns.projects.cis.ksu.edu/`

Random 1000 Random Formulas (415 without X)
Generated with the Test-Bench `lbtt`

- LTL to NBA translator: ltl2ba
- Generating det. Rabin automata (Safra's algorithm)

Results (without X)

Formulas	DRA	stuttered DRA	Time	
25 Literature	278	168 (-40%)	0.3s /	0.3s
30 Patterns	311	189 (-39%)	0.3s /	0.3s
415 Random	1,820	1,499 (-18%)	3.9s /	4.0s

Sum of the number of states of the automata, running time

Results (with partial stuttering)

Formulas	DRA	stuttered DRA	Time	
14 Literature	107	79 (-26%)	0.3s /	6.8s
25 Patterns	103,318	17,731 (-83%)	207.5s /	99.4s
585 Random	3,441	3,081 (-11%)	5.6s /	10.6s

Sum of the number of states of the automata, running time

Results (with partial stuttering)

Formulas	DRA	stuttered DRA	Time	
14 Literature	107	79 (-26%)	0.3s /	6.8s
25 Patterns	103,318	17,731 (-83%)	207.5s /	99.4s
585 Random	3,441	3,081 (-11%)	5.6s /	10.6s

Sum of the number of states of the automata, running time

Formulas	Calc. S	Calc. DRA	Avg. $\lvert S\rvert/\lvert\Sigma\rvert$
14 Literature	6.5s	0.3s	75.9%
25 Patterns	15.5s	83.9s	92.7%
585 Random	3.8s	6.8s	49.6%

Summary

- Use stutter insensitiveness to generate smaller deterministic ω-automata in practice
- Partial stuttering where not all symbols are stutter insensitive
- Calculating set S for formulas with X operator is expensive but feasible in practice

http://www.ltl2dstar.de/