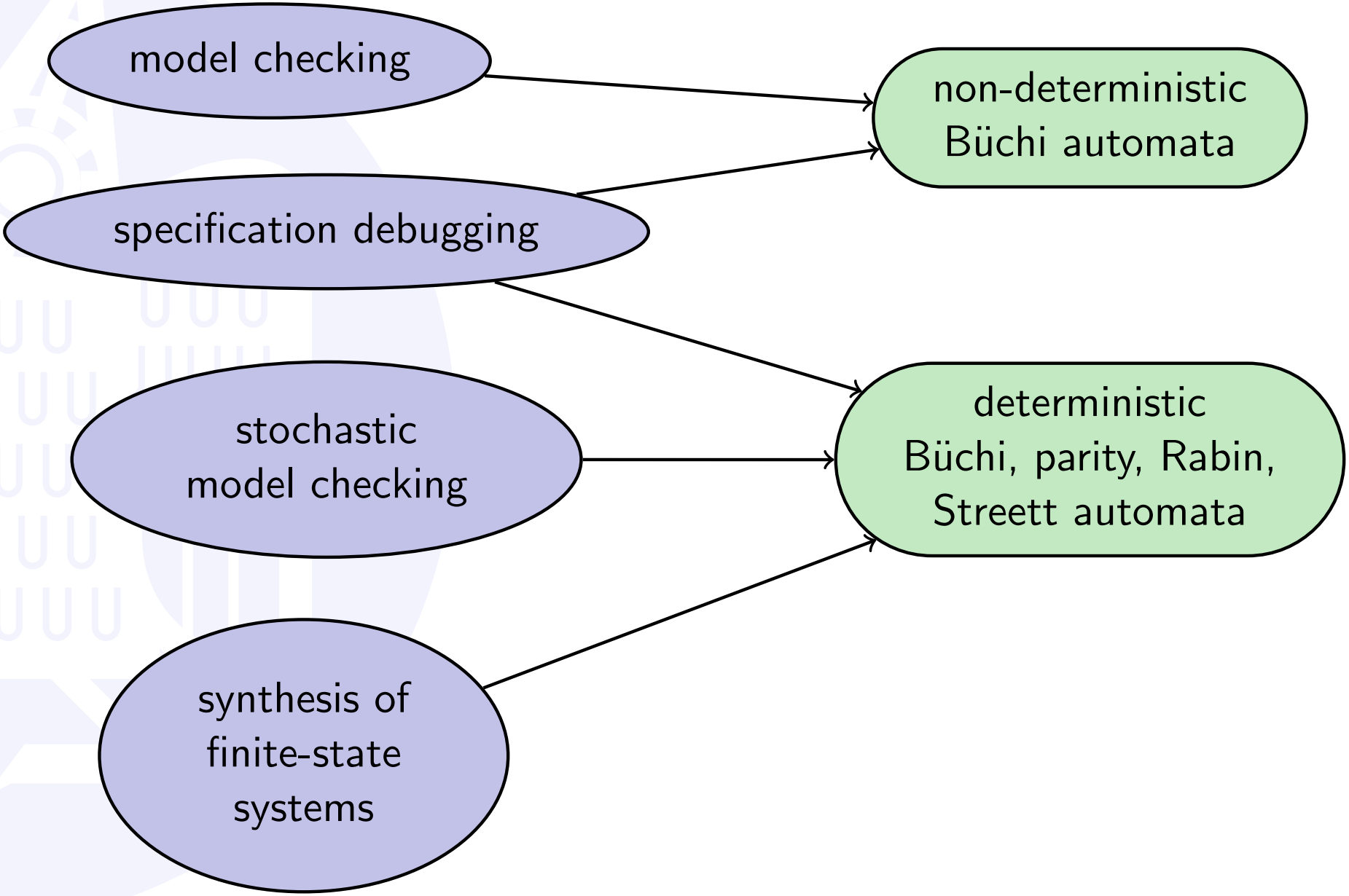
Minimising Deterministic Büchi Automata Precisely using SAT Solving

Rüdiger Ehlers

Saarland University, Reactive Systems Group

SAT 2010 – July 14, 2010

Formal methods & ω-automata theory



On ω-automata (1/2)

Purpose

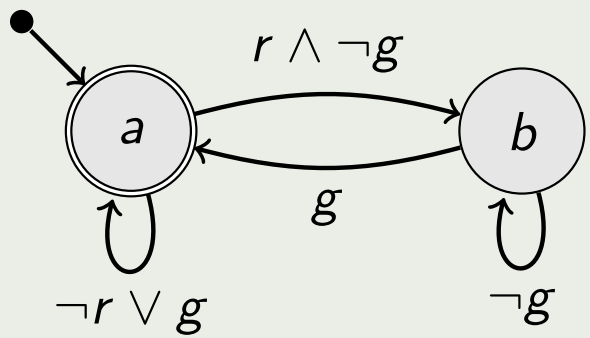
ω-automata accept or reject infinite words. In model checking, these normally correspond to runs of a system.

Example system

A run of a gate-keeper with $\Sigma = 2^{\{r,g\}}$:

$$w = \begin{pmatrix} r=1 \\ g=0 \end{pmatrix} \begin{pmatrix} r=0 \\ g=1 \end{pmatrix} \begin{pmatrix} r=0 \\ g=0 \end{pmatrix} \cdots$$

Example Büchi automaton



ω-automata types

Branching types: deterministic, non-deterministic, universal, alternating
Acceptance conditions: safety, Büchi, co-Büchi, parity, Rabin, Streett

Deterministic Büchi automata (DBAs)

Properties of DBAs

- Can be used for all purposes for which non-determinism is problematic (Stochastic model checking, synthesis, ...)
- Cannot express all ω-regular properties (8/12 in our benchmark suite [EH00])
- Non-deterministic Büchi automata can be exponentially more succinct.
- DBAs are guaranteed to be not larger than equivalent deterministic parity or Rabin automata.

DBAs in synthesis (from temporal logic formulas)

Heavily used in recent approaches:

- Generalized reactivity(1) synthesis [PPS06]
- Trigger property synthesis [KV10]

Minimisation of ω-automata

Complexity of the problems

Acc. cond. type	Non-deterministic	deterministic
Safety	PSPACE-comp.	in PTIME
Büchi	PSPACE-comp.	unknown
co-Büchi	PSPACE-comp.	unknown
Parity	PSPACE-comp.	unknown
Rabin	PSPACE-comp.	unknown
Streett	PSPACE-comp.	unknown

Minimisation of ω-automata

Complexity of the problems

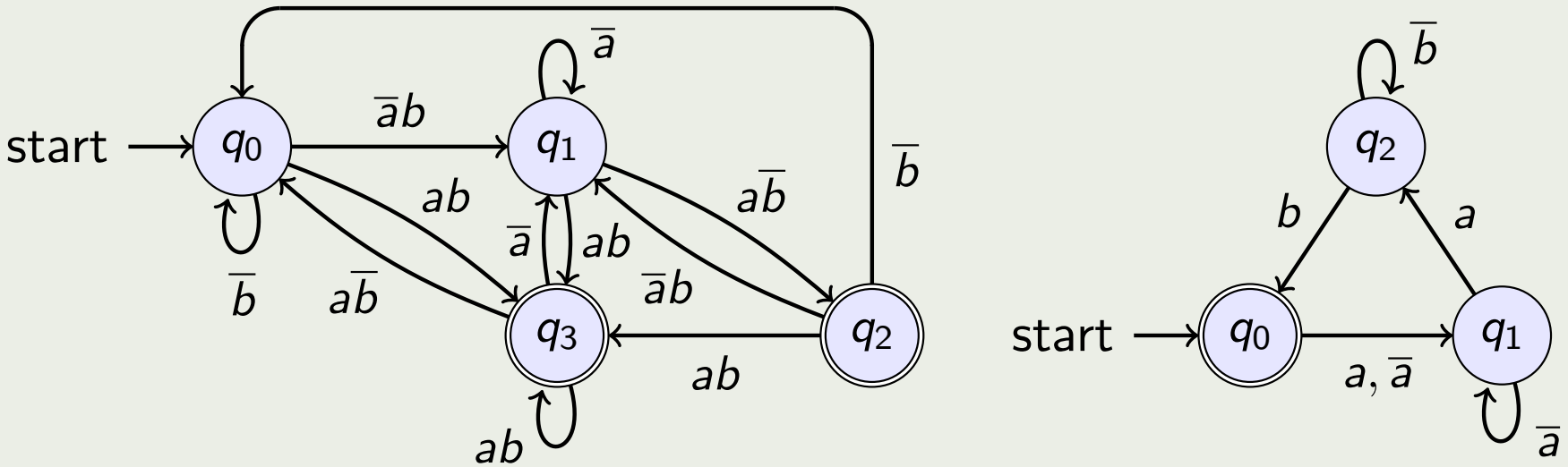
Acc. cond. type	Non-deterministic	deterministic
Safety	PSPACE-comp.	in PTIME
Büchi	PSPACE-comp.	in NP
co-Büchi	PSPACE-comp.	in NP
Parity	PSPACE-comp.	unknown
Rabin	PSPACE-comp.	unknown
Streett	PSPACE-comp.	unknown

Main contribution of this paper

For co-Büchi/Büchi automata, the problem is efficiently solvable by using a modern SAT solver!

Minimising DBA is difficult

Example for the LTL formula $(\mathrm{GF}a) \wedge (\mathrm{GF}b)$



Minimising DBAs using SAT solving

Reduction to a decision problem

Basic idea: Repeatedly ask the question whether there is an equivalent DBA with **one less** state.

Encoding the decision problem as a SAT instance

- **Guess** a smaller automaton
- **Guess** the reachable states in the product of the smaller automaton and the original automaton
- **Examine** if the product witnesses correctness
- Perform some **symmetry reduction**

The Encoding - Part 1 (smaller automaton)

Definition of DBA

Formal definition of a DBA: $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with:

- $\delta : Q \times \Sigma \to Q$
- $F \subseteq Q$
- W.l.o.g. we assume that $Q = \{0, 1, \ldots, n-1\}$ and $q_0 = 0$

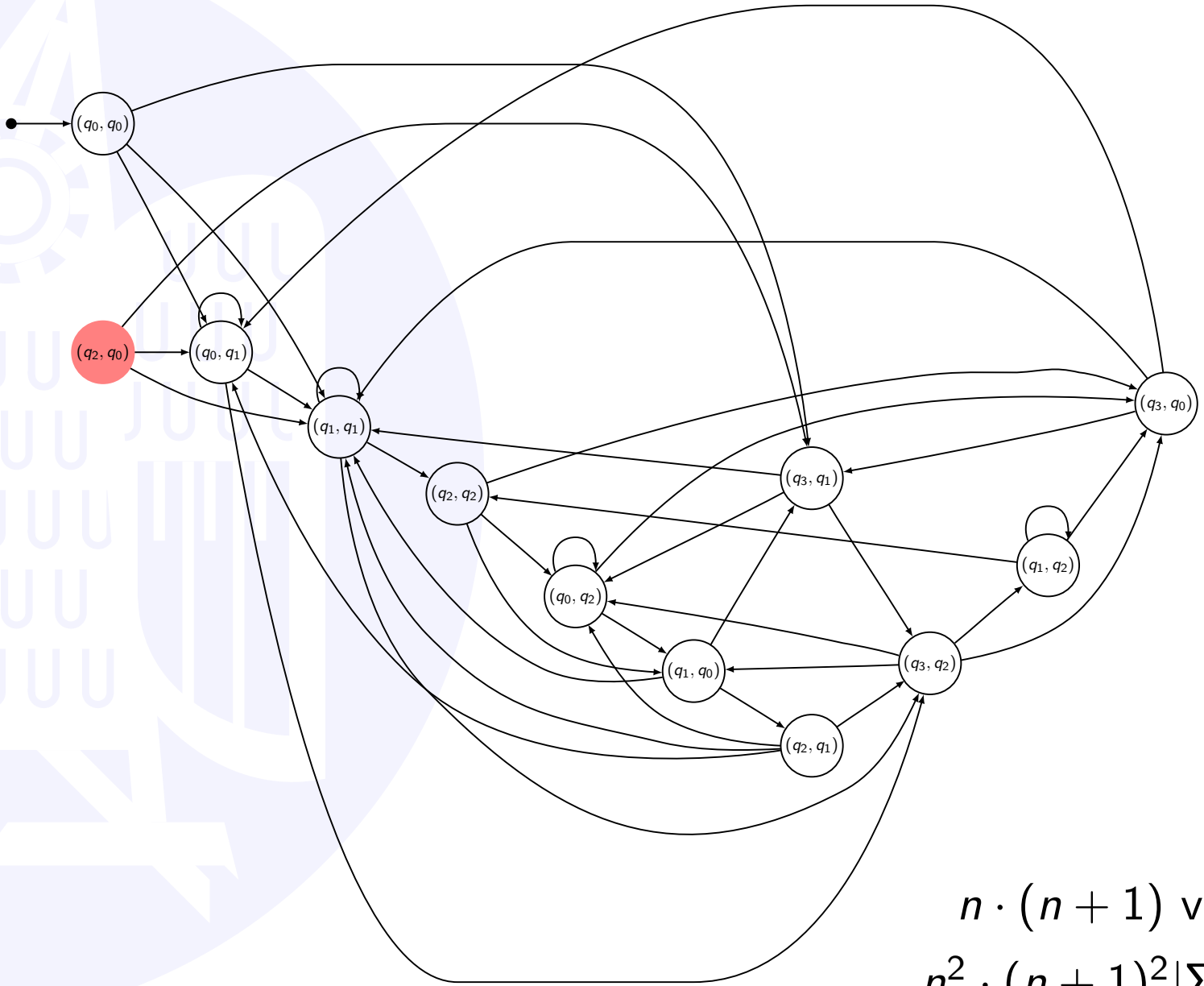
SAT Encoding

We have the following number of variables in the SAT instance:

$$\underbrace{n}_{\text{for } F} + \underbrace{n^2 \cdot |\Sigma|}_{\text{for } \delta}$$

We need $n \cdot |\Sigma|$ clauses for making sure that $\delta(q, x)$ is defined for all $x \in \Sigma$ and $q \in Q$.

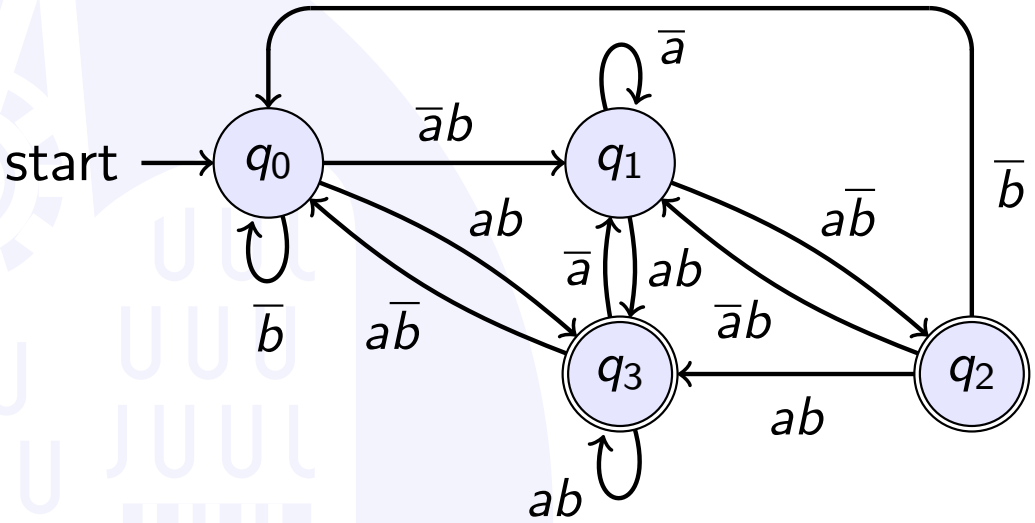
The Encoding - Part 2 (reachable states in the product)

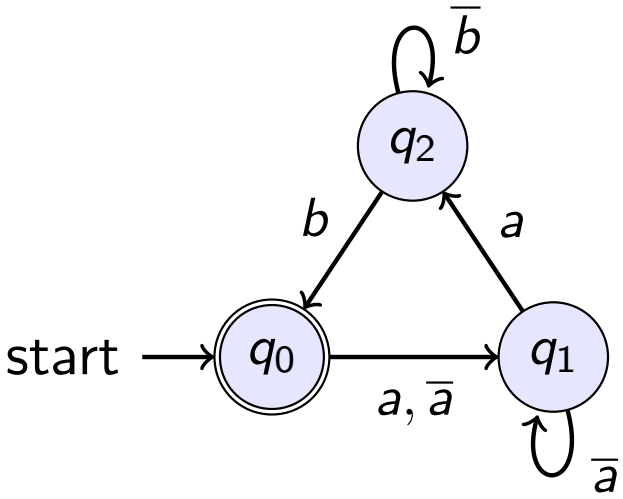


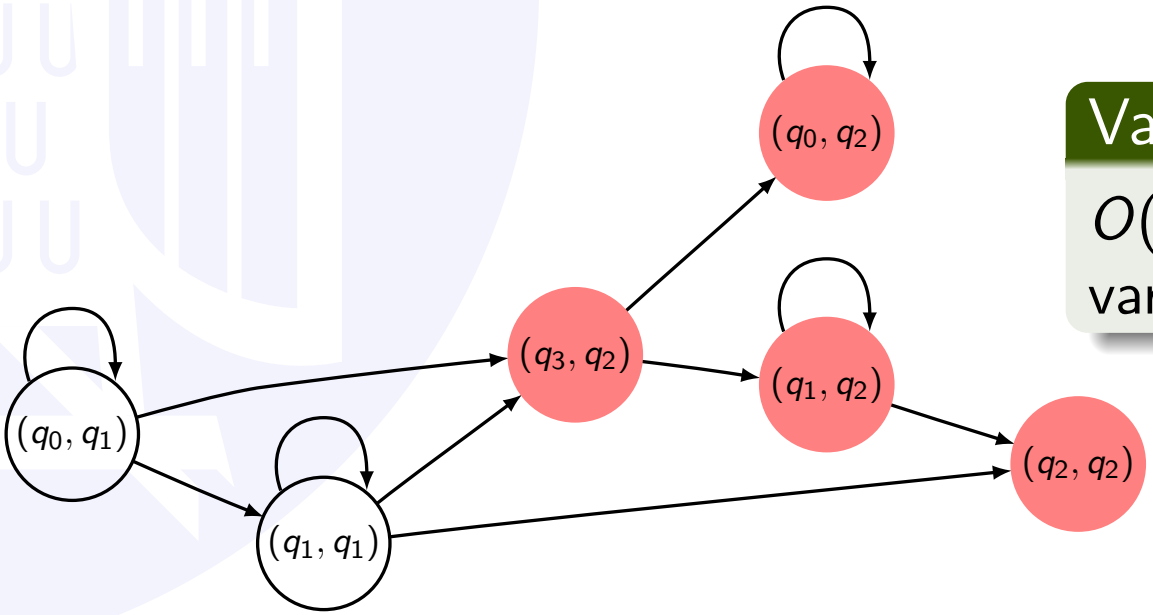
$n \cdot (n+1)$ variables

$n^2 \cdot (n+1)^2 |\Sigma|$ clauses

The Encoding - Part 3 (examining the product)







Var's and Clauses

$O(n^4)$ many clauses and variables

The Encoding - Summing it up

General properties

- Number of variables and clauses: $O(n^4 \cdot |\Sigma|)$
- "Almost of Horn-type": only step 1 leads to non-Horn clauses

Symmetry reduction

In addition, we perform some (cheap) symmetry reduction (adds no variables or non-unit clauses).

Experiments - Setting

Properties

Example specifications:

- 8/12 LTL properties from [EH00] representable as DBAs
- 13 additional examples from verification & synthesis

Tools

- Using `ltl2dstar v.0.5.1` [KB06] in conjunction with `ltl2ba v.1.1` [GO01] to obtain initial non-optimised deterministic Rabin automata equivalent to the input formulas.
- Converting these to Büchi automata whenever possible.
- Iteratively solving the resulting SAT instances with `picosat v.913` [Bie08]

Experiments - Results

LTL specification	# States		First instance		Total time
	From	To	# Vars	# Clauses	
$F(q \wedge X(pUr))$	3	3	94	697	0.01 s
$pUqUr \vee qUrUp \vee rUpUq$	3	3	112	699	0.01 s
$F(p \wedge X(q \wedge XFr))$	4	4	279	3785	0.01 s
$G(p \rightarrow qUr)$	5	3	852	13508	0.03 s
$pU(q \wedge X(rUs))$	5	5	780	26972	0.04 s
$F(p \wedge XF(q \wedge XF(r \wedge XFs)))$	5	5	780	26972	0.01 s
$GFp \wedge GFq \wedge GFr \wedge GFs \wedge GFu$	14	6	51467	13856026	63.03 s
$G(a \rightarrow Fb) \wedge Gc$	5	3	852	13508	0.03 s
$GF(a \rightarrow XXXb)$	7	2	3720	43457	0.08 s
$G(a \rightarrow Fb) \wedge G(\neg a \rightarrow F\neg b)$	8	4	5635	89511	0.14 s
$GF(a \leftrightarrow XXb)$	9	6	8760	168358	1.57 s
$G(a \rightarrow Fb) \wedge G(b \rightarrow Fc)$	10	5	14247	589925	0.98 s
$G(a \rightarrow XXXb)$	10	9	16623	295076	3.71 s
$G(a \rightarrow Fb) \wedge G(c \rightarrow Fd)$	15	6	72660	9926065	23.96 s
$GF(a \leftrightarrow XXXb)$	17	15	116912	4752970	3617.3 s*

Conclusion

Contributions

- Minimising DBAs can efficiently be done by using a SAT solver
- Bringing the benefits of SAT solving to areas in which they are not used so far (stochastic model checking, synthesis)
- Actually, the technique is applicable to parity, Streett and Rabin automata as well
- Tool and hard SAT instances available at:
  `http://react.cs.uni-saarland.de/tools/dbaminimizer` and
  `http://react.cs.uni-saarland.de/~ehlers/benchmarks`

A "wish list" for SAT solvers

- Allow streamed DIMACS input
- Delayed input instance simplification and symmetry breaking (during solving)

References

Armin Biere.
Picosat essentials.
JSAT, 4(2-4):75–97, 2008.

Kousha Etessami and Gerard J. Holzmann.
Optimizing Büchi automata.
In Catuscia Palamidessi, editor, *CONCUR*, volume 1877 of *LNCS*, pages 153–167. Springer, 2000.

Paul Gastin and Denis Oddoux.
Fast LTL to Büchi automata translation.
In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *CAV*, volume 2102 of *LNCS*, pages 53–65, 2001.

Joachim Klein and Christel Baier.
Experiments with deterministic ω-automata for formulas of linear temporal logic.
Theor. Comput. Sci., 363(2):182–195, 2006.

O. Kupferman and M.Y. Vardi.
Synthesis of trigger properties.
In *LPAR 2010*, 2010.

Nir Piterman, Amir Pnueli, and Yaniv Sa'ar.
Synthesis of reactive(1) designs.
In E. Allen Emerson and Kedar S. Namjoshi, editors, *VMCAI*, pages 364–380. Springer, 2006.