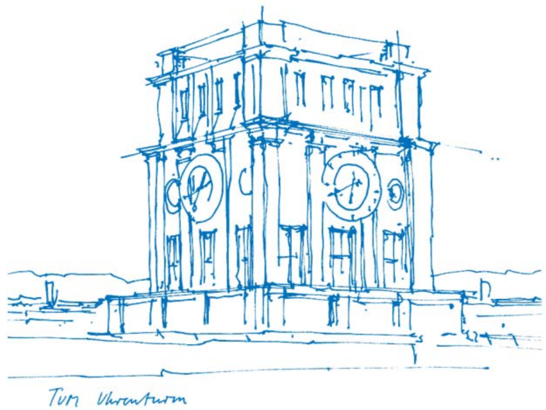
Limit-Deterministic Büchi Automata for Probabilistic Model Checking

Javier Esparza   Jan Křetínský

Stefan Jaax   Salomon Sickert

Technische Universität München



PROBABILISTIC MODEL CHECKING

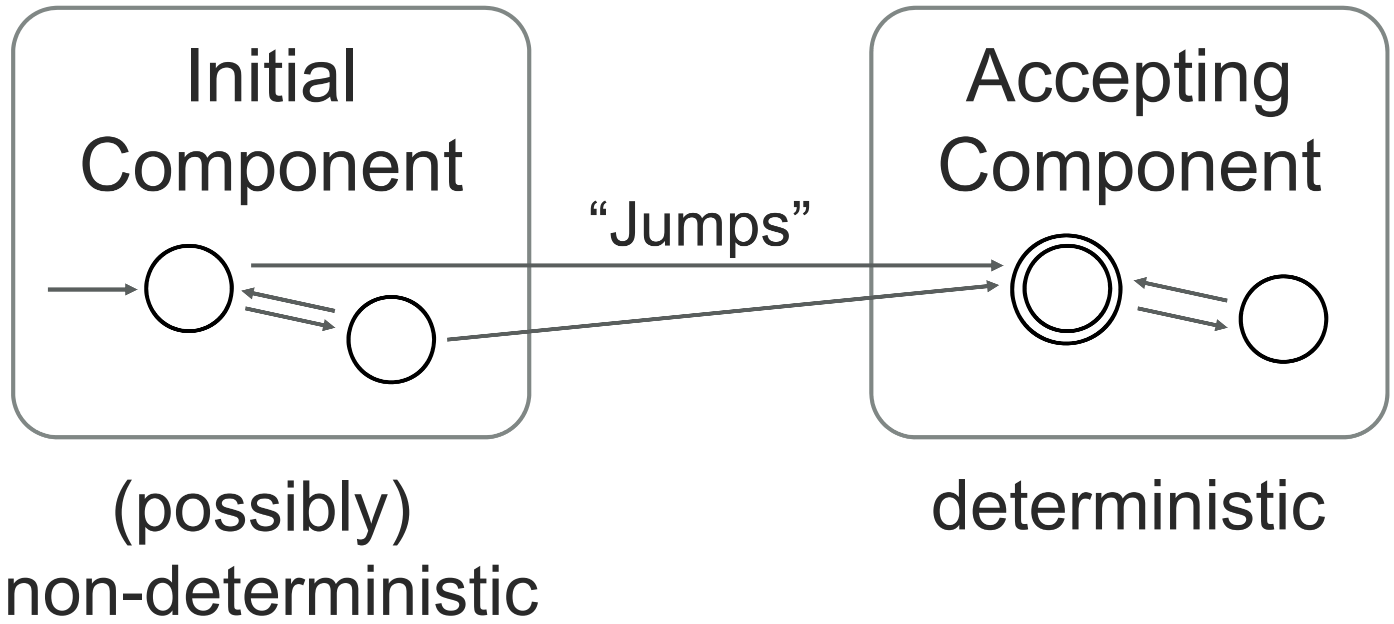
- Markov Decision Process (MDP) .

  At each state, a scheduler chooses a probability distribution, and then the next state is chosen stochastically according to the distribution.

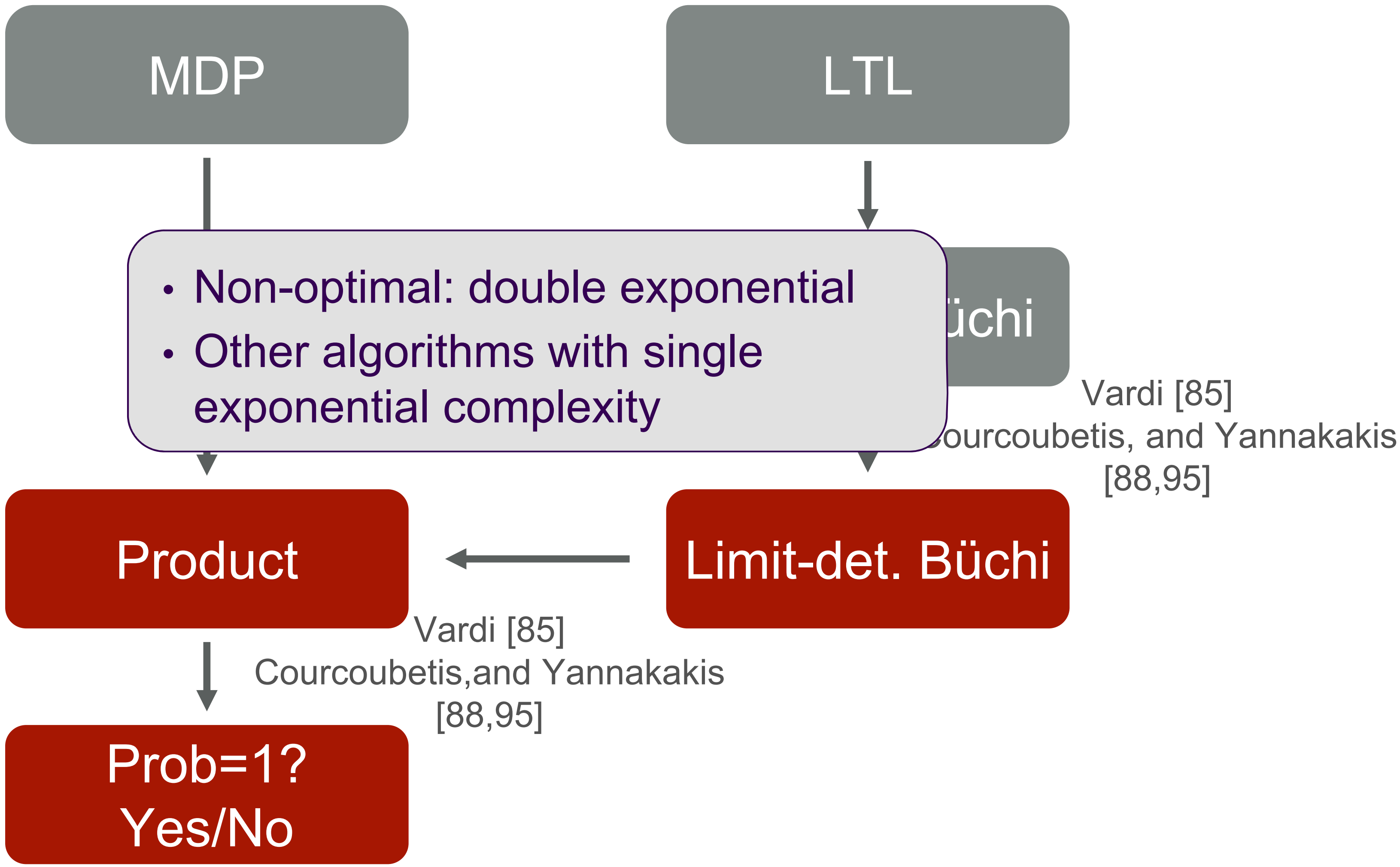
  Fixed scheduler: MDP → Markov chain

- Qualitative Model Checking:
  - Input: MDP, LTL formula
  - Does the formula hold for all schedulers with probability 1?
- Quantitative Model Checking:
  - Input: MDP, LTL formula, threshold c
  - Does the formula hold for all schedulers with probability at least c?

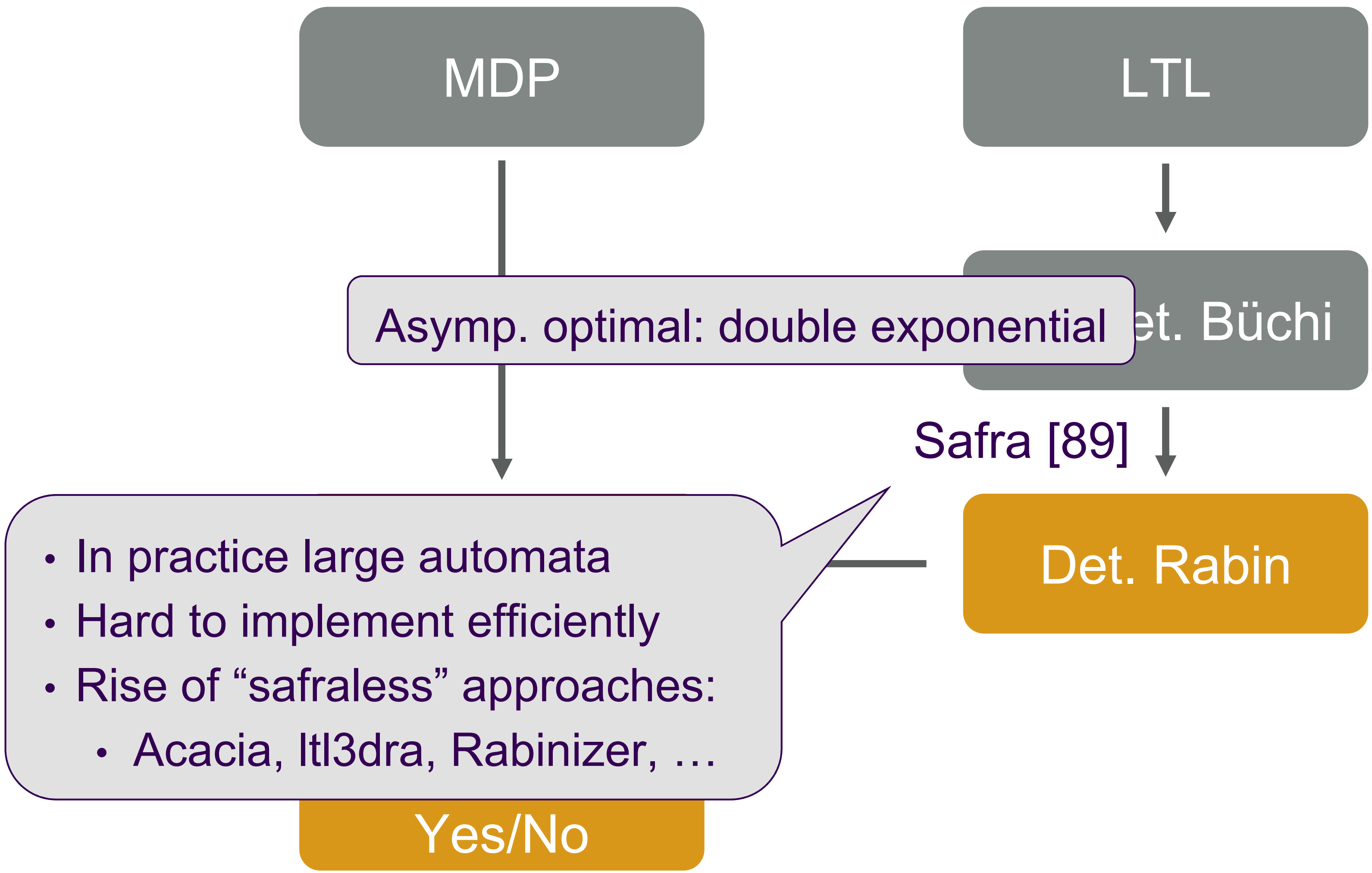
LIMIT-DETERMINISTIC BÜCHI AUTOMATA



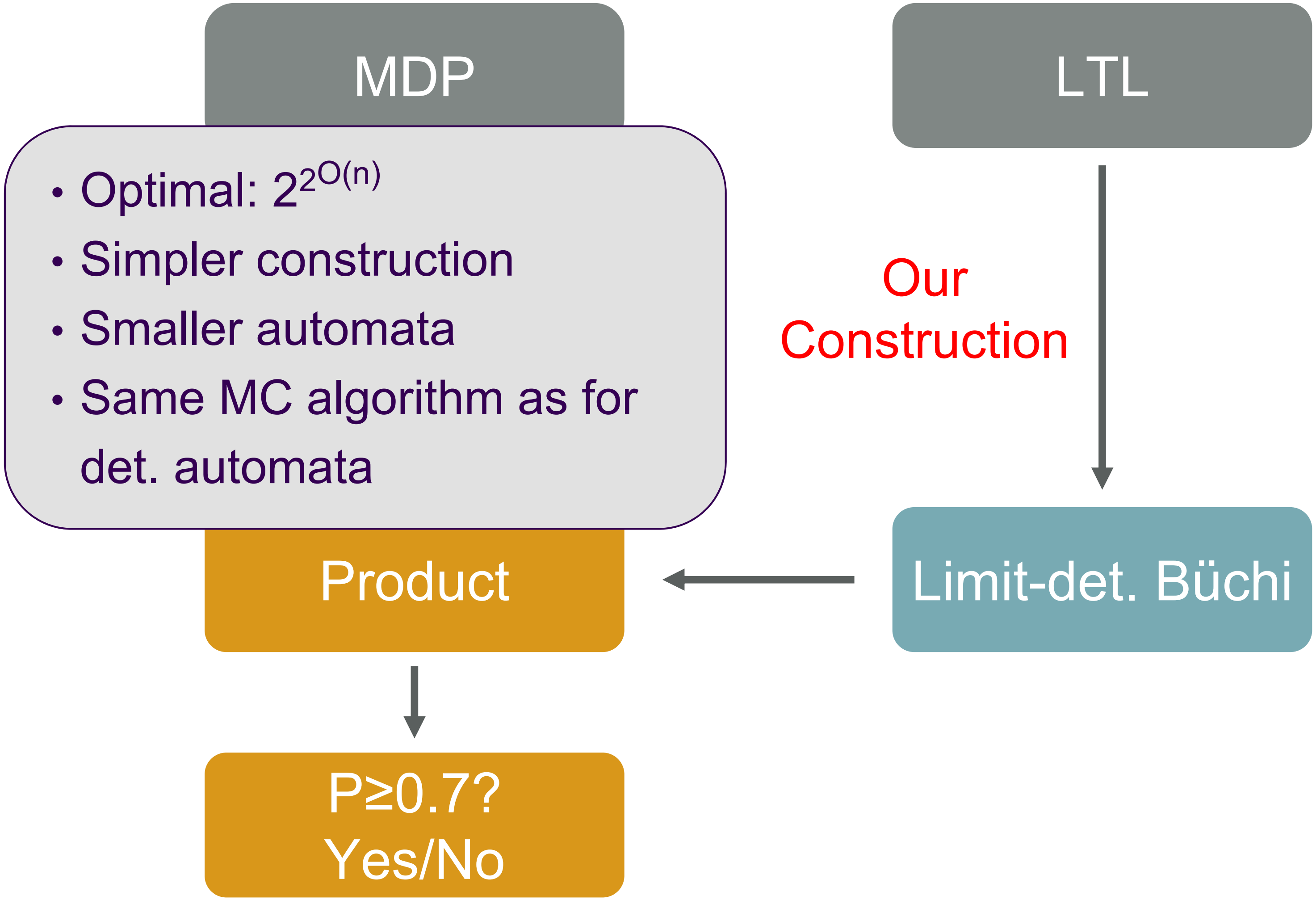
QUALITATIVE
PROB. MODEL CHECKING



QUANTITATIVE
PROB. MODEL CHECKING

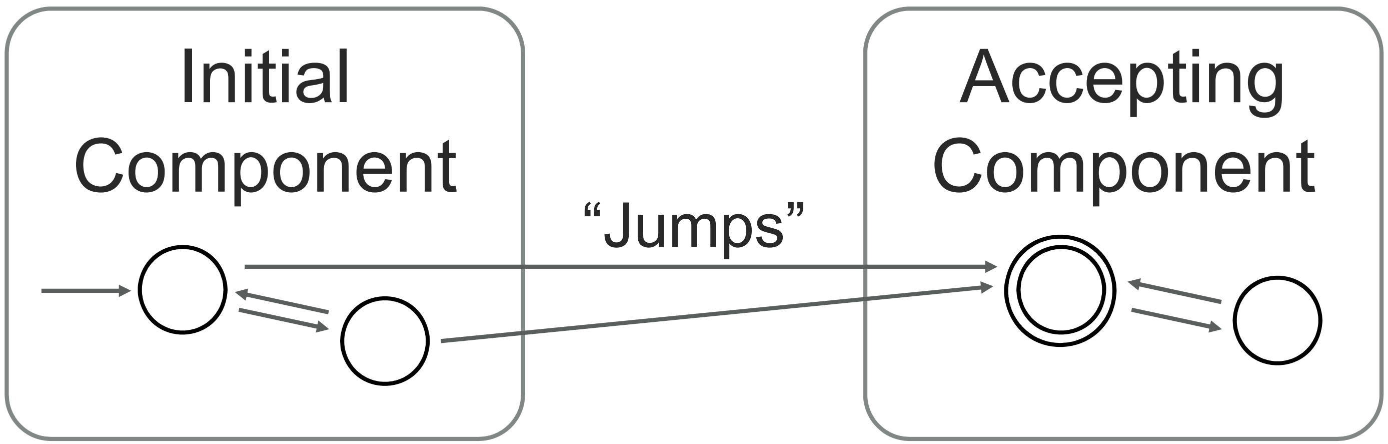


QUANTITATIVE PROB. MODEL CHECKING



LIMIT-DETERMINISM

In our construction:



~~non-deterministic~~ deterministic

deterministic

Every runs „uses“ nondeterminism at most once

PRELIMINARIES

- Linear Temporal Logic in Negation Normal Form

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \varphi\mathbf{U}\varphi \mid \mathbf{X}\varphi \mid \boxed{\mathbf{G}\varphi}$$

Only liveness operator.

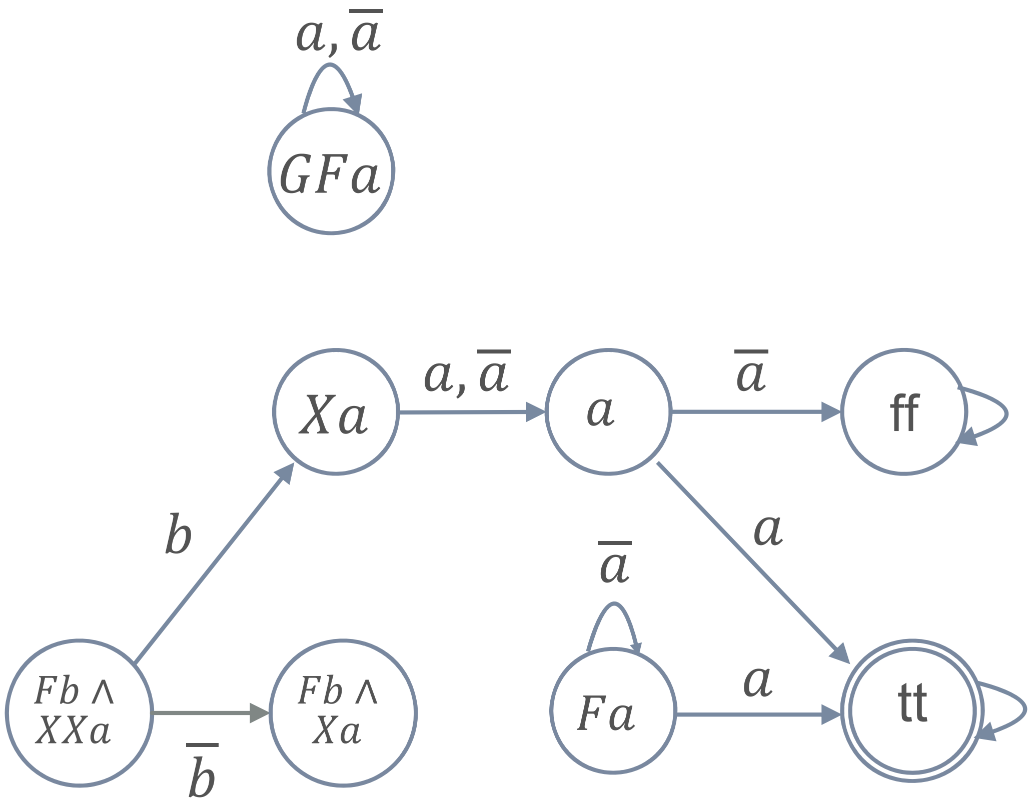
- Monotonicity of NNF:

if w satisfies φ

w' satisfies all the subformulas of φ satisfied by w, and perhaps more

then w' satisfies φ

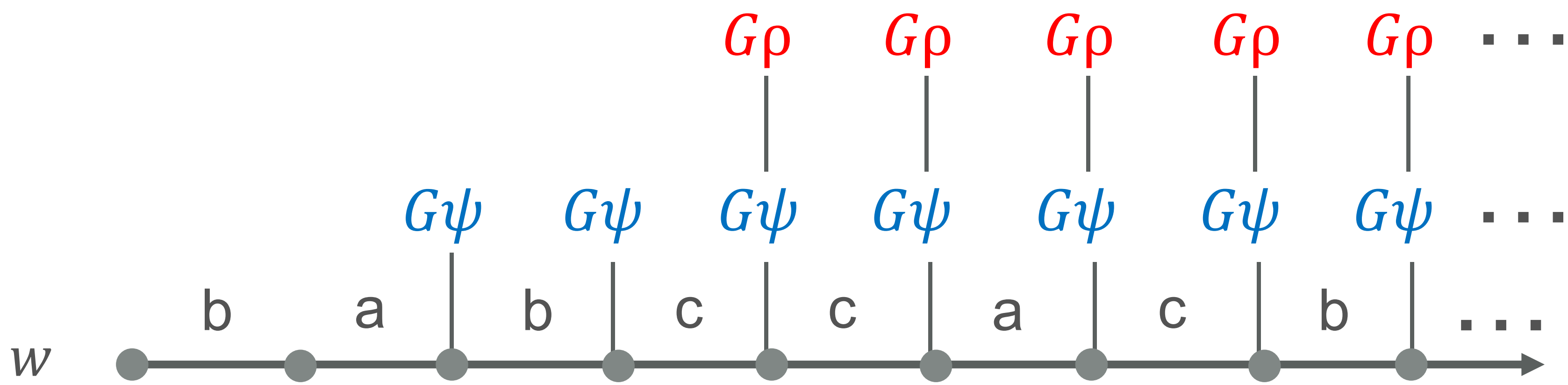
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON



- The automaton „tracks“ the property that must hold now for the original property to hold at the beginning
- Formulas with F, X, U: ✔
- Formulas with G: not good enough.

G-SUBFORMULAS

- Fix a formula φ and a word w.
  Let $G\psi$ be a G-subformula of φ.



- Informally: while reading the word w, the set of G-subformulas that hold cannot decrease, and eventually stabilizes to a set $\mathrm{True}Gs(w, \varphi)$.

SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to an accepting component.
- From each state ψ we add a jump for every set $\mathcal{G}$ of G-subformulas of ψ.
- „Meaning“ of a $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the word satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

  even if no other G-subformula of ψ ever becomes true.
- After the jump, the task of the accepting component is to „check that the guess is correct“, i.e., accept iff the guess is correct.

SECOND STEP: JUMPING

- „Meaning“ of the $\mathcal{G}$-jump at state ψ: The automaton „guesses“ that the rest of the run satisfies
  1. $\mathcal{G}$ (every formula of $\mathcal{G}$), and
  2. $\mathcal{G} \Rightarrow \psi$

  even if no other G-subformula of ψ ever becomes true.

- $w \vDash \varphi$ iff the automaton can make a right guess.
  - Right guess before suffix $\mathrm{w}' \rightarrow \mathrm{w}' \vDash \psi \rightarrow w \vDash \varphi$ (tracking!)
  - $w \vDash \varphi \rightarrow w' \vDash \mathrm{True}Gs(w, \varphi)$ for some suffix w' $\rightarrow$ jump before w' with $\mathcal{G} := \mathrm{True}Gs(w, \varphi)$ satisfies 1. and 2.

A DBA THAT CHECKS 1. & 2.

- Since DBA are closed under intersection, it suffices to construct two DBAs for 1. and 2.

CHECKING 2.

- „$\mathcal{G} \Rightarrow \psi$ holds even if no other G-subformula of ψ ever becomes true”
- Reduces to checking the G-free formula

$$\psi[\ \mathcal{G} \setminus \mathrm{tt}\ ,\ \overline{\mathcal{G}} \setminus \mathrm{ff}\ ]$$

- Example: $\psi = G(a \vee Fb) \wedge (Gc \vee Xd)$

$$\mathcal{G} = \{\, G(a \vee Fb)\, \}$$

  reduces to checking Xd

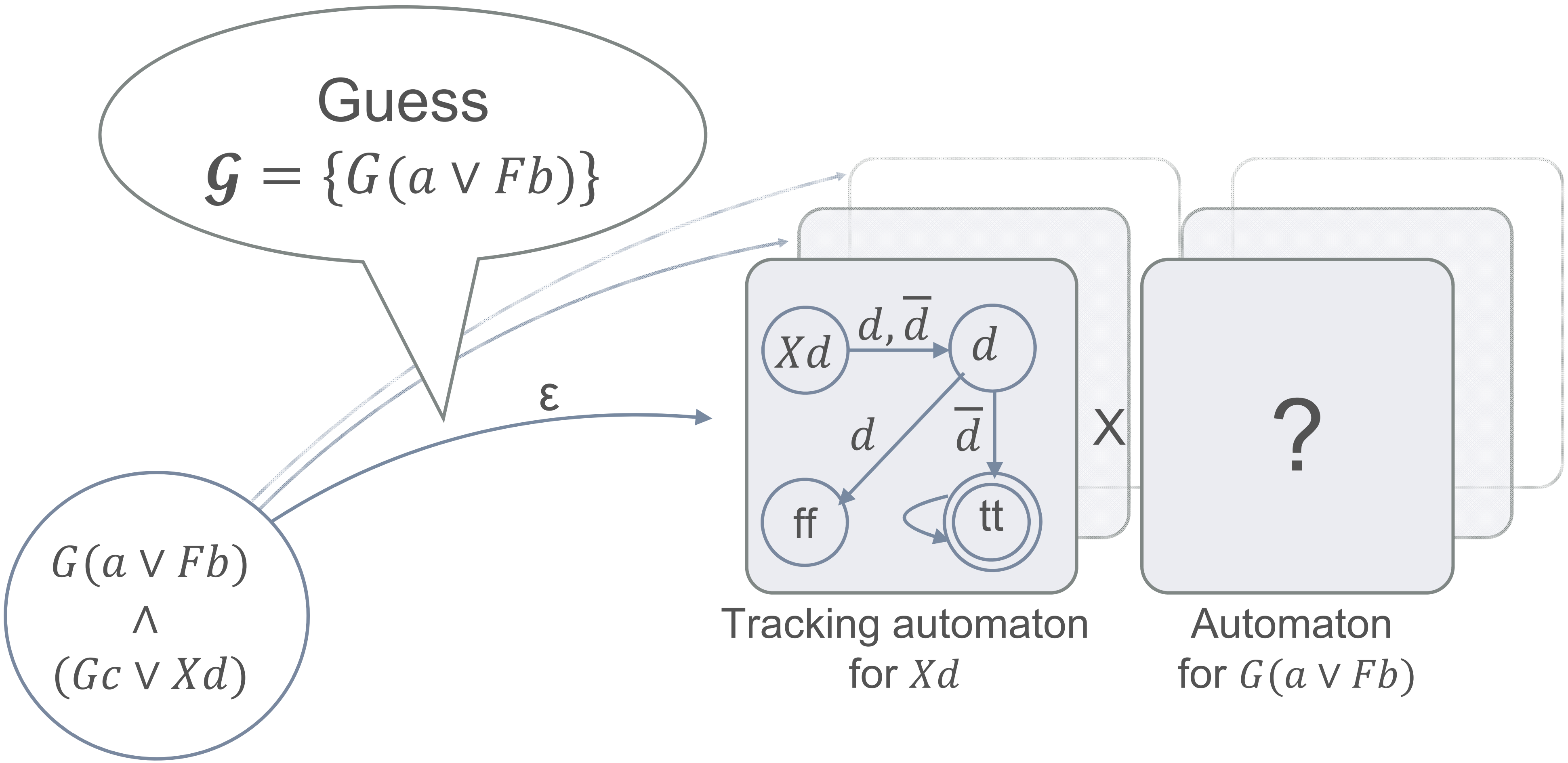
- Since the formula is G-free, use the tracking automaton.

CHECKING 1.

- „$\mathcal{G}$ holds even if no other G-subformula of ψ ever becomes true"
- Reduces to checking a formula $G\rho$ where ρ is G-free.
- Example: $\psi = Fc \wedge GF(a \wedge (Gb \vee FGc))$

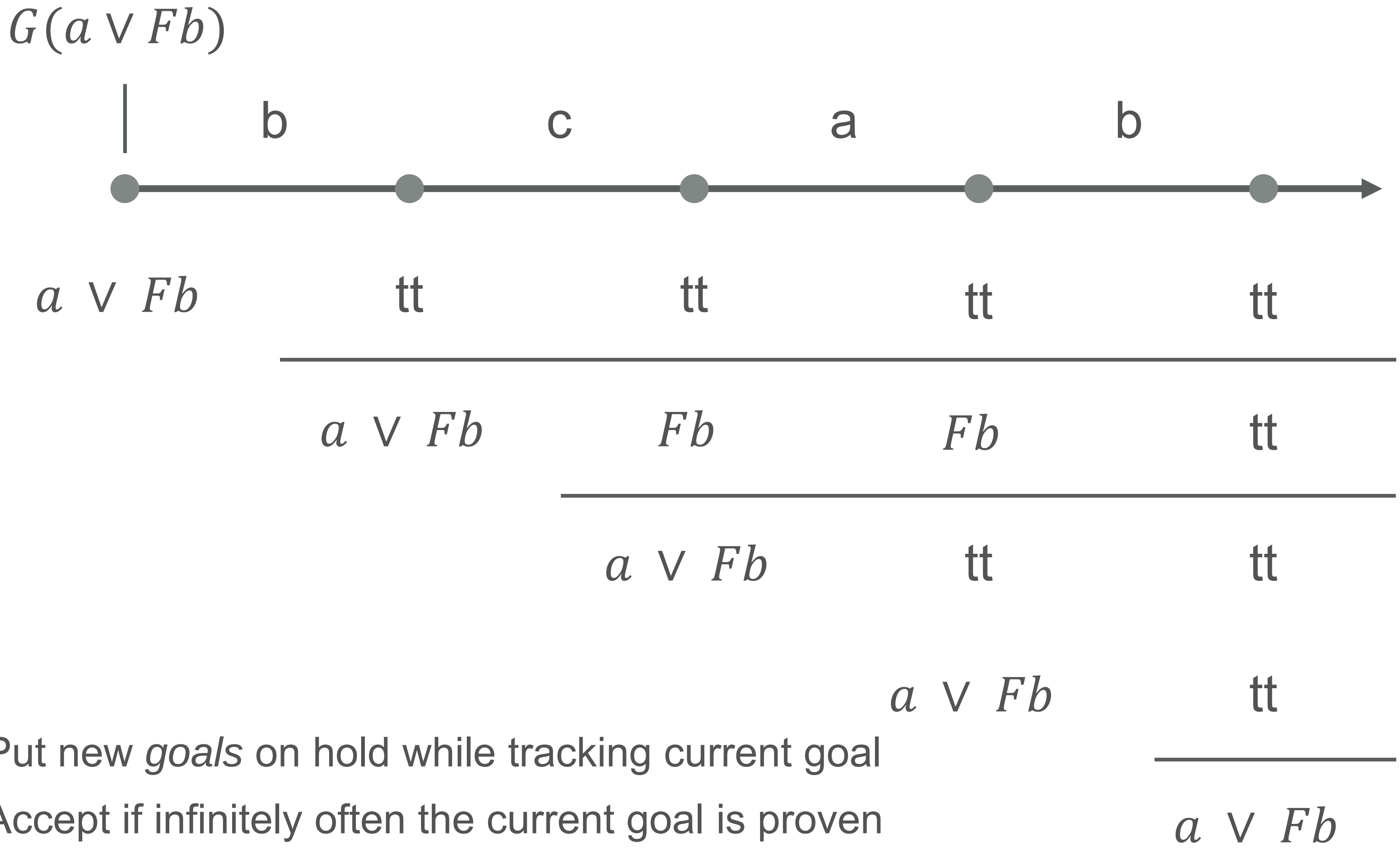
  $\mathcal{G} = \{ Gb \ , \ GF(a \wedge (Gb \vee FGc))\}$

  reduces to checking $Gb \wedge GFa \equiv G(b \wedge Fa)$



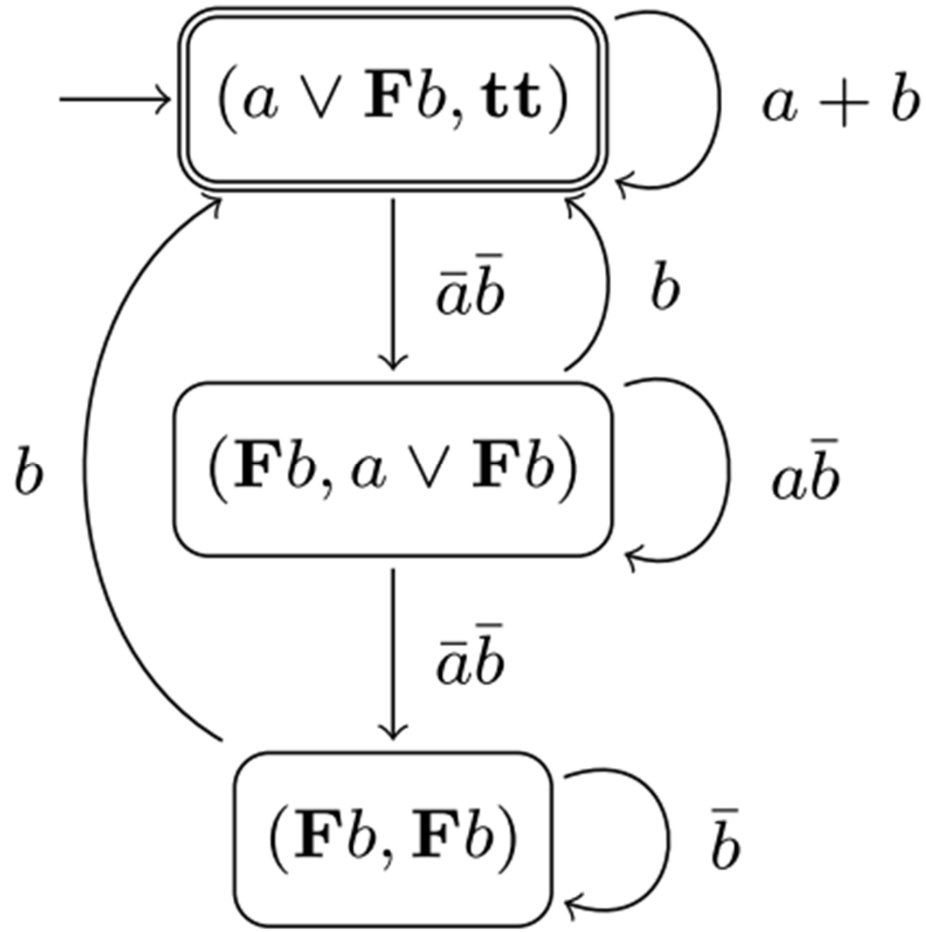
- We use the well-known breakpoint construction.

A DBA FOR $G(a \vee Fb)$



- Put new *goals* on hold while tracking current goal
- Accept if infinitely often the current goal is proven
- “Breakpoint Construction”

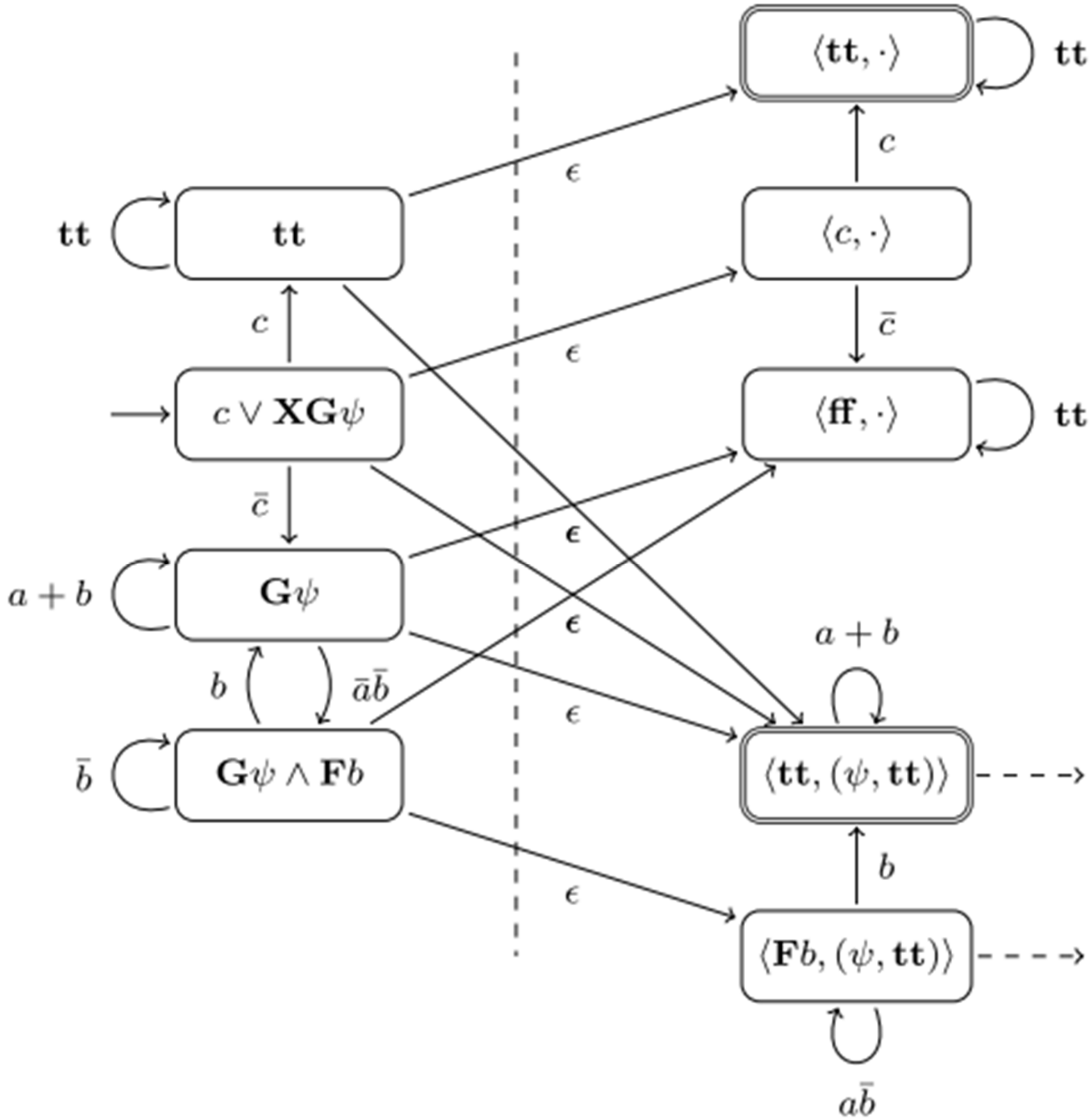
DBA FOR $G(a \vee Fb)$



COMPLETE LDBS FOR $\varphi = c \vee XG(a \vee Fb)$

1. Tracking DBA for φ (abbr. $\psi \coloneqq a \vee Fb$)

2. For every set $\mathcal{G}$ add a $\mathcal{G}$-jump to the product of the automata checking $\mathcal{G}$ and the $\mathcal{G}$ –remainder



LDBA SIZE FOR A FORMULA OF LENGTH N

Part	Size
Initial Component	$2^{2^{n}}$
G-Monitor	$2^{2^{n+1}}$
Accepting Component	$2^{2^{O(n)}}$
Total	$2^{2^{O(n)}}$

SIZES OF AUTOMATA

$\bigwedge_{i=1}^{j}(\mathbf{GF}a_i) \implies \bigwedge_{i=1}^{j}(\mathbf{GF}b_i)$

k: $\bigwedge_{i=1}^{k}(\mathbf{GF}a_i \vee \mathbf{FG}b_i)$

$f(0,j) = (\mathbf{GF}a_0)\mathbf{U}(\mathbf{X}^j b)$

$f(i+1,j) = (\mathbf{GF}a_{i+1})\mathbf{U}(\mathbf{G}f(i,j))$

	LDBA	Safra (spot+ltl2dstar)	Rabinizer
$j=1$	3	5	2
$j=2$	4	17	3
$j=3$	5	49	4
$j=4$	6	129	5
$k=2$	5	4385	13
$k=3$	9	*	198
$f(0,0)$	5	5	5
$f(0,2)$	10	10	7
$f(0,4)$	16	12	9
$f(1,0)$	6	196	17
$f(1,2)$	28	109839	33
$f(1,4)$	58	*	70
$f(2,0)$	10	99793	41
$f(2,2)$	46	*	94
$f(2,4)$	92	*	139

MODEL CHECKING RUNTIME
PNUELI-ZUCK MUTEX PROTOCOL

	#Clients	Our Implementation explicit, transition-based	ymbolic, ased	Rabinizer tate-based	C n-based
(6) $\mathbb{P}_{max=?}[(\mathbf{GF}p1{=}0 \vee \mathbf{FG}p2{\neq}0) \wedge (\mathbf{GF}p2{=}0 \vee \mathbf{FG}p3{\neq}0) \wedge (\mathbf{GF}p3{=}0 \vee \mathbf{FG}p1{\neq}0)]$	4	< 1	78	9	3
	5	10	1293	137	29
(7) $\mathbb{P}_{max=?}[(\mathbf{GF}p1{=}0 \vee \mathbf{FG}p1{\neq}0) \wedge (\mathbf{GF}p2{=}0 \vee \mathbf{FG}p2{\neq}0) \wedge (\mathbf{GF}p3{=}0 \vee \mathbf{FG}p3{\neq}0)]$	4	< 1	< 1	61	2
	5	1	< 1	1077	27
(8) $\mathbb{P}_{min=?}[(\mathbf{GF}p1{\neq}10 \vee \mathbf{GF}p1{=}0 \vee \mathbf{FG}p1{=}1) \wedge \mathbf{GF}p1{\neq}0 \wedge \mathbf{GF}p1{=}1]$	4	< 1	8	8	1
	5	1	145	190	16
(9) $\mathbb{P}_{max=?}[(\mathbf{G}p1{\neq}10 \vee \mathbf{G}p2{\neq}10 \vee \mathbf{G}p3{\neq}10) \wedge (\mathbf{FG}p1{\neq}1 \vee \mathbf{GF}p2{=}1 \vee \mathbf{GF}p3{=}1) \wedge (\mathbf{FG}p2{\neq}1) \vee \mathbf{GF}p1{=}1 \vee \mathbf{GF}p3{=}1)]$	4	5	-	1195	8
	5	99	-	-	125
(10) $\mathbb{P}_{min=?}[\mathbf{FG}p1{\neq}0 \vee \mathbf{FG}p2{\neq}0 \vee \mathbf{GF}p3{=}0 \vee (\mathbf{FG}p1{\neq}10 \wedge \mathbf{GF}p2{=}10 \wedge \mathbf{GF}p3{=}10)]$	4	1	728	33	79
	5	24	-	486	-
(11) $\mathbb{P}_{min=?}[f_{0,0}] = \mathbb{P}_{min=?}[(\mathbf{GF}p1{=}10)\mathbf{U}(p2{=}10)]$	4	< 1	17	40	2
	5	11	257	715	23
(12) $\mathbb{P}_{max=?}[f_{0,4}] = \mathbb{P}_{max=?}[(\mathbf{GF}p1{=}10)\mathbf{U}(\mathbf{XXXX}p2{=}10)]$	4	< 1	3	< 1	1
	5	5	20	2	20

CONCLUSION

- We have presented a translation from LTL to LDBA that
  - uses formulas as states
  - is modular
    - optimisations of any module helps to reduce state space!
  - yields in practice small ω-automata
  - is usable for quantitative prob. model checking without changing the algorithm!
- Website: https://www7.in.tum.de/~sickert/projects/ltl2ldba/