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.
- For a fixed scheduler:
  MDP $\rightarrow$ Markov chain
PROBABILISTIC MODEL CHECKING

• Qualitative Model Checking:
  • Input: MDP, LTL formula
  • Does the formula hold for all schedulers with probability 1?

[Diagram of a stochastic process with states A, B, and C, labeled with probabilities and transitions.]
PROBABILISTIC MODEL CHECKING

- 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

Initial Component

(possibly) non-deterministic

Accepting Component
deterministic

“Jumps”
AUTOMATA-BASED MODEL CHECKING

- Kripke struct.
- LTL

Product

Nondet. Büchi

Emptiness check

Yes/No

Vardi, Wolper middle 80s
QUALITATIVE
PROB. MODEL CHECKING

MDP

LTL

Product

Nondet. Büchi

Limit-det. Büchi

Prob=1?
Yes/No

Vardi and Wolper Courcoubetis, and Yannakakis [95]

Vardi and Wolper Courcoubetis, and Yannakakis [95]
QUALITATIVE
PROB. MODEL CHECKING

- Double exponential complexity in the formula, optimal.
- At the time: not applicable to the quantitative case.

Vardi and Wolper Courcoubetis, and Yannakakis [95]
Quantitative Prob. Model Checking

MDP

Product

P \geq 0.7?
Yes/No

LTL

Nondet. Büchi

Det. Rabin

Safra [89]
QUANTITATIVE
PROB. MODEL CHECKING

- Also double exponential complexity in the formula.
- Solves both the qualitative and quantitative case.
QUANTITATIVE PROB. MODEL CHECKING

- Also double exponential complexity in the formula.
- Solves both the qualitative and quantitative case.

In practice large automata
- Hard to implement efficiently
- Rise of “safraleess” approaches:
  - Acacia, ltl3dra, Rabinizer, …
Our Construction
Quantitative Prob. Model Checking

MDP

• Optimal: $2^{2O(n)}$
• Simpler construction
• Smaller automata
• Same MC algorithm as for det. automata

Product

P ≥ 0.7? Yes/No

LTL

Our Construction

Limit-det. Büchi
Our Construction

MDP
- Optimal: $2^{2^{O(n)}}$
- Simpler construction
- Smaller automata
- Same MC algorithm as for det. automata

Product

P ≥ 0.7?
Yes/No

LTL

Limit-det. Büchi
LIMIT-DETERMINISM

Initial Component

Accepting Component

non-deterministic

deterministic

"Jumps"
LIMIT-DETERMINISM

In our construction:

Every runs „uses“ nondeterminism at most once
PRELIMINARIES

- Linear Temporal Logic in Negation Normal Form

\[ \varphi ::= \mathsf{tt} \mid \mathsf{ff} \mid a \mid \lnot a \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \mathsf{F}\varphi \mid \varphi \mathsf{U}\varphi \mid \mathsf{X}\varphi \mid \mathsf{G}\varphi \]
PRELIMINARIES

• Linear Temporal Logic in Negation Normal Form

\[ \varphi ::= \text{tt} \mid \text{ff} \mid a \mid \neg a \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \text{F} \varphi \mid \varphi \text{U} \varphi \mid \text{X} \varphi \mid \text{G} \varphi \]

• Monotonicity of NNF:

\textbf{if} \ w \text{ satisfies } \varphi

\text{w’ satisfies all the subformulas of } \varphi \text{ satisfied by } w, \text{ and perhaps more}

\textbf{then } w’ \text{ satisfies } \varphi
PRELIMINARIES

• Linear Temporal Logic in Negation Normal Form

\[ \varphi ::= \texttt{tt} \mid \texttt{ff} \mid a \mid \neg a \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid F\varphi \mid \varphi U \varphi \mid X\varphi \mid G\varphi \]

• Monotonicity of NNF:

  if \( w \) satisfies \( \varphi \)

  \( w' \) satisfies all the subformulas of \( \varphi \) satisfied by \( w \), and perhaps more

  then \( w' \) satisfies \( \varphi \)
FIRST STEP: A DETERMINISTIC "TRACKING" AUTOMATON
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON
FIRST STEP: A DETERMINISTIC "TRACKING" AUTOMATON

The formula $af(\varphi, \nu)$ ("$\varphi$ after $\nu$") is defined by:

\[
\begin{align*}
af(tt, \nu) &= tt \\
af(ff, \nu) &= ff \\
af(a, \nu) &= \begin{cases} 
   tt & \text{if } a \in \nu \\
   ff & \text{if } a \notin \nu 
\end{cases} \\
af(\neg a, \nu) &= \begin{cases} 
   ff & \text{if } a \in \nu \\
   tt & \text{if } a \notin \nu 
\end{cases}
\end{align*}
\]

\[
af(\varphi \land \psi, \nu) = af(\varphi, \nu) \land af(\psi, \nu) \\
af(\varphi \lor \psi, \nu) = af(\varphi, \nu) \lor af(\psi, \nu) \\
af(X\varphi, \nu) = \varphi \\
af(G\varphi, \nu) = af(\varphi, \nu) \land G\varphi \\
af(F\varphi, \nu) = af(\varphi, \nu) \lor F\varphi \\
af(\varphi U \psi, \nu) = af(\psi, \nu) \lor (af(\varphi, \nu) \land \varphi U \psi)
\]
FIRST STEP: A DETERMINISTIC „TRACKING“ AUTOMATON

- The automaton „tracks“ the property that must hold now for the original property to hold at the beginning.
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 $F, X, U$: ✔

[Diagram of automaton with states and transitions]
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.
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 $\varphi$ and a word $w$. Let $G\psi$ be a $G$-subformula of $\varphi$. 
\( \mathcal{G} \)-SUBFORMULAS

- Fix a formula \( \varphi \) and a word \( w \). Let \( \mathcal{G} \psi \) be a \( \mathcal{G} \)-subformula of \( \varphi \).

![Diagram showing the word w and its subformula relationship with \( \varphi \)]
**G-SUBFORMULAS**

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

![Diagram showing a word $w$ and a subformula $G\psi$]
**G-SUBFORMULAS**

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

![Diagram with symbols and letters]
$G$-SUBFORMULAS

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

\[ \begin{array}{cccccccccc}
\text{w} & b & a & b & c & c & a & c & b & \ldots \\
\end{array} \]
**G**-SUBFORMULAS

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

- Informally: while reading the word $w$, the set of $G$-subformulas that hold cannot decrease, and eventually stabilizes to a set $\text{True}_G\varsigma(w, \varphi)$. 
SECOND STEP: JUMPING

• We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.
SECOND STEP: JUMPING

• We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.

• From each state $\psi$ we add a jump for every set $\mathcal{G}$ of $\mathcal{G}$-subformulas of $\psi$. 
SECOND STEP: JUMPING

- We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.
- From each state $\psi$ we add a jump for every set $G$ of $G$-subformulas of $\psi$.
- „Meaning“ of a $G$-jump at state $\psi$: The automaton „guesses“ that the rest of the word satisfies
SECOND STEP: JUMPING

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

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

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

  even if no other $G$-subformula of $\psi$ ever becomes true.
SECOND STEP: JUMPING

• We modify the tracking automaton so that at any moment it can nondeterministically jump to the accepting component.

• From each state $\psi$ we add a jump for every set $G$ of $G$-subformulas of $\psi$.

• „Meaning“ of a $G$-jump at state $\psi$: The automaton „guesses“ that the rest of the word satisfies

  1. $G$ (every formula of $G$), and

  2. $G \Rightarrow \psi$

    even if no other $G$-subformula of $\psi$ 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 $G$-jump at state $\psi$: The automaton „guesses“ that the rest of the run satisfies

1. $G$ (every formula of $G$), and

2. $G \Rightarrow \psi$

even if no other $G$-subformula of $\psi$ ever becomes true.
SECOND STEP: JUMPING

- „Meaning“ of the $G$-jump at state $\psi$ : The automaton „guesses“ that the rest of the run satisfies
  1. $G$ (every formula of $G$), and
  2. $G \Rightarrow \psi$
  even if no other $G$-subformula of $\psi$ ever becomes true.

- $w \models \varphi$ iff the automaton can guess correctly.
SECOND STEP: JUMPING

• „Meaning“ of the $\mathcal{G}$-jump at state $\psi$ : 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 $\mathcal{G}$-subformula of $\psi$ ever becomes true.

• $w \models \varphi$ iff the automaton can guess correctly.
  • If the correct guess is made at suffix $w'$ then $w' \models \psi$
    which implies $w \models \varphi$ (tracking!)
SECOND STEP: JUMPING

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

  even if no other $G$-subformula of $\psi$ ever becomes true.

• $w \models \varphi$ iff the automaton can guess correctly.

• If the correct guess is made at suffix $w'$ then $w' \models \psi$
  which implies $w \models \varphi$ (tracking!)

• If $w \models \varphi$ then $w' \models \text{TrueGs}(w, \varphi)$ for some suffix $w'$
  and so the jump before $w'$ that chooses $g := \text{TrueGs}(w, \varphi)$ satisfies 1. and 2.
A DBA THAT CHECKS 1. & 2.

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

- "\( G \Rightarrow \psi \) holds even if no other \( G \)-subformula of \( \psi \) ever becomes true"
CHECKING 2.

- "\( \mathcal{G} \Rightarrow \psi \) holds even if no other \( \mathcal{G} \)-subformula of \( \psi \) ever becomes true"

- Example: \[ \psi = \mathcal{G}(a \lor Fb) \land (\mathcal{G}c \lor Xd) \]
  \[ \mathcal{G} = \{ \mathcal{G}(a \lor Fb) \} \]

Reduces to checking \( Xd \)
CHECKING 2.

• "\( \mathcal{G} \Rightarrow \psi \) holds even if no other \( \mathcal{G} \)-subformula of \( \psi \) ever becomes true”

• Example: \( \psi = \mathcal{G}(a \lor Fb) \land (\mathcal{G}c \lor Xd) \)

\[
\mathcal{G} = \{ \mathcal{G}(a \lor Fb) \}
\]

Reduces to checking \( Xd \)

• Reduces to checking the \( \mathcal{G} \)-free formula

\[
\psi[\mathcal{G} \setminus tt, \overline{\mathcal{G}} \setminus ff]
\]
CHECKING 2.

• "\( \mathcal{G} \Rightarrow \psi \) holds even if no other \( \mathcal{G} \)-subformula of \( \psi \) ever becomes true”

• Example: \( \psi = \mathcal{G}(a \lor Fb) \land (\mathcal{G}c \lor Xd) \)

\[ \mathcal{G} = \{ \mathcal{G}(a \lor Fb) \} \]

Reduces to checking \( Xd \)

• Reduces to checking the \( \mathcal{G} \)-free formula

\[ \psi[\; \mathcal{G}\backslash tt, \; \overline{\mathcal{G}}\backslash ff \; ] \]

• Since the formula is \( \mathcal{G} \)-free, use the tracking automaton.
CHECKING 1.

• "\( G \) holds even if no other \( G \)-subformula of \( \psi \) ever becomes true"
CHECKING 1.

• 
  "\( G \) holds even if no other \( G \)-subformula of \( \psi \) ever becomes true"

• Example:
  \[
  \psi = Fc \land GF(a \land (Gb \lor FGC))
  \]

  \[
  G = \{ Gb, \ GF(a \land (Gb \lor FGC)) \}
  \]

  reduces to checking \( Gb \land GFa \equiv G(b \land Fa) \)
CHECKING 1.

• „$G$ holds even if no other $G$-subformula of $\psi$ ever becomes true”

• Example: $\psi = Fc \land GF(a \land (Gb \lor FGc))$

\[ G = \{ Gb, GF(a \land (Gb \lor FGc)) \} \]

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

• Reduces to checking a formula $G\rho$ where $\rho$ is $G$-free.
CHECKING 1.

- “\(G\) holds even if no other \(G\)-subformula of \(\psi\) ever becomes true”

- Example: \[\psi = Fc \land GF(a \land (Gb \lor FGe))\]

\[G = \{Gb, GF(a \land (Gb \lor FGe))\}\]

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

- Reduces to checking a formula \(G\rho\) where \(\rho\) is \(G\)-free.

- So we need DBAs for formulas \(G\rho\) where \(\rho\) is \(G\)-free.
Tracking automaton for $\varphi$

Accepting component for $G_1$

Accepting component for $G_n$
Tracking automaton for $\varphi$

Accepting component for $G_n$

Tracking automaton for $\psi [G_1 \setminus tt, \overline{G_1} \setminus ff ]$

Automaton for $G \rho$, where $\rho$ is $G$-free
$G(a \lor Fb) \\
\land \\
(Gc \lor Xd)$
Guess
\[ G = \{ G(a \lor Fb) \} \]
Guess \( G = \{G(a \lor Fb)\} \)

\[ G(a \lor Fb) \land (Gc \lor Xd) \]
Guess \( \mathcal{G} = \{G(a \lor Fb)\} \)

\( G(a \lor Fb) \land (Gc \lor Xd) \)

\( \varepsilon \)

Tracking automaton for \( Xd \)

Automaton for \( G(a \lor Fb) \)
Guess
\[ G = \{ G(a \lor Fb) \} \]

\[ G(a \lor Fb) \land (Gc \lorXd) \]

\[ \epsilon \]

Tracking automaton for \( Xd \)

Automaton for \( G(a \lor Fb) \)
A DBA FOR $G(a \lor Fb)$
A DBA FOR $G(a \lor Fb)$

$G(a \lor Fb)$

| c | a | b | c |

$a \lor Fb$

$a \lor Fb$

$a \lor Fb$

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

$G(a \lor Fb)$

\[\begin{array}{ccccc}
| & c & a & b & c \\
\hline
a \lor Fb & Fb & Fb & tt \\
\hline
a \lor Fb & tt & Fb \\
\hline
a \lor Fb & tt & Fb \\
\hline
a \lor Fb & Fb \\
\end{array}\]
A DBA FOR $G(a \lor Fb)$
a ∨ Fb
\( a \lor Fb \quad Fb \quad Fb \quad a \lor Fb \)
\( a \lor Fb \)
DBA FOR $G(a \lor Fb)$
COMPLETE LDBA FOR $\varphi = c \lor XG(a \lor Fb)$

1. Tracking DBA for $\varphi$
   (abbr. $\psi := a \lor Fb$)
COMPLETE LDBA FOR $\varphi = c \lor XG(a \lor Fb)$

1. Tracking DBA for $\varphi$
   (abbr. $\psi := a \lor Fb$)

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
COMPLETE LDBA FOR $\varphi = c \lor XG(a \lor Fb)$

1. Tracking DBA for $\varphi$
   (abbr. $\psi := a \lor Fb$)

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
COMPLETE LDBA FOR $\varphi = c \lor XG(a \lor Fb)$

1. Tracking DBA for $\varphi$  
   (abbr. $\psi := a \lor Fb$)

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
UPPER BOUND ON LDBA SIZE

• **Theorem:** Every formula obtained by „tracking \( \phi \)“ is a positive boolean combination of subformulas of \( \phi \).

\[
c \lor XG(a \lor Fb) \rightarrow G(a \lor Fb) \rightarrow G(a \lor Fb) \land Fb
\]
Theorem: Every formula obtained by „tracking $\varphi$“ is a positive boolean combination of subformulas of $\varphi$.

$$c \lor XG(a \lor Fb) \rightarrow G(a \lor Fb) \rightarrow G(a \lor Fb) \land Fb$$

Corollary: for a formula of length $n$ there are at most $2^{2^n}$ „tracking formulas“ up to equivalence, even if we leave temporal operators uninterpreted.

$$Fa \land (Fa \lor Gb) \equiv_p Fa \quad Fa \lor Ga \nleq_p Fa$$
UPPER BOUND ON LDBA SIZE

• **Theorem:** Every formula obtained by „tracking φ“ is a positive boolean combination of subformulas of φ.

\[ c \lor XG(a \lor Fb) \rightarrow G(a \lor Fb) \rightarrow G(a \lor Fb) \land Fb \]

• **Corollary:** for a formula of length \( n \) there are at most \( 2^{2^n} \) „tracking formulas“ up to equivalence, even if we leave temporal operators uninterpreted.

\[ Fa \land (Fa \lor Ga) =_p Fa \quad Fa \lor Ga \neq_p Fa \]

• This allows us to derive an upper bound on the size of the LDBA
### UPPER BOUND ON LDBA SIZE

<table>
<thead>
<tr>
<th>Part</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Component</td>
<td>$2^{2n}$</td>
</tr>
<tr>
<td>G-Monitor</td>
<td>$2^{2n+1}$</td>
</tr>
<tr>
<td>Accepting Component</td>
<td>$2^{2^{O(n)}}$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$2^{2^{O(n)}}$</td>
</tr>
</tbody>
</table>
### LDBA SIZE IN PRACTICE

<table>
<thead>
<tr>
<th></th>
<th>LDBA</th>
<th>Safra</th>
<th>Rabinizer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(spot+ltl2dstar)</td>
<td></td>
</tr>
<tr>
<td>$j = 1$</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$j = 2$</td>
<td>4</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>$j = 3$</td>
<td>5</td>
<td>49</td>
<td>4</td>
</tr>
<tr>
<td>$j = 4$</td>
<td>6</td>
<td>129</td>
<td>5</td>
</tr>
<tr>
<td>$k = 2$</td>
<td></td>
<td>4385</td>
<td>13</td>
</tr>
<tr>
<td>$k = 3$</td>
<td></td>
<td>*</td>
<td>198</td>
</tr>
<tr>
<td>$f(0,0)$</td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$f(0,2)$</td>
<td>10</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>$f(0,4)$</td>
<td>16</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>$f(1,0)$</td>
<td>6</td>
<td>196</td>
<td>17</td>
</tr>
<tr>
<td>$f(1,2)$</td>
<td>28</td>
<td>109839</td>
<td>33</td>
</tr>
<tr>
<td>$f(1,4)$</td>
<td>58</td>
<td>*</td>
<td>70</td>
</tr>
<tr>
<td>$f(2,0)$</td>
<td>10</td>
<td>99793</td>
<td>41</td>
</tr>
<tr>
<td>$f(2,2)$</td>
<td>46</td>
<td>*</td>
<td>94</td>
</tr>
<tr>
<td>$f(2,4)$</td>
<td>92</td>
<td>*</td>
<td>139</td>
</tr>
</tbody>
</table>
LDBA SIZE IN PRACTICE

\[ \bigwedge_{i=1}^{j}(\text{GF}a_i) \implies \bigwedge_{i=1}^{j}(\text{GF}b_i) \]

\[ k: \bigwedge_{i=1}^{k}(\text{GF}a_i \lor \text{FG}b_i) \]

\[ f(0, j) = (\text{GF}a_0)U(X^j b) \]

\[ f(i+1, j) = (\text{GF}a_{i+1})U(G f(i, j)) \]

<table>
<thead>
<tr>
<th>(j)</th>
<th>LDBA</th>
<th>Safra (spot+ltl2dstar)</th>
<th>Rabinizer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>49</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>129</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4385</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>*</td>
<td>198</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>196</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>109839</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>*</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>99793</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>46</td>
<td>*</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>92</td>
<td>*</td>
<td>139</td>
</tr>
</tbody>
</table>
### MODEL CHECKING RUNTIME

**PNUELI-ZUCK MUTEX PROTOCOL**

<table>
<thead>
<tr>
<th>#Clients</th>
<th>Our Imp. explicit</th>
<th>PRISM symb</th>
<th>PRISM+ Rabinizer symbolic</th>
<th>IscasMC explicit</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>78</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>1293</td>
<td>137</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>61</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1077</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>8</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>145</td>
<td>190</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>-</td>
<td>1195</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>99</td>
<td>-</td>
<td>-</td>
<td>125</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>728</td>
<td>33</td>
<td>79</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>-</td>
<td>486</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>17</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>257</td>
<td>715</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>20</td>
<td>2</td>
<td>20</td>
</tr>
</tbody>
</table>

### Example Expressions

- \( P_{max} = (G F p_1 = 0 \lor F G p_2 \neq 0) \land (G F p_2 = 0 \lor F G p_3 \neq 0) \land (G F p_3 = 0 \lor F G p_1 \neq 0) \)
- \( P_{min} = [G F p_1 \neq 10 \lor G F p_1 = 0 \land F G p_1 = 1] \land G F p_1 \neq 0 \land G F p_1 = 1 \)
- \( P_{max} = (G p_1 \neq 10 \land G p_2 \neq 10 \land G p_3 \neq 10) \land (G F p_1 \neq 1 \lor G F p_2 = 1 \lor G F p_3 = 1) \land (F G p_2 \neq 1 \lor V F G p_1 = 1 \lor V G F p_3 = 1) \)
- \( P_{min} = [V G F p_3 = 0 \lor (G F p_1 \neq 10)] \land G F p_2 = 10 \land G F p_3 = 10 \)
- \( P_{min} = P_{min} = \left[ (G F p_1 = 10) \cup (p_2 = 10) \right] \)
- \( P_{max} = \left[ G F p_1 = 10 \right] \cup (X X X X p_2 = 10) \)
## MODEL CHECKING RUNTIME

### PNUELI-ZUCK MUTEX PROTOCOL

<table>
<thead>
<tr>
<th>#Clients</th>
<th>Our Imp. explicit</th>
<th>PRISM symb</th>
<th>PRISM+ Rabinizer symbolic</th>
<th>IscasMC explicit</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>78</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td><strong>10</strong></td>
<td>1293</td>
<td>137</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>61</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>&lt; 1</td>
<td>1077</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>8</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>145</td>
<td>190</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>-</td>
<td>1195</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td><strong>99</strong></td>
<td>-</td>
<td>-</td>
<td>125</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>728</td>
<td>33</td>
<td>79</td>
</tr>
<tr>
<td>5</td>
<td><strong>24</strong></td>
<td>-</td>
<td>486</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>17</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td><strong>11</strong></td>
<td>257</td>
<td>715</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1</td>
<td>3</td>
<td>&lt; 1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>20</td>
<td><strong>2</strong></td>
<td>20</td>
</tr>
</tbody>
</table>
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS?

Uncontrolled system → Product → Parity game

LTL → Nondet. Büchi → Det. Parity
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS?

Uncontrolled system

Product

Parity game

LTL

Nondet. Büchi

Det. Parity

single exp.

Safra, single exp.
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS?

**Diagram:**
- Uncontrolled system
- LTL
- Limit-det. Büchi
- Product
- Det. Parity
- Parity game
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS?

Uncontrolled system → Product → Parity game

LTL → Limit-det. Büchi → Det. Parity

double exp.
single exp.
CAN LDBA BE ALSO USED FOR CONTROLLER SYNTHESIS?

- Uncontrolled system
  - Product
    - Parity game
- LTL
  - Limit-det. Büchi
    - Det. Parity

(double exp.)
THE NEW PICTURE

Probabilistic model checking for MCs

Nonamb. Büchi

Nondet. Büchi

Limit-det. Büchi

Det. Parity

LTL

Model checking

Probabilistic model checking for MDPs

Synthesis
THE NEW PICTURE

- **LTL**
  - Nondet. Büchi
  - Limit-det. Büchi
  - Det. Parity

- **Probabilistic model checking for MCs**
  - Nonamb. Büchi
  - Good for Games

- **Model checking**
  - Probabilistic model checking for MDPs

- **Synthesis**
CONCLUSION

• We have presented a translation from LTL to LDBA that
  • uses formulas as states
  • is modular
    • optimizations of any module helps to reduce state space!
  • yields in practice small $\omega$-automata
  • is usable for quantitative prob. model checking without changing the algorithm.
  • can be also used as intermediate step to synthesis.

• Website: https://www7.in.tum.de/~sickert/projects/ltl2ldba/