Decision Procedures for CTL

Oliver Friedmann$^1$  Markus Latte$^1$

$^1$ Dept. of Computer Science, Ludwig-Maximilians-University, Munich, Germany

CLoDeM

Edinburgh, 15 July 2010
Introduction to CTL*

Origin: Emerson and Halpern ’86

- supersedes the branching-time logic CTL and the linear-time logic LTL

\[
\mu\text{-calculus} \\
\uparrow \\
CTL^* \\
\downarrow \\
CTL \quad LTL
\]

- applied to specify and verify reactive and agent-based systems

- also applied to program synthesis

- however: decision procedures difficult to obtain

- worst case runtime: doubly exponential
  - lower bound: Vardi and Stockmeyer ’85
  - upper bound: Emerson and Sistla ’84; Emerson and Jutla ’00
Table of contents

1 Syntax and Semantic of CTL*

2 Emerson-Jutla Method

3 Reynolds’ Tableaux

4 Infinite Tableaux

5 Experimental Results
## Syntax of CTL

### Negation normal form

\[
\psi ::= q \mid \neg q \mid \psi \land \psi \mid \psi \lor \psi \mid X\psi \mid \psi U\psi \mid \psi R\psi \mid E\psi \mid A\psi
\]

where \( q \in \mathcal{P} \) are propositional constants
Syntax of CTL

Negation normal form

\[ \psi ::= q \mid \neg q \mid \psi \land \psi \mid \psi \lor \psi \mid X\psi \mid \psi U\psi \mid \psi R\psi \mid E\psi \mid A\psi \]

where \( q \in P \) are propositional constants

This talk: replace fixpoints \( \psi U\psi \), \( \psi R\psi \) by \( F\psi \), \( G\psi \).
Syntax of CTL*

Negation normal form

\[ \psi ::= q | \neg q | \psi \land \psi | \psi \lor \psi | \text{X} \psi | \text{F} \psi | \text{G} \psi | \text{E} \psi | \text{A} \psi \]

where \( q \in \mathcal{P} \) are propositional constants

**This talk**: replace fixpoints \( \psi \text{U} \psi, \psi \text{R} \psi \) by \( \text{F} \psi, \text{G} \psi \).
Interpretation

Transition systems

\[ T = (S, \rightarrow, \lambda) \]

- \((S, \rightarrow)\) directed, total graph
- \(\lambda : S \rightarrow 2^P\) labeling function
Interpretation

Transition systems

\[ T = (S, \rightarrow, \lambda) \]

- \( (S, \rightarrow) \) directed, total graph
- \( \lambda : S \rightarrow 2^P \) labeling function

Path \( \pi \): sequence \( (s_i)_{i \in \mathbb{N}} = s_0, s_1, \ldots \) of states respecting edges
Transition systems

TS $\mathcal{T} = (S, \rightarrow, \lambda)$ with
- $(S, \rightarrow)$ directed, total graph
- $\lambda : S \rightarrow 2^P$ labeling function

Path $\pi$: sequence $(s_i)_{i \in \mathbb{N}} = s_0, s_1, \ldots$ of states respecting edges

Notations: $\pi^i = s_i, s_{i+1}, \ldots$
### Semantics of Formulas

- $\mathcal{T}, \pi \models q$ iff $q \in \lambda(\pi(0))$
- $\mathcal{T}, \pi \models \neg q$ iff $q \not\in \lambda(\pi(0))$
- $\mathcal{T}, \pi \models \psi_1 \land \psi_2$ iff $\mathcal{T}, \pi \models \psi_1$ and $\mathcal{T}, \pi \models \psi_2$
- $\mathcal{T}, \pi \models \psi_1 \lor \psi_2$ iff $\mathcal{T}, \pi \models \psi_1$ or $\mathcal{T}, \pi \models \psi_2$
- $\mathcal{T}, \pi \models X\psi$ iff $\mathcal{T}, \pi^1 \models \psi$
- $\mathcal{T}, \pi \models F\psi$ iff $\mathcal{T}, \pi^i \models \psi$ for some $i \in \mathbb{N}$
- $\mathcal{T}, \pi \models G\psi$ iff $\mathcal{T}, \pi^i \models \psi$ for all $i \in \mathbb{N}$
- $\mathcal{T}, \pi \models E\psi$ iff $\mathcal{T}, \tilde{\pi} \models \psi$ for some $\tilde{\pi}$ with $\pi(0) = \tilde{\pi}(0)$
- $\mathcal{T}, \pi \models A\psi$ iff $\mathcal{T}, \tilde{\pi} \models \psi$ for all $\tilde{\pi}$ with $\pi(0) = \tilde{\pi}(0)$
A formula is a state formula iff $X$, $F$ and $G$ only occur under an $E$ or an $A$. Otherwise the formula is a path formula.

Property

For any state formula $\varphi$, any paths $\pi$ and $\pi'$ in some TS $\mathcal{T}$ we have:

$$\mathcal{T}, \pi \models \varphi \text{ iff } \mathcal{T}, \pi' \models \varphi$$

provided that $\pi(0) = \pi'(0)$.

Notation:

$\mathcal{T}, s \models \varphi$ abbreviates $\mathcal{T}, \pi \models \varphi$ for a path $\pi$ starting with $s$. 
Satisfiability Problem

Given a CTL* state formula $\vartheta$, decide whether there is a TS $\mathcal{T} = (S, \rightarrow, \lambda)$ and a state $s^* \in S$ s.t.

$$\mathcal{T}, s^* \models \varphi$$
**Satisfiability**

Given a CTL\(^*\) state formula \(\vartheta\), decide whether there is a TS \(\mathcal{T} = (S, \to, \lambda)\) and a state \(s^* \in S\) s.t.

\[ \mathcal{T}, s^* \models \vartheta \]

as opposed to the model checking problem

**Model Checking**

Given a CTL\(^*\) state formula \(\varphi\) and TS \(\mathcal{T} = (S, \to, \lambda)\) and a state \(s^* \in S\), decide whether

\[ \mathcal{T}, s^* \models \varphi \]
**Satisfiability Problem**

**Satisfiability**
Given a $\text{CTL}^*$ state formula $\vartheta$, decide whether there is a TS $T = (S, \rightarrow, \lambda)$ and a state $s^* \in S$ s.t.

$$T, s^* \models \vartheta$$

as opposed to the model checking problem

**Model Checking**
Given a $\text{CTL}^*$ state formula $\varphi$ and TS $T = (S, \rightarrow, \lambda)$ and a state $s^* \in S$, decide whether

$$T, s^* \models \varphi$$

*note*: there is no strong relationship between satisfiability and model checking decision procedures (in general)!
Running Example

Consider the formula

$$AFGp \land EGEF\neg p$$
Consider the formula

\[ \text{AFG}p \land \text{EGEF} \neg p \]

The following TS is a model of it.
Overview

Emerson-Jutla Method ('84)
- emptiness test of a tree automaton accepting all models
- **drawbacks**: no implementation, unintuitive proof structure, constant branching degree

Reynolds’ Tableaux ('09)
- exhaustive tableau-search restricted by small model property
- **drawbacks**: fairly slow in practice, no intrinsic detection of unfulfilled eventualities

Our System
- existence of infinite tableaux with global conditions
- **drawbacks**: requires automata deterministation for checking global conditions
Table of contents

1 Syntax and Semantic of CTL*
2 Emerson-Jutla Method
3 Reynolds’ Tableaux
4 Infinite Tableaux
5 Experimental Results
Given a $\text{CTL}^*$-formula $\vartheta$,

- normalise $\vartheta$ to a normal form $\psi$,

\[
\psi ::= E\lambda \mid A\lambda \mid AGE\lambda \mid \psi \land \psi \mid \psi \lor \psi \mid p \mid \neg p
\]

where $\lambda$ is a LTL-formula,

- construct a tree automaton which recognises tree-models of $\psi$, and

- test automaton for emptiness.
Given a $\text{CTL}^*$-formula $\vartheta$,

1. transform $\vartheta$ into negation form.
2. replace a subformula $Q\lambda$, $Q \in \{E, A\}$, by a fresh variable, say $p$.
3. attach $\land "AG(p \leftrightarrow Q\lambda)"$ to $\vartheta$.

"$AG(q \leftrightarrow E\lambda)" \equiv AGE(q \rightarrow \lambda) \land AG(\neg q \rightarrow \neg \lambda)"

"$AG(q \leftrightarrow A\lambda)" \equiv AG(q \rightarrow \lambda) \land AGE(\neg q \rightarrow \neg \lambda)"

4. iterate 2.–3. as long as possible.
Let $B_\lambda$ be a non-det. Büchi automaton for $\lambda$, (exp. size)
and $D_\lambda$ be a det. parity or Rabin automaton for $\lambda$. (2-exp. size)

A tree automata for $\varphi$

<table>
<thead>
<tr>
<th>$E \lambda$</th>
<th>Simulate $B_\lambda$ on a guessed path.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \lambda$</td>
<td>Simulate $D_\lambda$ on all paths.</td>
</tr>
</tbody>
</table>

**Note**: implicit quantifier in $B_\lambda$ does not commute with the path quantifier.

| $A G E \lambda$ | start a simulation of $B_\lambda$ everywhere. |
| $\varphi$       | follow the Boolean connectives.               |

**Note**: The connectives apply to the root only.
1. Normalize $\text{AFG}p \land \text{EGEF}\neg p$:
   
   $\text{AFG}p \land \text{EG}q \land \text{AGE}(q \implies F\neg p) \land \text{AG}(F\neg p \implies q)$

2. Build non-det. Büchi automata $B_{Gq}$ and $B_q \implies F\neg p$

3. Build det. Rabin automata $D_{FGp}$ and $D_{G(F\neg p \implies q)}$

4. Turn all four automata into deterministic tree automata

5. Use a crossproduct construction to get a tree automaton for the initial formula

6. Apply an emptiness test
Corollary

The decision procedure by Emerson, Sistla and Jutla is in $2\text{EXPTIME}$.

However, Emerson noted that . . .

“...[o]ne drawback to the use of automata is that, due to the delicate combinatorial constructions involved, there is usually no clear relationship between the structure of the automaton and the candidate formula.”

Corollary

The decision procedure by Emerson, Sistla and Jutla is in 2EXPTIME.

However, Emerson noted that . . .

“. . . [o]ne drawback to the use of automata is that, due to the delicate combinatorial constructions involved, there is usually no clear relationship between the structure of the automaton and the candidate formula.”


another drawback: fixed branching degree of final tree automaton
Table of contents

1 Syntax and Semantic of CTL*
2 Emerson-Jutla Method
3 Reynolds’ Tableaux
4 Infinite Tableaux
5 Experimental Results
Reynolds’ Tableaux

Structure

- finite tableaux with back-loops
- nodes labelled with colours: a set of hues
- hues – Hintikka-style sets – correspond to fullpaths in the intended model
- edges in a tableau correspond to proceeding in time by one step
- successors of a node depend on the contained intended fullpaths
Reynolds’ Tableaux (cont.)

<table>
<thead>
<tr>
<th>Correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>local</strong> conditions: node correctness and successor correctness</td>
</tr>
<tr>
<td>• <strong>global</strong> conditions: eventualities in hue threads have to be fulfilled</td>
</tr>
</tbody>
</table>

**Theorem**: Reynolds’ tableau system is sound and complete.
Reynolds’ Tableau for $AFGp \land EGEF\neg p$

**Relevant Hues**

$h_0 : \{AFGp \land EGEF\neg p, AFGp, FGp, EF\neg p, Gp, p, EGEF\neg p, GEF\neg p\}$

$h_1 : \{AFGp \land EGEF\neg p, AFGp, FGp, EF\neg p, F\neg p, p, EGEF\neg p, FAGp\}$

$h_2 : \{EGF\neg p \lor AFAGp, AFGp, FGp, EF\neg p, F\neg p, \neg p, AFAGp, FAGp\}$

$h_3 : \{EGF\neg p \lor AFAGp, AFGp, FGp, Gp, p, AFAGp, FAGp, AGp\}$
Reynolds’ Tableau for $\text{AFG}p \land \text{EGEF}\neg p$

Relevant Hues

$h_0 : \{\text{AFG}p \land \text{EGEF}\neg p, \text{AFG}p, \text{FG}p, \text{EF}\neg p, Gp, p, \text{EGEF}\neg p, \text{GEF}\neg p\}$

$h_1 : \{\text{AFG}p \land \text{EGEF}\neg p, \text{AFG}p, \text{FG}p, \text{EF}\neg p, F\neg p, p, \text{EGEF}\neg p, \text{FAG}p\}$

$h_2 : \{\text{EGF}\neg p \lor \text{AFAG}p, \text{AFG}p, \text{FG}p, \text{EF}\neg p, F\neg p, \neg p, \text{AFAG}p, \text{FAG}p\}$

$h_3 : \{\text{EGF}\neg p \lor \text{AFAG}p, \text{AFG}p, \text{FG}p, Gp, p, \text{AFAG}p, \text{FAG}p, \text{AG}p\}$
# Tableau Search

<table>
<thead>
<tr>
<th>Algorithmic Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableau-building</td>
</tr>
<tr>
<td>loop checking</td>
</tr>
<tr>
<td>backtracking</td>
</tr>
</tbody>
</table>
Tableau Search

Algorithmic Method

- tableau-building
- loop checking
- backtracking

Good Loops

- witness the fact that every eventually in the hue thread is satisfied after a finite number of steps
- checked by a model-checking style algorithm
Bad Loops

- occurring repetition but looping back results in **unfulfilled** eventualities
- **solution**: extend the branch instead of looping back
- **problem**: when do we stop to extend unfulfilled branches?
Bad Loops

- occurring repetition but looping back results in unfulfilled eventualities
- **solution**: extend the branch instead of looping back
- **problem**: when do we stop to extend unfulfilled branches?

When to stop?

- currently: use **small model property** to restrict the length of the branches
- but: small model property yields **doubly exponential bound**
based on Reynolds’ prototype implementation

- comparably slow as unprofitable branches are solely detected by hitting the length restriction
- running example: longer than one day; our system requires less than a second
Table of contents

1. Syntax and Semantic of CTL*
2. Emerson-Jutla Method
3. Reynolds’ Tableaux
4. Infinite Tableaux
5. Experimental Results
A Tableau for CTL

A tableau for $\vartheta$ is a tree which imitates a potential model of $\vartheta$. 
A Tableau for CTL*

A tableau for $\vartheta$ is a tree which imitates a potential model of $\vartheta$.

A pre-tableau for a formula $\vartheta$ is an infinite tree s.th.

- it is finitely branching,
- each node is labelled with a goal (as a set),

\[
\begin{align*}
A\Sigma_1, \ldots, A\Sigma_n, & \quad E\Pi_1, \ldots, E\Pi_m, \\
\Lambda & \\
\end{align*}
\]

Example: \{A\{\neg p \lor q\}, E\{Xp, Fq\}, \neg p, \neg q\}.

Sloppy writing: $A(\neg p \lor q)$ or $E(Xp, \Pi)$, e.g.
A tableau for $\vartheta$ is a tree which imitates a potential model of $\vartheta$.

A pre-tableau for a formula $\vartheta$ is an infinite tree s.th.

- it is finitely branching,
- each node is labelled with a goal (as a set),

\[
\begin{align*}
A \Sigma_1, \ldots, A \Sigma_n, & \quad E \Pi_1, \ldots, E \Pi_m, \quad \Lambda
\end{align*}
\]

Example: \{A\{\neg p \lor q\}, E\{Xp, Fq\}, \neg p, \neg q\}.

Sloppy writing: $A(\neg p \lor q)$ or $E(Xp, \Pi)$, e.g.
A Tableau for CTL*

A tableau for $\vartheta$ is a tree which imitates a potential model of $\vartheta$.

A pre-tableau for a formula $\vartheta$ is an infinite tree s.th.

- it is finitely branching,
- each node is labelled with a goal (as a set),

$$A\Sigma_1, \ldots, A\Sigma_n, E\Pi_1, \ldots, E\Pi_m, \Lambda$$

$$\bigwedge_{i=1}^n A(\bigvee \Sigma_i) \land \bigwedge_{i=1}^m E(\bigwedge \Pi_i) \land \Lambda$$

Example: $\{A\{\neg p \lor q\}, E\{Xp, Fq\}, \neg p, \neg q\}$.
Sloppy writing: $A(\neg p \lor q)$ or $E(Xp, \Pi)$, e.g.
A Tableau for CTL*

A tableau for $\vartheta$ is a tree which imitates a potential model of $\vartheta$.

A pre-tableau for a formula $\vartheta$ is an infinite tree s.th.

- it is finitely branching,
- each node is labelled with a goal (as a set),
  \( A\Sigma_1, \ldots, A\Sigma_n, E\Pi_1, \ldots, E\Pi_m, \Lambda \)
- nodes are locally consistent, i.e.
  - does not contain a literal together with its negation, and
  - does not contain $A\emptyset$.
- root is labelled with $E\{\vartheta\}$,
- nodes follow the following rules . . .
Exemplary Rules

\[
\begin{align*}
(E \lor) & \quad \frac{E(\varphi, \Pi), \Phi \mid E(\psi, \Pi), \Phi}{E(\varphi \lor \psi, \Pi), \Phi} \\
(E \land) & \quad \frac{E(\varphi, \psi, \Pi), \Phi}{E(\varphi \land \psi, \Pi), \Phi} \\
(EF) & \quad \frac{E(\psi, \Pi), \Phi \mid E(X(F\psi), \Pi), \Phi}{E(F\psi, \Pi), \Phi} \\
(AF) & \quad \frac{A(\psi, X(F\psi), \Sigma), \Phi}{A(F\psi, \Sigma), \Phi} \\
(X_1) & \quad \frac{E\Pi_1, A\Sigma_1, \ldots, A\Sigma_m, \Phi \mid \ldots \mid E\Pi_n, A\Sigma_1, \ldots, A\Sigma_m, \Phi}{EX\Pi_1, \ldots, EX\Pi_n, AX\Sigma_1, \ldots, AX\Sigma_m, \Lambda, \Phi}
\end{align*}
\]
Traces

- A trace is an infinite sequence of connected blocks.
- A trace is an A- resp. E- trace iff the block quantifier eventually remains A resp. E.

Thread

- A thread is an infinite sequence of connected formulas.
- A thread is an F- resp. G-thread iff there is some $\psi$ s.t. the thread finally alternates between $F\psi$ or $XF\psi$ (resp. G...).
Pre-tableaux are insufficient – an informal discussion

- In the intended model
  - every formula on a $F$-thread is false, and
  - every formula on a $G$-thread is true.

- Blocks in an $E$-trace is understood as a conjunction.
  - Avoid $F$-threads.

- Blocks in an $A$-trace is understood as a disjunction.
  - Assure a $G$-thread.

Definition

A tableau for $\vartheta$ is a pre-tableau for $\vartheta$ iff on every branch we have
- every $E$-trace does not contain an $F$-thread, and
- every $A$-trace contains a $G$-thread.

Such traces and branches are called good.
Successful Tableau for $\text{AFG}p \land \text{GEF}\neg p$

\[
\frac{\text{A}(Gp, XFGp)}{\text{A}(Gp, FGp)}
\]
\[
\frac{\text{A}(Gp, XFGp), A(XGp, XFGp)}{\rightarrow \text{A}(Gp, XFGp)}
\]
\[
\frac{\text{A}(FGp), A(Gp, FGp)}{\text{A}(XFGp), A(XGp, XFGp), \neg p}
\]
\[
\frac{\text{A}(p, XFGp), A(XGp, XFGp), \neg p}{\text{A}(Gp, XFGp), E(\neg p)}
\]
\[
\frac{\text{A}(Gp, XFGp), E(\neg p)}{\text{A}(Gp, FGp), E(F\neg p)}
\]
\[
\frac{\text{A}(Gp, XFGp), E(\neg p)}{\text{A}(Gp, FGp), E(\text{GEF}\neg p)}
\]
\[
\frac{\text{A}(Gp, XFGp), E(\text{EF}\neg p, XGEF\neg p)}{\text{A}(Gp, XFGp), E(\text{GEF}\neg p)}
\]
\[
\frac{\text{A}(Gp, XFGp), E(\text{EF}\neg p, XGEF\neg p)}{\rightarrow \text{A}(Gp, XFGp), E(\text{GEF}\neg p)}
\]
\[
\frac{\text{A}(FGp), E(\text{GEF}\neg p)}{\rightarrow \text{E}(\text{AFG}p, \text{GEF}\neg p)}
\]
\[
\frac{\text{E}(\text{AFG}p, \text{GEF}\neg p)}{\rightarrow \text{E}(\text{AFG}p \land \text{GEF}\neg p)}
\]

$\models \text{AFG}p \land \text{GEF}\neg p$
Successful Tableau for $\text{AFG}p \land \text{EGEF}\neg p$
Successful Tableau for $\text{AFG}p \land \text{EGEF} \neg p$
Given a $\text{CTL}^*$-formula $\vartheta$, decide whether $\vartheta$ is satisfiable.
Decision Procedure

Given a \( \text{CTL}^* \)-formula \( \vartheta \), decide whether there is a tableau for \( \vartheta \).
Decision Procedure

Given a \( \text{CTL}^* \)-formula \( \vartheta \), decide whether \( \vartheta \) is satisfiable.

Idea: treat a tableau as a parity game.
Reduction to Parity Games

The tableaux as a game

- **Nodes** are the goals for $\varphi$.
- **Proponent (player 0)** chooses a rule application if neither $(X_0)$ nor $(X_1)$ is applicable.
- **Opponent (player 1)** chooses a rule application and a premise if $(X_0)$ or $(X_1)$ is applicable.
Reduction to Parity Games

The tableaux as a game

- **Nodes** are the goals for $\vartheta$.
- **Proponent (player 0)** chooses a rule application if neither $(X_0)$ nor $(X_1)$ is applicable.
- **Opponent (player 1)** chooses a rule application and a premise if $(X_0)$ or $(X_1)$ is applicable.

Problem

This game defines a **pre-tableaux** but not a tableaux.

Observation

The property separating pre-tableau and tableaux is $\omega$-regular.
1 Syntax and Semantic of CTL*
2 Emerson-Jutla Method
3 Reynolds’ Tableaux
4 Infinite Tableaux
5 Experimental Results
Implementation – Our vs. Reynold

Note: Reynold’s implementation is a proof-of-concept in Java but compiled with gcj.

- Formula \((\AG(p \rightarrow EXr) \land \AG(r \rightarrow EXp)) \rightarrow (p \rightarrow \EG(Fp \land Fr))\)

<table>
<thead>
<tr>
<th></th>
<th>formula</th>
<th>negated formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reynold</td>
<td>&gt; 10h</td>
<td>&gt; 10h</td>
</tr>
<tr>
<td>Ours</td>
<td>0s</td>
<td>15s</td>
</tr>
</tbody>
</table>

- Formula \(\AG((p \land X\neg p \land \neg q \land \neg r) \lor (\neg p \land Xp \land q \land \neg r) \lor (\neg p \land Xp \land \neg q \land r)) \land \EG(Fq \land Fr)\)

<table>
<thead>
<tr>
<th></th>
<th>formula</th>
<th>negated formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reynold</td>
<td>17s</td>
<td>&gt; 10h</td>
</tr>
<tr>
<td>Ours</td>
<td>0s</td>
<td>0s</td>
</tr>
</tbody>
</table>
## Concluding Comparison

<table>
<thead>
<tr>
<th>Aspect / Method</th>
<th>Emerson et. al.</th>
<th>Reynolds</th>
<th>ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept</td>
<td>tree-automata</td>
<td>tableau</td>
<td>tableau</td>
</tr>
<tr>
<td>Worst-case complexity</td>
<td>2EXPTIME</td>
<td>2EXPTIME</td>
<td>2EXPTIME</td>
</tr>
<tr>
<td>Implementation available</td>
<td>no</td>
<td>not public</td>
<td>yes</td>
</tr>
<tr>
<td>Model construction</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Finite representation by</td>
<td>rabin</td>
<td>small. mod. p.</td>
<td>parity</td>
</tr>
<tr>
<td>Out-degree</td>
<td>fix., lin. bounded</td>
<td>var., lin. bounded</td>
<td>var., lin. bounded</td>
</tr>
<tr>
<td>Req. small model property</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Derives small model prop.</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Needs Büchi determ.</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>