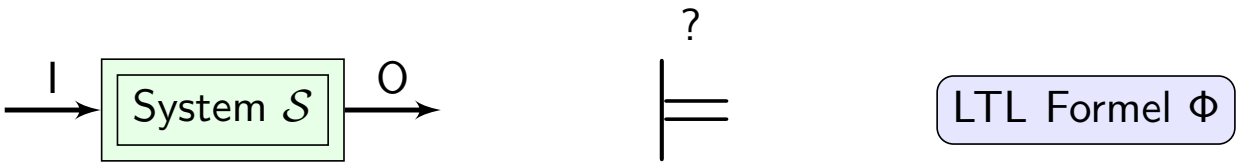
Symbolic Controller Synthesis for LTL Specifications

Andreas Morgenstern

12. Februar 2010

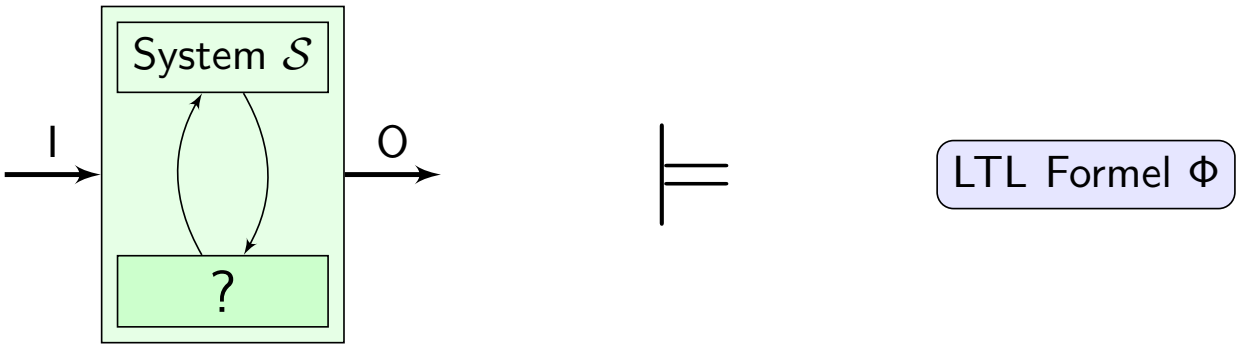
Übersicht

Modellprüfung



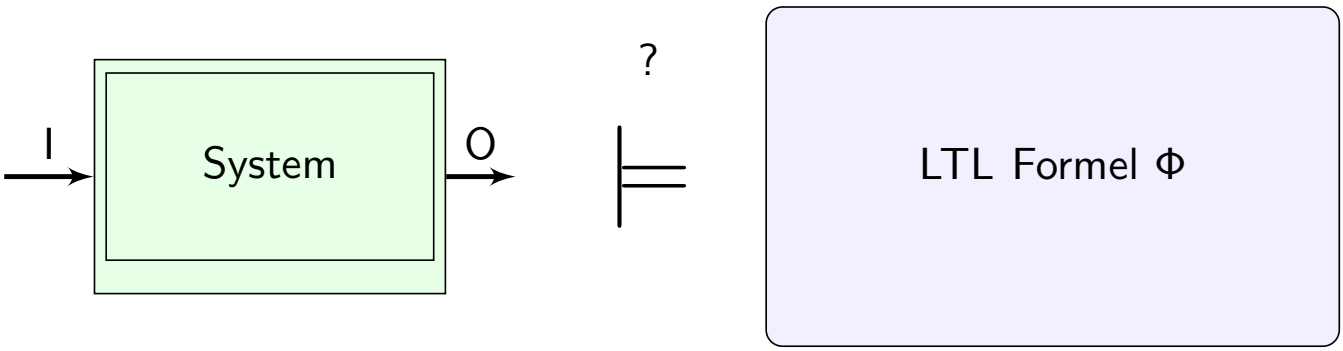
- Gegeben: Modell eines Systems $\mathcal{S}$
- Spezifikation: Formel Φ in der Temporallogik LTL
- Fragestellung: $\mathcal{S} \models \Phi$?

Controller Synthese



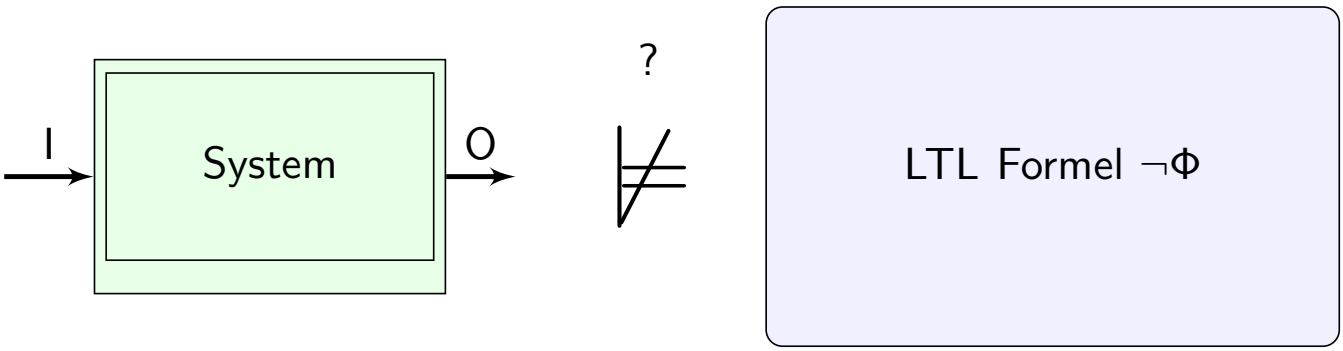
- Gegeben: Modell eines Systems $\mathcal{S}$
- Spezifikation: Formel Φ in der Temporallogik LTL
- Fragestellung: $\exists$ Controller $\mathcal{C}$. $\quad \mathcal{C} \times \mathcal{S} \models \Phi$?

Modellprüfung



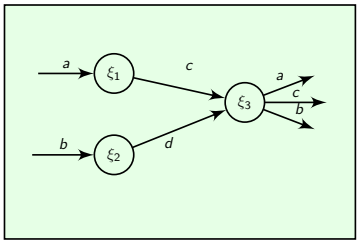
Fragestellung: $\mathcal{S} \models \Phi$?

Modellprüfung

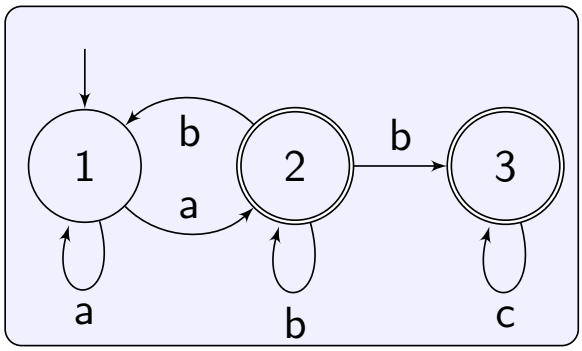


$$(\mathcal{S} \models \Phi) \leftrightarrow (\mathcal{S} \not\models \neg\Phi) \ ?$$

Automatenbasierte Modellprüfung



$\times$



- nicht-terminierende Systeme !

Büchi-Automaten

- Automaten lesen unendliche Worte
- Automat akzeptiert, falls ein $\mathcal{F}$ Zustand ∞ oft besucht wird !

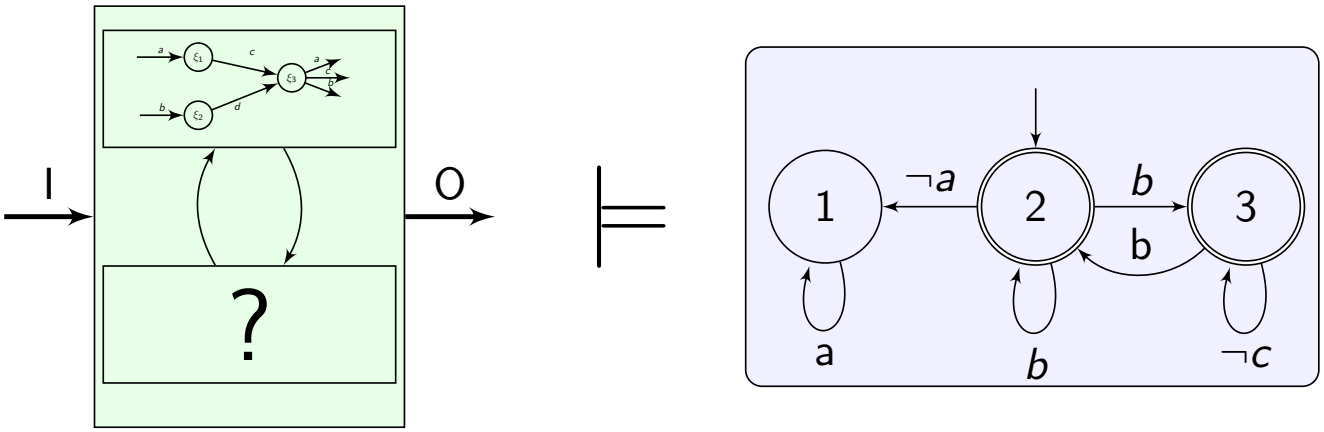
- $(\mathcal{S} \models \Phi) \leftrightarrow (\mathcal{S} \not\models \neg\Phi) \leftrightarrow \mathcal{L}(\mathcal{S} \times \mathcal{A}_{\neg\Phi}) = \emptyset)$?
- Graphsuche nach akzeptierendem Lauf für Negation !

Symbolische Modellprüfung

$$\mathcal{R} = \begin{array}{l} p_0 \wedge a \vee \\ p_1 \wedge \neg b \\ \ldots \end{array} \quad \bigwedge \quad \mathcal{R} = \begin{array}{l} r_0 \leftrightarrow (a \vee \neg b) \wedge \\ r_1 \leftrightarrow (c \vee d) \wedge \\ \ldots \end{array}$$

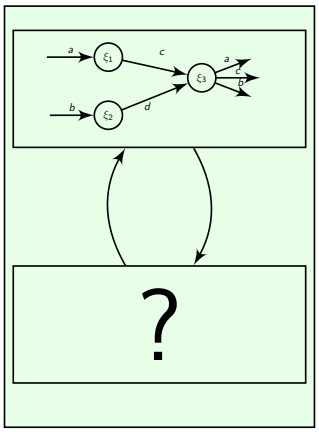
- aussagenlogische Repräsentation von
  - System und
  - Büchi Automat
- Vorteile:
  - Formeln repräsentieren große Zustandsräume
  - effiziente Verfahren wie BDD / SAT
- Probleme aus Industrie lösbar:
  - Bsp: Verifikation bei Intel

Automatenbasierte Controller Synthese

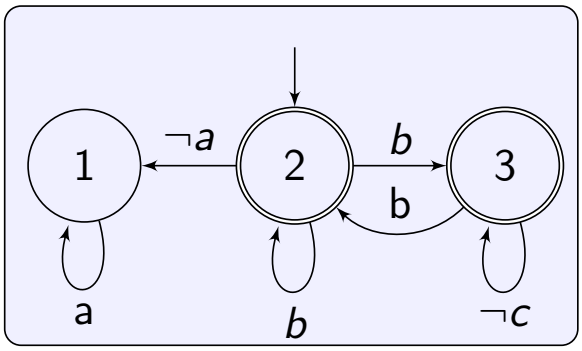


- $(\exists \mathcal{C}.\ \mathcal{S} \times \mathcal{C} \models \Phi) \leftrightarrow \exists \mathcal{C}.\ \mathcal{L}(\mathcal{S} \times \mathcal{C}) \subseteq \mathcal{L}(\mathfrak{A}_\Phi)$

Automatenbasierte Controller Synthese

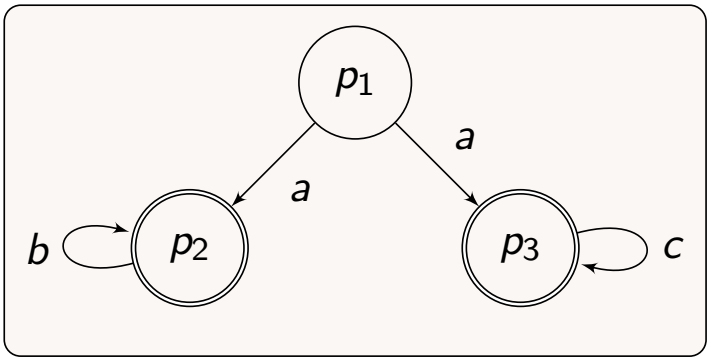


$\times$



- $(\exists \mathcal{C}.\ \mathcal{S} \times \mathcal{C} \models \Phi) \leftrightarrow \exists \mathcal{C}.\ \mathcal{L}(\mathcal{S} \times \mathcal{C}) \subseteq \mathcal{L}(\mathfrak{A}_\Phi)$
- Idee: Suche erfüllenden Teilautomaten für alle Eingaben auf dem Produktautomaten.

Automatenbasierte Controller Synthese

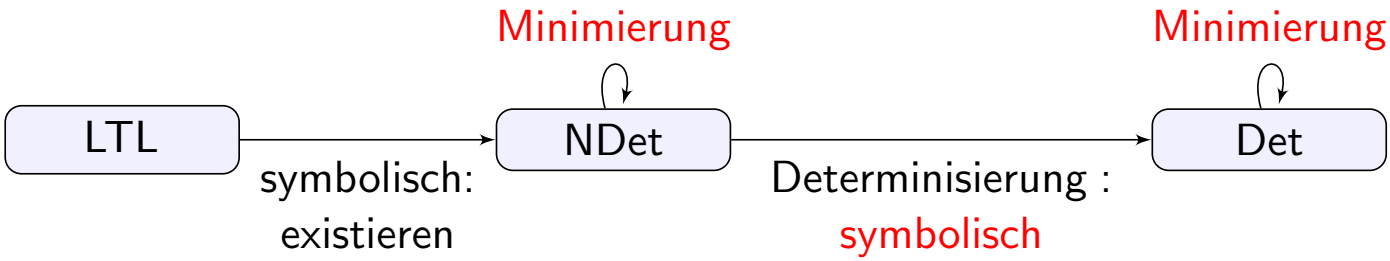


- Idee: Suche erfüllenden Teilautomaten für alle Eingaben auf Produktautomat!
- **Problem: Nichtdeterminismus**
- intuitiv: Zukunft unbekannt: kommt b oder c als nächstes?
- deterministischer Controller aus nichtdeterminisistschem Produktautomat ↯

Determinisierung von Büchi Automaten: Fakten

- “Normale“ Teilmengenkonstruktion genügt nicht!
- Es existieren Verfahren zur Determinisierung von Büchi Automaten (Safra, 1988).
- erste Implementierung : 2006
- keine **symbolische** Implementierung bekannt
- deshalb: nur kleine Probleme handhabbar

Beiträge der Dissertation



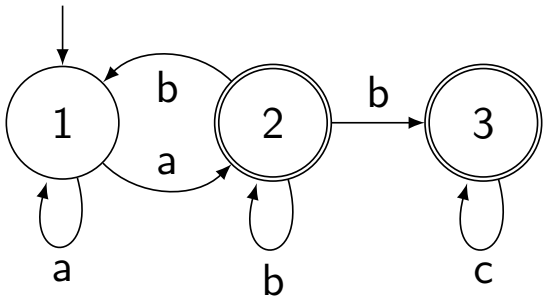
- symbolische Übersetzung LTL $\rightarrow$ NDet $\surd$
- symbolische Algorithmen, um Controller aus deterministischem System zu gewinnen $\surd$
- Minimierung für Automaten existieren $\surd$
  - Minimierung verbessert (im folgenden nicht weiter betrachtet)

Fehlt:

- symbolische Determinisierungsverfahren

Übersicht

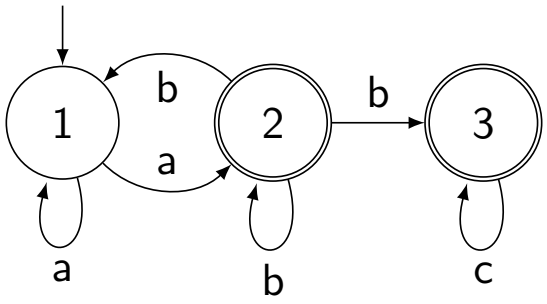
ω-Automaten



ω-Automaten

- ω-Automaten lesen unendliche Worte.
- unterschiedliche Akzeptanzbedingungen:
  - Büchi : $\mathcal{F}$ Zustände unendlich oft besuchen

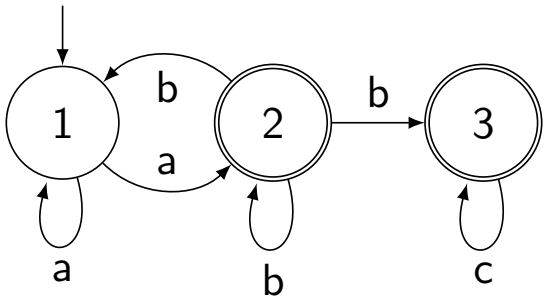
ω-Automaten



ω-Automaten

- ω-Automaten lesen unendliche Worte.
- unterschiedliche Akzeptanzbedingungen:
  - Büchi : $\mathcal{F}$ Zustände unendlich oft besuchen
  - Co-Büchi: $\neg\mathcal{F}$ Zustände endlich oft besuchen

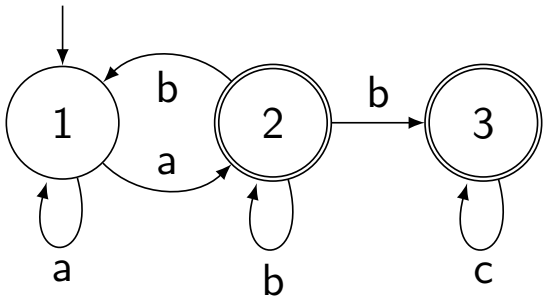
ω-Automaten



ω-Automaten

- ω-Automaten lesen unendliche Worte.
- unterschiedliche Akzeptanzbedingungen:
  - Büchi : $\mathcal{F}$ Zustände unendlich oft besuchen
  - Co-Büchi: $\neg\mathcal{F}$ Zustände endlich oft besuchen
  - Streett: Boolsche Kombination von Büchi und Co-Büchi (in Normalform)

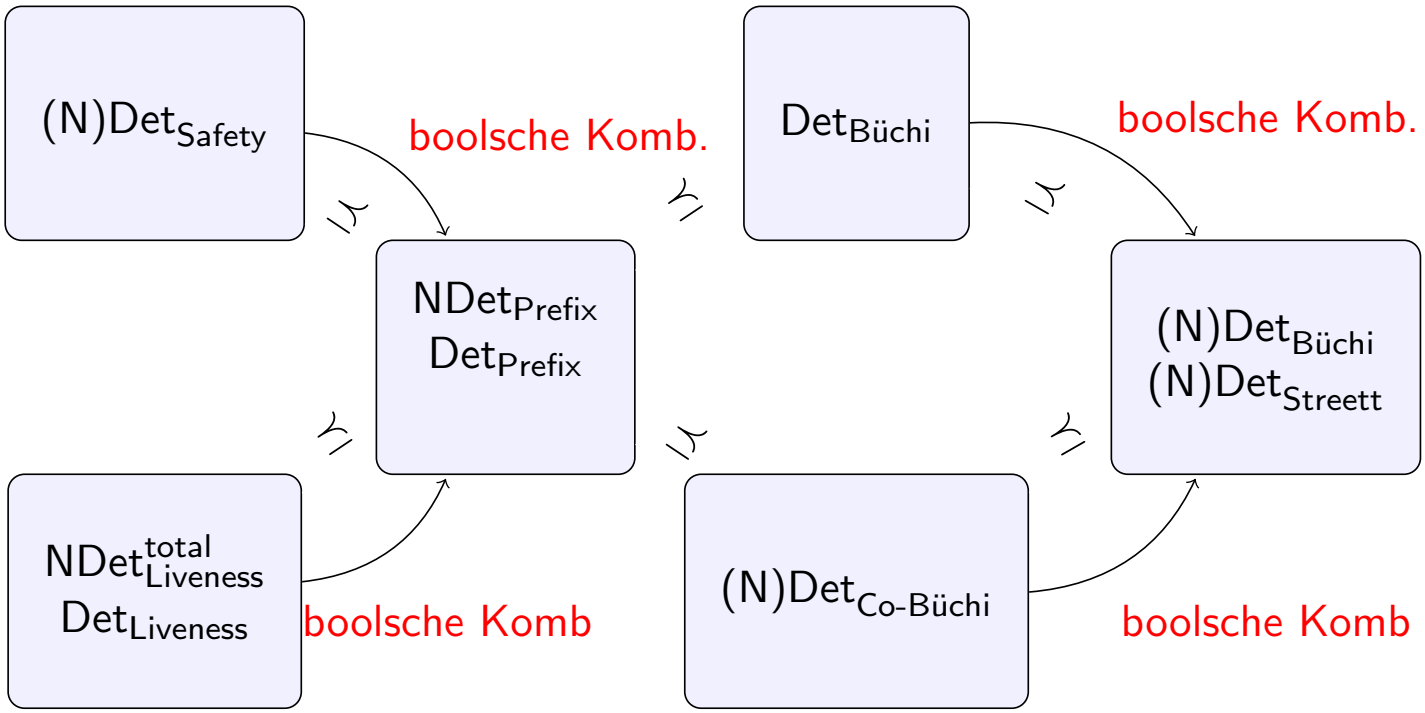
ω-Automaten



ω-Automaten

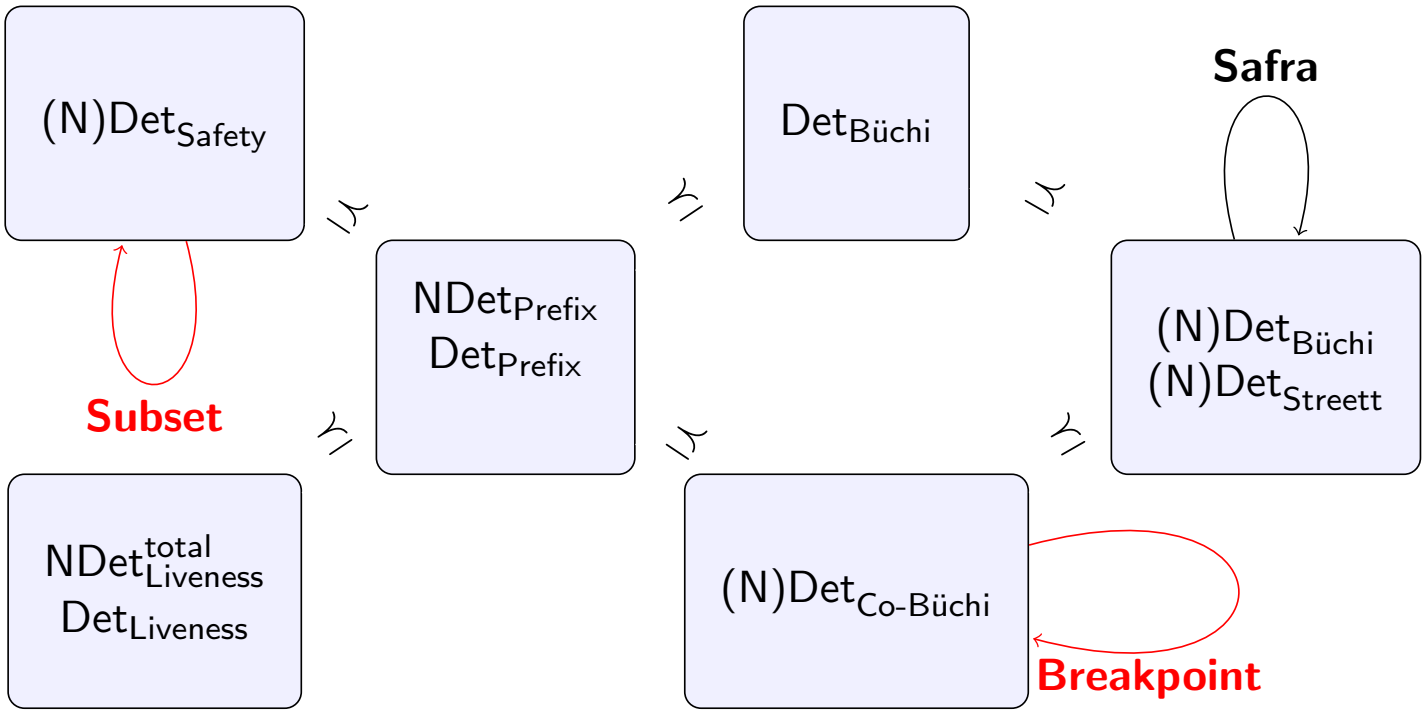
- ω-Automaten lesen unendliche Worte.
- unterschiedliche Akzeptanzbedingungen:
  - Büchi : $\mathcal{F}$ Zustände unendlich oft besuchen
  - Co-Büchi: $\neg\mathcal{F}$ Zustände endlich oft besuchen
  - Streett: Boolsche Kombination von Büchi und Co-Büchi (in Normalform)
  - Safety : nur $\mathcal{F}$ Zustände besuchen
  - Liveness : nur $\mathcal{F}$ Zustände einmal besuchen
  - Prefix : Boolsche Kombination von Safety und Liveness (in Normalform)

Die Automatenhierarchie (Wagner, 1979)



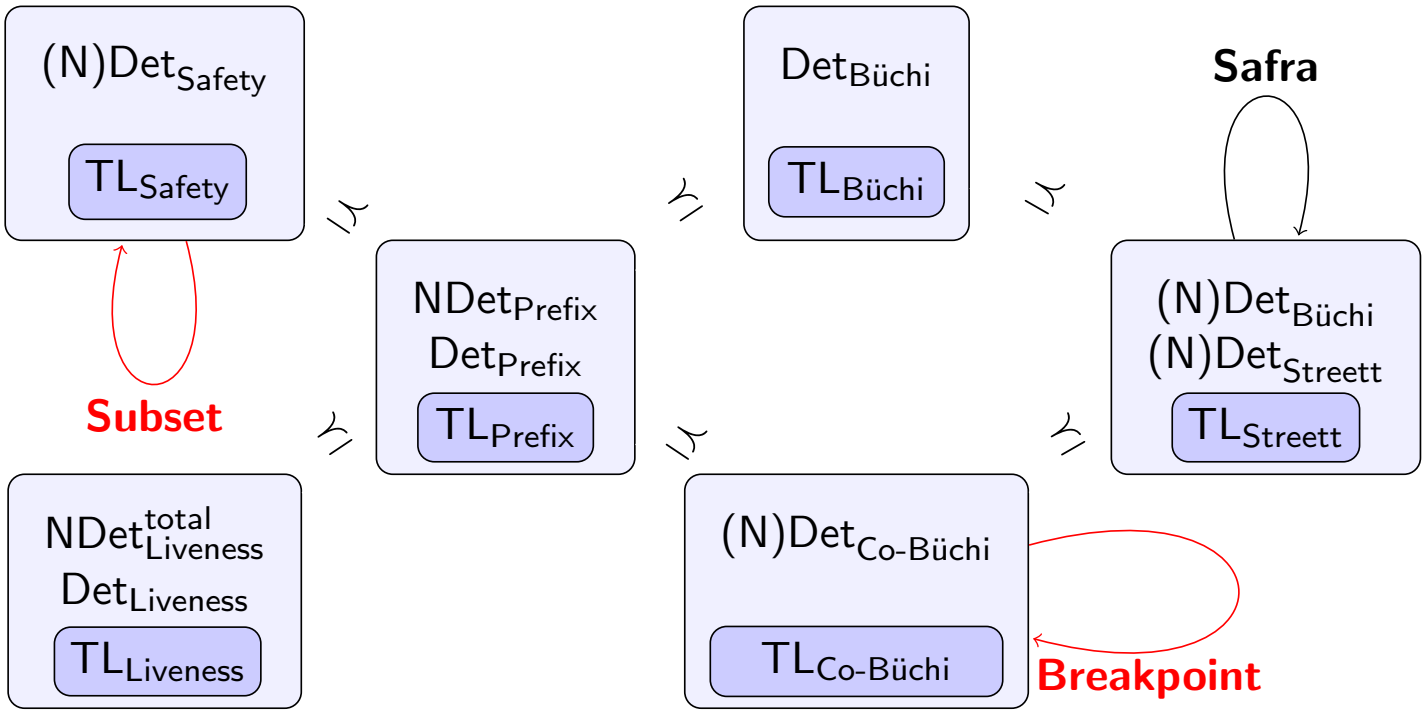
$\mathfrak{C}_1 \preceq \mathfrak{C}_2 :=$ Automat aus $\mathfrak{C}_1$ übersetzbar in Automat aus $\mathfrak{C}_2$

Die Automatenhierarchie (Wagner, 1979)



$\mathfrak{C}_1 \preceq \mathfrak{C}_2 :=$ Automat aus $\mathfrak{C}_1$ übersetzbar in Automat aus $\mathfrak{C}_2$

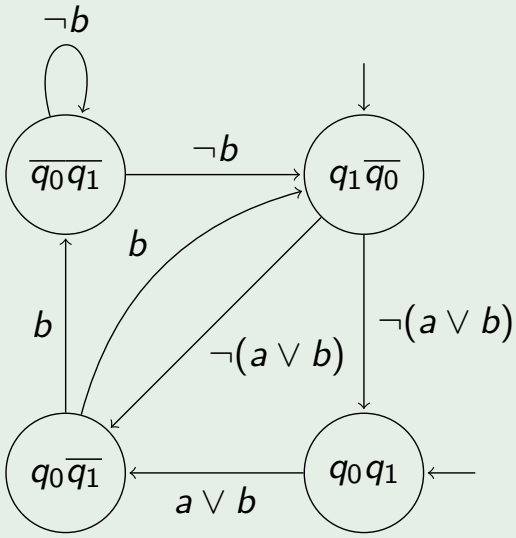
Die Temporallogikhierarchie (Manna&Pnueli, 1987)



$\mathfrak{C}_1 \preceq \mathfrak{C}_2 :=$ Automat aus $\mathfrak{C}_1$ übersetzbar in Automat aus $\mathfrak{C}_2$

Symbolische Subset-Konstruktion: 1. Onehot-Kodierung

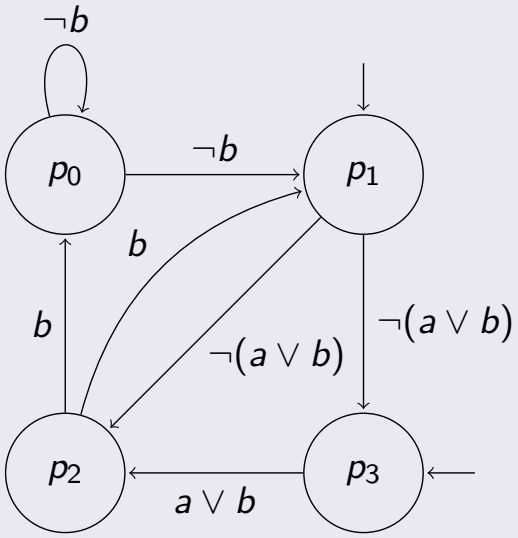
Nichtdeterministischer Automat



$$\mathcal{R} = \begin{array}{c} (q_0 \leftrightarrow b \vee a \wedge {q_0}') \wedge \\ (q_1 \leftrightarrow {q_0}') \end{array}$$

4

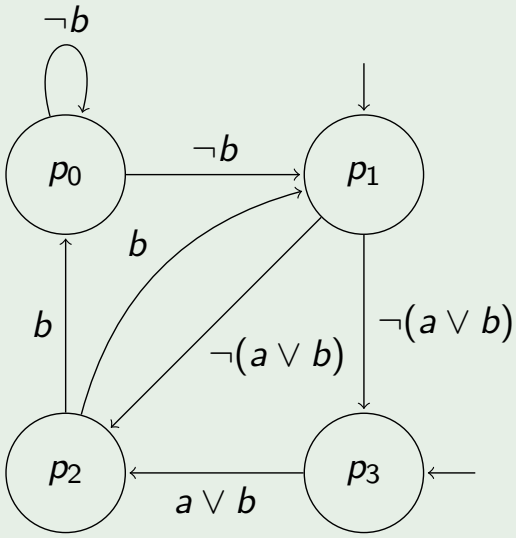
Onehot-kodierter Automat



$$\mathcal{R} = \begin{array}{c} (p_0 \wedge \neg b \vee p_2 \wedge b) \wedge p_1' \vee \\ (p_1 \wedge (\neg(a \vee b))) \wedge p_3' \vee \\ \ldots \end{array}$$

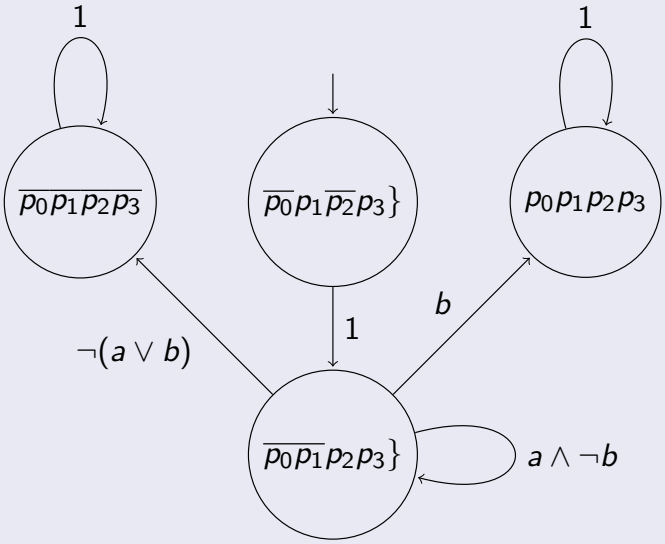
Symbolische Subset-Konstruktion: 2. Determinisierung

Onehot-kodierter Automat



$$\mathcal{R} = \begin{array}{l}(p_0 \wedge \neg b \vee p_2 \wedge b) \wedge p_1' \vee \\ (p_1 \wedge (\neg(a \vee b))) \wedge p_3' \vee \\ \ldots\end{array}$$

Deterministischer Automat



$$\mathcal{R} = \begin{array}{l}(p_0 \wedge \neg b \vee p_2 \wedge b) \leftrightarrow p_1' \vee \\ (p_1 \wedge (\neg(a \vee b))) \leftrightarrow p_3' \vee \\ \ldots\end{array}$$

Symbolische Subset-Konstruktion: BDD-Implementierung

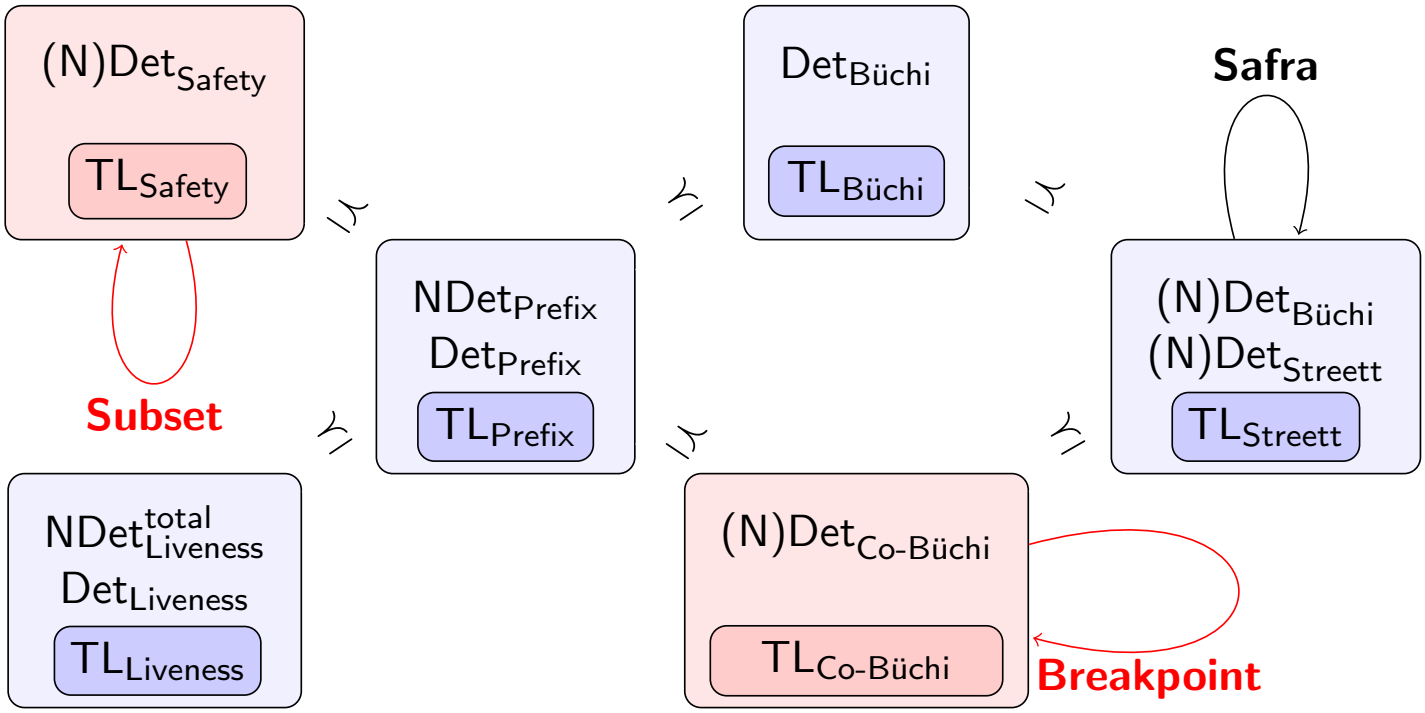
Definition (Symbolische Subset Construction)

- $\mathcal{H} :\equiv \bigvee_{j=1}^{n} p_j \wedge \mathrm{mt}_{Q_{\mathrm{det}}}(\vartheta_j)$
- $\mathcal{I}_{\mathrm{det}} :\equiv \bigwedge_{i=1}^{n} p_i \leftrightarrow \exists q_1 \ldots q_m.\ \mathrm{mt}_{Q_{\mathrm{det}}}(\vartheta_i) \wedge \mathcal{I}$
- $\mathcal{F}_{\mathrm{det}} :\equiv \exists q_1 \ldots q_m.\ \mathcal{H} \wedge \mathcal{F}$
- $\mathcal{R}_{\mathrm{det}} :\equiv \bigwedge_{i=1}^{n} p_i{}' \leftrightarrow \eta_i$, mit

  $\eta_i :\equiv \exists q_1 \ldots q_m q_1{}' \ldots q_m{}'.\ \mathcal{H} \wedge \mathcal{R} \wedge [\mathrm{mt}_{Q_{\mathrm{det}}}(\vartheta_i)]_{q_1',\ldots,q_m'}^{q_1,\ldots,q_m}$

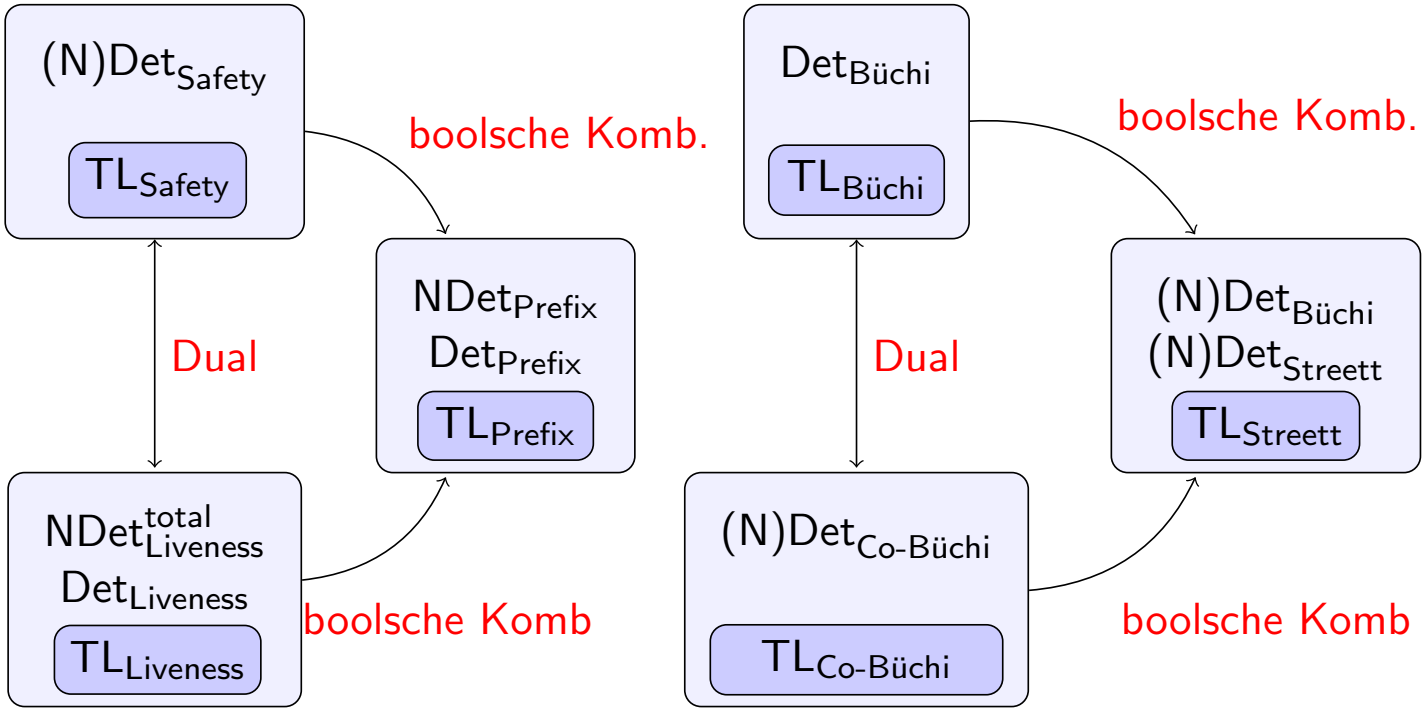
- nicht-symbolischer Schritt: Aufzählen der n nichtdeterministischen Zustände.

Symbolische Determinisierung via Automatenhierarchie



Symbolische Automaten für $\mathrm{TL}_{\mathrm{Safety}}$ und $\mathrm{TL}_{\text{Co-Büchi}}$

Symbolische Determinisierung via Automatenhierarchie



Symbolische Automaten für $TL_{Liveness}$, TL_{Prefix}, $TL_{Büchi}$, $TL_{Streett}$

Determinisierung via Automatenhierarchie: Zusammenfassung

Grundlage

- Subset (Breakpoint) Konstruktion symbolisch
- Boolsche Kombination der Formeln/ der det. Automaten

Vorteile

- deterministischer Automat niemals explizit
- kein Aufzählen der deterministischen Zustände
- effizient: wegen boolscher Kombination Teilautomaten sehr klein (< 20 ndet Zustände)
- praktisch alle Formeln sind aus $\mathrm{TL}_{\mathrm{Streett}}$

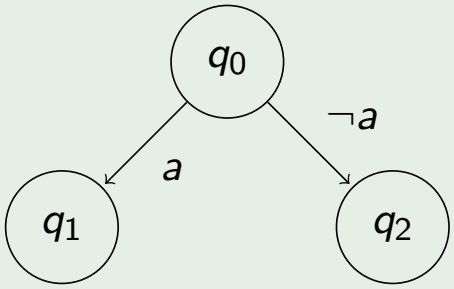
Nachteile

- **Nicht alle Formeln liegen in** $\mathrm{TL}_{\mathrm{Streett}}$

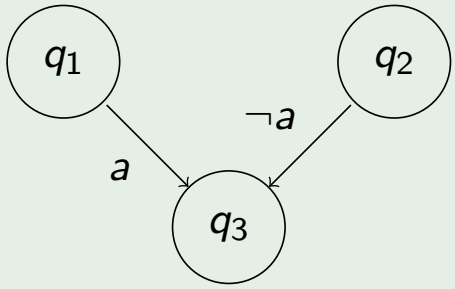
Übersicht

Grundlage: Rückwärts-Determinismus

Determinismus



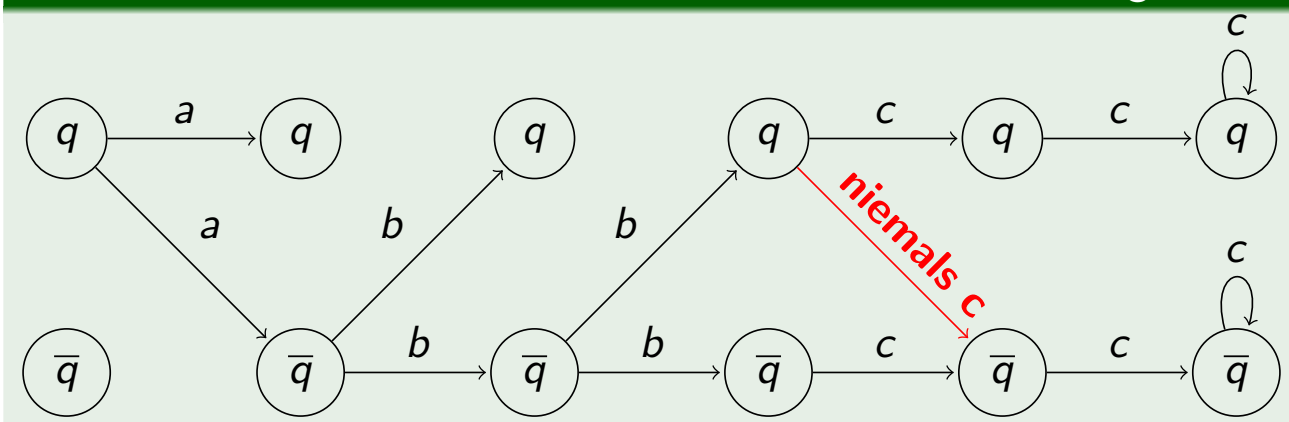
Rückwärts-Determinismus



- jeder Büchi Automat aus LTL ist rückwärts-deterministisch !

Läufe sind eindeutig

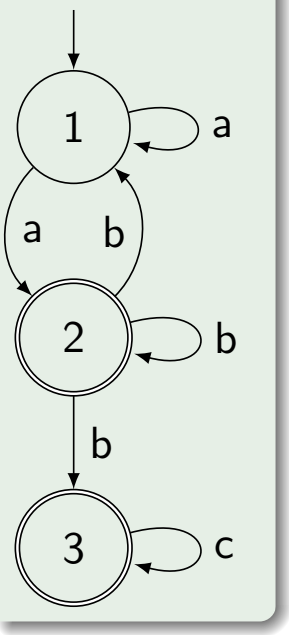
Läufe rückwärts-deterministischer Automaten sind eindeutig



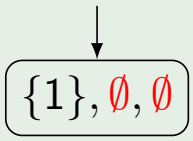
- jeder Büchi Automat aus LTL ist rückwärts-deterministisch !
- dadurch: Läufe sind eindeutig (unambiguous) bestimmt durch letzten Zustand!
- Idee: mehrere parallele Subset-Konstruktionen
- Mengen kodieren ob $\mathcal{F}$ Zustand besucht wurde !

Skizzierung des Determinisierungsverfahrens

NDet



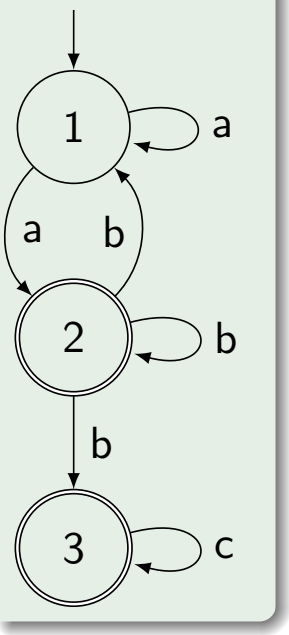
Deterministischer Automat



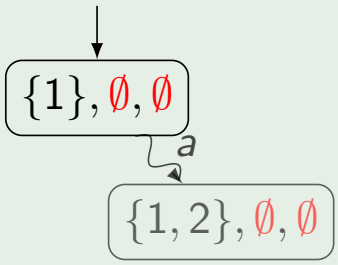
- Parallele Subset-Konstruktionen in Mengen $S_0 \ldots S_{n-1}$

Skizzierung des Determinisierungsverfahrens

NDet



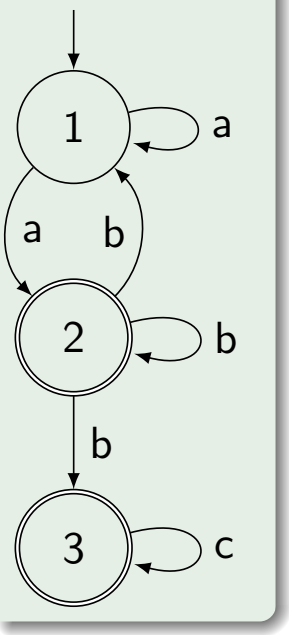
Deterministischer Automat



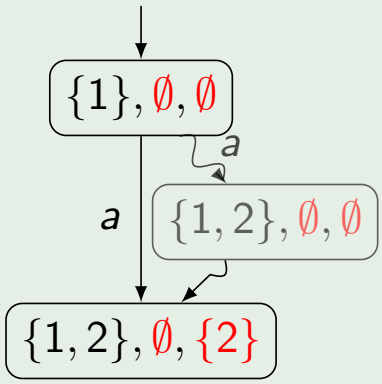
- Dead-Ends → rote Markierung

Skizzierung des Determinisierungsverfahrens

NDet



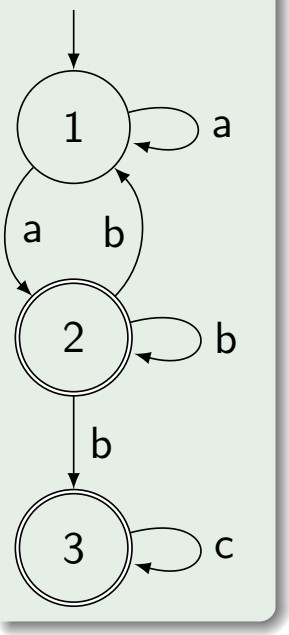
Deterministischer Automat



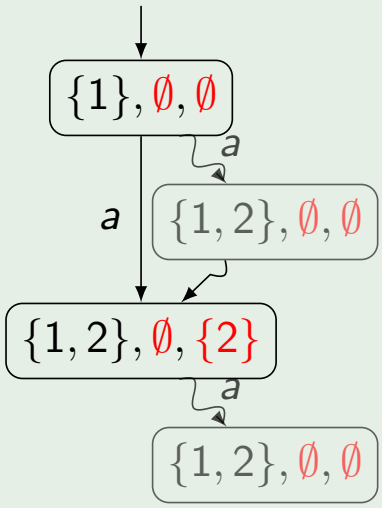
- $\mathcal{F}$ Zustände $\rightarrow$ neue Subsetkonstruktion rechts

Skizzierung des Determinisierungsverfahrens

NDet



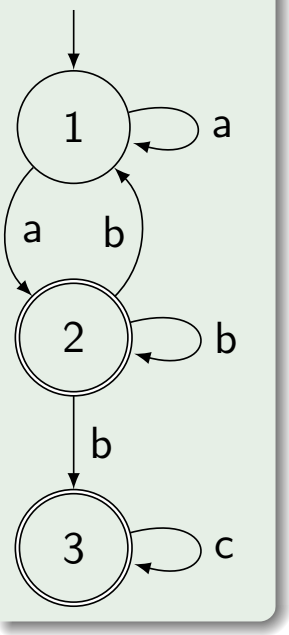
Deterministischer Automat



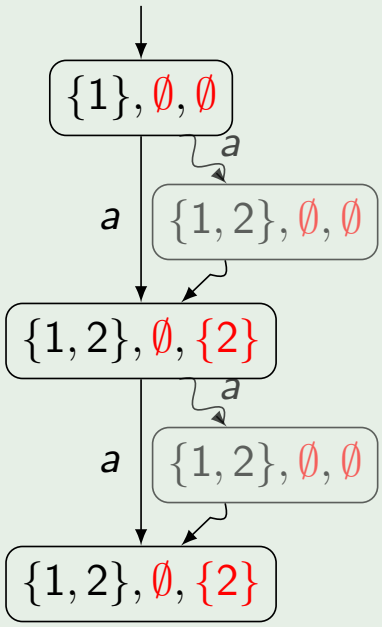
- Dead-Ends → rote Markierung

Skizzierung des Determinisierungsverfahrens

NDet



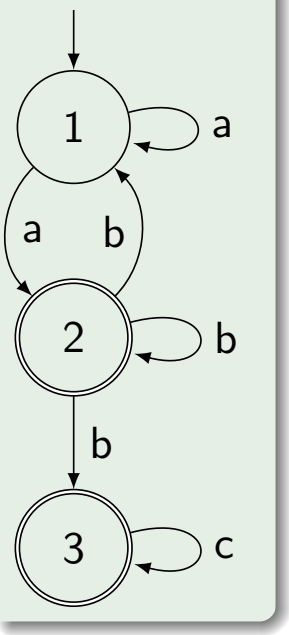
Deterministischer Automat



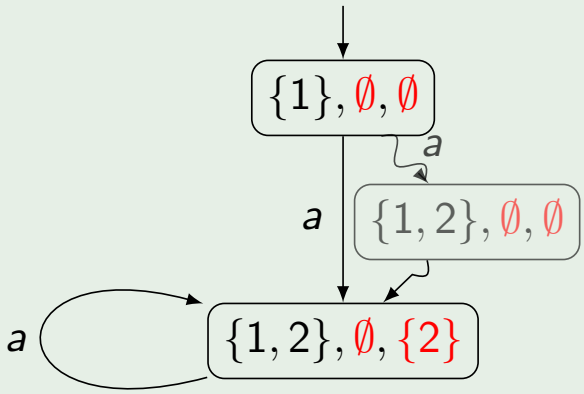
- $\mathcal{F}$ Zustände abspalten $\rightarrow$ gleicher Zustand !

Skizzierung des Determinisierungsverfahrens

NDet



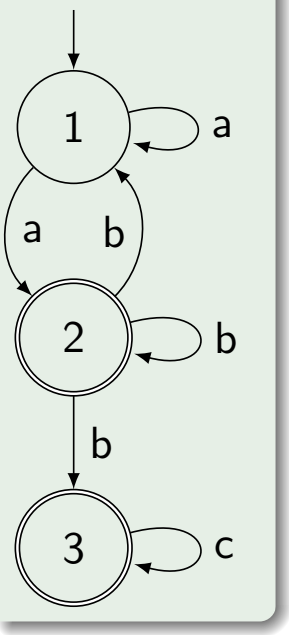
Deterministischer Automat



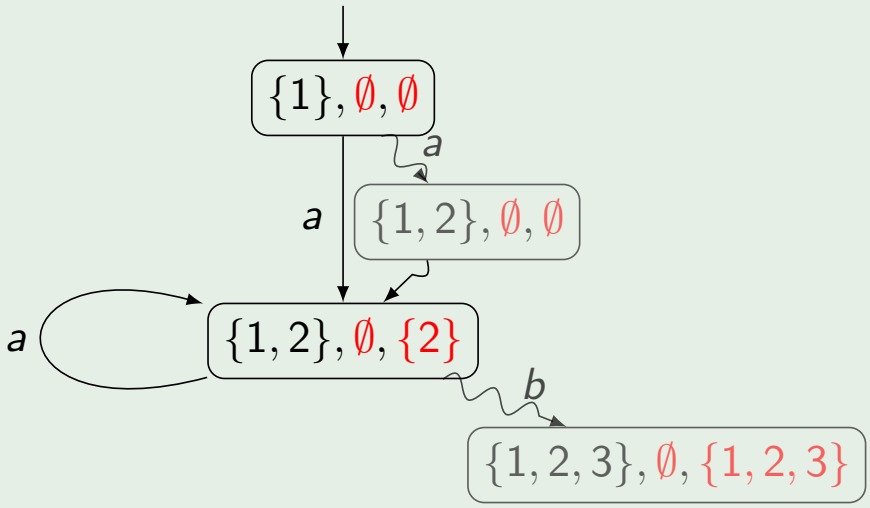
- gleicher Zustand!

Skizzierung des Determinisierungsverfahrens

NDet



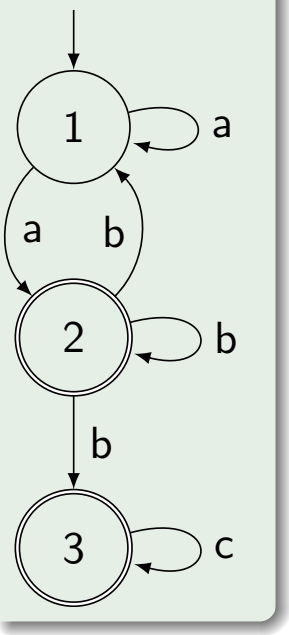
Deterministischer Automat



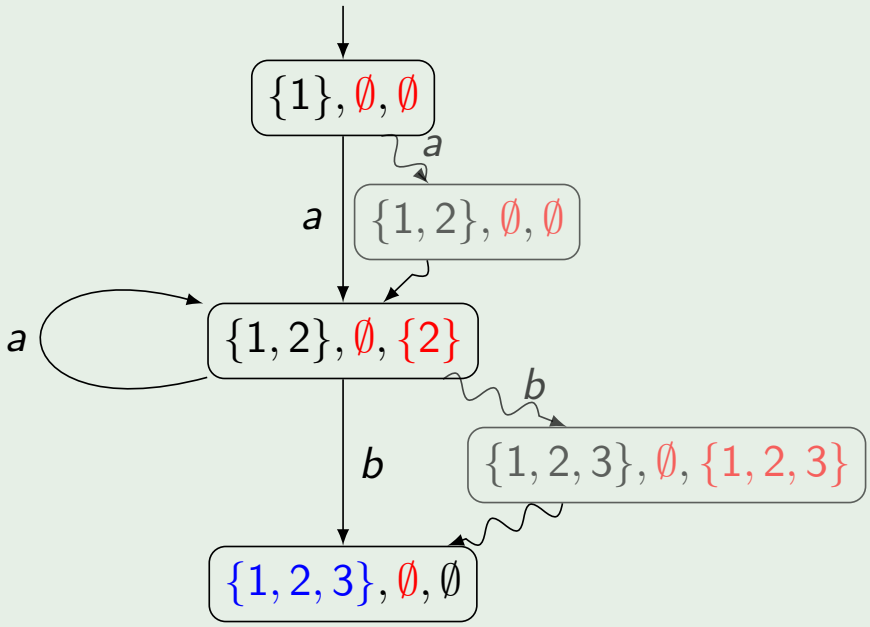
- parallele Subset-Konstruktion!

Skizzierung des Determinisierungsverfahrens

NDet



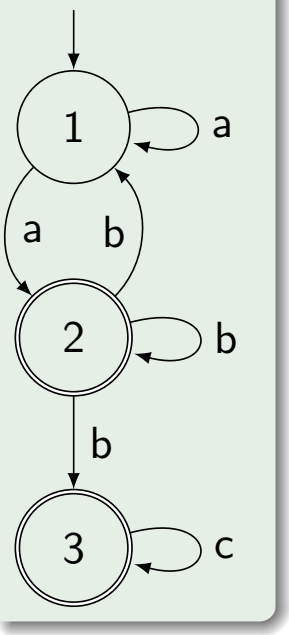
Deterministischer Automat



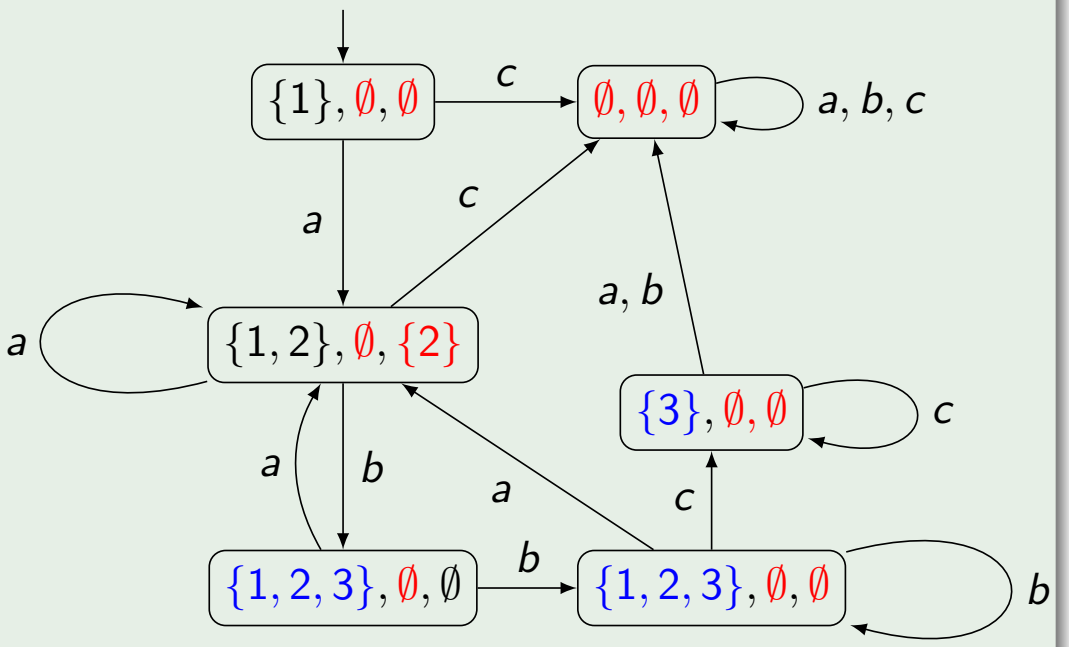
- Läufe eindeutig → blaue Markierung und Entfernen rechts

Skizzierung des Determinisierungsverfahrens

NDet



Deterministischer Automat



- Akzeptiere, falls Menge ($\neg\infty$ oft rot) und (∞ blau) !

Det. unambiguous Automaten:Zusammenfassung

Grundlage

- LTL Formeln liefern “unambiguous“ Automaten
- Läufe sind eindeutig
- n parallele Subset-Konstruktionen

Vorteile

- symbolische Implementierung
- lösbar bis ca. 20 nichtdeterministische Zustände
- erneut: boolsche Kombinationen det. Automaten

Nachteile

- BDDs explodieren für große Beispiele, da große Variablenzahl

Übersicht

Implementierung: Vergleich mit anderen Tools

Opal [Morgenstern]: Tool zur Controller Synthese

- symbolische Determinisierungsverfahren für ganz LTL
- symbolische Minimierung für unterschiedliche ω-Automaten
- symbolischer Controller Synthese Algorithmus für generalisierte Parity Spiele [Chatterjee et al, 2007]

Implementierung: Vergleich mit anderen Tools

Opal [Morgenstern]: Tool zur Controller Synthese

- symbolische Determinisierungsverfahren für ganz LTL
- symbolische Minimierung für unterschiedliche ω-Automaten
- symbolischer Controller Synthese Algorithmus für generalisierte Parity Spiele [Chatterjee et al, 2007]

Tools zur reinen LTL Synthese

Lily [Jobstmann & Bloem, 2006]

- statt Determinisierung universelle Baumautomaten
- explizite Implementierung
- erstes Tool, welches ganz LTL beherrscht

Implementierung: Vergleich mit anderen Tools

Opal [Morgenstern]: Tool zur Controller Synthese

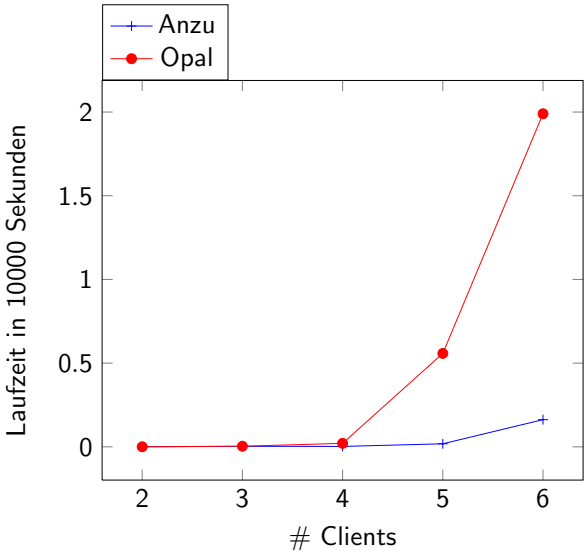
- symbolische Determinisierungsverfahren für ganz LTL
- symbolische Minimierung für unterschiedliche ω-Automaten
- symbolischer Controller Synthese Algorithmus für generalisierte Parity Spiele [Chatterjee et al, 2007]

Tools zur reinen LTL Synthese

Anzu [Jobstmann et al., 2007]

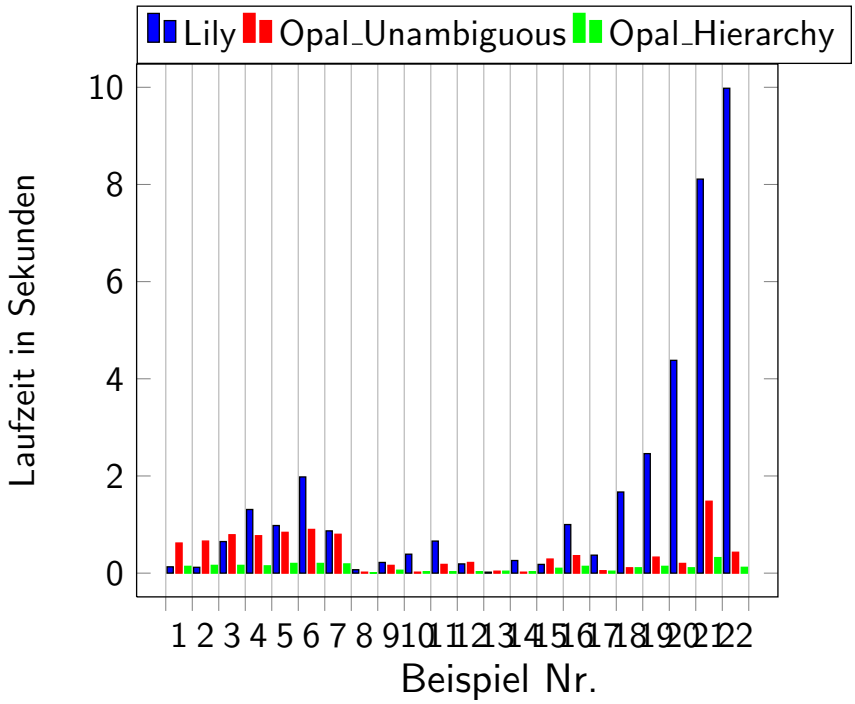
- symbolische Implementierung
- eingeschränkte Spezifikation (Implikation von Büchi-Akzeptanzbedingungen)
- Erweiterung: Determinisiere Büchi Automaten von Hand
- dennoch: Stärker eingeschränkt als $\text{TL}_{\text{Streett}}$

LTL Synthese: Industriespezifikation AMBA Arbiter



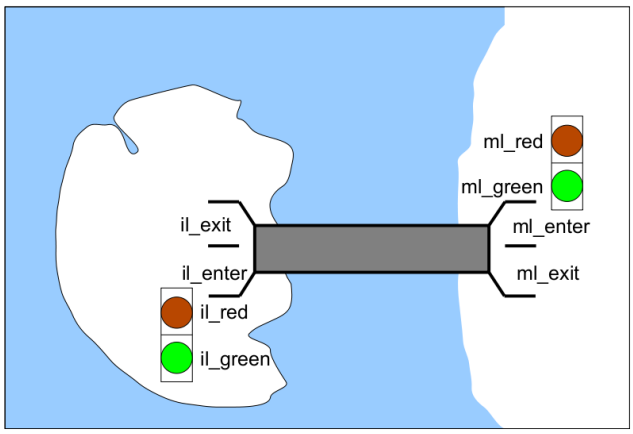
- Spezifikation Arbiter für AMBA Bus: Caches, Prozessoren, ...
- Je Client 10 Safety/ 1 Büchi Eigenschaft
- Insgesamt ca. 40 Temporaloperatoren/Client
- Anzu:
  Einfache Spezifikationen $\rightarrow$ einfache Synthesealgorithmen

LTL Synthese: Die Lily Beispiele

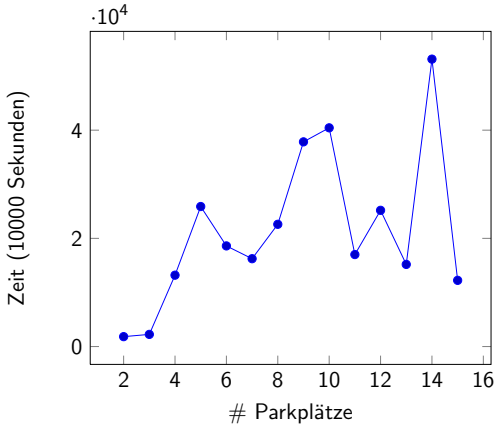


- 23 Spezifikationen: Arbiter, Mutex
- Temporaloperatoren: $4 - 40$, Boolsche Operatoren: $3 - 38$

Controller Synthese: Island Traffic Control



(a) Problem



(b) Laufzeit

- Zugangs-Controller für einspurigen Tunnel zu Insel
- Sicherheit: niemals grün auf beiden Seiten
- Fairness: kein unendlich langes Warten auf einer Seite
- Anzahl der Parkplätze auf Insel beschränkt
- Gleichzeitig große Umgebung / große Spezifikation

Zusammenfassung

Experimente

- Laufzeit wenige Sekunden bis einige Stunden
- **symbolische Implementierung schlägt explizite Implementierung eindeutig**
- viel Verbesserungspotential: AMBA-Beispiel
- **Controller Synthese für ganz LTL ist möglich!**

Ausblick

- Situation ähnlich Model Checking vor 20 Jahren
- kompositionale/ inkrementelle Controller Synthese
- bessere Strategiesynthesealgorithmen nötig
- Controller Synthese = Hilfsmittel für schwere Teilprobleme
- Industrieeinsatz ?

Danke