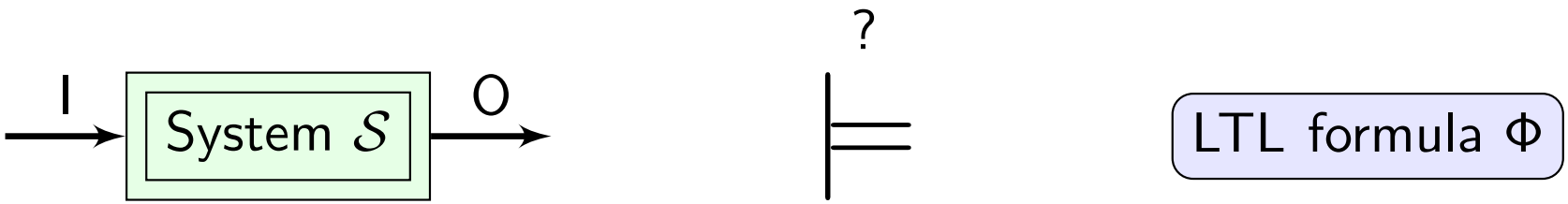
Exploiting the Temporal Logic Hierarchy and the Non-Confluence Property for Efficient LTL Synthesis

Andreas Morgenstern

GandALF 2010

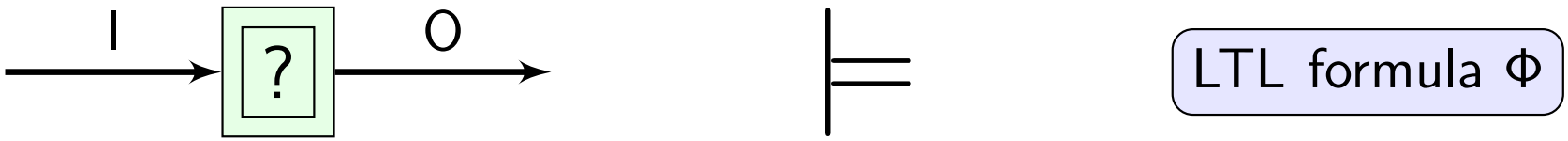
Overview

Model Checking



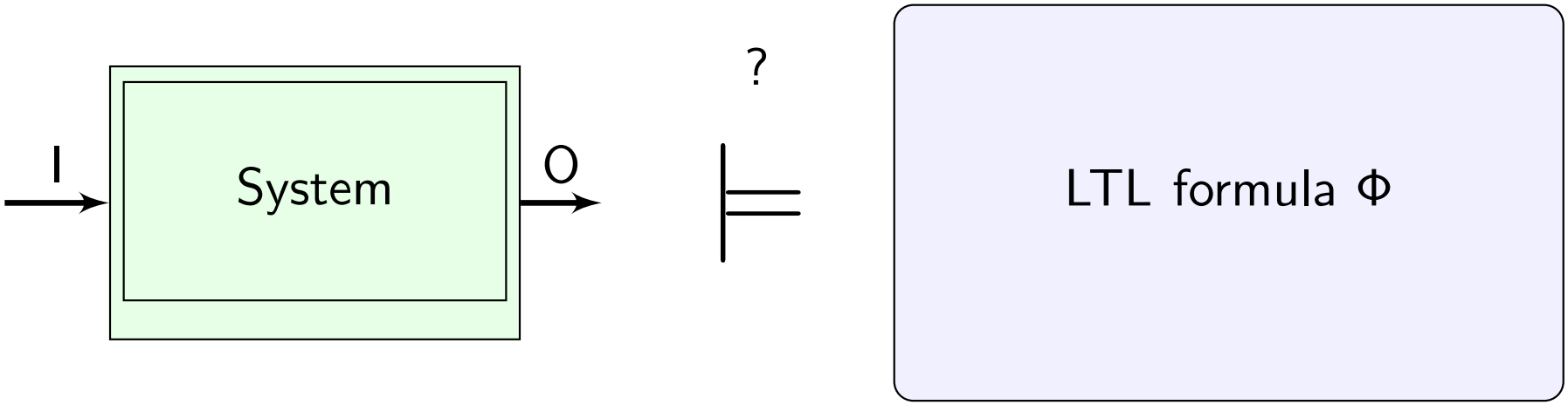
- Specification: Formula Φ in Temporal-Logic LTL
- Question: $\mathcal{S} \models \Phi$?

LTL Synthesis



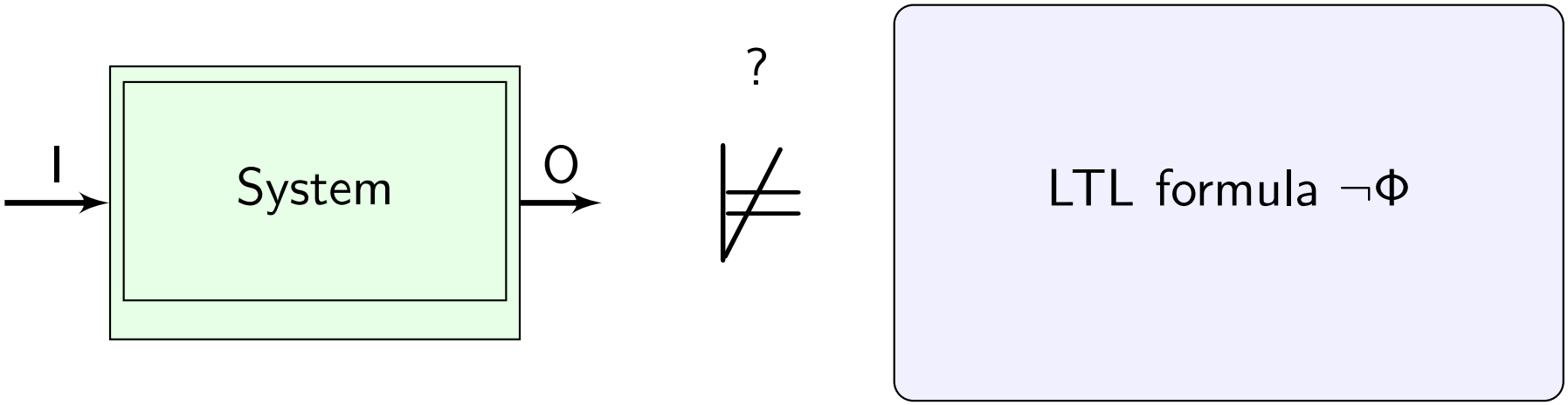
- Specification: Formula Φ in Temporal-Logic LTL
- Question: $\exists$ System $\mathcal{S}$. $\mathcal{S} \models \Phi$?

Model Checking



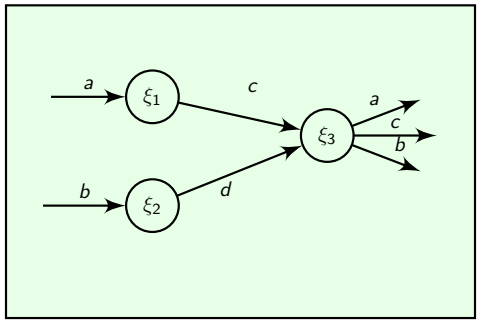
Question: $\mathcal{S} \models \Phi$?

Model Checking

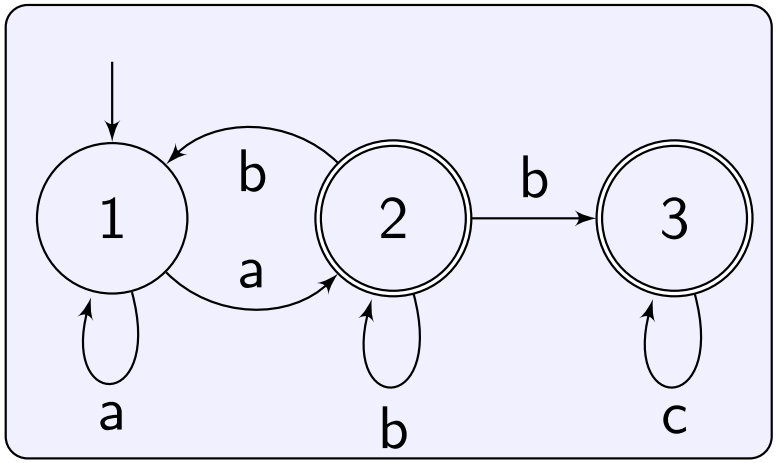


$$(\mathcal{S} \models \Phi) \leftrightarrow (\mathcal{S} \not\models \neg\Phi) \ ?$$

Automata based Model Checking



×



- Non-terminating systems !

Büchi-Automata

- Automata read infinite words
- Automata accept, whenever a $\mathcal{F}$ state is visited ∞ often !

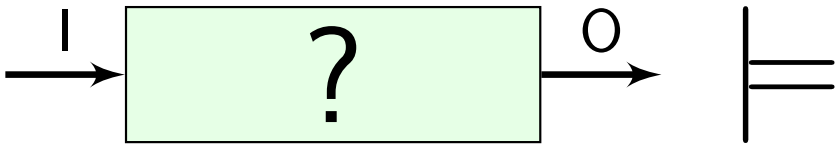
- Graphsearch for one Non-Accepting run !
- $(\mathcal{S} \models \Phi) \leftrightarrow (\mathcal{S} \not\models \neg\Phi) \leftrightarrow \mathcal{L}(\mathcal{S} \times \mathcal{A}_{\neg\Phi}) = \emptyset)$?

Symbolic Model Checking

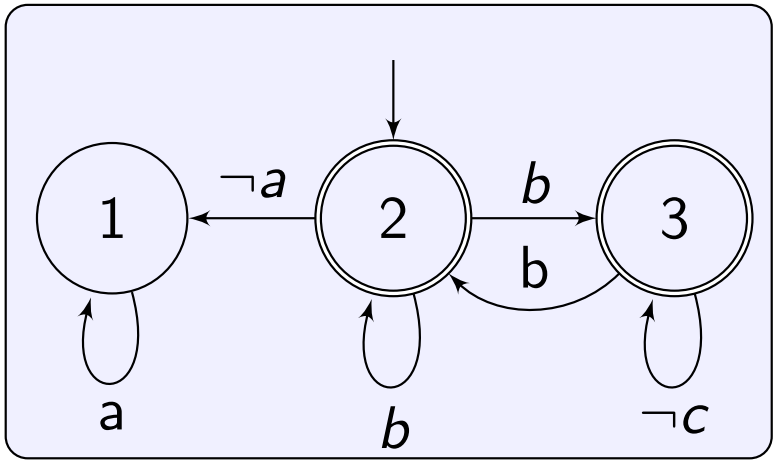
$$\mathcal{R} = \begin{array}{l} p_0 \wedge a \vee \\ p_1 \wedge \neg b \\ \ldots \end{array} \quad \bigwedge \quad \mathcal{R} = \begin{array}{l} r_0 \leftrightarrow (a \vee \neg b) \wedge \\ r_1 \leftrightarrow (c \vee d) \wedge \\ \ldots \end{array}$$

- Using propositional logic to represent
  - System and
  - Büchi Automata
- Advantages:
  - Represent large state spaces
  - Efficient methods like BDD / SAT
- Industry-sized problems managable:
  - Verification at Intel

Automata based LTL Synthesis

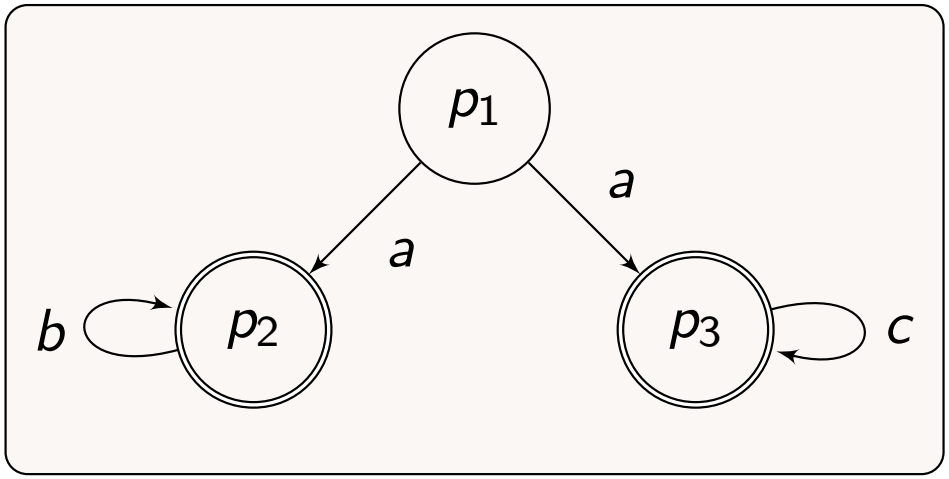


$\models$



- $(\exists \mathcal{S}.\ \mathcal{S} \models \Phi) \leftrightarrow \exists \mathcal{S}.\ \mathcal{L}(\mathcal{S}) \subseteq \mathcal{L}(\mathfrak{A}_\Phi)$
- Idea: Search for valid sub-automaton for each input on Büchi automaton!
- Infinite Game between Environment and System !

Automata based LTL Synthesis

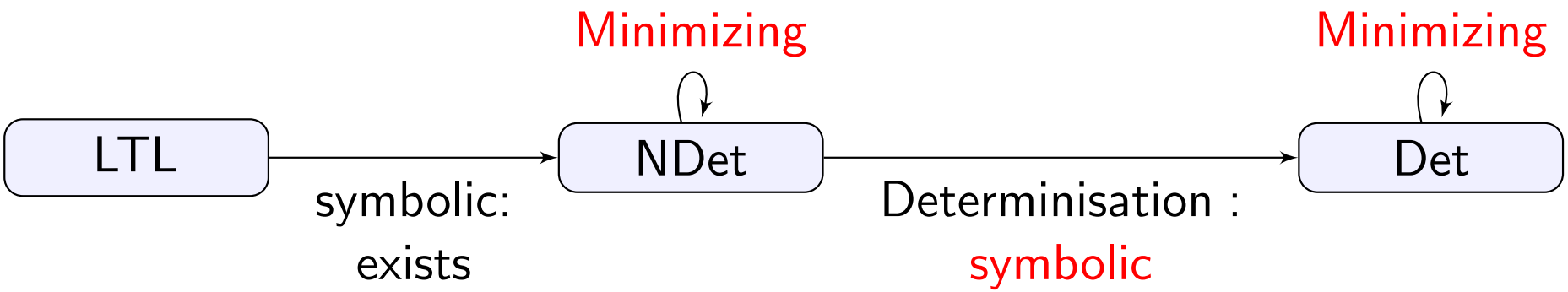


- Idea: Search for satisfying automaton for each input on specification automaton!
- **Problem: nondeterminism**
- intuitively: a priori not known whether b or c comes
- deterministic system from nondeterministic Büchi automaton ↯

Determinisation of Büchi Automata: Facts

- Rabin-Scott Subset construction not sufficient !
- Safra (1988): Determinisation of Büchi automata
- First Implementation : 2006
- State space: Trees of sets of states
- No fully symbolic implementation known
- Only small examples managable

Core of this Work

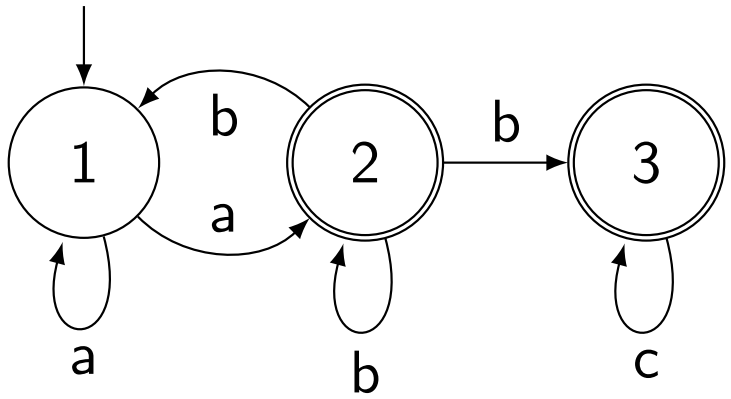


- symbolic translation LTL $\rightarrow$ NDet $\surd$
- symbolic Algorithms for infinite games $\surd$
- minimizing automata symbolically $\surd$
- symbolic determinization (shown in [MoSc08,MoSc08a])

How well does it work in practice ?

Overview

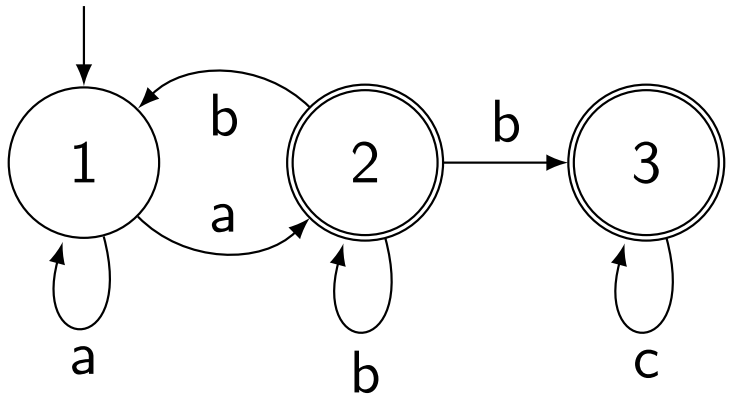
ω-Automata



ω-Automata

- ω-Automata read infinite Worte.
- Different acceptance conditions:
  - Büchi : visit $\mathcal{F}$ states infinitely often

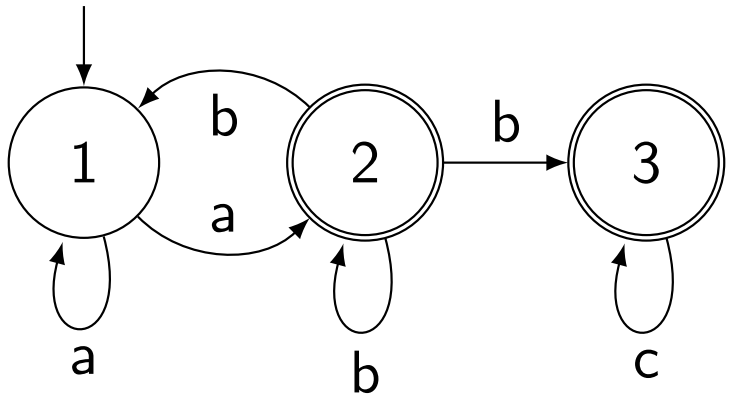
ω-Automata



ω-Automata

- ω-Automata read infinite Worte.
- Different acceptance conditions:
  - Büchi : visit $\mathcal{F}$ states infinitely often
  - Co-Büchi: visit $\neg\mathcal{F}$ states finitely often

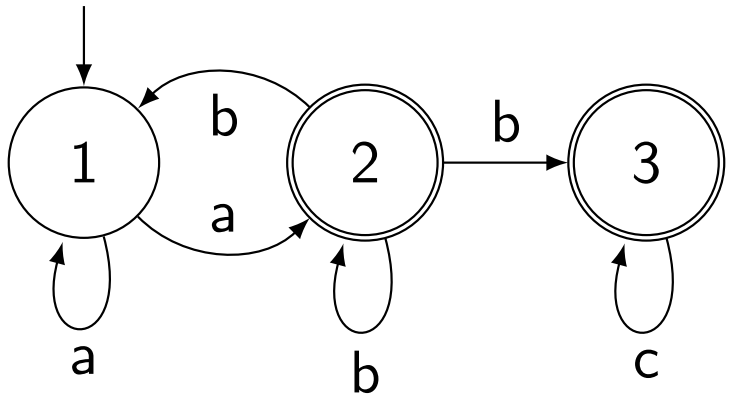
ω-Automata



ω-Automata

- ω-Automata read infinite Worte.
- Different acceptance conditions:
  - Büchi : visit $\mathcal{F}$ states infinitely often
  - Co-Büchi: visit $\neg\mathcal{F}$ states finitely often
  - Streett: boolean combination of (co)-Büchi (in Normalform)

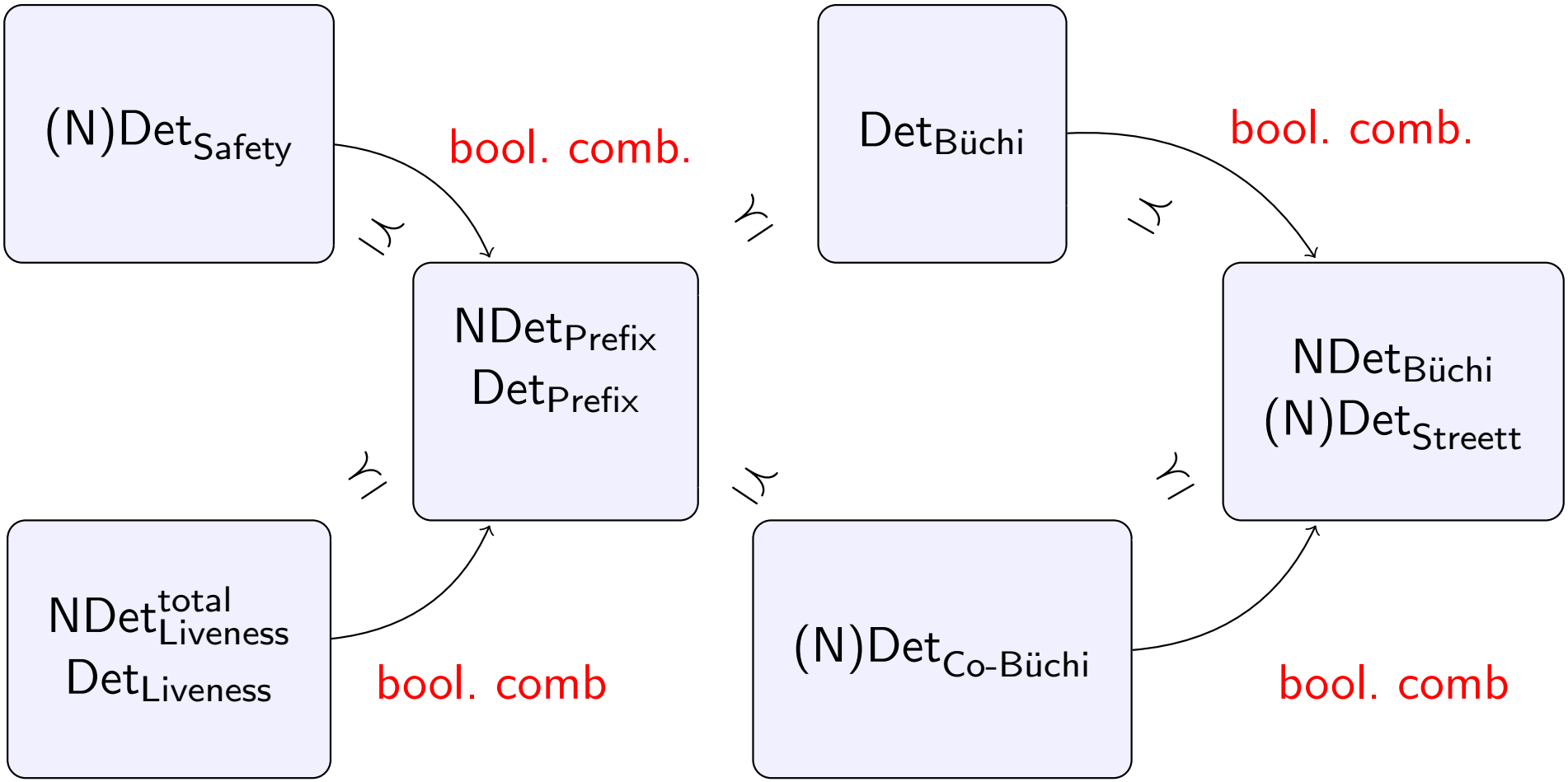
ω-Automata



ω-Automata

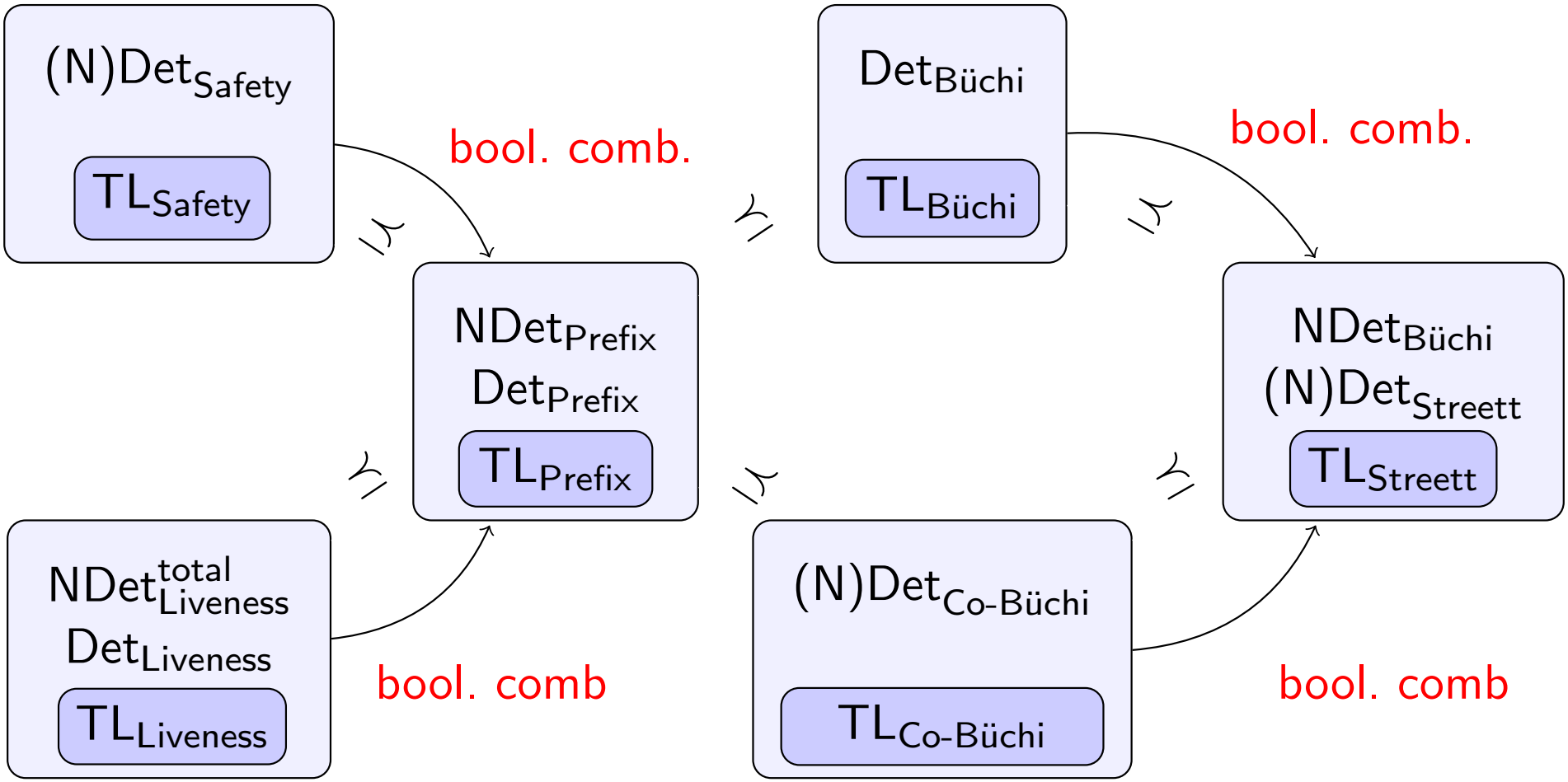
- ω-Automata read infinite Worte.
- Different acceptance conditions:
  - Büchi : visit $\mathcal{F}$ states infinitely often
  - Co-Büchi: visit $\neg\mathcal{F}$ states finitely often
  - Streett: boolean combination of (co)-Büchi (in Normalform)
  - Safety : visit only $\mathcal{F}$ states
  - Liveness : visit $\mathcal{F}$ states at least once
  - Prefix : boolean combination of Safety und Liveness (in Normalform)

The Automata Hierarchy (Wagner, 1979)



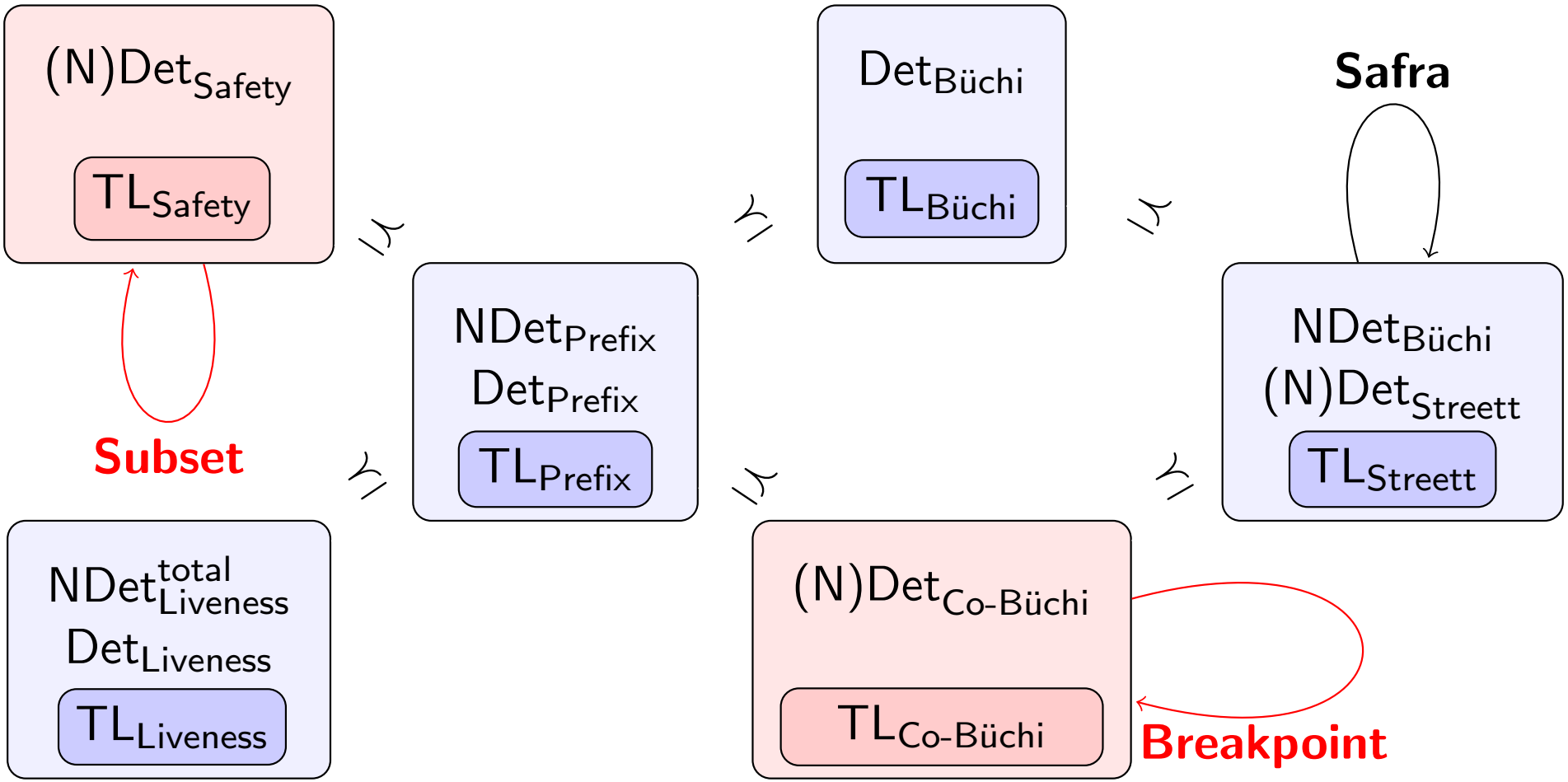
$\mathfrak{C}_1 \preceq \mathfrak{C}_2 :=$ automaton from $\mathfrak{C}_1$ can be translated to one from $\mathfrak{C}_2$

The Temporallogic Hierarchy (Manna&Pnueli, 1987)



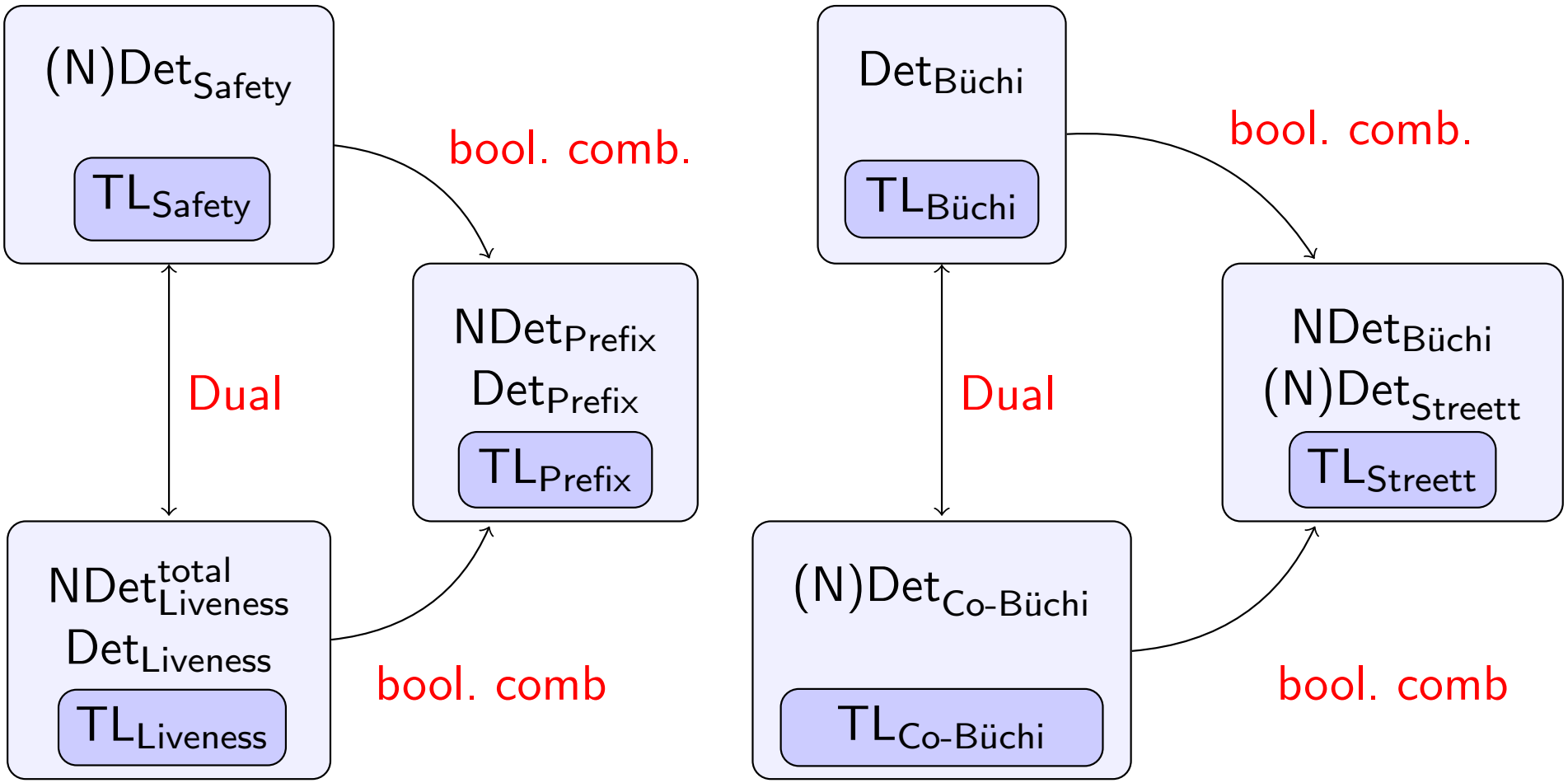
$\mathfrak{C}_1 \preceq \mathfrak{C}_2 :=$ automaton from $\mathfrak{C}_1$ can be translated to one from $\mathfrak{C}_2$

Symbolic Determinisation via Automata Hierarchy



BDD-represented Automata for $\mathrm{TL}_{\mathrm{Safety}}$ and $\mathrm{TL}_{\mathrm{Co\text{-}Büchi}}$

Symbolic Determinisation via Automata Hierarchy



BDD-represented Automata for $TL_{Liveness}$, TL_{Prefix} $TL_{Büchi}$ und $TL_{Streett}$

Determinisation via Automata Hierarchy: Conclusion

Main Idea

- Locate formula syntactically in Hierarchy
- Subset (Breakpoint) construction symbolically
- boolean combination of Formulas / Automata

Advantages

- Deterministic automata never explicitely represented
- Efficient: due to boolean combination subautomata very small (less than < 20 ndet states)
- Nearly all formula belong to $\mathrm{TL}_{\mathrm{Streett}}$

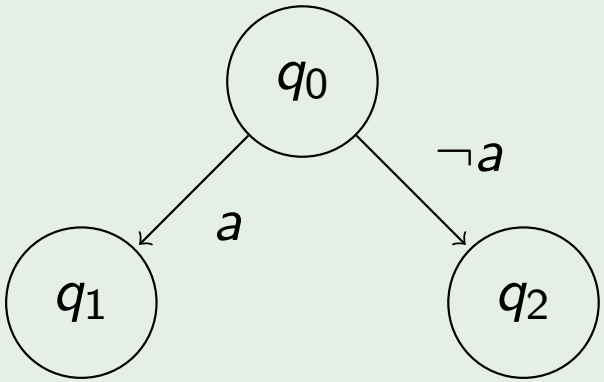
Disadvantages

- **Not every formula is in $\mathrm{TL}_{\mathrm{Streett}}$!**

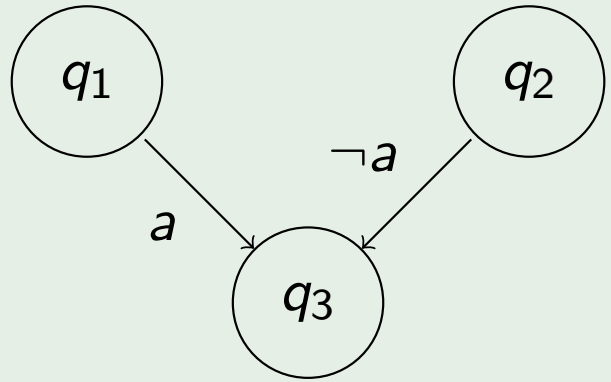
Overview

Backwards-Determinism

Determinism

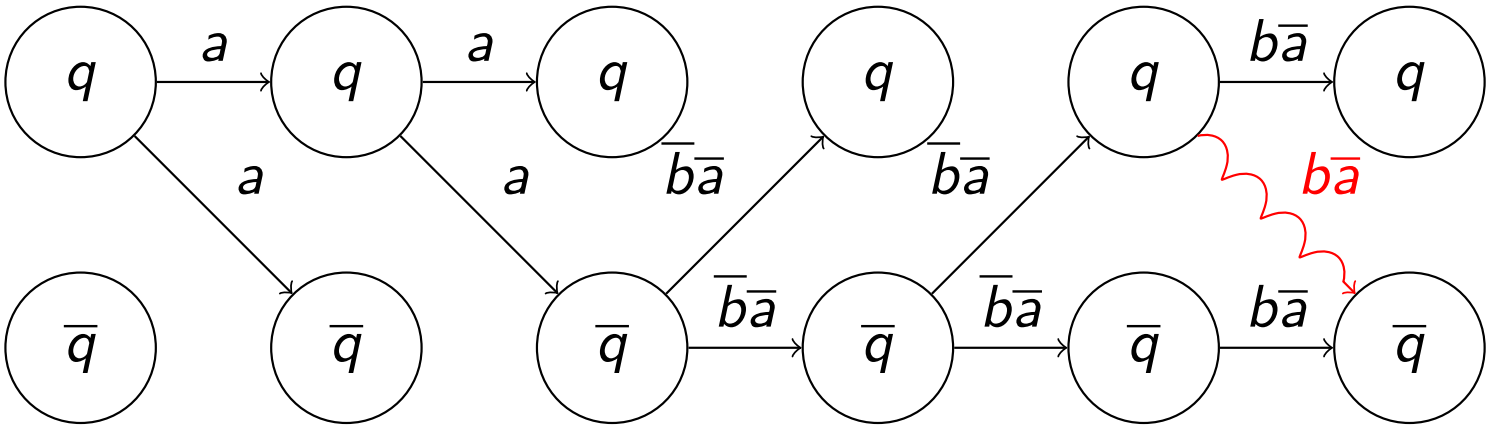


Backwards Determinism



- every Büchi automaton from LTL is backwards deterministic !

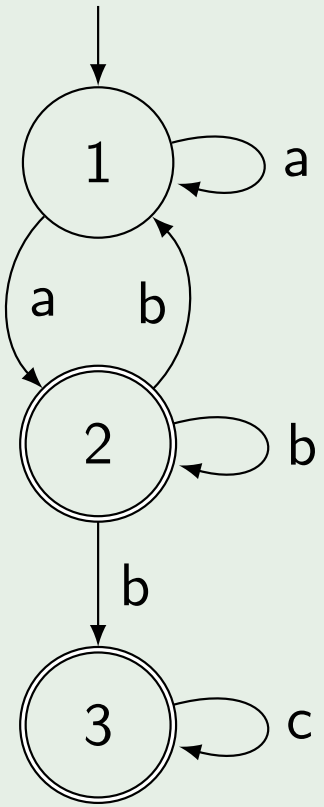
Non-Confluent Automata



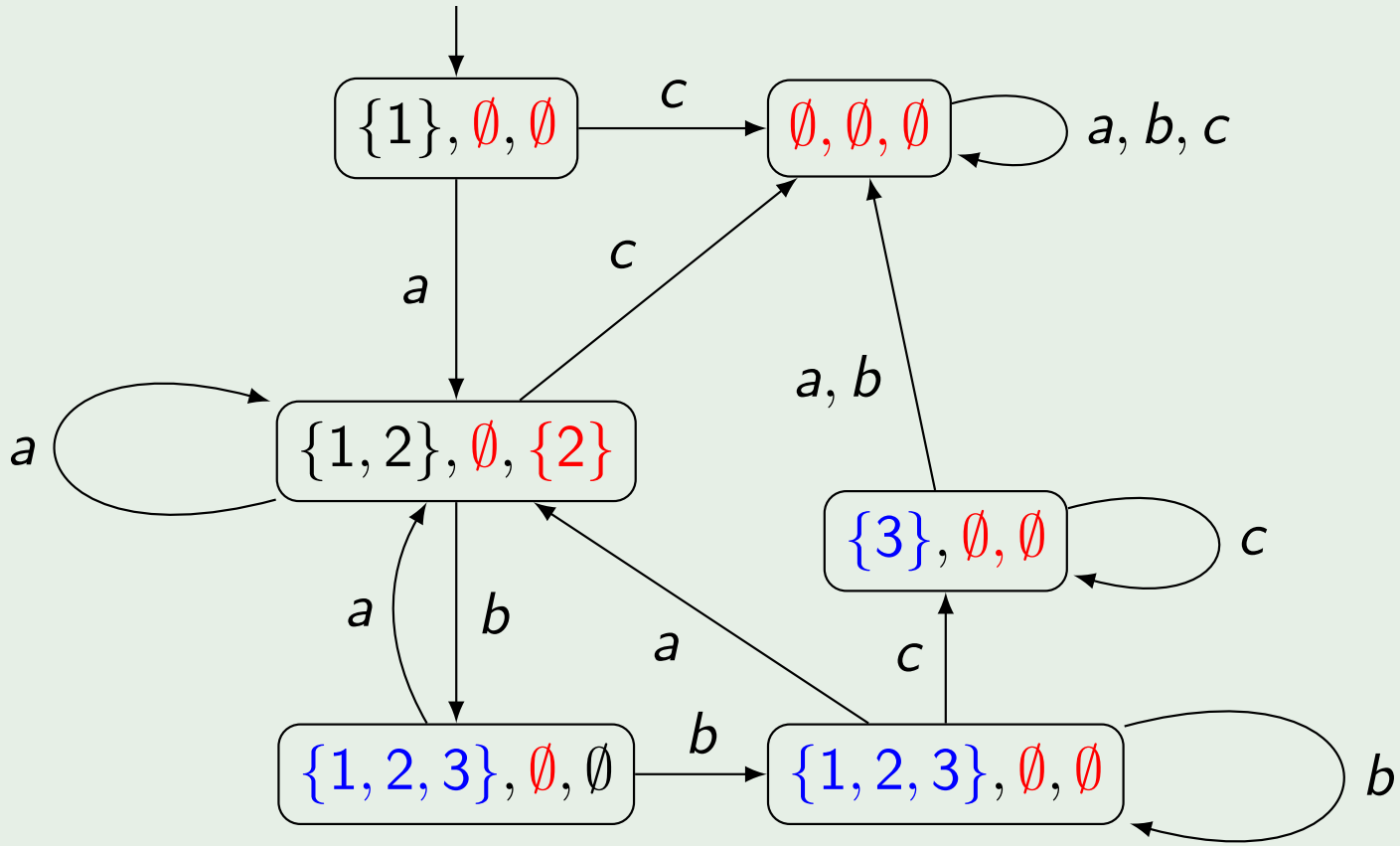
- Backwards determinism implies non-confluence
- Non-Confluence: two different runs never merge
- Therefore: Every run uniquely determined by last visited state
- No Need for Tree Structure in Deterministic Automaton !

Determinization: Example

NDet



Deterministic Automaton



- Accept, whenever some set ($\neg\infty$ often red) and (∞ blue) !

Overview

Other Tools

Opal [Morgenstern]

- Symbolic determinisation for full LTL
- Symbolic minimization for different ω-automata
- Symbolic synthesis algorithm for generalized Parity Games[Chatterjee et al, 2007]

Other Tools

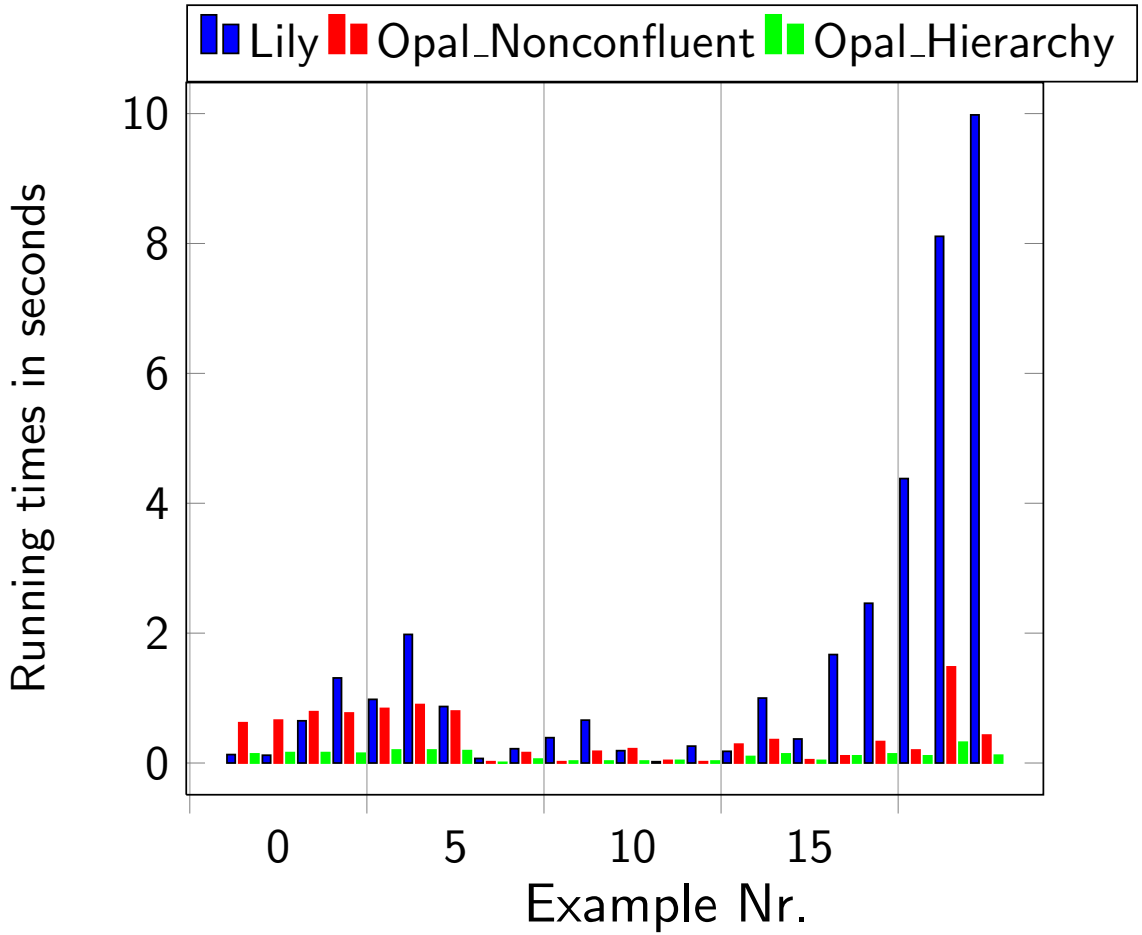
Opal [Morgenstern]

- Symbolic determinisation for full LTL
- Symbolic minimization for different ω-automata
- Symbolic synthesis algorithm for generalized Parity Games[Chatterjee et al, 2007]

Lily[Jobstmann et al.]

- Based on Safraless approach of Vardi and Kupferman
- replace determinisim by universal tree automata
- explicit implementation
- first Tool for full LTL

Lily Examples



- 23 Medium-Sized Spezifications: Arbiter, Mutex
- Temporal operators: $4 - 40$, boolean operators: $3 - 38$

Other Tools

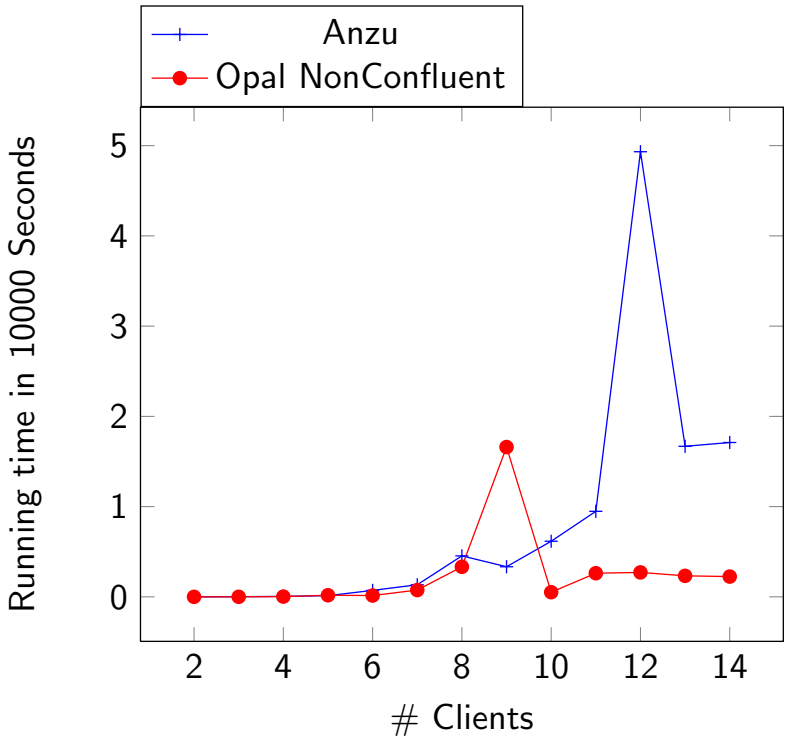
Opal [Morgenstern]

- Symbolic determinisation for full LTL
- Symbolic minimization for different ω-automata
- Symbolic synthesis algorithm for generalized Parity Games[Chatterjee et al, 2007]

Anzu[Jobstmann et al.]

- Symbolic implementation of Generalised Streett (1) Approach (Piterman et al.)
- Limited set of specifications (implication of Büchi-conditions)
- Possible: determinise Büchi automata manually
- nevertheless: much more restricted than $\mathrm{TL}_{\mathrm{Streett}}$

Industry Specification: AMBA Arbiter



- Arbiter for AMBA Bus: connects Caches, CPUs, ...
- Each Client 10 Safety / 1 Streett Condition
- Total: 40 Temporal operators / Client

Conclusion

Summary

- Presented a tool for LTL Synthesis starting with plain formula
- Running time competitive to anzu
  (where deterministic automata are generated manually)
- LTL Synthesis for full LTL is possible !
- Future ?

Thank you!

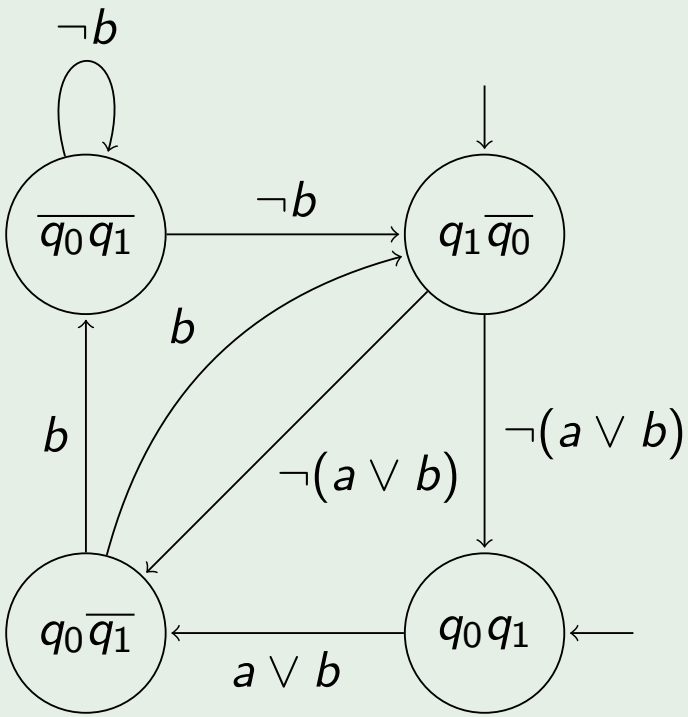
Temporallogic-Hierarchy

$P_G ::= V_\Sigma$ $\mid \neg P_F \mid P_G \wedge P_G \mid P_G \vee P_G$ $\mid \mathsf{X} P_G \mid \mathsf{G} P_G$ $\mid [P_G \mathsf{U} P_G]$ + Past	$P_F ::= V_\Sigma$ $\mid \neg P_G \mid P_F \wedge P_F \mid P_F \vee P_F$ $\mid \mathsf{X} P_F \mid \mathsf{F} P_F$ $\mid [P_F \underline{\mathsf{U}} P_F]$ + Past
$P_{\text{Prefix}} ::= P_G \mid P_F$ $\mid \neg P_{\text{Prefix}} \mid P_{\text{Prefix}} \wedge P_{\text{Prefix}} \mid P_{\text{Prefix}} \vee P_{\text{Prefix}}$ + Past	
$P_{GF} ::= P_{\text{Prefix}}$ $\mid \neg P_{FG} \mid P_{GF} \wedge P_{GF} \mid P_{GF} \vee P_{GF}$ $\mid \mathsf{X} P_{GF} \mid \mathsf{G} P_{GF}$ $\mid [P_{GF} \mathsf{U} P_{GF}] \mid [P_{GF} \underline{\mathsf{U}} P_F]$ + Past	$P_{FG} ::= P_{\text{Prefix}}$ $\mid \neg P_{GF} \mid P_{FG} \wedge P_{FG} \mid P_{FG} \vee P_{FG}$ $\mid \mathsf{X} P_{FG} \mid \mathsf{F} P_{FG}$ $\mid [P_{FG} \underline{\mathsf{U}} P_{FG}] \mid [P_G \mathsf{U} P_{FG}]$ + Past
$P_{\text{Streett}} ::= P_{GF} \mid P_{FG}$ $\mid \neg P_{\text{Streett}} \mid P_{\text{Streett}} \wedge P_{\text{Streett}} \mid P_{\text{Streett}} \vee P_{\text{Streett}}$ + Past	

Example for LTL, not $\text{TL}_{\text{Streett}}$: $\mathsf{GF}[\varphi \mathsf{U} \psi] \equiv \mathsf{FG}[\varphi \mathsf{U} \psi] \wedge \mathsf{GF}\psi$

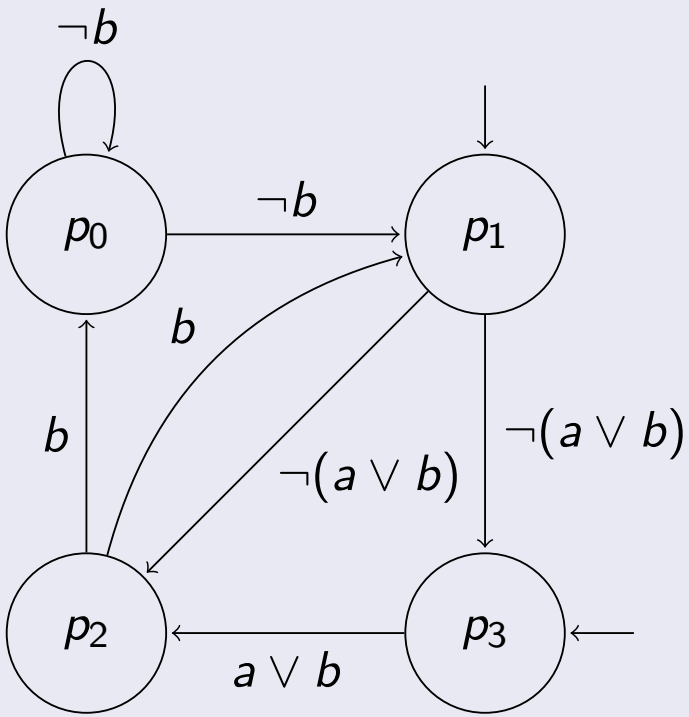
Symbolic Subset-Construktion: 1. Onehot-Kodierung

Nondeterministic automaton



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

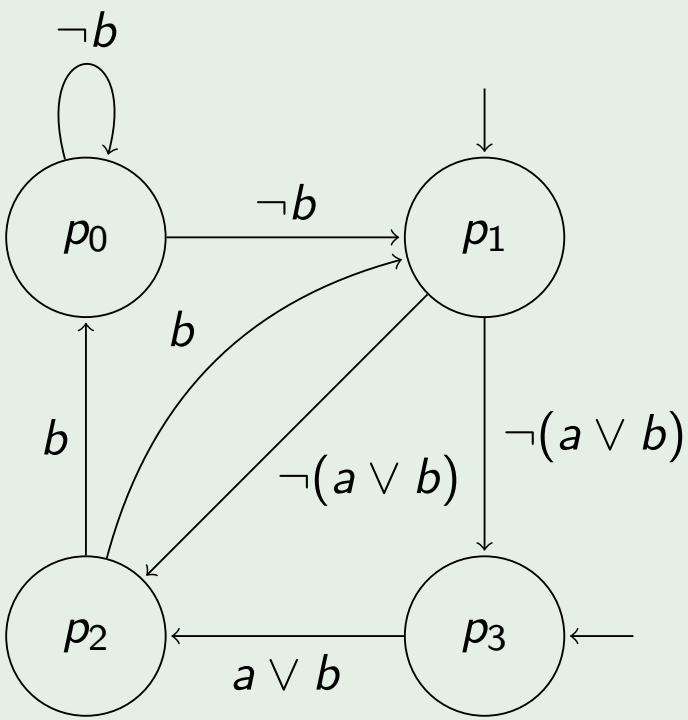
Onehot encoded automaton



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

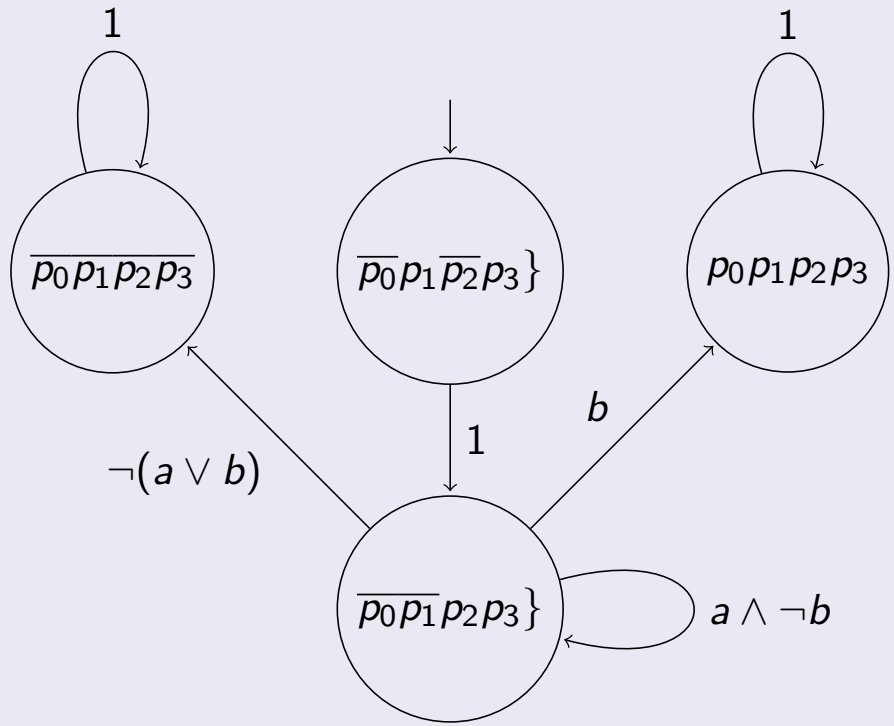
Symbolic Subset-Construktion: 2. Determinisation

Onehot-kodierter Automat



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

Deterministic automaton



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

Symbolic Subset-Construktion: BDD-Implementation

Definition (Symbolic Subset Construction)

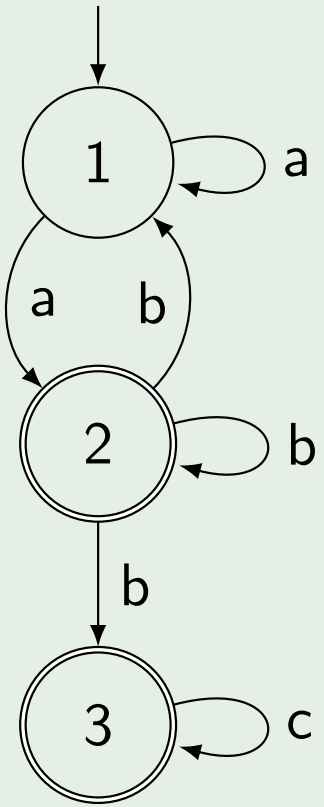
- $\mathcal{H} :\equiv \bigvee_{j=1}^{n} p_j \wedge \mathrm{mt}_{Q_{\mathrm{det}}}(\vartheta_j)$
- $\mathcal{I}_{\mathrm{det}} :\equiv \bigwedge_{i=1}^{n} p_i \leftrightarrow \exists q_1 \ldots q_m.\ \mathrm{mt}_{Q_{\mathrm{det}}}(\vartheta_i) \wedge \mathcal{I}$
- $\mathcal{F}_{\mathrm{det}} :\equiv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

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

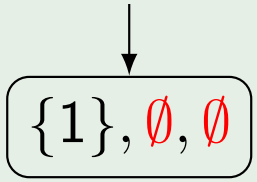
- nicht-symbolischer Schritt: Aufzählen der n nichtdeterministischen Zustände

Determinizing Non-Confluent Automata

NDet



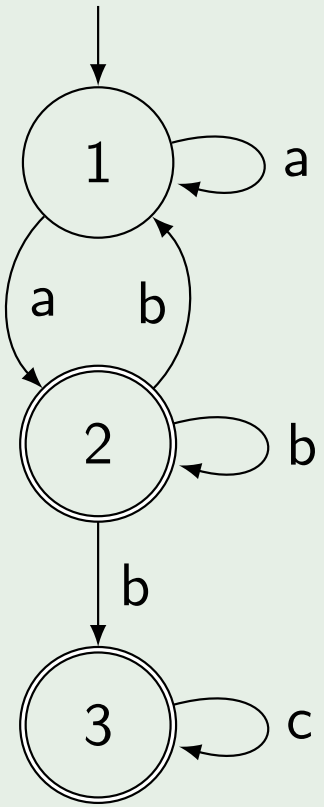
Deterministic automaton



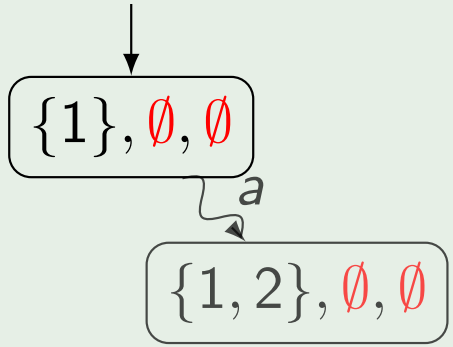
- Parallel Subset-Construktionen in sets $S_0 \ldots S_{n-1}$

Determinizing Non-Confluent Automata

NDet



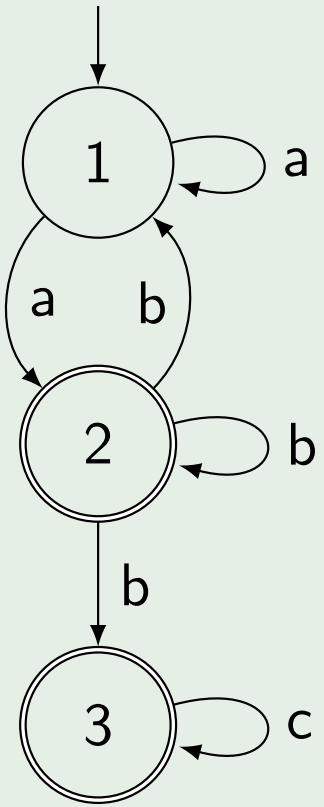
Deterministic automaton



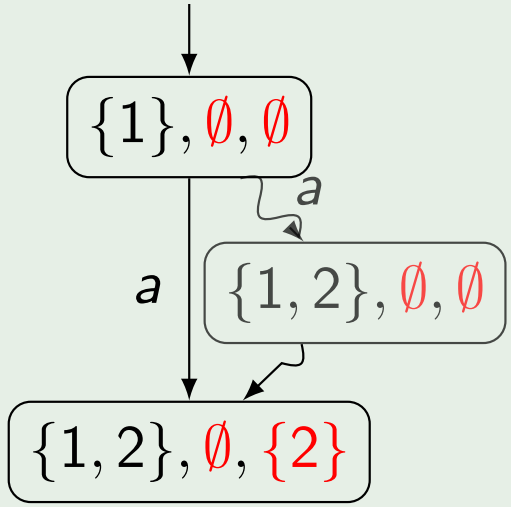
- Dead-Ends → red marking

Determinizing Non-Confluent Automata

NDet



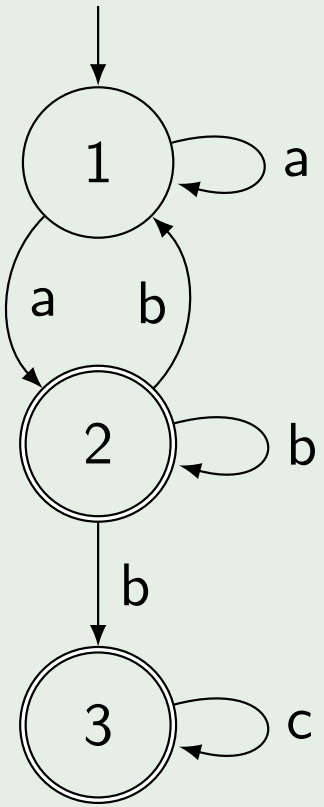
Deterministic automaton



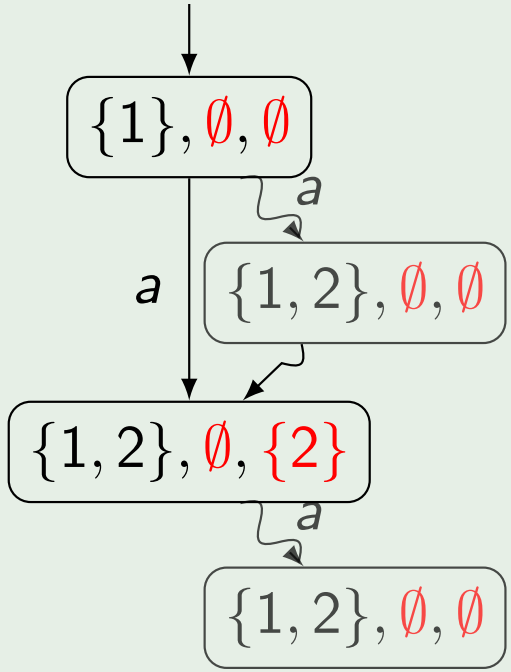
- $\mathcal{F}$ states $\rightarrow$ new Subset-Construktion rightt

Determinizing Non-Confluent Automata

NDet



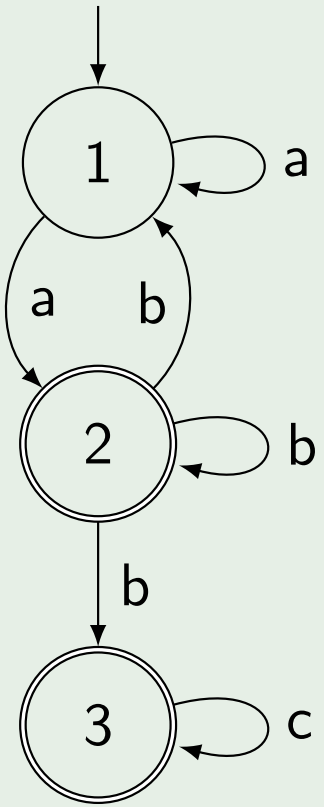
Deterministic automaton



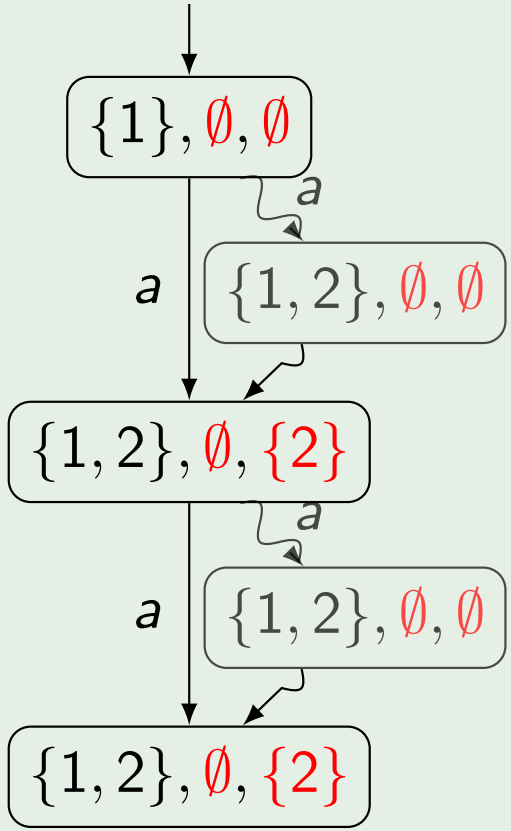
- Dead-Ends $\rightarrow$ red marking

Determinizing Non-Confluent Automata

NDet



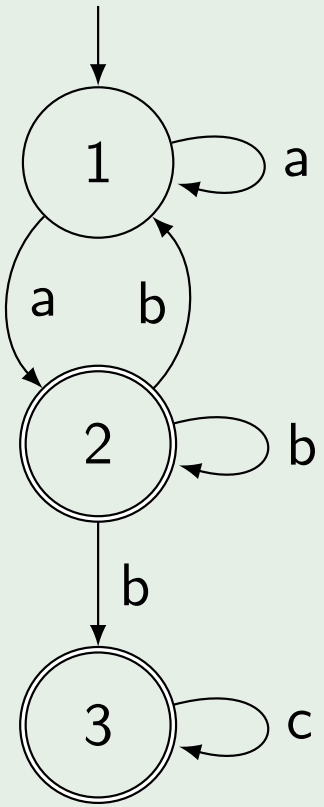
Deterministic automaton



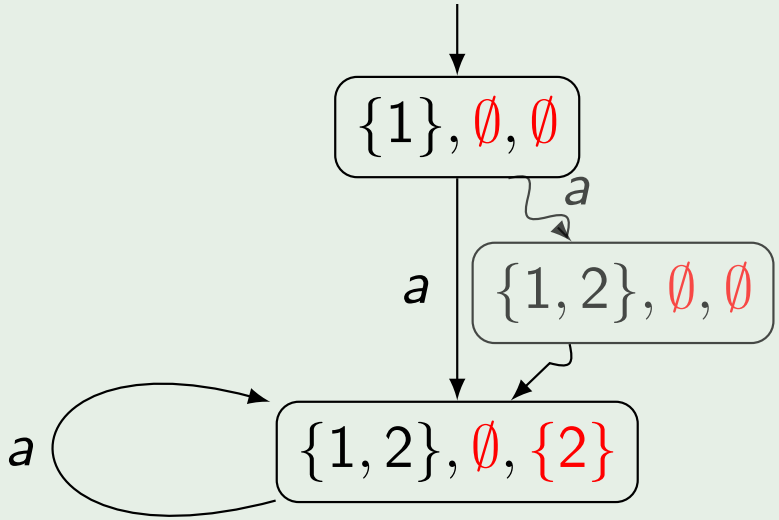
- split off $\mathcal{F}$ states $\rightarrow$ same state !

Determinizing Non-Confluent Automata

NDet



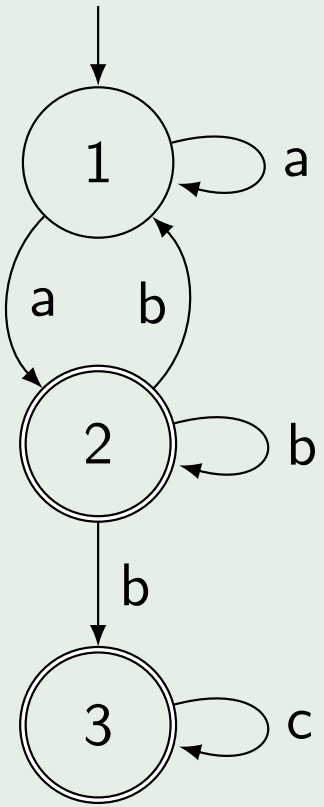
Deterministic automaton



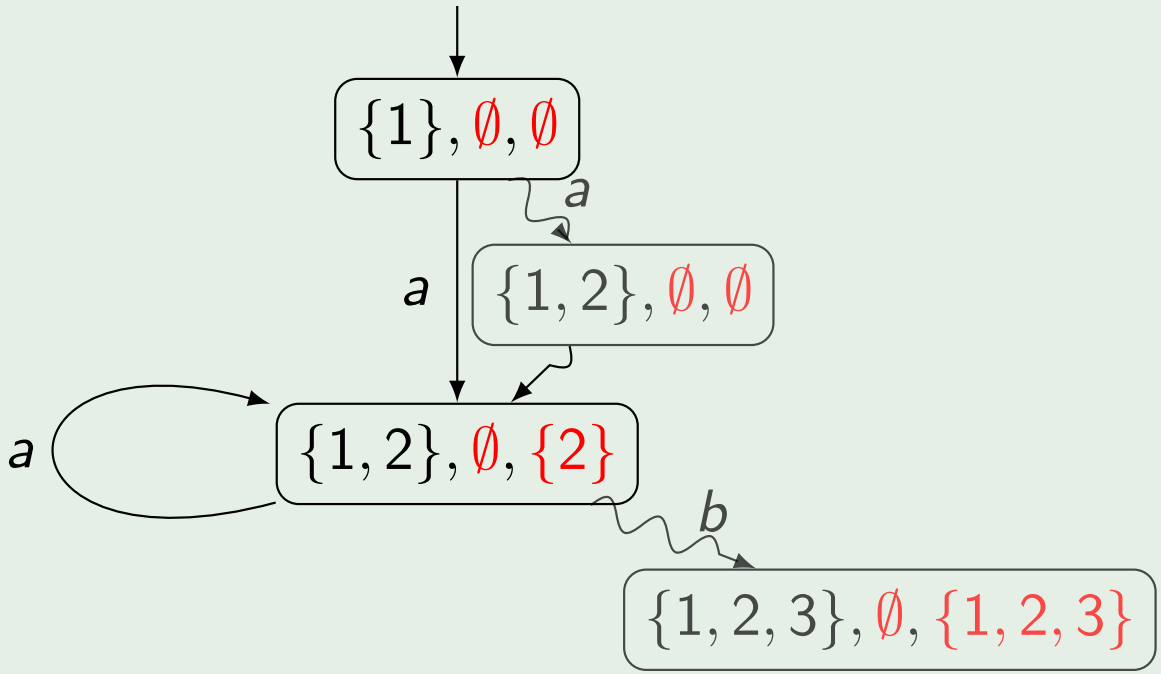
- same state!

Determinizing Non-Confluent Automata

NDet



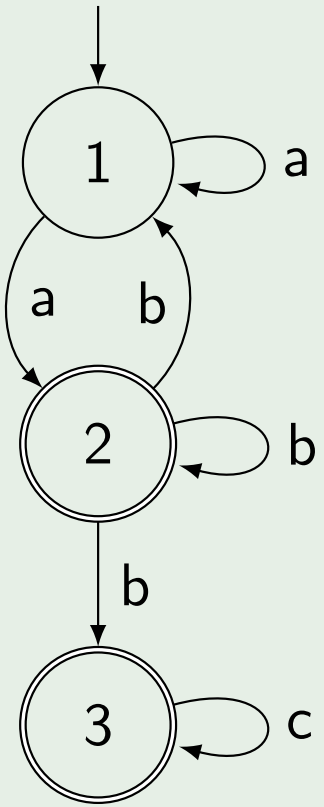
Deterministic automaton



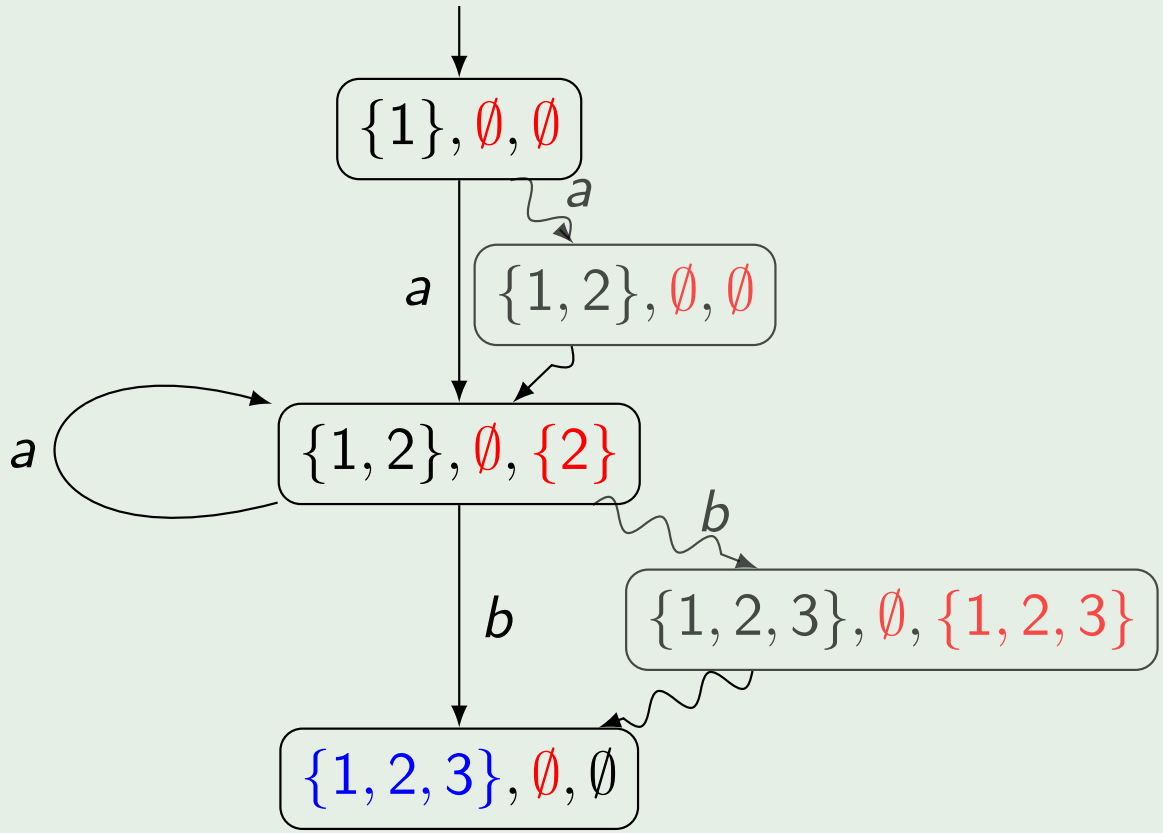
- parallel Subset-Construktion!

Determinizing Non-Confluent Automata

NDet



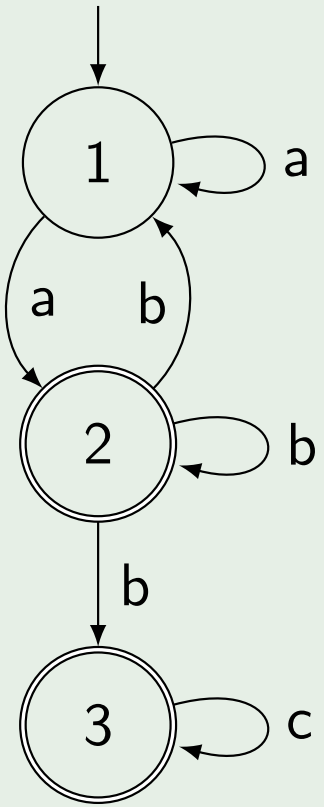
Deterministic automaton



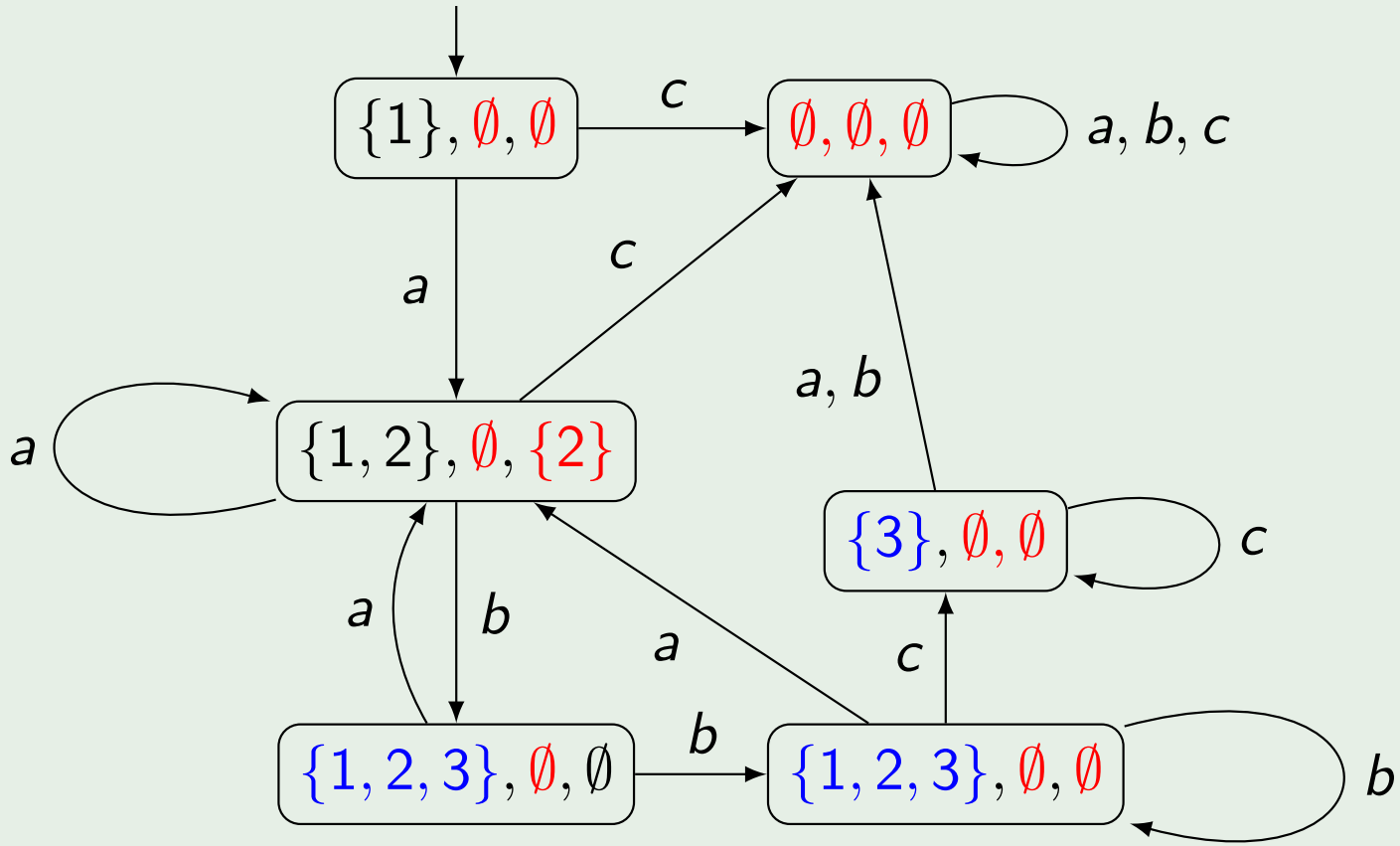
- runs unique → blue marking and removal right

Determinizing Non-Confluent Automata

NDet



Deterministic automaton



- Accept, whenever some set ($\neg\infty$ often red) and (∞ blue) !