Synchronizing Automata and the Černý Conjecture

Alexander Weinert

RWTH Aachen University

Seminar on Automata Theory

DFA

Definition

A DFA is a 5-tupel $(Q, \Sigma, \delta, q_0, F)$

with

- Q a finite set of states
- Σ an alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function
- $q_0 \in Q$ the starting state
- $F \subseteq Q$ a set of final states

DFA

Definition

A DFA is a ~~5~~3-tupel $(Q, \Sigma, \delta$~~$, q_0, F$~~$)$

with

- Q a finite set of states
- Σ an alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function
- ~~$q_0 \in Q$ the starting state~~
- ~~$F \subseteq Q$ a set of final states~~

DFA

Definition

A DFA is a ~~5~~3-tupel $(Q, \Sigma, \delta\sout{, q_0, F})$

with

- Q a finite set of states
- Σ an alphabet
- $\delta : \mathcal{P}(Q) \times \Sigma^* \to Q$ the transition function
- ~~$q_0 \in Q$ the starting state~~
- ~~$F \subseteq Q$ a set of final states~~

Synchronising DFA

Definition

A DFA is a 3-tupel $(Q, \Sigma, \delta : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q))$.

Definition

A word is *synchronising* with respect to an automaton $\mathcal{A} = (Q, \Sigma, \delta)$ if it leaves the automaton in a certain state, no matter which state the automaton started in.

Synchronising DFA

Definition

A DFA is a 3-tupel $(Q, \Sigma, \delta : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q))$.

Definition

A word is *synchronising* with respect to an automaton $\mathcal{A} = (Q, \Sigma, \delta)$ if it leaves the automaton in a certain state, no matter which state the automaton started in.

w is synchronising in $\mathcal{A}$ $\Leftrightarrow$ For all $q \in Q$ it holds that $\delta(q, w) = q_0$

Synchronising DFA

Definition

A DFA is a 3-tupel $(Q, \Sigma, \delta : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q))$.

Definition

A word is *synchronising* with respect to an automaton $\mathcal{A} = (Q, \Sigma, \delta)$ if it leaves the automaton in a certain state, no matter which state the automaton started in.

w is synchronising in $\mathcal{A}$ $\Leftrightarrow$ For all $q \in Q$ it holds that $\delta(q, w) = q_0$

Definition

A *synchronising* DFA is a DFA

Synchronising DFA

Definition

A DFA is a 3-tupel $(Q, \Sigma, \delta : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q))$.

Definition

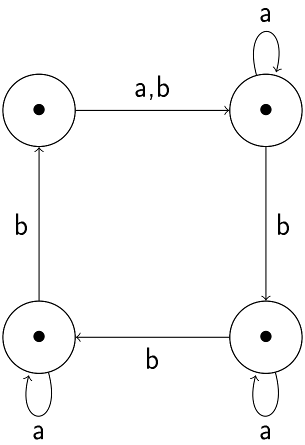
A word is *synchronising* with respect to an automaton $\mathcal{A} = (Q, \Sigma, \delta)$ if it leaves the automaton in a certain state, no matter which state the automaton started in.

w is synchronising in $\mathcal{A}$ $\Leftrightarrow$ For all $q \in Q$ it holds that $\delta(q, w) = q_0$

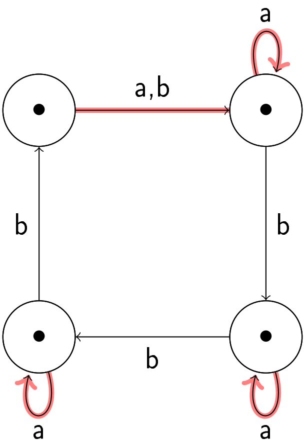
Definition

A *synchronising* DFA is a DFA that has a synchronising word.

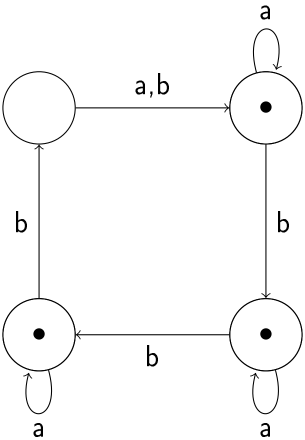
Example



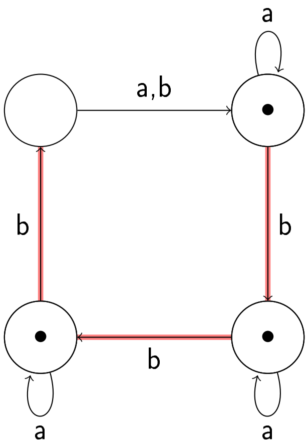
Example



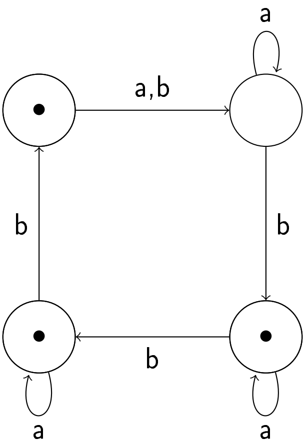
Example



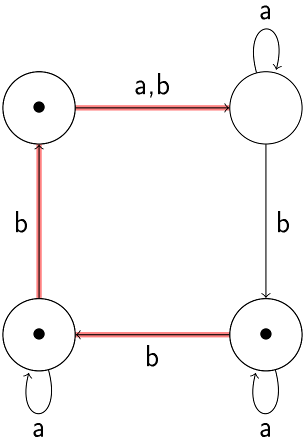
Example



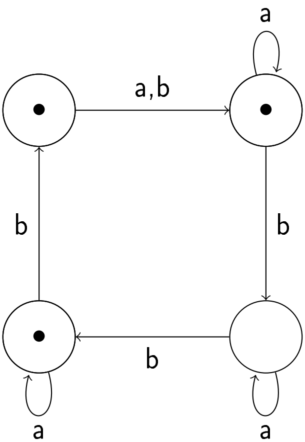
Example



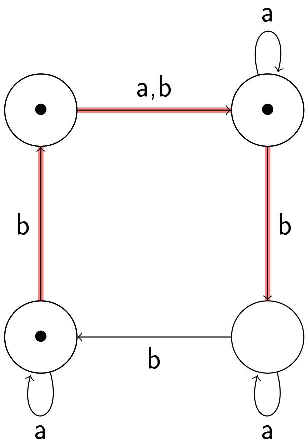
Example



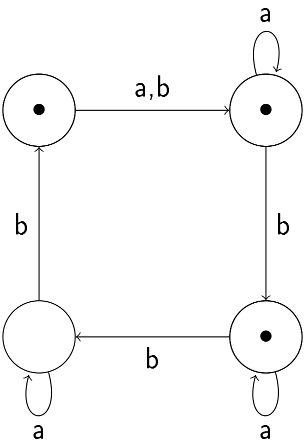
Example



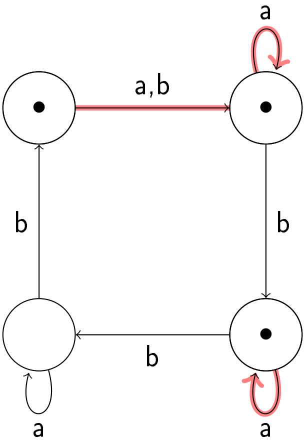
Example



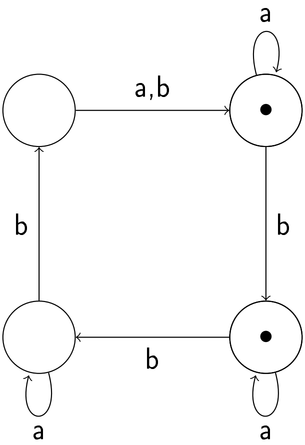
Example



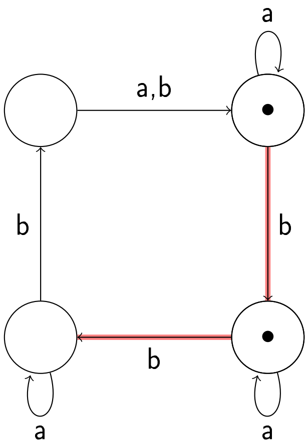
Example



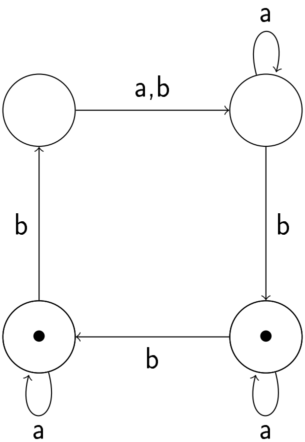
Example



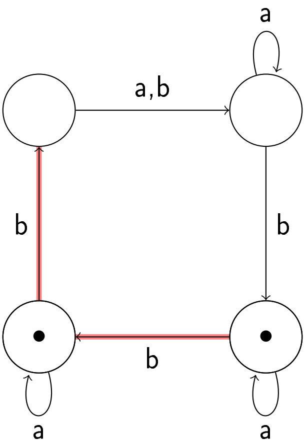
Example



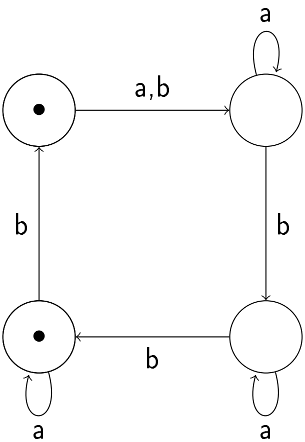
Example



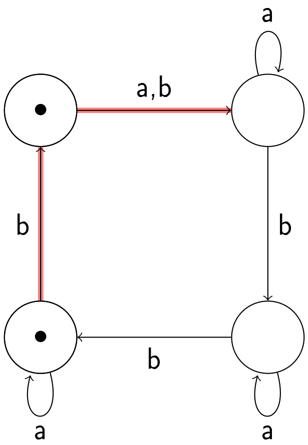
Example



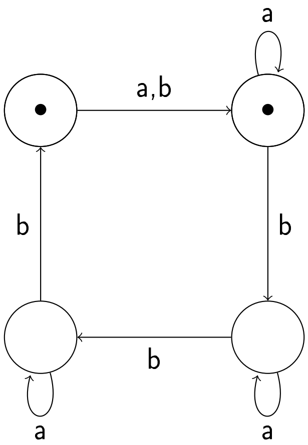
Example



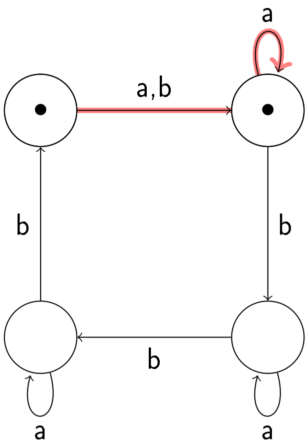
Example



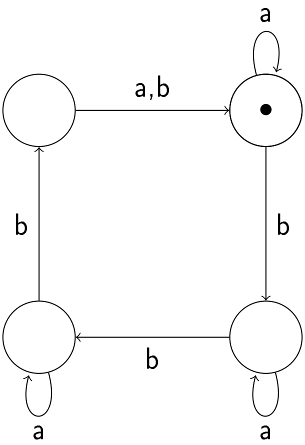
Example



Example



Example



Extension

Observation

w is synchronising $\Rightarrow u \cdot w \cdot v$ is synchronising for all $u, v \in \Sigma^*$

Orientation on Conveyor Belts

- Problem: Orienting parts on conveyor belts [Ananichev and Volkov, 2004]

Orientation on Conveyor Belts

- Problem: Orienting parts on conveyor belts [Ananichev and Volkov, 2004]

- Tools: Two types of barriers
  - High: Turns part by 90°
  - Low: Only turns part if the "nose" is down

Orientation on Conveyor Belts

- Problem: Orienting parts on conveyor belts [Ananichev and Volkov, 2004]

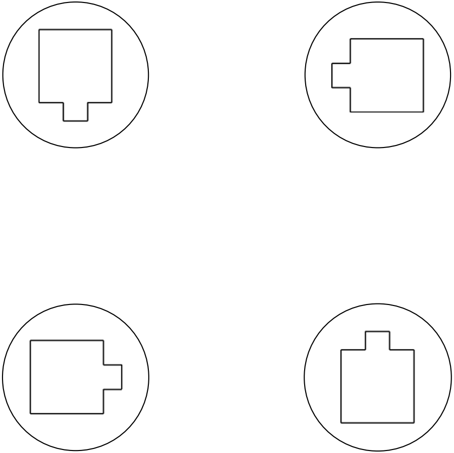
- Tools: Two types of barriers
  - High: Turns part by 90°
  - Low: Only turns part if the "nose" is down
- First solution: Sensors
  - High need for maintenance
  - High production costs

Orientation on Conveyor Belts

- Problem: Orienting parts on conveyor belts [Ananichev and Volkov, 2004]

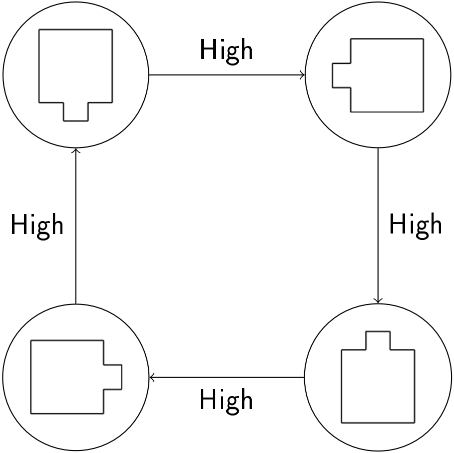
- Tools: Two types of barriers
  - High: Turns part by 90°
  - Low: Only turns part if the "nose" is down
- First solution: Sensors
  - High need for maintenance
  - High production costs
- Better solution: Synchronising automata

Orientation on Conveyor Belts

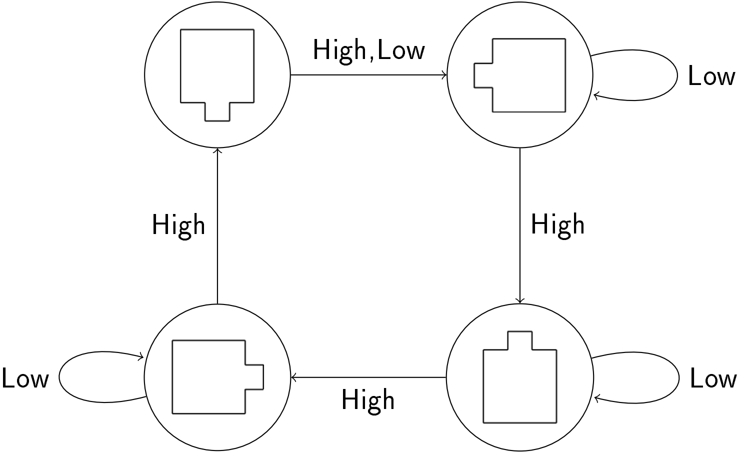


Example from [Ananichev and Volkov, 2004]

Orientation on Conveyor Belts



Example from [Ananichev and Volkov, 2004]



Example from [Ananichev and Volkov, 2004]

Biological Automata

- [Benenson et al., 2003] presents biological DFA
- "Soup of automata" containing $3 \cdot 10^{12}$ automata per μl
- Input: DNA molecules
- Possibility for highly parallel computing

Biological Automata

- [Benenson et al., 2003] presents biological DFA
- "Soup of automata" containing $3 \cdot 10^{12}$ automata per μl
- Input: DNA molecules
- Possibility for highly parallel computing
- Problem: Bring automata back to starting state
- Solution: Build synchronising automata

Checking for synchronising words

- $L_{sync} := \{\mathcal{A} \mid \mathcal{A} \text{ is a synchronising DFA}\}$
- Decision problem: Input DFA $\mathcal{A}$, Output $\mathcal{A} \in L_{sync}$

Checking for synchronising words

- $L_{sync} := \{\mathcal{A} \mid \mathcal{A}$ is a synchronising DFA$\}$
- Decision problem: Input DFA $\mathcal{A}$, Output $\mathcal{A} \in L_{sync}$
- Idea: Keep track of states we can possibly be in

$\Rightarrow$ Power automaton

Checking for synchronising words

Input: A DFA $\mathcal{A} = (Q, \Sigma, \delta)$

Checking for synchronising words

Input: A DFA $\mathcal{A} = (Q, \Sigma, \delta)$

1. Construct power automaton $\mathcal{P}(\mathcal{A}) = (\mathcal{P}(Q), \Sigma, \delta')$
   - $\delta' : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q), \delta'(P, a) := \{\delta(p, a) \mid p \in P\}$
2. Traverse DFA from $Q \in \mathcal{P}(Q)$ via BFS
3. If some state P with $|P| = 1$ is reached, return true, false otherwise

Checking for synchronising words

Input: A DFA $\mathcal{A} = (Q, \Sigma, \delta)$

1. Construct power automaton $\mathcal{P}(\mathcal{A}) = (\mathcal{P}(Q), \Sigma, \delta')$
   - $\delta' : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q), \delta'(P, a) := \{\delta(p, a) \mid p \in P\}$
2. Traverse DFA from $Q \in \mathcal{P}(Q)$ via BFS
3. If some state P with $|P| = 1$ is reached, return true, false otherwise

Output: *true* if the automaton is synchronising, *false* otherwise

Checking for synchronising words

Input: A DFA $\mathcal{A} = (Q, \Sigma, \delta)$

1. Construct power automaton $\mathcal{P}(\mathcal{A}) = (\mathcal{P}(Q), \Sigma, \delta')$
   - $\delta' : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q), \delta'(P, a) := \{\delta(p, a) \mid p \in P\}$
2. Traverse DFA from $Q \in \mathcal{P}(Q)$ via BFS
3. If some state P with $|P| = 1$ is reached, return true, false otherwise

Output: *true* if the automaton is synchronising, *false* otherwise

Runtime

Construction	$\mathcal{O}(\lvert\mathcal{P}(Q)\rvert) = \mathcal{O}(2^{\lvert Q\rvert})$
BFS	$\mathcal{O}(\lvert\mathcal{P}(Q)\rvert) = \mathcal{O}(2^{\lvert Q\rvert})$
Sum	$\mathcal{O}(2 \cdot \lvert\mathcal{P}(Q)\rvert) = \mathcal{O}(2^{\lvert Q\rvert})$

Checking for synchronising words

Input: A DFA $\mathcal{A} = (Q, \Sigma, \delta)$

1. Construct power automaton $\mathcal{P}(\mathcal{A}) = (\mathcal{P}(Q), \Sigma, \delta')$
   - $\delta' : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q), \delta'(P, a) := \{\delta(p, a) \mid p \in P\}$
2. Traverse DFA from $Q \in \mathcal{P}(Q)$ via BFS
3. If some state P with $|P| = 1$ is reached, return true, false otherwise

Output: *true* if the automaton is synchronising, *false* otherwise

Runtime

Construction	$\mathcal{O}(\lvert\mathcal{P}(Q)\rvert) = \mathcal{O}(2^{\lvert Q\rvert})$
BFS	$\mathcal{O}(\lvert\mathcal{P}(Q)\rvert) = \mathcal{O}(2^{\lvert Q\rvert})$
Sum	$\mathcal{O}(2 \cdot \lvert\mathcal{P}(Q)\rvert) = \mathcal{O}(2^{\lvert Q\rvert})$

Exponential runtime

Acceleration

Acceleration

Lemma

DFA $\mathcal{A}$ is synchronising $\Leftrightarrow$ *For each pair of states p, q, there is a word w such that $\delta(p, w) = \delta(q, w)$*

Acceleration

Lemma

DFA $\mathcal{A}$ is synchronising $\Leftrightarrow$ *For each pair of states p, q, there is a word w such that $\delta(p, w) = \delta(q, w)$*

Proof.

"$\Rightarrow$"

Acceleration

Lemma

DFA $\mathcal{A}$ is synchronising $\Leftrightarrow$ *For each pair of states p, q, there is a word w such that $\delta(p, w) = \delta(q, w)$*

Proof.

"$\Rightarrow$" Choose synchronising word of $\mathcal{A}$ as w.

Acceleration

Lemma

DFA $\mathcal{A}$ is synchronising $\Leftrightarrow$ *For each pair of states p, q, there is a word w such that $\delta(p, w) = \delta(q, w)$*

Proof.

"$\Rightarrow$" Choose synchronising word of $\mathcal{A}$ as w.

"$\Leftarrow$"

- Start in P
- Go from $P \subseteq Q$ to $P' \subseteq Q$ with $|P'| < |P|$
- Possible since there are $p, q \in P$ that can be unified
- Repeat until $|P'| = 1$

Acceleration

Lemma

DFA $\mathcal{A}$ is synchronising $\Leftrightarrow$ *For each pair of states p, q, there is a word w such that $\delta(p, w) = \delta(q, w)$*

Proof.

"$\Rightarrow$" Choose synchronising word of $\mathcal{A}$ as w.

"$\Leftarrow$"

- Start in P
- Go from $P \subseteq Q$ to $P' \subseteq Q$ with $|P'| < |P|$
- Possible since there are $p, q \in P$ that can be unified
- Repeat until $|P'| = 1$
- $\Rightarrow$ Singleton state reachable from Q in power automaton
- $\Rightarrow$ $\mathcal{A}$ is synchronising

□

Polynomial algorithm

- New idea: Check reachability of singleton state from all $P \in \mathcal{P}(Q)$ with $|P| = 2$

Input: DFA $\mathcal{A}$

Polynomial algorithm

- New idea: Check reachability of singleton state from all $P \in \mathcal{P}(Q)$ with $|P| = 2$

Input: DFA $\mathcal{A}$

1. Construct $\mathcal{P}^{[2]}(\mathcal{A}) := (Q_{pot}, \Sigma, \delta')$
   - $Q_{pot} \subseteq \mathcal{P}(Q)$, contains all states of cardinality 1 and 2
2. $W := \emptyset$

Polynomial algorithm

- New idea: Check reachability of singleton state from all $P \in \mathcal{P}(Q)$ with $|P| = 2$

Input: DFA $\mathcal{A}$

1. Construct $\mathcal{P}^{[2]}(\mathcal{A}) := (Q_{pot}, \Sigma, \delta')$
   - $Q_{pot} \subseteq \mathcal{P}(Q)$, contains all states of cardinality 1 and 2
2. $W := \emptyset$
3. For each singleton $P \in Q_{pot}$
   1. Get the states R it is reachable from
   2. Set $W := W \cup (R \cap \{S \in Q_{pot} \mid |S| = 2\})$

Polynomial algorithm

- New idea: Check reachability of singleton state from all $P \in \mathcal{P}(Q)$ with $|P| = 2$

Input: DFA $\mathcal{A}$

1. Construct $\mathcal{P}^{[2]}(\mathcal{A}) := (Q_{pot}, \Sigma, \delta')$
   - $Q_{pot} \subseteq \mathcal{P}(Q)$, contains all states of cardinality 1 and 2
2. $W := \emptyset$
3. For each singleton $P \in Q_{pot}$
   1. Get the states R it is reachable from
   2. Set $W := W \cup (R \cap \{S \in Q_{pot} \mid |S| = 2\})$
4. If $R = \{P \in Q_{pot} \mid |P| = 2\}$, return *true*, else return *false*

Polynomial algorithm

- New idea: Check reachability of singleton state from all $P \in \mathcal{P}(Q)$ with $|P| = 2$

Input: DFA $\mathcal{A}$

1. Construct $\mathcal{P}^{[2]}(\mathcal{A}) := (Q_{pot}, \Sigma, \delta')$
   - $Q_{pot} \subseteq \mathcal{P}(Q)$, contains all states of cardinality 1 and 2
2. $W := \emptyset$
3. For each singleton $P \in Q_{pot}$
   1. Get the states R it is reachable from
   2. Set $W := W \cup (R \cap \{S \in Q_{pot} \mid |S| = 2\})$
4. If $R = \{P \in Q_{pot} \mid |P| = 2\}$, return *true*, else return *false*

Output: *true* if the automaton is synchronising, *false* otherwise

Complexity

Construction of automaton		
	States:	$\mathcal{O}(\frac{\lvert Q\rvert\lvert Q-1\rvert}{2}) = \mathcal{O}(\lvert Q\rvert^2)$
	Transitions:	$\mathcal{O}(\lvert Q\rvert^4)$
	Total:	$\mathcal{O}(\lvert Q\rvert^4)$

Complexity

Construction of automaton	States:	$\mathcal{O}(\frac{\|Q\|\|Q-1\|}{2}) = \mathcal{O}(\|Q\|^2)$
	Transitions:	$\mathcal{O}(\|Q\|^4)$
	Total:	$\mathcal{O}(\|Q\|^4)$
BFS	$\mathcal{O}($	$\|Q_{pot}\|) = \mathcal{O}(\|Q\|^2)$

Complexity

Construction of automaton	States:	$\mathcal{O}(\frac{\lvert Q\rvert\lvert Q-1\rvert}{2}) = \mathcal{O}(\lvert Q\rvert^2)$
	Transitions:	$\mathcal{O}(\lvert Q\rvert^4)$
	Total:	$\mathcal{O}(\lvert Q\rvert^4)$
BFS for each singleton state	$\mathcal{O}(\lvert Q_{pot}\rvert \cdot \lvert Q_{pot}\rvert) = \mathcal{O}(\lvert Q\rvert^4)$	

Complexity

Construction of automaton	States:	$\mathcal{O}(\frac{\lvert Q\rvert\lvert Q-1\rvert}{2}) = \mathcal{O}(\lvert Q\rvert^2)$
	Transitions:	$\mathcal{O}(\lvert Q\rvert^4)$
	Total:	$\mathcal{O}(\lvert Q\rvert^4)$
BFS for each singleton state	$\mathcal{O}(\lvert Q_{pot}\rvert \cdot \lvert Q_{pot}\rvert) = \mathcal{O}(\lvert Q\rvert^4)$	
Accumulated	$\mathcal{O}(\lvert Q\rvert^4)$	

Complexity

Construction of automaton	States:	$\mathcal{O}(\frac{\|Q\|\|Q-1\|}{2}) = \mathcal{O}(\|Q\|^2)$
	Transitions:	$\mathcal{O}(\|Q\|^4)$
	Total:	$\mathcal{O}(\|Q\|^4)$
BFS for each singleton state	$\mathcal{O}(\|Q_{pot}\| \cdot \|Q_{pot}\|) = \mathcal{O}(\|Q\|^4)$	
Accumulated	$\mathcal{O}(\|Q\|^4)$	

No information about length of shortest synchronising word anymore

Length of synchronising words

$$L_{ShortResetWord} := \{(\mathcal{A}, l) \mid \text{DFA } \mathcal{A} \text{ has synchronising word of length } l\}$$

$$L_{ShortestResetWord} := \{(\mathcal{A}, l) \mid \text{ Minimal synchronising word of DFA } \mathcal{A} \text{ has length } l\}$$

NP-completeness

Lemma

$L_{ShortResetWord}$ *is NP-complete*

NP-completeness

Lemma

$L_{ShortResetWord} =: L_{SRW}$ *is NP-complete*

NP-completeness

Lemma

$L_{ShortResetWord} =: L_{SRW}$ is NP-complete

Proof. (I)

$L_{SRW} \in$ NP: Guess w of length l nondeterministically, check in $\mathcal{O}(|Q| \cdot |w|)$

NP-completeness

Lemma

$L_{ShortResetWord} =: L_{SRW}$ is NP-complete

Proof. (I)

$L_{SRW} \in$ NP: Guess w of length l nondeterministically, check in $\mathcal{O}(|Q| \cdot |w|)$
L_{SRW} NP-hard: Reduce from 3-SAT.

NP-completeness

Lemma

$L_{ShortResetWord} =: L_{SRW}$ *is NP-complete*

Proof. (I)

$L_{SRW} \in$ NP: Guess w of length l nondeterministically, check in $\mathcal{O}(|Q| \cdot |w|)$
L_{SRW} NP-hard: Reduce from 3-SAT.

Idea

3-SAT	L_{SRW}
Given a formula,	Given an automaton,
does there exist	*does there exist*
an assignment that makes	a word that maps
all clauses	*all states*
true?	to the same state?

NP-completeness

Lemma

$L_{ShortResetWord} =: L_{SRW}$ *is NP-complete*

Proof. (I)

$L_{SRW} \in$ NP: Guess w of length l nondeterministically, check in $\mathcal{O}(|Q| \cdot |w|)$
L_{SRW} NP-hard: Reduce from 3-SAT.

Idea

3-SAT		L_{SRW}
Given a formula,		Given an automaton,
does there exist	$\Leftrightarrow$	*does there exist*
an assignment that makes		a word that maps
all clauses	$\Leftrightarrow$	*all states*
true?		to the same state?

Reduction from 3-SAT

Input: Formula $\Phi = \bigwedge_{i=1}^{n} \phi_i$, $\phi_i = \bigvee_{j=1}^{3} x_{ij}$

Reduction from 3-SAT

Input: Formula $\Phi = \bigwedge_{i=1}^{n} \phi_i$, $\phi_i = \bigvee_{j=1}^{3} x_{ij}$

1. $Var(\Phi)$:= Variables in Φ, $Lit(\phi)$:= Literals in ϕ
   - $\phi := (x_1, \neg x_1, x_2) \Rightarrow Var(\phi) = \{x_1, x_2\}, Lit(\phi) = \{x_1, \neg x_1, x_2\}$

Reduction from 3-SAT

Input: Formula $\Phi = \bigwedge_{i=1}^{n} \phi_i$, $\phi_i = \bigvee_{j=1}^{3} x_{ij}$

1. $Var(\Phi) :=$ Variables in Φ, $Lit(\phi) :=$ Literals in ϕ
   - $\phi := (x_1, \neg x_1, x_2) \Rightarrow Var(\phi) = \{x_1, x_2\}, Lit(\phi) = \{x_1, \neg x_1, x_2\}$
2. $Q := \{\phi_i \mid 1 \leq i \leq n\} \cup Var(\Phi) \cup \{\mathrm{SAT}\}$
3. $\Sigma := \{x, \neg x \mid x \in Var(\Phi)\}$

Reduction from 3-SAT

Input: Formula $\Phi = \bigwedge_{i=1}^{n} \phi_i$, $\phi_i = \bigvee_{j=1}^{3} x_{ij}$

1. $Var(\Phi)$:= Variables in Φ, $Lit(\phi)$:= Literals in ϕ
   - $\phi := (x_1, \neg x_1, x_2) \Rightarrow Var(\phi) = \{x_1, x_2\}, Lit(\phi) = \{x_1, \neg x_1, x_2\}$
2. $Q := \{\phi_i \mid 1 \leq i \leq n\} \cup Var(\Phi) \cup \{\text{SAT}\}$
3. $\Sigma := \{x, \neg x \mid x \in Var(\Phi)\}$
4. $\delta(q, a) := \begin{cases} \text{SAT}, & q = \text{SAT} \\ & \\ & \end{cases}$

Reduction from 3-SAT

Input: Formula $\Phi = \bigwedge_{i=1}^{n} \phi_i$, $\phi_i = \bigvee_{j=1}^{3} x_{ij}$

1. $Var(\Phi)$:= Variables in Φ, $Lit(\phi)$:= Literals in ϕ
   - $\phi := (x_1, \neg x_1, x_2) \Rightarrow Var(\phi) = \{x_1, x_2\}, Lit(\phi) = \{x_1, \neg x_1, x_2\}$
2. $Q := \{\phi_i \mid 1 \leq i \leq n\} \cup Var(\Phi) \cup \{\text{SAT}\}$
3. $\Sigma := \{x, \neg x \mid x \in Var(\Phi)\}$
4. $\delta(q, a) := \begin{cases} \text{SAT}, & q \text{ represents clause } \bigvee_{i=1}^{3} x_i \text{ and } a = x_i \\ q, & q \text{ represents clause } \bigvee_{i=1}^{3} x_i \text{ and } a \neq x_i \\ & \\ & \end{cases}$

Reduction from 3-SAT

Input: Formula $\Phi = \bigwedge_{i=1}^{n} \phi_i$, $\phi_i = \bigvee_{j=1}^{3} x_{ij}$

1. $Var(\Phi) :=$ Variables in Φ, $Lit(\phi) :=$ Literals in ϕ
   - $\phi := (x_1, \neg x_1, x_2) \Rightarrow Var(\phi) = \{x_1, x_2\}, Lit(\phi) = \{x_1, \neg x_1, x_2\}$
2. $Q := \{\phi_i \mid 1 \leq i \leq n\} \cup Var(\Phi) \cup \{\text{SAT}\}$
3. $\Sigma := \{x, \neg x \mid x \in Var(\Phi)\}$
4. $$\delta(q, a) := \begin{cases} \\ \\ \\ \text{SAT}, & q \in Var(\Phi), a = q \text{ or } a = \neg q \\ q, & q \in Var(\Phi), a \neq q \text{ and } a \neq \neg q \end{cases}$$

Reduction from 3-SAT

Input: Formula $\Phi = \bigwedge_{i=1}^{n} \phi_i$, $\phi_i = \bigvee_{j=1}^{3} x_{ij}$

1. $Var(\Phi) :=$ Variables in Φ, $Lit(\phi) :=$ Literals in ϕ
   - $\phi := (x_1, \neg x_1, x_2) \Rightarrow Var(\phi) = \{x_1, x_2\}, Lit(\phi) = \{x_1, \neg x_1, x_2\}$
2. $Q := \{\phi_i \mid 1 \leq i \leq n\} \cup Var(\Phi) \cup \{\text{SAT}\}$
3. $\Sigma := \{x, \neg x \mid x \in Var(\Phi)\}$
4. $$\delta(q, a) := \begin{cases} \text{SAT}, & q = \text{SAT} \\ \text{SAT}, & q \text{ represents clause } \bigvee_{i=1}^{3} x_i \text{ and } a = x_i \\ q, & q \text{ represents clause } \bigvee_{i=1}^{3} x_i \text{ and } a \neq x_i \\ \text{SAT}, & q \in Var(\Phi), a = q \text{ or } a = \neg q \\ q, & q \in Var(\Phi), a \neq q \text{ and } a \neq \neg q \end{cases}$$

Output: Automaton $\mathcal{A} = (Q, \Sigma, \delta)$, $(\mathcal{A}, \qquad) \in L_{SRW} \Leftrightarrow \Phi \in 3-SAT$

Reduction from 3-SAT

Input: Formula $\Phi = \bigwedge_{i=1}^{n} \phi_i$, $\phi_i = \bigvee_{j=1}^{3} x_{ij}$

1. $Var(\Phi) :=$ Variables in Φ, $Lit(\phi) :=$ Literals in ϕ
   - $\phi := (x_1, \neg x_1, x_2) \Rightarrow Var(\phi) = \{x_1, x_2\}, Lit(\phi) = \{x_1, \neg x_1, x_2\}$
2. $Q := \{\phi_i \mid 1 \leq i \leq n\} \cup Var(\Phi) \cup \{\text{SAT}\}$
3. $\Sigma := \{x, \neg x \mid x \in Var(\Phi)\}$
4. $$\delta(q, a) := \begin{cases} \text{SAT}, & q = \text{SAT} \\ \text{SAT}, & q \text{ represents clause } \bigvee_{i=1}^{3} x_i \text{ and } a = x_i \\ q, & q \text{ represents clause } \bigvee_{i=1}^{3} x_i \text{ and } a \neq x_i \\ \text{SAT}, & q \in Var(\Phi), a = q \text{ or } a = \neg q \\ q, & q \in Var(\Phi), a \neq q \text{ and } a \neq \neg q \end{cases}$$

Output: Automaton $\mathcal{A} = (Q, \Sigma, \delta)$, $(\mathcal{A}, |Var(\Phi)|) \in L_{SRW} \Leftrightarrow \Phi \in 3-SAT$

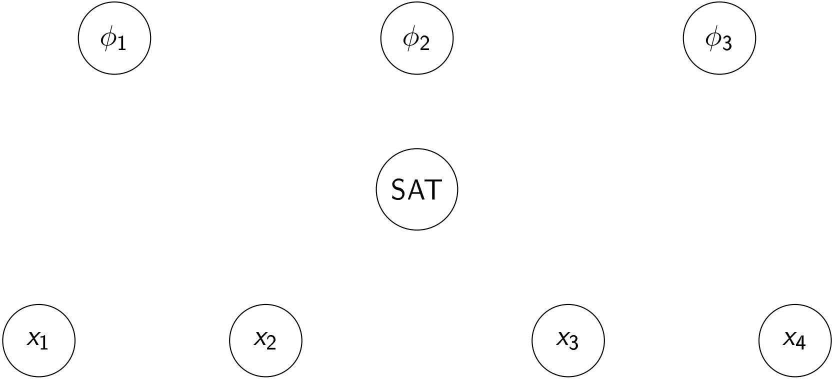
Example

$$\Phi = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_1 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee \neg x_2)$$
$$\phi_1 = (x_1 \vee \neg x_3 \vee x_4), \phi_2 = (x_2 \vee \neg x_1 \vee x_4), \phi_3 = (\neg x_3 \vee x_1 \vee \neg x_2)$$

Example

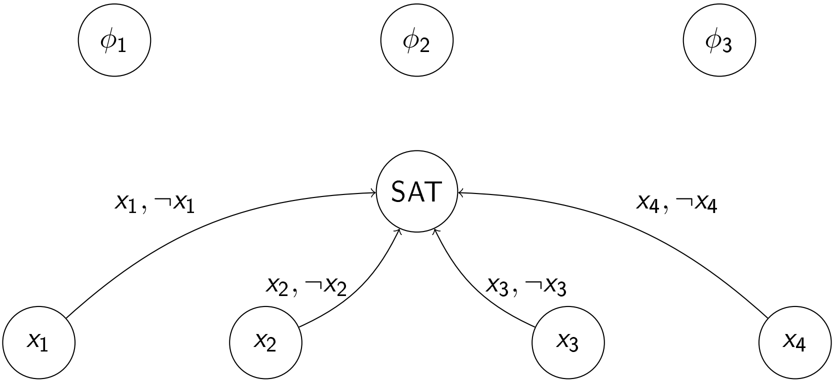
$$\Phi = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_1 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee \neg x_2)$$

$$\phi_1 = (x_1 \vee \neg x_3 \vee x_4), \phi_2 = (x_2 \vee \neg x_1 \vee x_4), \phi_3 = (\neg x_3 \vee x_1 \vee \neg x_2)$$



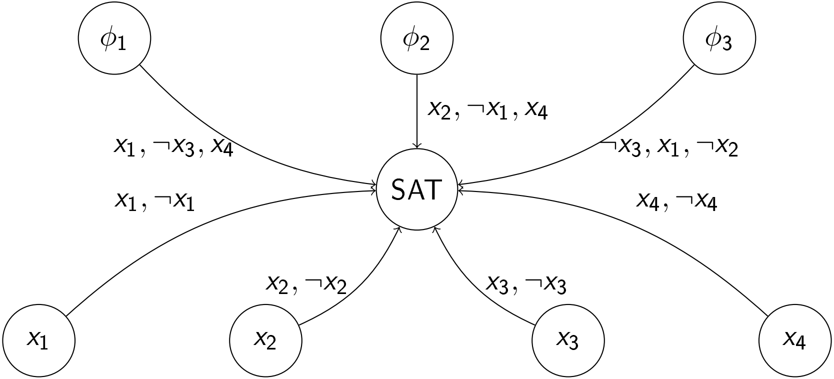
Example

$$\Phi = (x_1 \lor \neg x_3 \lor x_4) \land (x_2 \lor \neg x_1 \lor x_4) \land (\neg x_3 \lor x_1 \lor \neg x_2)$$
$$\phi_1 = (x_1 \lor \neg x_3 \lor x_4), \phi_2 = (x_2 \lor \neg x_1 \lor x_4), \phi_3 = (\neg x_3 \lor x_1 \lor \neg x_2)$$



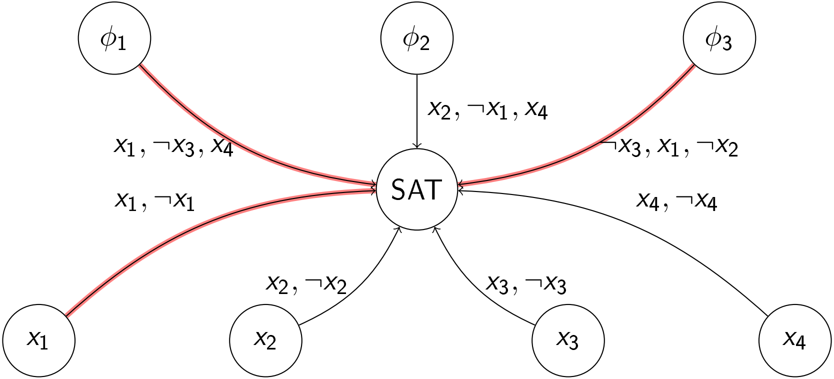
Example

$$\Phi = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_1 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee \neg x_2)$$
$$\phi_1 = (x_1 \vee \neg x_3 \vee x_4), \phi_2 = (x_2 \vee \neg x_1 \vee x_4), \phi_3 = (\neg x_3 \vee x_1 \vee \neg x_2)$$



Example

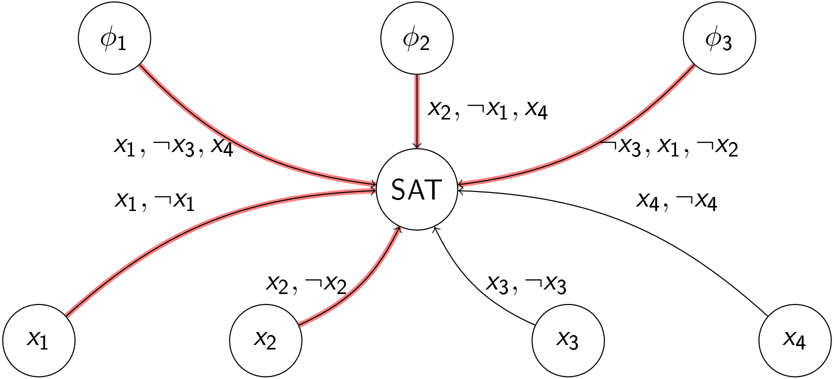
$$\Phi = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_1 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee \neg x_2)$$
$$\phi_1 = (x_1 \vee \neg x_3 \vee x_4), \phi_2 = (x_2 \vee \neg x_1 \vee x_4), \phi_3 = (\neg x_3 \vee x_1 \vee \neg x_2)$$



$$\alpha(x_1) = 1$$

Example

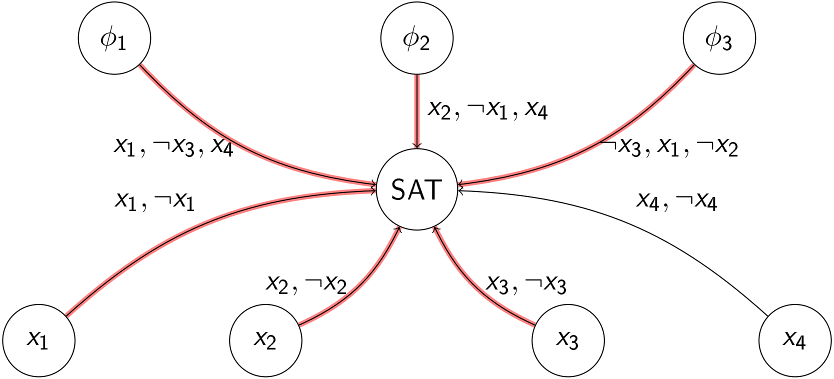
$$\Phi = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_1 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee \neg x_2)$$
$$\phi_1 = (x_1 \vee \neg x_3 \vee x_4), \phi_2 = (x_2 \vee \neg x_1 \vee x_4), \phi_3 = (\neg x_3 \vee x_1 \vee \neg x_2)$$



$$\alpha(x_1) = 1 \ \alpha(x_2) = 1$$

Example

$$\Phi = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_1 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee \neg x_2)$$
$$\phi_1 = (x_1 \vee \neg x_3 \vee x_4), \phi_2 = (x_2 \vee \neg x_1 \vee x_4), \phi_3 = (\neg x_3 \vee x_1 \vee \neg x_2)$$

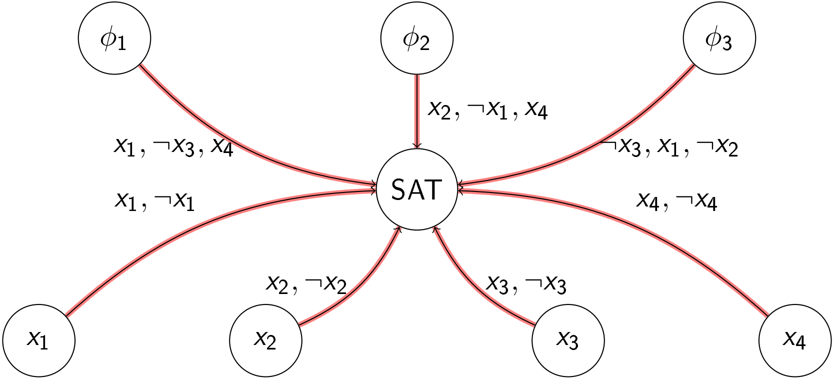


$$\alpha(x_1) = 1 \;\alpha(x_2) = 1 \;\alpha(x_3) = 0$$

Example

$$\Phi = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_1 \vee x_4) \wedge (\neg x_3 \vee x_1 \vee \neg x_2)$$

$$\phi_1 = (x_1 \vee \neg x_3 \vee x_4), \phi_2 = (x_2 \vee \neg x_1 \vee x_4), \phi_3 = (\neg x_3 \vee x_1 \vee \neg x_2)$$



$$\alpha(x_1) = 1 \;\; \alpha(x_2) = 1 \;\; \alpha(x_3) = 0 \;\; \alpha(x_4) = 0$$

Runtime

States	$\lvert\{\phi_i\}\rvert + \lvert\{Var(\Phi)\}\rvert + 1$

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$	$\mathcal{O}(n)$

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$	$\mathcal{O}(n)$
Alphabet	$2 \cdot \lvert Var(\Phi) \rvert$	

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$	$\mathcal{O}(n)$
Alphabet	$\mathcal{O}(3n)$	

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$	$\mathcal{O}(n)$
Alphabet	$\mathcal{O}(3n)$	$\mathcal{O}(n)$

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$	$\mathcal{O}(n)$
Alphabet	$\mathcal{O}(3n)$	$\mathcal{O}(n)$
Transitions	$\lvert Q\rvert \cdot \lvert\Sigma\rvert$	

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$	$\mathcal{O}(n)$
Alphabet	$\mathcal{O}(3n)$	$\mathcal{O}(n)$
Transitions	$\mathcal{O}(n) \cdot \mathcal{O}(n)$	

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$	$\mathcal{O}(n)$
Alphabet	$\mathcal{O}(3n)$	$\mathcal{O}(n)$
Transitions	$\mathcal{O}(n) \cdot \mathcal{O}(n)$	$\mathcal{O}(n^2)$

Runtime

States	$\mathcal{O}(n) + \mathcal{O}(3n)$	$\mathcal{O}(n)$
Alphabet	$\mathcal{O}(3n)$	$\mathcal{O}(n)$
Transitions	$\mathcal{O}(n) \cdot \mathcal{O}(n)$	$\mathcal{O}(n^2)$
Accumulated	$\mathcal{O}(n^2)$	

Correctness

$\Phi \in 3 - SAT \Rightarrow (\mathcal{A}, |Var(\Phi)|) \in L_{SRW}$.

- $\Phi = \bigwedge_{i=1}^{n} \phi_i \in 3 - SAT$
- $\Rightarrow$ There is an assignment α making at least one literal in every clause true
- Pick $w := a_1 \ldots a_n$ with $a_i = \begin{cases} x_i, & \alpha(x_i) = 1 \\ \neg x_i, & \text{else} \end{cases}$

Correctness

$\Phi \in 3 - SAT \Rightarrow (\mathcal{A}, |Var(\Phi)|) \in L_{SRW}$.

- $\Phi = \bigwedge_{i=1}^{n} \phi_i \in 3 - SAT$
- $\Rightarrow$ There is an assignment α making at least one literal in every clause true
- Pick $w := a_1 \ldots a_n$ with $a_i = \begin{cases} x_i, & \alpha(x_i) = 1 \\ \neg x_i, & \text{else} \end{cases}$
    - w maps every state ϕ_i to SAT
    - w maps every literal x_i to SAT
    - $\Rightarrow$ w is synchronising

Correctness

$\Phi \in 3 - SAT \Rightarrow (\mathcal{A}, |Var(\Phi)|) \in L_{SRW}$.

- $\Phi = \bigwedge_{i=1}^{n} \phi_i \in 3 - SAT$
- $\Rightarrow$ There is an assignment α making at least one literal in every clause true
- Pick $w := a_1 \ldots a_n$ with $a_i = \begin{cases} x_i, & \alpha(x_i) = 1 \\ \neg x_i, & \text{else} \end{cases}$
  - w maps every state ϕ_i to SAT
  - w maps every literal x_i to SAT
  - $\Rightarrow w$ is synchronising
- $\Rightarrow \mathcal{A}$ has a synchronising word of length $|Var(\Phi)|$
- $\Rightarrow (\mathcal{A}, |Var(\Phi)|) \in L_{SRW}$

□

Correctness (cont.)

$(\mathcal{A}, |Var(\Phi)|) \in L_{SRW} \Rightarrow \Phi \in 3-SAT.$

- $(\mathcal{A}, |Var(\Phi)|) \in L_{SRW}$
- $\Rightarrow \mathcal{A}$ has a synchronising word w of length $|Var(\Phi)|$

Correctness (cont.)

$(\mathcal{A}, |Var(\Phi)|) \in L_{SRW} \Rightarrow \Phi \in 3-SAT$.

- $(\mathcal{A}, |Var(\Phi)|) \in L_{SRW}$
- $\Rightarrow \mathcal{A}$ has a synchronising word w of length $|Var(\Phi)|$
  - w does not describe a valid assignment $\Rightarrow$ w is not synchronising
  - $\Rightarrow$ w describes a valid assignment

Correctness (cont.)

$(\mathcal{A}, |Var(\Phi)|) \in L_{SRW} \Rightarrow \Phi \in 3-SAT.$

- $(\mathcal{A}, |Var(\Phi)|) \in L_{SRW}$
- $\Rightarrow \mathcal{A}$ has a synchronising word w of length $|Var(\Phi)|$
  - w does not describe a valid assignment $\Rightarrow$ w is not synchronising
  - $\Rightarrow$ w describes a valid assignment

  - Implied assignment α does not satisfy Φ
  - $\Rightarrow$ one of the states ϕ_i is not mapped to SAT
  - $\Rightarrow$ w is not synchronising

Correctness (cont.)

$(\mathcal{A}, |Var(\Phi)|) \in L_{SRW} \Rightarrow \Phi \in 3-SAT$.

- $(\mathcal{A}, |Var(\Phi)|) \in L_{SRW}$
- $\Rightarrow \mathcal{A}$ has a synchronising word w of length $|Var(\Phi)|$
  - w does not describe a valid assignment $\Rightarrow$ w is not synchronising
  - $\Rightarrow$ w describes a valid assignment

  - Implied assignment α does not satisfy Φ
  - $\Rightarrow$ one of the states ϕ_i is not mapped to SAT
  - $\Rightarrow$ w is not synchronising
- $\Rightarrow$ w describes a satisfying assignment α for Φ
- $\Rightarrow \Phi \in$ 3-SAT

□

NP-Completeness

Proof.

- Function $f(\Phi)$ s.t. $\Phi \in$ 3-SAT $\Leftrightarrow f(\Phi) \in L_{SRW}$
- f computable in polynomial time
- $\Rightarrow L_{SRW}$ NP-hard
- $L_{SRW} \in$ NP $\Rightarrow L_{SRW}$ NP-complete

$\square$

Remarks

Lemma

$L_{ShortResetWord}$ is NP-complete.

Remark

$L_{ShortestResetWord}$ is NP-hard and coNP-hard
[Olschewski and Ummels, 2010] $\Rightarrow L_{ShortestResetWord} \notin NP$, unless NP = coNP.

Černý conjecture

$$W_{sync}(\mathcal{A}) := \text{synchronising words of } \mathcal{A}$$

$$C(n) := \max\{ \min_{w \in W_{sync}(\mathcal{A})} (|w|) \mid \mathcal{A} \text{ is a DFA with } n \text{ states } \}$$

Černý conjecture

$$W_{sync}(\mathcal{A}) := \text{synchronising words of } \mathcal{A}$$

$$C(n) := \max\{ \min_{w \in W_{sync}(\mathcal{A})} (|w|) \mid \mathcal{A} \text{ is a DFA with } n \text{ states } \}$$

Lemma

$$C(n) \geq (n-1)^2$$

[Černý, 1964]

Černý conjecture

$$W_{sync}(\mathcal{A}) := \text{synchronising words of } \mathcal{A}$$

$$C(n) := \max\{ \min_{w \in W_{sync}(\mathcal{A})} (|w|) \mid \mathcal{A} \text{ is a DFA with } n \text{ states } \}$$

Lemma

$$C(n) \geq (n-1)^2$$

[Černý, 1964]

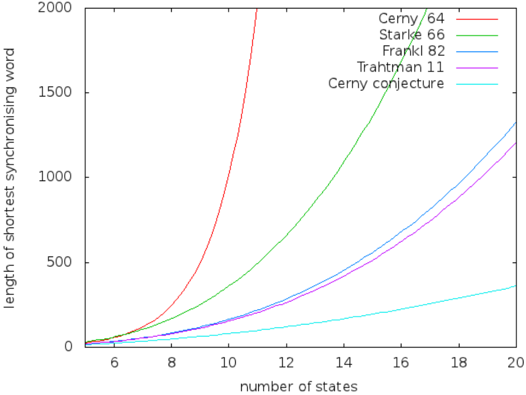
Černý conjecture

$$C(n) = (n-1)^2$$

Proven for several subsets of automata

Known results

- First polynomial bound: $C(n) \leq 1 + \frac{n(n-1)(n-2)}{2}$ [Starke, 1966]
- Simple bound in $\mathcal{O}(n^3)$: $\frac{n^3-n}{6}$ [Pin, 1983] and [Frankl, 1982]
- Recent improvement by $\frac{1}{8}$: $\frac{n(7n^2+6n-16)}{48}$ [Trahtman, 2011]



Conclusion

- Synchronising DFA: DFA with simple extra property
- Has applications in several other fields
- Check if given automaton is synchronising: polynomial time
- Check if given automaton has synchronising word of given length: NP-complete
- Question about shortest synchronising word for given number of states: Still open, possibly in $\mathcal{O}(n^2)$

Ananichev, D. and Volkov, M. (2004).
Some results on Černý type problems for transformation subgroups.
In Araújo, I. M., Branco, M., J. J., Fernandes, V. H., and Gomes, G. M. S., editors, *Proceedings of the Workshop Semigroups and Languages*, pages 23 – 42. World Scientific Publishing Singapore / New Jersey / London.

Benenson, Y., Adar, R., Paz-Elizur, T., Livneh, Z., and Shapiro, E. (2003).
Dna molecule provides a computing machine with both data and fuel.
In Dervan, P. B., editor, *Proceedings of the National Academy of Sciences of the United States of America*, volume 100, pages 2191 – 2196. National Academy of Sciences Washington.

Černý, J. (1964).
Poznámka k. homogénnym experimentoms konecnými automatmi.
Matematicko-fyzikalny Časopis Slovensk. Akad. Vied, 14(3):208–216.

Frankl, P. (1982).
An extremal problem for two families of sets.

European Journal of Combinatorics, 3:125 – 127.

Olschewski, J. and Ummels, M. (2010).
The complexity of finding reset words in finite automata.
In Goos, G., Hartmanis, J., and van Leeuwen, J., editors, *Lecture Notes in Computer Science*, volume 6281, pages 568 – 579. Springer Berlin / Heidelberg.

Pin, J.-É. (1983).
On two combinatorial problems arising from automata theory.
In *Annals of Discrete Mathematics*.

Starke, P. (1966).
Eine Bemerkung über homogene Experimente.
Elektronische Informationsverarbeitung und Kybernetik, 2.

Trahtman, A. (2011).
Modifying the upper bound on the length of minimal synchronizing word.

In Owe, O., Steffen, M., and Telle, J. A., editors, *Lecture Notes in Computer Science*, volume 6914, pages 173–180. Springer Berlin / Heidelberg.

Volkov, M. (2008).
Synchronizing automata and the Černý conjecture.
In Martín-Vide, C., Otto, F., and Fernau, H., editors, *Language and Automata Theory and Applications*, volume 5196 of *Lecture Notes in Computer Science*, pages 11–27. Springer Berlin / Heidelberg.