Complexity of Büchi automata minimization

Dejan Kostyszyn

Chair of Software Engineering

February 3, 2018

Short overview

- Proof by Sven Schewe in 2010

Short overview

- Proof by Sven Schewe in 2010
- Minimization of deterministic Büchi automata (MIN) is NP-complete

Short overview

- Proof by Sven Schewe in 2010
- Minimization of deterministic Büchi automata (MIN) is NP-complete
- Reduction from vertex cover problem to MIN

Short overview

- Proof by Sven Schewe in 2010
- Minimization of deterministic Büchi automata (MIN) is NP-complete
- Reduction from vertex cover problem to MIN

Roadmap

- Foundations

Roadmap

- Foundations
  - Deterministic Büchi automata

Roadmap

- Foundations
  - Deterministic Büchi automata
  - NP-completeness

Roadmap

- Foundations
  - Deterministic Büchi automata
  - NP-completeness
  - The vertex cover problem

Roadmap

- Foundations
  - Deterministic Büchi automata
  - NP-completeness
  - The vertex cover problem
- Definitions & Constructions
  - 'Nice graph G_{v_0}'
  - Language of the nice graph $L(G_{v_0})$
  - DBA that recognises $L(G_{v_0})$

Roadmap

- Foundations
  - Deterministic Büchi automata
  - NP-completeness
  - The vertex cover problem
- Definitions & Constructions
  - 'Nice graph G_{v_0}'
  - Language of the nice graph $L(G_{v_0})$
  - DBA that recognises $L(G_{v_0})$
- The proof

Deterministic Büchi automata (DBA)

Deterministic Büchi automaton $\mathcal{B} := (\Sigma, Q, q_0, \delta, F)$, where

$\Sigma =$ finite set of symbols

$Q =$ finite set of states

$Q_+ = Q \cup \{\bot, \top\}$

$q_0 \in Q_+$ is initial state

$\delta : Q_+ \times \Sigma \to Q_+ \quad , \delta(\bot, a) = \bot \wedge \delta(\top, a) = \top, a \in \Sigma$

$F \subseteq Q_+,$ finite set of final states.

Deterministic Büchi automata (DBA)

Deterministic Büchi automaton $\mathcal{B} := (\Sigma, Q, q_0, \delta, F)$, where

$$\Sigma = \text{finite set of symbols}$$
$$Q = \text{finite set of states}$$
$$Q_+ = Q \cup \{\bot, \top\}$$
$$q_0 \in Q_+ \text{ is initial state}$$
$$\delta : Q_+ \times \Sigma \to Q_+ \quad , \delta(\bot, a) = \bot \wedge \delta(\top, a) = \top, a \in \Sigma$$
$$F \subseteq Q_+, \text{ finite set of final states.}$$

$\rho = q_0 q_1 q_2 \ldots$, where $i \in \mathbb{N}_0 \wedge q_i \in Q_+$, a run.

$\mathcal{B}$ **accepts** exactly those runs in which at least one of the infinitely often occurring states is in F

Deterministic Büchi automata (DBA)

Σ^* is infinite set of **finite** words.

Contains all possible finite combinations of symbols in Σ

Deterministic Büchi automata (DBA)

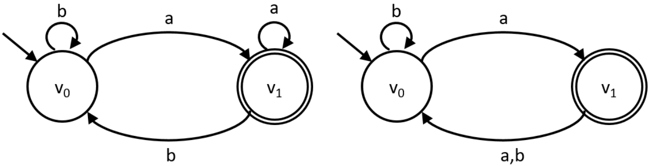
Σ^* is infinite set of **finite** words.

Contains all possible finite combinations of symbols in Σ

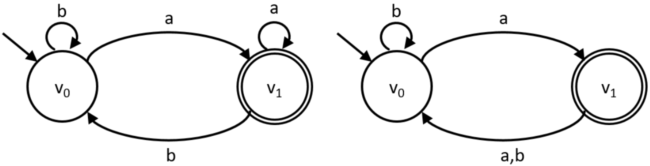
Σ^ω is infinite set of **infinite** words.

Contains all possible infinite combinations of symbols in Σ

Deterministic Büchi automata (DBA)

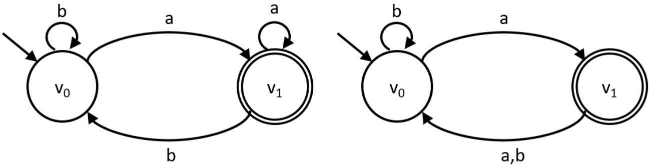


Deterministic Büchi automata (DBA)



$$L = \{w \in \Sigma^\omega | w \text{ contains infinitely many } a's\}$$

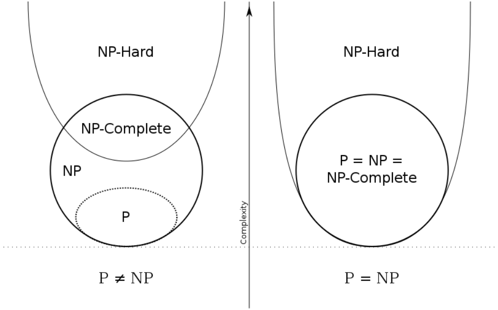
Deterministic Büchi automata (DBA)



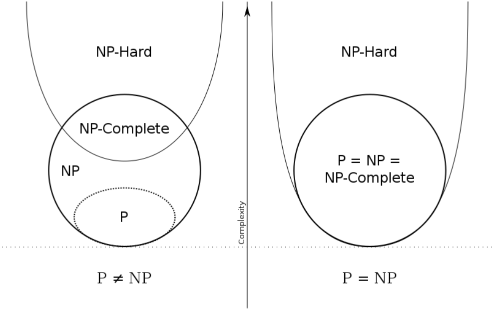
$L = \{w \in \Sigma^\omega | w \text{ contains infinitely many } a's\}$

$\Rightarrow$ Minimal, equivalent, but non-isomorphic

NP-completeness



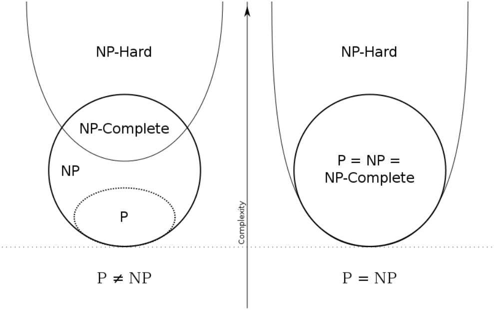
NP-completeness



By Behnam Esfahbod, CC BY-SA 3.0, https:commons.wikimedia.org/w/index.php?curid=3532181

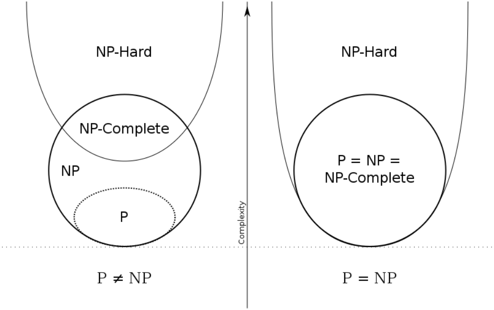
NP is the set of problems that can be solved in non-deterministic polynomial time.

NP-completeness



NP is the set of problems that can be solved in non-deterministic polynomial time.
A problem H is **NP-hard** if every problem $L \in \text{NP}$ can be reduced in polynomial time to H.

NP-completeness



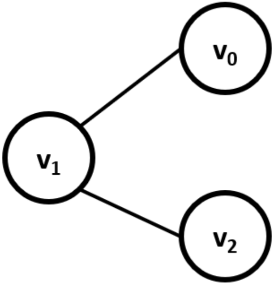
NP is the set of problems that can be solved in non-deterministic polynomial time.
A problem H is **NP-hard** if every problem $L \in \text{NP}$ can be reduced in polynomial time to H.
A problem is **NP-complete** if it belongs to NP and NP-hard.

Vertex cover problem

Let $G = (E, V)$ be an undirected graph.
$S \subseteq V$ is called a **vertex cover** if
$(u, v) \in E \Rightarrow u \in S \vee v \in S$.

Vertex cover problem

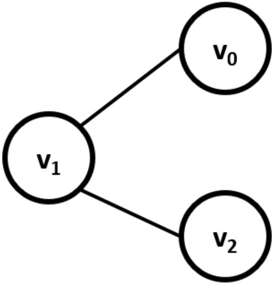
Let $G = (E, V)$ be an undirected graph.
$S \subseteq V$ is called a **vertex cover** if
$(u, v) \in E \Rightarrow u \in S \vee v \in S$.



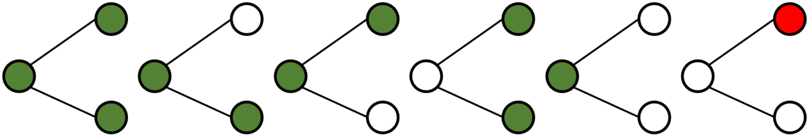
A **minimal vertex cover** (MCOVER) is a vertex cover of minimal size.

Vertex cover problem

Let $G = (E, V)$ be an undirected graph.
$S \subseteq V$ is called a **vertex cover** if
$(u, v) \in E \Rightarrow u \in S \vee v \in S$.



A **minimal vertex cover** (MCOVER) is a vertex cover of minimal size.



Next steps

- Definition 'nice graph'

Next steps

- Definition 'nice graph'
- Definition characteristic language of nice graph $L(G_{v_0})$

Next steps

- Definition 'nice graph'
- Definition characteristic language of nice graph $L(G_{v_0})$
- Construction DBA that recognises $L(G_{v_0})$

Definition of a nice graph

We call a non-trivial ($|V| > 1$) simple connected graph $G_{v_0} = (V, E)$ with a distinguished initial vertex $v_0 \in V$ **nice**.

Lemma (1)

The problem of checking whether a nice graph G_{v_0} has a **vertex cover** *of size k is NP-complete.*

Definition of the characteristic language of the nice graph

We define the *characteristic language* $L(G_{v_0})$ of a nice graph $G_{v_0} = (V, E)$ as the ω-language over $V_\# = V \uplus \{\#\}$.

$\#$ indicates a stop

Definition of the characteristic language of the nice graph

$L(G_{v_0})$ consists of:

- trace words:
  all ω-words of the form $v_0^* v_1^+ v_2^+ v_3^+ v_4^+ \cdots \in V^\omega$ with $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$

Definition of the characteristic language of the nice graph

$L(G_{v_0})$ consists of:

- trace words:
  all ω-words of the form $v_0^* v_1^+ v_2^+ v_3^+ v_4^+ \cdots \in V^\omega$ with $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$
- $\#$-words ('stop'-words):
  all words **starting** with $v_0^* v_1^+ v_2^+ \ldots v_n^+ \# v_n \in V_\#^*$ with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$.

Definition of the characteristic language of the nice graph

$L(G_{v_0})$ consists of:

- trace words:
  all ω-words of the form $v_0^* v_1^+ v_2^+ v_3^+ v_4^+ \cdots \in V^\omega$ with $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$
- $\#$-words ('stop'-words):
  all words **starting** with $v_0^* v_1^+ v_2^+ \ldots v_n^+ \# v_n \in V_\#^*$ with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$.

Trace words are in V^ω and $\#$-words are in $V_\#^\omega \setminus V^\omega$

Definition of DBA that recognises $L(G_{v_0})$

DBA $\mathcal{B} = (V, Q, q_0, \delta, F)$, nice graph $G_{v_0} = (V, E)$.

Definition of DBA that recognises $L(G_{v_0})$

DBA $\mathcal{B} = (V, Q, q_0, \delta, F)$, nice graph $G_{v_0} = (V, E)$.

The states of $\mathcal{B}$ are called

- *v-state* if it can be reached upon an input word $v_0^* v_1^+ v_2^+ \dots v_n^+ \in V^*$, with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$, that ends in $v = v_n$.

Definition of DBA that recognises $L(G_{v_0})$

DBA $\mathcal{B} = (V, Q, q_0, \delta, F)$, nice graph $G_{v_0} = (V, E)$.

The states of $\mathcal{B}$ are called

- *v-state* if it can be reached upon an input word $v_0^* v_1^+ v_2^+ \dots v_n^+ \in V^*$, with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$, that ends in $v = v_n$.
- *$v\#$-state* if it can be reached from a v-state upon reading a $\#$ sign.

Definition of DBA that recognises $L(G_{v_0})$

DBA $\mathcal{B} = (V, Q, q_0, \delta, F)$, nice graph $G_{v_0} = (V, E)$.

The states of $\mathcal{B}$ are called

- *v-state* if it can be reached upon an input word $v_0^* v_1^+ v_2^+ \dots v_n^+ \in V^*$, with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$, that ends in $v = v_n$.
- *$v\#$-state* if it can be reached from a v-state upon reading a $\#$ sign.

vertex-states = set of v-states.

Definition of DBA that recognises $L(G_{v_0})$

DBA $\mathcal{B} = (V, Q, q_0, \delta, F)$, nice graph $G_{v_0} = (V, E)$.

The states of $\mathcal{B}$ are called

- *v-state* if it can be reached upon an input word $v_0^* v_1^+ v_2^+ \dots v_n^+ \in V^*$, with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$, that ends in $v = v_n$.
- *$v\#$-state* if it can be reached from a v-state upon reading a $\#$ sign.

vertex-states = set of v-states.
$\#$-states = set of $v\#$-states.

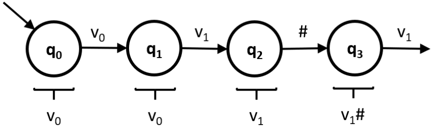
Definition of DBA that recognises $L(G_{v_0})$

DBA $\mathcal{B} = (V, Q, q_0, \delta, F)$, nice graph $G_{v_0} = (V, E)$.

The states of $\mathcal{B}$ are called

- *v-state* if it can be reached upon an input word $v_0^* v_1^+ v_2^+ \dots v_n^+ \in V^*$, with $n \in \mathbb{N}_0$ and $\{v_{i-1}, v_i\} \in E$ for all $i \in \mathbb{N}$, that ends in $v = v_n$.
- *v#-state* if it can be reached from a v-state upon reading a $\#$ sign.

vertex-states = set of v-states.
#-states = set of $v\#$-states.



Definition of DBA, that recognises $L(G_{v_0})$

Lemma (2)

$\mathcal{B}$ has the following properties:

1. *The vertex- and #-states of $\mathcal{B}$ are disjoint.*

Definition of DBA, that recognises $L(G_{v_0})$

Lemma (2)

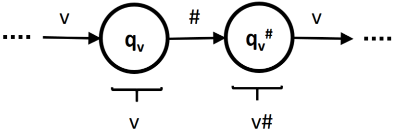
$\mathcal{B}$ has the following properties:

1. *The vertex- and $\#$-states of $\mathcal{B}$ are disjoint.*

Proof.

Let $q_v^{\#}$ be a $v\#$-state and q a vertex-state.
As $\mathcal{B}$ recognises $L(G_{v_0})$, $\mathcal{B}_{q_v^{\#}}$ must accept v^{ω}, while $\mathcal{B}_q$ must reject it. □

trace-words:
$v_0^* v_1^+ v_2^+ v_3^+ v_4^+ \cdots \in V^{\omega}$



Definition of DBA, that recognises $L(G_{v_0})$

Lemma (2)

$\mathcal{B}$ has the following properties:

1. *The vertex- and $\#$-states of $\mathcal{B}$ are disjoint.*
2. *$\forall v, w \in V$ with $v \neq w$ the v-states and w-states are disjoint.*

Definition of DBA, that recognises $L(G_{v_0})$

Lemma (2)

$\mathcal{B}$ has the following properties:

1. *The vertex- and $\#$-states of $\mathcal{B}$ are disjoint.*
2. *$\forall v, w \in V$ with $v \neq w$ the v-states and w-states are disjoint.*
3. *$\forall v, w \in V$ with $v \neq w$ the $v\#$-states and $w\#$-states are disjoint.*

Definition of DBA, that recognises $L(G_{v_0})$

Lemma (2)

$\mathcal{B}$ has the following properties:

1. *The vertex- and $\#$-states of $\mathcal{B}$ are disjoint.*
2. $\forall v, w \in V$ *with* $v \neq w$ *the* v*-states and* w*-states are disjoint.*
3. $\forall v, w \in V$ *with* $v \neq w$ *the* $v\#$*-states and* $w\#$*-states are disjoint.*
4. *For each vertex* $v \in V$*, there is a* $v\#$*-state.*

Definition of DBA, that recognises $L(G_{v_0})$

Lemma (2)

$\mathcal{B}$ has the following properties:

1. *The vertex- and #-states of $\mathcal{B}$ are disjoint.*
2. *$\forall v, w \in V$ with $v \neq w$ the v-states and w-states are disjoint.*
3. *$\forall v, w \in V$ with $v \neq w$ the $v\#$-states and $w\#$-states are disjoint.*
4. *For each vertex $v \in V$, there is a $v\#$-state.*
5. *For each vertex $v \in V$, there is a rejecting v-state.*

Definition of DBA, that recognises $L(G_{v_0})$

Lemma (2)

$\mathcal{B}$ has the following properties:

1. *The vertex- and #-states of $\mathcal{B}$ are disjoint.*
2. $\forall v, w \in V$ *with* $v \neq w$ *the* v*-states and* w*-states are disjoint.*
3. $\forall v, w \in V$ *with* $v \neq w$ *the* $v\#$*-states and* $w\#$*-states are disjoint.*
4. *For each vertex* $v \in V$*, there is a* $v\#$*-state.*
5. *For each vertex* $v \in V$*, there is a rejecting* v*-state.*
6. *For every edge* $\{v, w\} \in E$*, there is an accepting* v*-state or an accepting* w*-state.*

Definition of DBA, that recognises $L(G_{v_0})$

6. For every edge $\{v, w\} \in E$, there is an accepting v-state or an accepting w-state.

$$\Rightarrow$$

The set C of vertices with an accepting vertex-state is a **vertex cover** of $G = (V, E)$.

Definition of DBA, that recognises $L(G_{v_0})$

Corollary (1)

For a DBA $\mathcal{B}$ that recognises the characteristic language of a nice graph $G_{v_0} = (V, E)$ with initial vertex v_0, the set $C = \{v \in V |$ there is an accepting v-state$\}$ is a **vertex cover** *of G_{v_0}, and $\mathcal{B}$ has at least $2|V| + |C|$ states.*

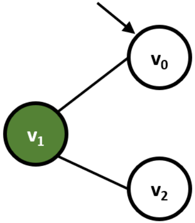
Definition of DBA, that recognises $L(G_{v_0})$

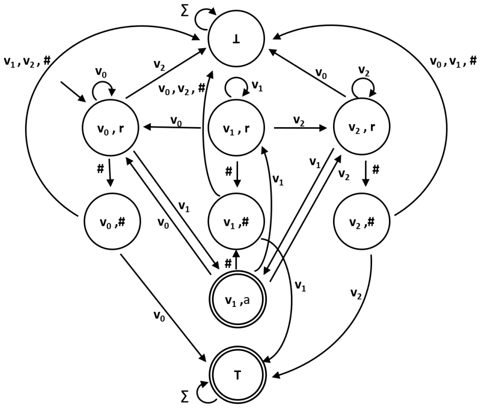
$$\mathcal{B}' = (V_{\#}, (V \times \{r, \#\}) \uplus (C \times \{a\}), (v_0, r), \delta, (C \times \{a\}) \uplus \{\top\}).$$

- $\delta((v, r), v') = (v', a)$ if $\{v, v'\} \in E$ and $v' \in C$,
  $\delta((v, r), v') = (v', r)$ if $\{v, v'\} \in E$ and $v' \in C$,
  $\delta((v, r), v') = (v, r)$ if $v = v'$,
  $\delta((v, r), v') = (v, \#)$ if $v = \#$,
  $\delta((v, r), v') = \bot$ otherwise;
- $\delta((v, a), v') = \delta((v, r), v\#)$, and
- $\delta((v, \#), v) = \top$ and $\delta((v, \#), v') = \bot$ for $v\# \neq v$.

Example of DBA, that recognises $L(G_{v_0})$

$\delta((v, r), v') = (v', a)$ if $\{v, v'\} \in E$ and $v' \in C$,
$\delta((v, r), v') = (v', r)$ if $\{v, v'\} \in E$ and $v' \in C$,
$\delta((v, r), v') = (v, r)$ if $v = v'$,
$\delta((v, r), v') = (v, \#)$ if $v = \#$,
$\delta((v, r), v') = \bot$ otherwise;
$\delta((v, a), v') = \delta((v, r), v\#)$,
$\delta((v, \#), v) = \top$ and $\delta((v, \#), v') = \bot$ for $v\# \neq v$.





Definition of DBA, that recognises $L(G_{v_0})$

Lemma (3)

For a nice graph $G_{v_0} = (V, E)$ with initial vertex v_0 and vertex cover C, $\mathcal{B}'$ recognises the characteristic language of G_{v_0}.

Definition of DBA, that recognises $L(G_{v_0})$

Corollary (1)

For a DBA $\mathcal{B}$ that recognises the characteristic language of a nice graph $G_{v_0} = (V, E)$ with initial vertex v_0, the set $C = \{v \in V |$ there is an accepting v-state$\}$ is a **vertex cover** *of G_{v_0}, and $\mathcal{B}$ has at least $2|V| + |C|$ states.*

Lemma (3)

For a nice graph $G_{v_0} = (V, E)$ with initial vertex v_0 and vertex cover C, $\mathcal{B}'$ recognises the characteristic language of G_{v_0}.

$$\Rightarrow$$

Corollary (2)

Let C be a MCOVER of a nice graph $G_{v_0} = (V, E)$. Then $\mathcal{B}'$ is a minimal DBA that recognises the characteristic language of G_{v_0}.

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof: Containment in NP.

For containment in NP, we can simply use non-determinism to guess such an automaton. Because the equivalence test can be done in polynomial time, the problem must be in NP. □

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof: Containment in NP-hard.

$G_v = (V, E)$

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof: Containment in NP-hard.

$G_v = (V, E)$
trivial vertex cover $C = V, |V| = m$

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof: Containment in NP-hard.

$G_v = (V, E)$
trivial vertex cover $C = V, |V| = m$
Construction
$\mathcal{B}'$ has $2|V| + |C| = 2m + m = 3m$ states

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof: Containment in NP-hard.

$G_v = (V, E)$
trivial vertex cover $C = V, |V| = m$
Construction
$\mathcal{B}'$ has $2|V| + |C| = 2m + m = 3m$ states
Question 1: $\exists$ vertex cover of size k?

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof: Containment in NP-hard.

$G_v = (V, E)$
trivial vertex cover $C = V, |V| = m$
Construction
$\mathcal{B}'$ has $2|V| + |C| = 2m + m = 3m$ states
Question 1: $\exists$ vertex cover of size k?
Question 2: $\exists$ DBA $\mathcal{B}$ with $2m + k$ states?

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof: Containment in NP-hard.

$G_v = (V, E)$
trivial vertex cover $C = V, |V| = m$
Construction
$\mathcal{B}'$ has $2|V| + |C| = 2m + m = 3m$ states
Question 1: $\exists$ vertex cover of size k?
Question 2: $\exists$ DBA $\mathcal{B}$ with $2m + k$ states?

Corollary 2:
If C is MCOVER,
then $\mathcal{B}$ is minimal.

Proof of Theorem

Theorem

The problem of whether there is, for a given DBA, a language equivalent Büchi automaton with at most k states is NP-complete.

Proof: Containment in NP-hard.

$G_v = (V, E)$
trivial vertex cover $C = V, |V| = m$
Construction
$\mathcal{B}'$ has $2|V| + |C| = 2m + m = 3m$ states
Question 1: $\exists$ vertex cover of size k?
Question 2: $\exists$ DBA $\mathcal{B}$ with $2m + k$ states?

Corollary 2:
If C is MCOVER,
then $\mathcal{B}$ is minimal.

$\Rightarrow$ NP-complete

$\square$

Sources

Thank you for listening!

Schewe, Sven. 2010. „Minimisation of Deterministic Parity and Buchi Automata and Relative Minimisation of Deterministic Finite Automata“. arXiv:1007.1333 [cs], Juli. http://arxiv.org/abs/1007.1333.