The Tractability Frontier for NFA Minimization

Henrik Björklund Wim Martens

TU Dortmund

Notation

- NFA: (Non-Deterministic) Finite State Automata
- DFA: Deterministic Finite State Automata
- UFA: Unambiguous Finite State Automata

Unambiguous = at most one accepting run per string

Notation

- NFA: (Non-Deterministic) Finite State Automata
- DFA: Deterministic Finite State Automata
- UFA: Unambiguous Finite State Automata

Unambiguous = at most one accepting run per string

Definition ($X \rightarrow Y$ Minimization standard version)

- Input: Automaton A from class X
- Output: Automaton B in class Y such that
  - A and B are equivalent
  - B is minimal in class Y

Notation

Definition ($X \to Y$ Minimization standard version)

- Input: Automaton A from class X
- Output: Automaton B in class Y such that
  - A and B are equivalent
  - B is minimal in class Y

Example

- DFA $\to$ DFA $\approx$ classical DFA minimization problem
- DFA $\to$ NFA $\approx$ given a DFA, compute minimal NFA

Notation

In this paper we'll use the decision version of state minimization

Definition ($X \to Y$ Minimization decision version)

- Input: Automaton A from class X, integer n in binary
- Output: Does there exist an automaton B in class Y such that
  - A and B are equivalent and
  - B has at most n states?

Observation

Lower bounds for decision version imply lower bounds for standard version

DFA Minimization

- An old-school problem
- Algorithms for minimizing DFAs are in every undergraduate CS curriculum
- If not, they should be

[Huffmann 1954, Moore 1956, Hopcroft 1971]

DFA $\rightarrow$ DFA Minimization is in $\mathcal{O}(n \log n)$

But What About NFAs?

In practice: Bisimulation Minimization [Paige, Tarjan 1987]

- efficient
- usually makes the input automaton smaller

In general, NFA $\rightarrow$ NFA Minimization is **PSPACE**-complete

But What About NFAs?

Further Results

[Jiang, Ravikumar 1993]:

- UFA $\longrightarrow$ UFA Minimization is **NP**-complete
- DFA $\longrightarrow$ UFA Minimization is **NP**-complete
- DFA $\longrightarrow$ NFA Minimization is **PSPACE**-complete

But What About NFAs?

Further Results

[Malcher 2003]: Minimization is **NP**-complete for

- DFA $\rightarrow$ k-MDFA for all $k \geq 2$
- DFA $\rightarrow$ NFA(branching k) for all $k \geq 3$

k-MDFA: Possibly ambiguous automata with k initial states, but otherwise a deterministic transition function

NFA(branching k): NFAs with k possible computations per string

Several (technical) different techniques are used for lower bounds

But What About NFAs?

Question [Malcher 2003]

Are there any classes of non-DFAs with efficient minimization?

But What About NFAs?

Question [Malcher 2003]

Are there any classes of non-DFAs with efficient minimization?

The short answer

[Here]: No

But What About NFAs?

Question [Malcher 2003]

Are there any classes of non-DFAs with efficient minimization?

The long answer

[Here]: OK, yes. But we don't think they'll be very useful

So What's the Result?

Definition (δNFA)

The class of NFAs that

- have at most one pair (q, a) such that $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$
- have one start state
- are unambiguous
- do not loop

So What's the Result?

Definition (δNFA)

The class of NFAs that

- have at most one pair (q, a) such that $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$
- have one start state
- are unambiguous
- do not loop

Theorem

For every class $\mathcal{N}$ of NFAs such that $\delta NFA \subseteq \mathcal{N}$:

$$DFA \to \mathcal{N} \text{ } \textit{Minimization is } \mathbf{NP}\textit{-hard}$$

So What's the Result?

Definition (δNFA)

The class of NFAs that

- have at most one pair (q, a) such that $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$
- have one start state
- are unambiguous
- do not loop

Theorem

For every class $\mathscr{N}$ of NFAs such that $\delta NFA \subseteq \mathscr{N}$:

$DFA \to \mathscr{N}$ Minimization is **NP**-*hard*

One **NP** lower bound proof that unifies and strengthens all previous cases

Outline

1. Some Technical Details
2. Closer to Determinism?
3. Concluding Remarks

A Proof Revisited (Jiang, Ravikumar 1993)

Definition (Vertex Cover)

$G = (V, E)$ graph

$V' \subseteq V$ Vertex Cover of $G \Leftrightarrow \forall (v_1, v_2) \in E$, $\{v_1, v_2\} \cap V' \neq \emptyset$

Definition (Set Basis)

$\mathscr{B}$, $\mathscr{C}$ finite collections of finite sets

$\mathscr{B}$ Set Basis of $\mathscr{C} \Leftrightarrow \forall C \in \mathscr{C}\ \exists B_C \subseteq \mathscr{B}$: $\displaystyle\bigcup_{B \in B_C} B = C$

A Proof Revisited (Jiang, Ravikumar 1993)

Definition (Set Basis)

$\mathscr{B}$, $\mathscr{C}$ finite collections of finite sets

$\mathscr{B}$ Set Basis of $\mathscr{C}$ $\Leftrightarrow$ $\forall C \in \mathscr{C}\ \exists B_C \subseteq \mathscr{B}$: $\bigcup_{B \in B_C} B = C$

Definition (Separable Normal Set Basis)

$\mathscr{B}$, $\mathscr{C}$ finite collections of finite sets

$\mathscr{B}$ Separable Normal Set Basis of $\mathscr{C}$ $\Leftrightarrow$ $\forall C \in \mathscr{C}\ \exists B_C \subseteq \mathscr{B}$:

- $\biguplus_{B \in B_C} B = C$
- the sets in B_C are disjoint
- B_C contains at most two sets

A Proof Revisited (Jiang, Ravikumar 1993)

Decision Problems

- Vertex Cover:

  Given $G = (V, E)$ and integer k,
  does there exist a Vertex Cover with at most k nodes?

- Separable Normal Set Basis:

  Given collection $\mathscr{C}$ and integer s,
  does there exist a Separable Normal Set Basis $\mathscr{B}$ with at most s sets?

A Proof Revisited (Jiang, Ravikumar 1993)

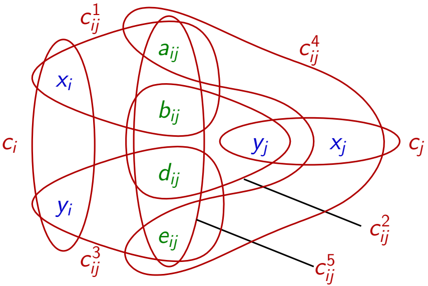
Lemma

(Separable) Normal Set Basis is **NP***-complete*

Proof Idea

Reduction from Vertex Cover

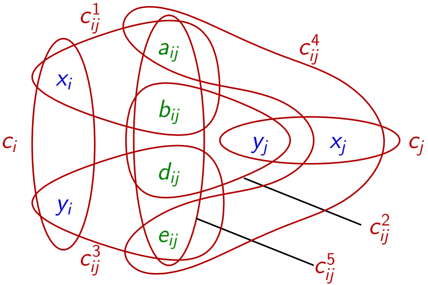
Translate each edge (v_i, v_j) in graph G into the collection



A Proof Revisited (Jiang, Ravikumar 1993)

Proof Idea

Translate each edge (v_i, v_j) in graph G into the collection

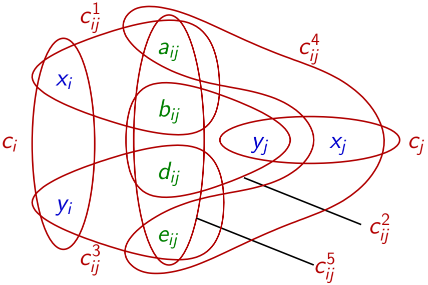


This collection has $|V| + 5|E|$ sets

A Proof Revisited (Jiang, Ravikumar 1993)

Proof Idea

Translate each edge (v_i, v_j) in graph G into the collection



This collection has $|V| + 5|E|$ sets

G has a Vertex Cover of size k $\Leftrightarrow$
this collection has a (Sep.) Normal Set Basis with $|V| + 4|E| + k$ sets $\square$

Strengthening the Jiang-Ravikumar Result

Lemma (Set Basis = Sep.Norm.Set Basis on some NP-complete instances)

For the above reduction from Vertex Cover to Sep. NSB we also have that

G *has a Vertex Cover of size* k

$\Leftrightarrow$ *the collection has a Separable NSB with* $|V| + 4|E| + k$ *sets*

$\Leftrightarrow$ *the collection has a Set Basis with* $|V| + 4|E| + k$ *sets*

Proof.

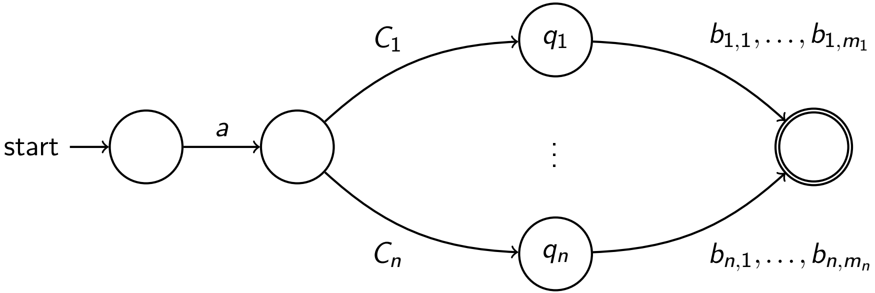
If there is a Set Basis,

show with a case study that there is also a Separable NSB $\square$

From Sep. Normal Set Basis to Automata Minimization

Let $\mathscr{C} = \{C_1, \ldots, C_n\}$ be a collection of n sets, $C_i = \{b_{i,1}, \ldots, b_{i,m_i}\}$

A is the DFA for $\{aCb \mid C \in \mathscr{C} \text{ and } b \in C\}$



From Sep. Normal Set Basis to Automata Minimization

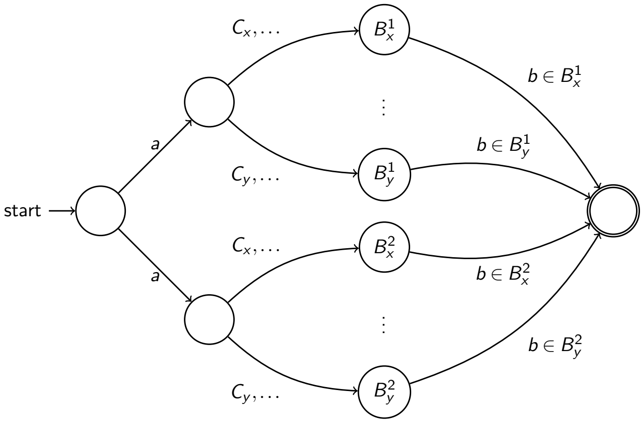
If there is a Separable NSB $\mathscr{B} = \{B_1, \ldots, B_\ell\}$ for $\mathscr{C}$, then

fix, for every $C_x \in \mathscr{C}$,

$$B_x^1 \text{ and } B_x^2 \in \mathscr{B} \text{ s.t. } C_x = B_x^1 \uplus B_x^2$$

From Sep. Normal Set Basis to Automata Minimization

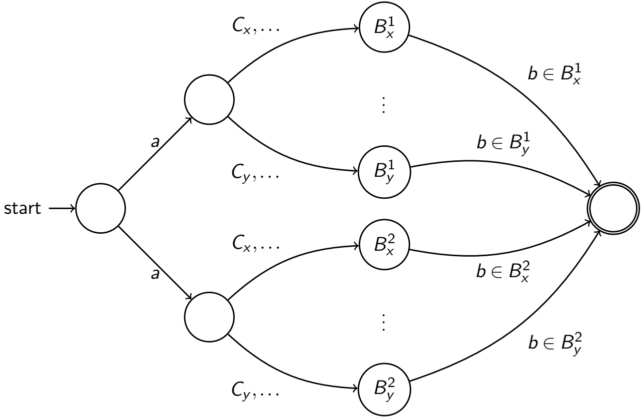
If there is a Separable NSB $\mathscr{B} = \{B_1, \ldots, B_\ell\}$ for $\mathscr{C}$, then



is a δNFA for $\{aCb \mid C \in \mathscr{C}$ and $b \in C\}$ of size $\ell + 4$

From Sep. Normal Set Basis to Automata Minimization

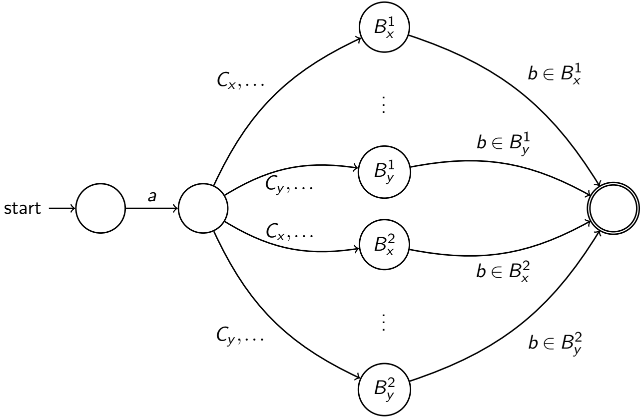
There is a Separable NSB $\mathscr{B} = \{B_1, \ldots, B_\ell\}$ for $\mathscr{C}$ if and only if



is a δNFA for $\{aCb \mid C \in \mathscr{C} \text{ and } b \in C\}$ of size $\ell + 4$

From Sep. Normal Set Basis to Automata Minimization

There is a Separable NSB $\mathscr{B} = \{B_1, \ldots, B_\ell\}$ for $\mathscr{C}$ if and only if



is an NFA for $\{aCb \mid C \in \mathscr{C} \text{ and } b \in C\}$ of size $\ell + 3$

So we just proved ...

Lemma

The following are equivalent:

- $\mathscr{C}$ *has a Sep. NSB of at most* ℓ *sets*
- *there is a* δ*NFA for* $L(A)$ *of size at most* $\ell + 4$
- *there is an NFA for* $L(A)$ *of size at most* $\ell + 3$

Corollary

There exists a set of regular languages $\mathscr{L}$ *such that*

- *DFA* $\rightarrow$ δ*NFA Minimization is* **NP**-*complete for DFAs accepting* $\mathscr{L}$
- *for each* $L \in \mathscr{L}$, *the minimal NFA for* L *has one state less than the minimal* δ*NFA for* L

Putting Things Together

Theorem

Let $\mathscr{N}$ be a class of NFAs.
*If δNFA $\subseteq \mathscr{N}$ then DFA $\rightarrow \mathscr{N}$ Minimization is **NP**-hard.*

Proof.

We gave a reduction from Vertex Cover to DFA $\rightarrow$ δNFA Minimization

Putting Things Together

Theorem

Let $\mathscr{N}$ be a class of NFAs.
If δNFA $\subseteq \mathscr{N}$ then DFA $\rightarrow \mathscr{N}$ Minimization is **NP**-*hard.*

Proof.

We gave a reduction from Vertex Cover to DFA $\rightarrow \delta$NFA Minimization

Let $\mathscr{N}$ be a class s.t. δNFA $\subseteq \mathscr{N} \subseteq$ NFA

Putting Things Together

Theorem

Let $\mathscr{N}$ be a class of NFAs.
If $\delta NFA \subseteq \mathscr{N}$ then $DFA \to \mathscr{N}$ Minimization is **NP***-hard.*

Proof.

We gave a reduction from Vertex Cover to DFA $\to$ δNFA Minimization

Let $\mathscr{N}$ be a class s.t. $\delta\text{NFA} \subseteq \mathscr{N} \subseteq \text{NFA}$

A decision algorithm for DFA $\to \mathscr{N}$ Minimization can approximate DFA $\to$ δNFA Minimization within a term 1

Putting Things Together

Theorem

Let $\mathscr{N}$ be a class of NFAs.
*If δNFA $\subseteq \mathscr{N}$ then DFA $\to \mathscr{N}$ Minimization is **NP**-hard.*

Proof.

We gave a reduction from Vertex Cover to DFA $\to$ δNFA Minimization

Let $\mathscr{N}$ be a class s.t. δNFA $\subseteq \mathscr{N} \subseteq$ NFA

A decision algorithm for DFA $\to \mathscr{N}$ Minimization can approximate DFA $\to$ δNFA Minimization within a term 1

The approximation for DFA $\to \mathscr{N}$ Minimization can be adapted to an approximation of Vertex Cover within a term 1

Putting Things Together

Theorem

Let $\mathscr{N}$ be a class of NFAs.
If $\delta NFA \subseteq \mathscr{N}$ then DFA $\rightarrow \mathscr{N}$ Minimization is **NP***-hard.*

Proof.

We gave a reduction from Vertex Cover to DFA $\rightarrow \delta$NFA Minimization

Let $\mathscr{N}$ be a class s.t. $\delta\text{NFA} \subseteq \mathscr{N} \subseteq \text{NFA}$

A decision algorithm for DFA $\rightarrow \mathscr{N}$ Minimization can approximate
DFA $\rightarrow \delta$NFA Minimization within a term 1

The approximation for DFA $\rightarrow \mathscr{N}$ Minimization can be adapted to
an approximation of Vertex Cover within a term 1

Approximating Vertex Cover within a constant term is **NP**-complete
$\Rightarrow$ DFA $\rightarrow \mathscr{N}$ Minimization is **NP**-hard $\square$

Outline

Are All Classes of non-DFAs hard to Minimize?

(non-DFAs: Classes $\mathscr{N}$ such that DFA $\subseteq \mathscr{N}$ but not $\mathscr{N} \subseteq$ DFA)

Are All Classes of non-DFAs hard to Minimize?

Answer

Of course not!

Example (Infinitely many classes between DFA and δNFA)

Take the class of DFAs, and add a single δNFA

$\Rightarrow$ Minimization in **P**!

Are All Classes of non-DFAs hard to Minimize?

Let's look at a more interesting example

Definition (δ'NFA)

The class of NFAs that

- have at most one pair (q, a) such that $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$
- have one start state
- are unambiguous
- for each input w, have at most one rejecting run

(For each input w there are at most 2 runs: 1 accepting and 1 rejecting)

Are All Classes of non-DFAs hard to Minimize?

Let's look at a more interesting example

Definition (δ'NFA)

The class of NFAs that

- have at most one pair (q, a) such that $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$
- have one start state
- are unambiguous
- for each input w, have at most one rejecting run

(For each input w there are at most 2 runs: 1 accepting and 1 rejecting)

Observation

- δ'NFAs can be minimized in **P**

Are All Classes of non-DFAs hard to Minimize?

Let's look at a more interesting example

Definition (δ'NFA)

The class of NFAs that

- have at most one pair (q, a) such that $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$
- have one start state
- are unambiguous
- for each input w, have at most one rejecting run

(For each input w there are at most 2 runs: 1 accepting and 1 rejecting)

Observation

- δ'NFAs can be minimized in **P**
- but the minimal δ'NFAs are the DFAs!

δ'NFA can be minimized in **PTIME**

Take δ'NFA A that's not a DFA, let (q, a) be the unique state,label with

$$q \xrightarrow{a} q_1 \qquad q \xrightarrow{a} q_2$$

Let w be a string that leads A to q

δ'NFA can be minimized in **PTIME**

Take δ'NFA A that's not a DFA, let (q, a) be the unique state,label with

$$q \xrightarrow{a} q_1 \qquad q \xrightarrow{a} q_2$$

Let w be a string that leads A to q

As A is a δ'NFA, it must accept all waw'

(otherwise there are two rejecting runs)

δ'NFA can be minimized in **PTIME**

Take δ'NFA A that's not a DFA, let (q, a) be the unique state,label with

$$q \xrightarrow{a} q_1 \qquad q \xrightarrow{a} q_2$$

Let w be a string that leads A to q

As A is a δ'NFA, it must accept all waw'
(otherwise there are two rejecting runs)

So A can be made smaller by merging q_1 and q_2 into new state q_3
from which A accepts everything

A becomes deterministic this way

So, determinization followed by standard minimization is a **P** algorithm

Outline

Concluding Remarks

What did we do?

- State minimization is hard for all finite automata classes that include δNFAs
- One proof unifying and strengthening previous approaches
- The minimization tractability frontier is between δNFA and δ'NFA

Concluding Remarks

Is everything solved yet?

- What we didn't consider yet: fixed alphabet size
- What about approximations?

Thank you for listening