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 \((X \rightarrow Y \text{ Minimization standard version})\)

- **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 \(\rightarrow\) DFA \(\approx\) classical DFA minimization problem
- DFA \(\rightarrow\) NFA \(\approx\) given a DFA, compute minimal NFA
In this paper we’ll use the **decision version of state minimization**

**Definition (X → 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 → 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 → NFA Minimization is $\text{PSPACE}$-complete
But What About NFAs?

Further Results

[Jiang, Ravikumar 1993]:

- UFA $\rightarrow$ UFA Minimization is $\text{NP}$-complete
- DFA $\rightarrow$ UFA Minimization is $\text{NP}$-complete
- DFA $\rightarrow$ NFA Minimization is $\text{PSPACE}$-complete
But What About NFAs?

Further Results

[Malcher 2003]: Minimization is \textbf{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
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 ($\delta$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
- do not loop

Theorem

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

$$DFA \rightarrow \mathcal{N} \text{ Minimization is NP-hard}$$
So What’s the Result?

Definition ($\delta$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
- do not loop

Theorem

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

$$\text{DFA} \rightarrow \mathcal{N} \text{ Minimization is NP-hard}$$

One NP lower bound proof that unifies and strengthens all previous cases
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 \( G \) \( \iff \forall (v_1, v_2) \in E, \{v_1, v_2\} \cap V' \neq \emptyset \)

Definition (Set Basis)

\( \mathcal{B}, \mathcal{C} \) finite collections of finite sets

\( \mathcal{B} \) Set Basis of \( \mathcal{C} \) \( \iff \forall C \in \mathcal{C} \exists B_C \subseteq \mathcal{B}: \bigcup_{B \in B_C} B = C \)
Definition (Set Basis)

$B, C$ finite collections of finite sets

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

Definition (Separable Normal Set Basis)

$B, C$ finite collections of finite sets

$B$ Separable Normal Set Basis of $C$ $\iff$ $\forall C \in C \exists B_C \subseteq B$:

- $\biguplus_{B \in B_C} B = C$
- the sets in $B_C$ are disjoint
- $B_C$ contains at most two sets
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 $\mathcal{C}$ and integer $s$, 
  does there exist a Separable Normal Set Basis $\mathcal{B}$ with at most $s$ sets?
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

\[
\begin{align*}
    c_i, c_j, x_i, y_i, y_j, x_j, a_{ij}, b_{ij}, d_{ij}, e_{ij}, c^1_{ij}, c^2_{ij}, c^3_{ij}, c^4_{ij}, c^5_{ij}
\end{align*}
\]
Proof Idea

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

This collection has \(|V| + 5|E|\) sets
Proof Idea

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

\[
\begin{align*}
&\text{This collection has } |V| + 5|E| \text{ sets} \\
&G \text{ has a Vertex Cover of size } k \iff \\
&\text{this collection has a (Sep.) Normal Set Basis with } |V| + 4|E| + k \text{ sets}
\end{align*}
\]
Lemma (Set Basis = Sep.Norm.Set Basis on some \( \text{NP} \)-complete instances)

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

\[ G \text{ has a Vertex Cover of size } k \]
\[ \iff \text{the collection has a Separable NSB with } |V| + 4|E| + k \text{ sets} \]
\[ \iff \text{the collection has a Set Basis with } |V| + 4|E| + k \text{ sets} \]

Proof.

If there is a Set Basis,

show with a case study that there is also a Separable NSB

□
Let \( \mathcal{C} = \{C_1, \ldots, C_n\} \) be a collection of \( n \) sets, \( C_i = \{b_{i,1}, \ldots, b_{i,m_i}\} \).

\( A \) is the DFA for \( \{aCb \mid C \in \mathcal{C} \text{ and } b \in C\} \).
If there is a Separable NSB $\mathcal{B} = \{B_1, \ldots, B_\ell\}$ for $\mathcal{C}$, then fix, for every $C_x \in \mathcal{C}$,

$$B^1_x \text{ and } B^2_x \in \mathcal{B} \text{ s.t. } C_x = B^1_x \cup B^2_x$$
If there is a Separable NSB $\mathcal{B} = \{B_1, \ldots, B_\ell\}$ for $\mathcal{C}$, then is a $\delta$NFA for $\{aCb \mid C \in \mathcal{C} \text{ and } b \in C\}$ of size $\ell + 4$.
There is a Separable NSB $\mathcal{B} = \{B_1, \ldots, B_\ell\}$ for $\mathcal{C}$ if and only if

is a $\delta$NFA for $\{aCb \mid C \in \mathcal{C} \text{ and } b \in C\}$ of size $\ell + 4$
There is a Separable NSB $\mathcal{B} = \{B_1, \ldots, B_\ell\}$ for $\mathcal{C}$ if and only if

is an NFA for $\{aCb \mid C \in \mathcal{C} \text{ and } b \in C\}$ of size $\ell + 3$
So we just proved …

Lemma

The following are equivalent:

- $\mathcal{C}$ has a Sep. NSB of at most $\ell$ sets
- there is a $\delta$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 $\mathcal{L}$ such that

- DFA → $\delta$NFA Minimization is NP-complete for DFAs accepting $\mathcal{L}$
- for each $L \in \mathcal{L}$, the minimal NFA for $L$ has one state less than the minimal $\delta$NFA for $L$
Putting Things Together

Theorem

Let $\mathcal{N}$ be a class of NFAs. If $\delta_{\text{NFA}} \subseteq \mathcal{N}$ then $\text{DFA} \rightarrow \mathcal{N}$ Minimization is $\text{NP}$-hard.

Proof.

We gave a reduction from Vertex Cover to $\text{DFA} \rightarrow \delta_{\text{NFA}}$ Minimization. Approximating Vertex Cover within a constant term is $\text{NP}$-complete $\Rightarrow$ $\text{DFA} \rightarrow \mathcal{N}$ Minimization is $\text{NP}$-hard.
Theorem

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

Proof.

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

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

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

Proof.

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

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

A decision algorithm for $\text{DFA} \rightarrow \mathcal{N}$ Minimization can approximate

$\text{DFA} \rightarrow \delta\text{NFA}$ Minimization within a term 1
Theorem

Let $\mathcal{N}$ be a class of NFAs.

If $\delta_{\text{NFA}} \subseteq \mathcal{N}$ then $\text{DFA} \rightarrow \mathcal{N}$ Minimization is $\text{NP}$-hard.

Proof.

We gave a reduction from Vertex Cover to $\text{DFA} \rightarrow \delta_{\text{NFA}}$ Minimization.

Let $\mathcal{N}$ be a class s.t. $\delta_{\text{NFA}} \subseteq \mathcal{N} \subseteq \text{NFA}$

A decision algorithm for $\text{DFA} \rightarrow \mathcal{N}$ Minimization can approximate $\text{DFA} \rightarrow \delta_{\text{NFA}}$ Minimization within a term 1.

The approximation for $\text{DFA} \rightarrow \mathcal{N}$ Minimization can be adapted to an approximation of Vertex Cover within a term 1.
**Theorem**

Let $\mathcal{N}$ be a class of NFAs.

If $\delta_{\text{NFA}} \subseteq \mathcal{N}$ then $\text{DFA} \rightarrow \mathcal{N}$ Minimization is \textbf{NP}-hard.

**Proof.**

We gave a reduction from Vertex Cover to $\text{DFA} \rightarrow \delta_{\text{NFA}}$ Minimization.

Let $\mathcal{N}$ be a class s.t. $\delta_{\text{NFA}} \subseteq \mathcal{N} \subseteq \text{NFA}$

A decision algorithm for $\text{DFA} \rightarrow \mathcal{N}$ Minimization can approximate $\text{DFA} \rightarrow \delta_{\text{NFA}}$ Minimization within a term 1.

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

Approximating Vertex Cover within a constant term is \textbf{NP}-complete.

$\Rightarrow$ $\text{DFA} \rightarrow \mathcal{N}$ Minimization is \textbf{NP}-hard.
Outline

1. Some Technical Details

2. Closer to Determinism?

3. Concluding Remarks
Are All Classes of non-DFAs hard to Minimize?

(non-DFAs: Classes $\mathcal{N}$ such that DFA $\subseteq \mathcal{N}$ but not $\mathcal{N} \subseteq$ DFA)
Are All Classes of non-DFAs hard to Minimize?

Answer

Of course not!

Example (Infinitely many classes between DFA and $\delta$NFA)

Take the class of DFAs, and add a single $\delta$NFA

$\Rightarrow$ 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 \(\mathbf{P}\)
- but the minimal δ’NFAs are the DFAs!
NFA can be minimized in \textbf{PTIME}

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

\[
\begin{align*}
q &\xrightarrow{a} q_1 \\
q &\xrightarrow{a} q_2
\end{align*}
\]

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

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

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)
\( \delta' \)NFA can be minimized in \textbf{PTIME}

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

\[
q \xrightarrow{a} q_1 \quad q \xrightarrow{a} q_2
\]

Let \( w \) be a string that leads \( A \) to \( q \)

As \( A \) is a \( \delta' \)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 \( \mathbf{P} \) algorithm
1. Some Technical Details

2. Closer to Determinism?

3. Concluding Remarks
Concluding Remarks

What did we do?

- State minimization is hard for all finite automata classes that include $\delta$NFA$s$
- One proof unifying and strengthening previous approaches
- The minimization tractability frontier is between $\delta$NFA and $\delta'$NFA
Concluding Remarks

Is everything solved yet?

- What we didn’t consider yet: fixed alphabet size
- What about approximations?
Thank you for listening