From NFA to minimal DFA

Bas Ploeger\(^1\)    Rob van Glabbeek\(^2\)    Jan Friso Groote\(^1\)

\(^1\) Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
The Netherlands

\(^2\) National ICT Australia
Sydney, Australia

ProSe, 19 October 2006
Overview

Introduction

Problem

Solution 1

Solution 2

Conclusions
Preliminaries

Finite automata

- NFA is a tuple $(S, \Sigma, \rightarrow, i, F)$
- DFA: every state has at most one outgoing $a$-transition for every $a \in \Sigma$
Preliminaries

Finite automata

- NFA is a tuple \((S, \Sigma, \rightarrow, i, F)\)
- DFA: every state has at most one outgoing \(a\)-transition for every \(a \in \Sigma\)

Language semantics

- Language of a state \(s\): \(L(s) = \{\sigma \in \Sigma^* \mid \exists f \in F . s \xrightarrow{\sigma} f\}\)
- Language preorder and equivalence on states \(s, s'\):
  - \(s \sqsubseteq_L s' \iff L(s) \subseteq L(s')\)
  - \(s \equiv_L s' \iff L(s) = L(s')\)
Canonization

Problem

Given an NFA, find the smallest, language equivalent DFA.
Problem
Given an NFA, find the smallest, language equivalent DFA.

Solution
1. Determinize NFA (subset construction)
2. Minimize DFA (Hopcroft)
Canonization

Problem
Given an NFA, find the smallest, language equivalent DFA.

Solution
1. Determinize NFA (subset construction) \textit{EXPTIME}
2. Minimize DFA (Hopcroft) \textit{PTIME}
Canonsization

Problem
Given an NFA, find the smallest, language equivalent DFA.

Solution
1. Determinize NFA (*subset construction*) EXPTIME
2. Minimize DFA (*Hopcroft*) PTIME

Minimal DFA can be exponentially larger than NFA
Example
Subset construction

NFA

DFA

\{0\}
Example
Subset construction

NFA

DFA
Example

Subset construction

NFA

DFA
Example
Subset construction

NFA

1 ---- a ----> 0
     \      |
      \     | a
       \   |    a, b
        3 ---- b -----> 2

DFA

{0} ---- a ----> {1, 2}
         | b ----> {2}
Example
Subset construction
Example

Subset construction

NFA

DFA
Example

Subset construction

NFA

DFA
Example
Minimization

NFA

DFA
Example
Minimization

NFA

DFA
Relevance to Process Theory (1)

Labelled Transition Systems

- Process modelled by LTS \((S, \Sigma, \rightarrow, i)\)
- No final states (computation “never” stops)
Relevance to Process Theory (1)

Labelled Transition Systems

- Process modelled by LTS \((S, \Sigma, \rightarrow, i)\)
- No final states (computation “never” stops)

Trace semantics

- Traces of a state \(s\): \(\text{Tr}(s) = \{\sigma \in \Sigma^* | \exists f \in S . s \xrightarrow{\sigma} f\}\)
- Trace equivalence: \(s \equiv_T s' \iff \text{Tr}(s) = \text{Tr}(s')\)
Relevance to Process Theory (1)

Labelled Transition Systems

- Process modelled by LTS $(S, \Sigma, \rightarrow, i)$
- No final states (computation “never” stops)

Trace semantics

- Traces of a state $s$: $\text{Tr}(s) = \{ \sigma \in \Sigma^* \mid \exists f \in S. s \xrightarrow{\sigma} f \}$
- Trace equivalence: $s \equiv_T s'$ if $\text{Tr}(s) = \text{Tr}(s')$

Bisimulation semantics

- Bisimulation is a relation $R$ on states satisfying, for $a \in \Sigma$:
  - if $s R t$ and $s \xrightarrow{a} s'$, then $\exists t'. t \xrightarrow{a} t'$ and $s' R t'$;
  - if $s R t$ and $t \xrightarrow{a} t'$, then $\exists s'. s \xrightarrow{a} s'$ and $s' R t'$;
- Bisimulation equivalence: $s \leftrightarrow s'$ if there exists a bisimulation $R$ with $s R s'$
Relevance to Process Theory (2)

Problem
Given an LTS, minimize it under trace semantics
Relevance to Process Theory (2)

Problem
Given an LTS, minimize it under trace semantics

Facts

- Deciding trace equivalence is PSPACE-complete
Relevance to Process Theory (2)

Problem
Given an LTS, minimize it under trace semantics

Facts
- Deciding trace equivalence is PSPACE-complete
- Deciding bisimulation equivalence is in PTIME
Relevance to Process Theory (2)

Problem
Given an LTS, minimize it under trace semantics

Facts
- Deciding trace equivalence is PSPACE-complete
- Deciding bisimulation equivalence is in PTIME
- If LTS is deterministic: $\equiv_T$ equals $\leftrightarrow$
Relevance to Process Theory (2)

Problem
Given an LTS, minimize it under trace semantics

Facts
- Deciding trace equivalence is PSPACE-complete
- Deciding bisimulation equivalence is in PTIME
- If LTS is deterministic: $\equiv_T$ equals $\leftrightarrow$

Solution

1. Determinize LTS (subset construction)
2. Minimize LTS under bisimulation semantics (Paige-Tarjan)
Problem

Overview

- NFA → DFA via subset construction
- DFA → mDFA via minimize

Questions
- What if DFA is much larger than mDFA?
- Can we avoid the generation of redundant states?
- Space efficiency (average case)
Problem

Overview

NFA \rightarrow \text{subset construction} \rightarrow \text{DFA} \rightarrow \text{minimize} \rightarrow \text{mDFA}

Questions

▶ What if DFA is much larger than mDFA?
Problem

Overview

NFA \xrightarrow{\text{subset construction}} \text{DFA} \xrightarrow{\text{minimize}} \text{mDFA}

Questions

▶ What if DFA is much larger than mDFA?
▶ Can we avoid the generation of redundant states?
Problem

Overview

Questions

▶ What if DFA is much larger than mDFA?
▶ Can we avoid the generation of redundant states?
Problem

Overview

NFA ? mDFA

Questions

➤ What if DFA is much larger than mDFA?
➤ Can we avoid the generation of redundant states?
➤ Space efficiency (average case)
Solution 1

Subset construction

Input: \( \mathcal{N} = (S_N, \Sigma_N, \rightarrow_N, i_N, F_N) \)

Output: \( \mathcal{D} = (S_D, \Sigma_D, \rightarrow_D, i_D, F_D) \)

Every DFA state \( P \in S_D \) is a set of NFA states
Solution 1

Subset construction

Input: \( \mathcal{N} = (S_N, \Sigma_N, \rightarrow_N, i_N, F_N) \)
Output: \( \mathcal{D} = (S_D, \Sigma_D, \rightarrow_D, i_D, F_D) \)

Every DFA state \( P \in S_D \) is a set of NFA states

Basic idea

- Add “irrelevant” NFA states to \( P \)
- State is irrelevant if it does not alter \( L(P) \)
Solution 1

Subset construction

Input: \( \mathcal{N} = (S_N, \Sigma_N, \rightarrow_N, i_N, F_N) \)

Output: \( \mathcal{D} = (S_D, \Sigma_D, \rightarrow_D, i_D, F_D) \)

Every DFA state \( P \in S_D \) is a set of NFA states

Basic idea

- Add “irrelevant” NFA states to \( P \)
- State is irrelevant if it does not alter \( L(P) \)

\( L(P) \)?
Solution 1

Subset construction

Input: \( \mathcal{N} = (S_N, \Sigma_N, \rightarrow_N, i_N, F_N) \)
Output: \( \mathcal{D} = (S_D, \Sigma_D, \rightarrow_D, i_D, F_D) \)

Every DFA state \( P \in S_D \) is a set of NFA states

Basic idea

- Add “irrelevant” NFA states to \( P \)
- State is irrelevant if it does not alter \( L(P) \)

\( L(P) \)?

- Language in NFA: \( L_N(P) = \bigcup_{p \in P} L_N(p) \)
Solution 1

Subset construction

Input: NFA $\mathcal{N} = (S_\mathcal{N}, \Sigma_\mathcal{N}, \rightarrow_\mathcal{N}, i_\mathcal{N}, F_\mathcal{N})$

Output: DFA $\mathcal{D} = (S_\mathcal{D}, \Sigma_\mathcal{D}, \rightarrow_\mathcal{D}, i_\mathcal{D}, F_\mathcal{D})$

Every DFA state $P \in S_\mathcal{D}$ is a set of NFA states

Basic idea

- Add “irrelevant” NFA states to $P$
- State is irrelevant if it does not alter $L(P)$

$L(P)$?

- Language in NFA: $L_\mathcal{N}(P) = \bigcup_{p \in P} L_\mathcal{N}(p)$
- Language in DFA: $L_\mathcal{D}(P)$, defined as usual
Solution 1

Subset construction

Input: \( \mathcal{N} = (S_N, \Sigma_N, \rightarrow_N, i_N, F_N) \)

Output: \( \mathcal{D} = (S_D, \Sigma_D, \rightarrow_D, i_D, F_D) \)

Every DFA state \( P \in S_D \) is a set of NFA states

Basic idea

- Add “irrelevant” NFA states to \( P \)
- State is irrelevant if it does not alter \( L(P) \)

\( L(P) ? \)

- Language in NFA: \( L_N(P) = \bigcup_{p \in P} L_N(p) \)
- Language in DFA: \( L_D(P) \), defined as usual
- Lemma: \( L_N(P) = L_D(P) \) for every \( P \subseteq S_N \)
Solution 1

Closure

▶ For any set $P \subseteq S_N$ define: $\overline{P} = \{ p \in S_N \mid L_N(p) \subseteq L_N(P) \}$
Solution 1

Closure

- For any set $P \subseteq S_N$ define: $\overline{P} = \{p \in S_N \mid p \sqsubseteq_L P\}$
Solution 1

Closure

- For any set $P \subseteq S_N$ define: $\overline{P} = \{ p \in S_N \mid p \sqsubseteq L P \}$
- Proposition: $P \equiv_L \overline{P}$ for any $P \subseteq S_N$
Solution 1

Closure

- For any set $P \subseteq S_N$ define: $\bar{P} = \{ p \in S_N \mid p \sqsubseteq_L P \}$
- Proposition: $P \equiv_L \bar{P}$ for any $P \subseteq S_N$

Algorithm

- Normal subset construction, but . . .
- Replace every generated set $P$ by $\bar{P}$
Solution 1

Closure

- For any set $P \subseteq S_N$ define: $\overline{P} = \{ p \in S_N \mid p \sqsubseteq_L P \}$
- Proposition: $P \equiv_L \overline{P}$ for any $P \subseteq S_N$

Algorithm

- Normal subset construction, but . . .
- Replace every generated set $P$ by $\overline{P}$

Main Theorem
Given an NFA, the algorithm constructs the minimal, language equivalent DFA
Example

NFA

0
(a|b)a∗b

1
a
a,b

2
a
a

3
{λ}

DFA

{0}
Example

NFA

DFA
Example

NFA

DFA
Example

NFA

DFA

\[(a|b)a^*b\]
Example

NFA

DFA

\[(a|b)a^*b\]

\[
\begin{align*}
\text{NFA} & : \\
0 & \rightarrow a, a,b \\
1 & \rightarrow a, a,b, a^*b \\
2 & \rightarrow a \\
3 & \rightarrow \{\lambda\} \\
\text{DFA} & : \\
\{0\} & \rightarrow a,b \\
\{1, 2\} & \rightarrow a,b
\end{align*}
\]
Example

NFA

DFA
Example

NFA

DFA

Complexity issues

Closure

- Language inclusion is PSPACE-complete
Complexity issues

Closure

- Language inclusion is PSPACE-complete

Simulation semantics

- Simulation is a relation $R$ on states satisfying, for $a \in \Sigma_N$:
  - if $s R t$ and $s \xrightarrow{a} s'$, then $\exists t'. t \xrightarrow{a} t'$ and $s' R t'$
  - if $s R t$ then $s \in F \Rightarrow t \in F$
- Simulation preorder: $s \subseteq s'$ if there exists a simulation $R$ with $s R s'$. 
Complexity issues

Closure

- Language inclusion is PSPACE-complete

Simulation semantics

- Simulation is a relation $R$ on states satisfying, for $a \in \Sigma_N$:
  - if $s R t$ and $s \xrightarrow{a} s'$, then $\exists t'. t \xrightarrow{a} t'$ and $s' R t'$
  - if $s R t$ then $s \in F \Rightarrow t \in F$

- Simulation preorder: $s \sqsubseteq s'$ if there exists a simulation $R$ with $s R s'$.

- Simulation is in PTIME
Implementation

Trade-off

- Use $\subseteq$ instead of $\subseteq_L$ in closure
Implementation

Trade-off

▶ Use $\subseteq$ instead of $\subseteq_L$ in closure
▶ Resulting DFA is not minimal
▶ But at most as large as the DFA produced by subset construction
Preliminary results

<table>
<thead>
<tr>
<th></th>
<th>NFA</th>
<th>min. DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>states</td>
<td>transitions</td>
</tr>
<tr>
<td>1</td>
<td>236</td>
<td>456</td>
</tr>
<tr>
<td>2</td>
<td>1.438</td>
<td>2.821</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>states</th>
<th>transitions</th>
<th>memory</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - normal</td>
<td></td>
<td></td>
<td>5.90 MB</td>
<td>0.200 sec</td>
</tr>
<tr>
<td>1 - solution 1</td>
<td></td>
<td></td>
<td>4.39 MB</td>
<td>0.844 sec</td>
</tr>
<tr>
<td>2 - normal</td>
<td></td>
<td></td>
<td>468.05 MB</td>
<td>72.468 sec</td>
</tr>
<tr>
<td>2 - solution 1</td>
<td></td>
<td></td>
<td>63.57 MB</td>
<td>561.022 sec</td>
</tr>
</tbody>
</table>
### Preliminary results

<table>
<thead>
<tr>
<th>NFA</th>
<th>min. DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>states</td>
</tr>
<tr>
<td>1</td>
<td>236</td>
</tr>
<tr>
<td>2</td>
<td>1.438</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DFA</th>
<th>states</th>
<th>transitions</th>
<th>memory</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - normal</td>
<td>42.665</td>
<td>83.416</td>
<td>5,90 MB</td>
<td>0,200 sec</td>
</tr>
<tr>
<td>1 - solution 1</td>
<td>4.712</td>
<td>9.252</td>
<td>4,39 MB</td>
<td>0,844 sec</td>
</tr>
<tr>
<td>2 - normal</td>
<td>7.403.224</td>
<td>14.616.424</td>
<td>468,05 MB</td>
<td>72,468 sec</td>
</tr>
<tr>
<td>2 - solution 1</td>
<td>176.105</td>
<td>348.005</td>
<td>63,57 MB</td>
<td>561,022 sec</td>
</tr>
</tbody>
</table>
Solution 2

Idea

- Why not remove irrelevant states from a set $P$?
Solution 2

Idea

- Why not remove irrelevant states from a set $P$?
- A $p \in P$ is irrelevant if: $\exists q \in P. p \neq q \land p \sqsubseteq_L q$
Solution 2

Idea

- Why not **remove** irrelevant states from a set $P$?
- A $p \in P$ is irrelevant if: $\exists Q \subseteq P. p \not\in Q \land p \sqsubseteq_L Q$
Solution 2

Idea

- Why not remove irrelevant states from a set $P$?
- A $p \in P$ is irrelevant if: $\exists Q \subseteq P. \ p \not\in Q \land p \sqsubseteq_{L} Q$

Better idea

- Replace $P$ by the set of transitions of the states in $P$:

$$T = \{(a, q) \in \Sigma \times S \mid \exists p \in P. \ p \xrightarrow{a} q\}$$
Solution 2

Idea

▶ Why not remove irrelevant states from a set $P$?
▶ A $p \in P$ is irrelevant if: $\exists Q \subseteq P . p \not\in Q \land p \sqsubseteq_L Q$

Better idea

▶ Replace $P$ by the set of transitions of the states in $P$:

$$T = \{(a, q) \in \Sigma \times S \mid \exists p \in P . p \xrightarrow{a} q\}$$

▶ Remove irrelevant transitions from $T$
▶ A $t \in T$ is irrelevant if: $\exists u \in T . t \neq u \land t \sqsubseteq_L u$
Definitions

**Language semantics**

For transitions $t = (a, q)$ and $u = (b, r)$:

- **Language of $t$:** $L(t) = \{ a\sigma \in \Sigma^+ | \sigma \in L(q) \}$
Definitions

Language semantics

For transitions \( t = (a, q) \) and \( u = (b, r) \):

- Language of \( t \): \( L(t) = \{ a\sigma \in \Sigma^+ \mid \sigma \in L(q) \} \)
- \( t \sqsubseteq_L u \iff a = b \land q \sqsubseteq_L r \)
Definitions

Language semantics

For transitions $t = (a, q)$ and $u = (b, r)$:

- Language of $t$: $L(t) = \{a\sigma \in \Sigma^+ \mid \sigma \in L(q)\}$
- $t \sqsubseteq_L u \iff a = b \land q \sqsubseteq_L r$

Compression

For any set of transitions $T$ define:

\[\downarrow T = \begin{cases} 
T & \text{if } \neg \exists t, u \in T \cdot t \neq u \land t \sqsubseteq_L u \\
\downarrow (T - \{t\}) & \text{where } t \in T \text{ and } \exists u \in T \cdot t \neq u \land t \sqsubseteq_L u, \text{ otherwise}
\end{cases}\]
Compression

Language equivalence

For a set of transitions \( T \):
  
  \( \text{Language of } T: \ L(T) = \bigcup_{t \in T} L(t) \)
Compression

Language equivalence

For a set of transitions $T$:

- Language of $T$: $L(T) = \bigcup_{t \in T} L(t)$
- Proposition: $T \equiv_L \downarrow T$
Compression

Language equivalence
For a set of transitions $T$:

- Language of $T$: $L(T) = \bigcup_{t \in T} L(t)$
- Proposition: $T \equiv_L \downarrow T$

Uniqueness?

- $\downarrow T$ is not unique for a given $T$
Compression

Language equivalence
For a set of transitions $T$:
- Language of $T$: $L(T) = \bigcup_{t \in T} L(t)$
- Proposition: $T \equiv_L \downarrow T$

Uniqueness?
- $\downarrow T$ is not unique for a given $T$
- $\downarrow T$ is unique if no two states in the NFA are language equivalent
Sets of transitions

Language equivalence

- For any state \( p \), \( T_p \) is the set of outgoing transitions of \( p \)
Sets of transitions

Language equivalence

- For any state \( p \), \( T_p \) is the set of outgoing transitions of \( p \)
- \( L(p) = L(T_p) \)
Sets of transitions

Language equivalence

- For any state $p$, $T_p$ is the set of outgoing transitions of $p$
- $L(p) = L(T_p) \cup \begin{cases} \{ \lambda \} & \text{if } p \in F \\ \emptyset & \text{if } p \notin F \end{cases}$
Sets of transitions

Language equivalence

- For any state \( p \), \( T_p \) is the set of outgoing transitions of \( p \)
- \( L(p) = L(T_p) \cup \begin{cases} \{\lambda\} & \text{if } p \in F \\ \emptyset & \text{if } p \notin F \end{cases} \)
- For any set of states \( P \): \( L(P) = \bigcup_{p \in P} L(p) \)
Sets of transitions

Language equivalence

▶ For any state $p$, $T_p$ is the set of outgoing transitions of $p$

▶ $L(p) = L(T_p) \cup \begin{cases} \{\lambda\} & \text{if } p \in F \\ \emptyset & \text{if } p \notin F \end{cases}$

▶ For any set of states $P$:

$$L(P) = \bigcup_{p \in P} L(T_p) \cup \begin{cases} \{\lambda\} & \text{if } p \in F \\ \emptyset & \text{if } p \notin F \end{cases}$$
Sets of transitions

Language equivalence

- For any state \( p \), \( T_p \) is the set of outgoing transitions of \( p \)
- \( L(p) = L(T_p) \cup \begin{cases} \{\lambda\} & \text{if } p \in F \\ \emptyset & \text{if } p \notin F \end{cases} \)

- For any set of states \( P \):
  \[
  L(P) = \bigcup_{p \in P} L(T_p) \cup \begin{cases} \{\lambda\} & \text{if } p \in F \\ \emptyset & \text{if } p \notin F \end{cases}
  \]

DFA state

- A set of transitions \( T \)
- \( T \) does not determine whether the state is final
- Add a boolean
Solution 2

Algorithm

- Minimize NFA under language semantics
- Subset construction; DFA states are tuples \((T, b)\)
- Replace every \((T, b)\) by \((\downarrow T, b)\)
Solution 2

Algorithm

- Minimize NFA under language semantics
- Subset construction; DFA states are tuples \((T, b)\)
- Replace every \((T, b)\) by \((\downarrow T, b)\)

Implementation

- Minimize NFA under \textit{simulation} semantics
- Use \(\subseteq\) for compression
Conclusions

Done:

- Two algorithms for NFA $\rightarrow$ minimal DFA
- Aim: improve space efficiency (average case)
Conclusions

Done:

- Two algorithms for NFA $\rightarrow$ minimal DFA
- Aim: improve space efficiency (average case)
- Implementation trade-off: simulation semantics
- Suboptimal results

To do:

- Implement Solution 2
- Investigate performance gain in practice (benchmarking!)
- Compare to other tools
- Finish paper
Conclusions

Done:

- Two algorithms for NFA $\rightarrow$ minimal DFA
- Aim: improve space efficiency (average case)
- Implementation trade-off: simulation semantics
- Suboptimal results

To do:

- Implement Solution 2
Conclusions

Done:
- Two algorithms for NFA → minimal DFA
- Aim: improve space efficiency (average case)
- Implementation trade-off: simulation semantics
- Suboptimal results

To do:
- Implement Solution 2
- Investigate performance gain in practice (benchmarking!)
- Compare to other tools
Conclusions

Done:

- Two algorithms for NFA → minimal DFA
- Aim: improve space efficiency (average case)
- Implementation trade-off: simulation semantics
- Suboptimal results

To do:

- Implement Solution 2
- Investigate performance gain in practice (benchmarking!)
- Compare to other tools
- Finish paper