

From NFA to minimal DFA

Bas Ploeger¹ Rob van Glabbeek² Jan Friso Groote¹

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' :

$$s \sqsubseteq_L s' \Leftrightarrow L(s) \subseteq L(s')$$

$$s \equiv_L s' \Leftrightarrow L(s) = L(s')$$

Canonization

Problem

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

Canonization

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*) **EXPTIME**
2. Minimize DFA (*Hopcroft*) **PTIME**

Canonization

Problem

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

Solution

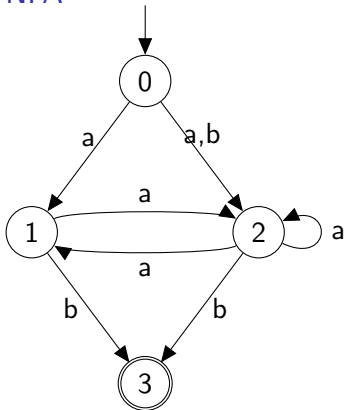
1. Determinize NFA (*subset construction*) **EXPTIME**
2. Minimize DFA (*Hopcroft*) **PTIME**

Minimal DFA can be exponentially larger than NFA

Example

Subset construction

NFA



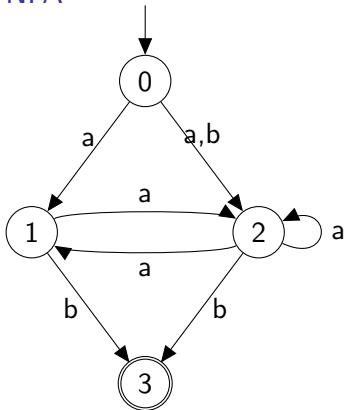
DFA



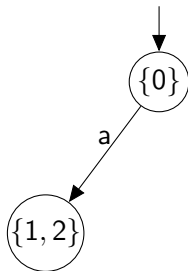
Example

Subset construction

NFA



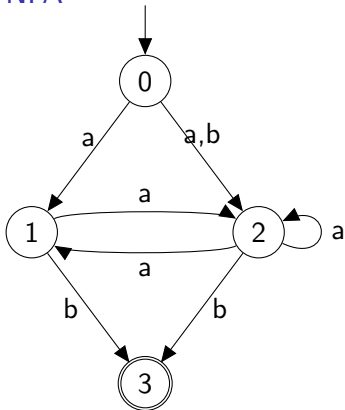
DFA



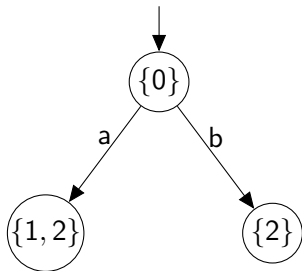
Example

Subset construction

NFA



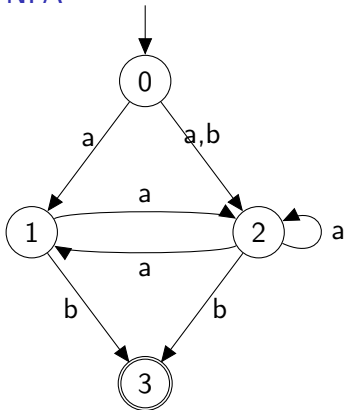
DFA



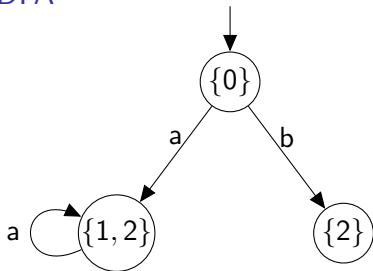
Example

Subset construction

NFA



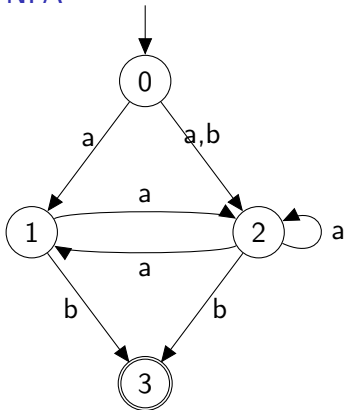
DFA



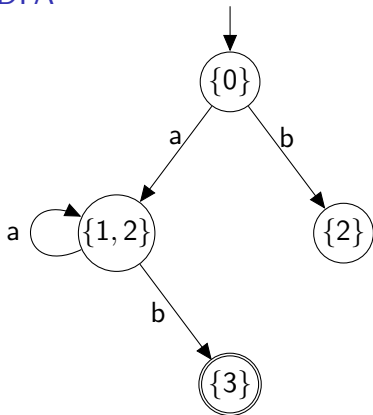
Example

Subset construction

NFA



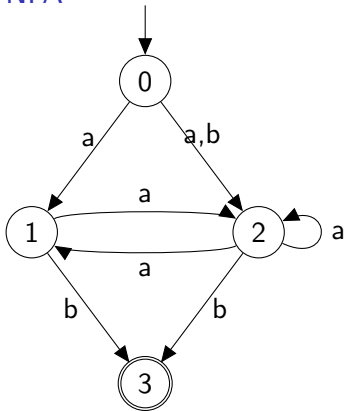
DFA



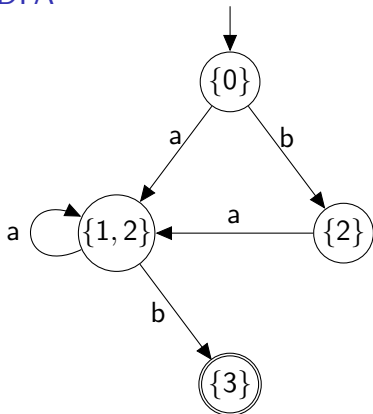
Example

Subset construction

NFA



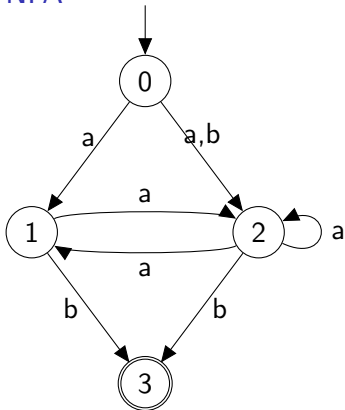
DFA



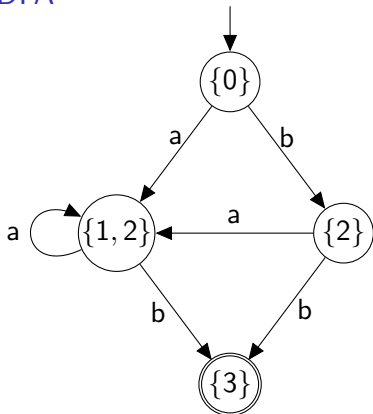
Example

Subset construction

NFA



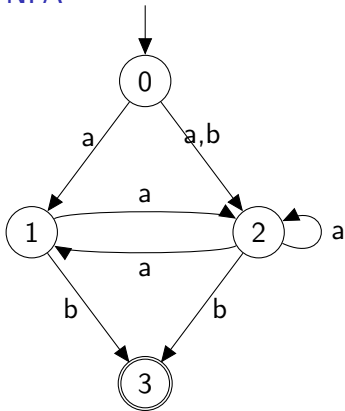
DFA



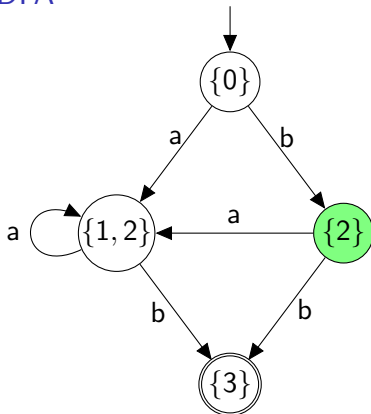
Example

Minimization

NFA



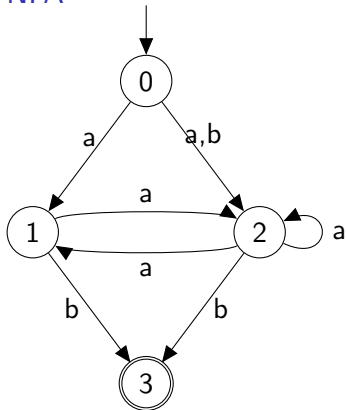
DFA



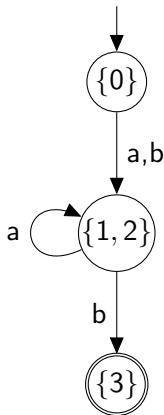
Example

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 : $\text{Tr}(s) = \{\sigma \in \Sigma^* \mid \exists f \in S. s \xrightarrow{\sigma} f\}$
- ▶ 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: $s \equiv_T s' \iff \text{Tr}(s) = \text{Tr}(s')$

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**: $\equiv_{\mathcal{T}}$ equals \leftrightarrow

Relevance to Process Theory (2)

Problem

Given an LTS, minimize it under trace semantics

Facts

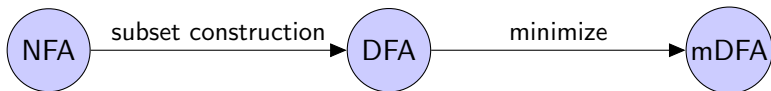
- ▶ Deciding trace equivalence is PSPACE-complete
- ▶ Deciding bisimulation equivalence is in PTIME
- ▶ If LTS is **deterministic**: $\equiv_{\mathcal{T}}$ equals \leftrightarrow

Solution

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

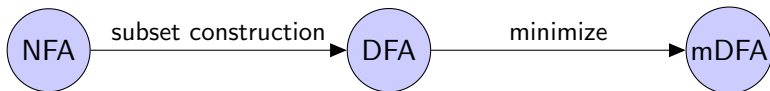
Problem

Overview



Problem

Overview

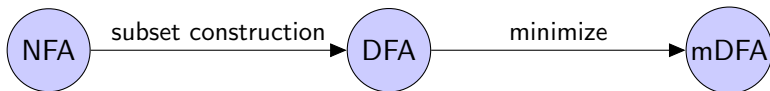


Questions

- ▶ What if DFA is much larger than mDFA?

Problem

Overview

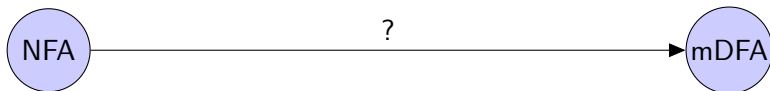


Questions

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

Problem

Overview

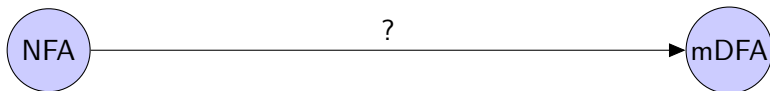


Questions

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

Problem

Overview



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: 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

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 $L(P)$

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 $L(P)$

$L(P)$?

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: $L_{\mathcal{N}}(P) = \bigcup_{p \in P} L_{\mathcal{N}}(p)$

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: 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
- ▶ Lemma: $L_{\mathcal{N}}(P) = L_{\mathcal{D}}(P)$ for every $P \subseteq S_{\mathcal{N}}$

Solution 1

Closure

- ▶ For any set $P \subseteq S_{\mathcal{N}}$ define: $\overline{P} = \{p \in S_{\mathcal{N}} \mid L_{\mathcal{N}}(p) \subseteq L_{\mathcal{N}}(P)\}$

Solution 1

Closure

- ▶ For any set $P \subseteq S_{\mathcal{N}}$ define: $\overline{P} = \{p \in S_{\mathcal{N}} \mid p \sqsubseteq_L P\}$

Solution 1

Closure

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

Solution 1

Closure

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

Algorithm

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

Solution 1

Closure

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

Algorithm

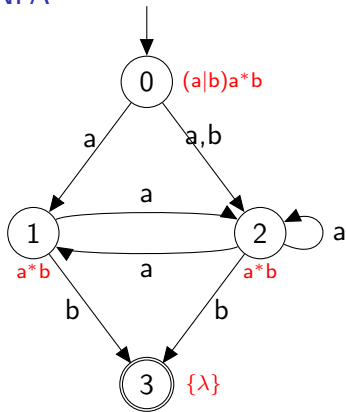
- ▶ Normal subset construction, but ...
- ▶ Replace every generated set P by \bar{P}

Main Theorem

Given an NFA, the algorithm constructs the minimal, language equivalent DFA

Example

NFA

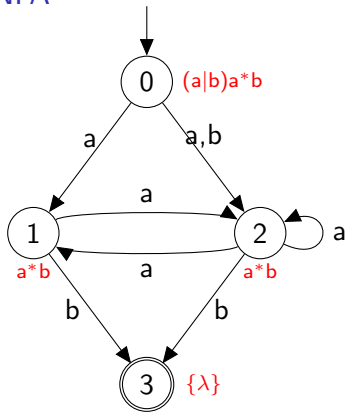


DFA

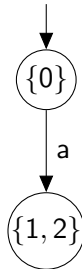


Example

NFA

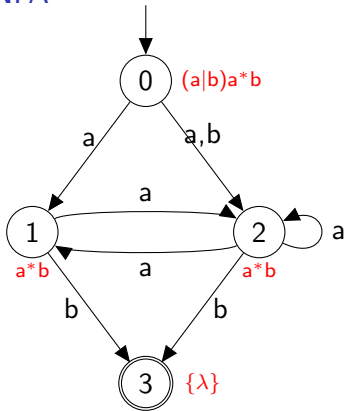


DFA

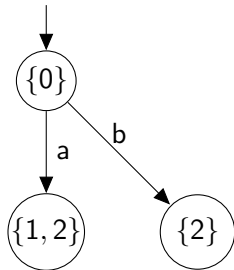


Example

NFA

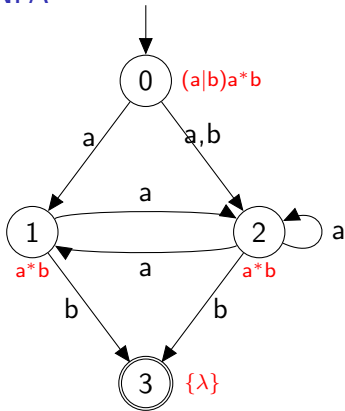


DFA

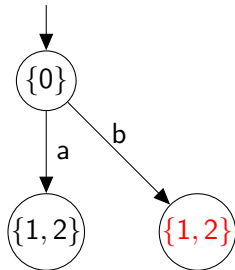


Example

NFA

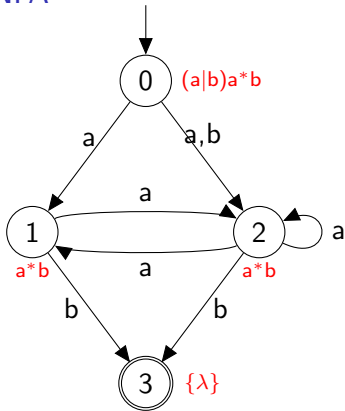


DFA

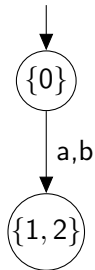


Example

NFA

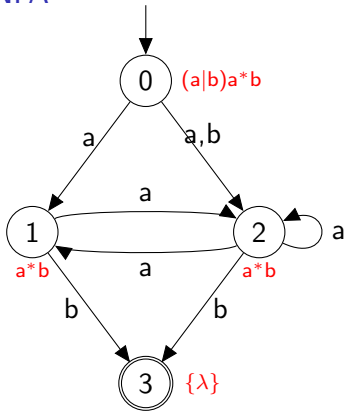


DFA

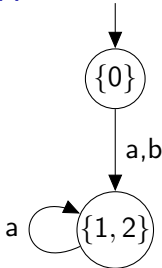


Example

NFA

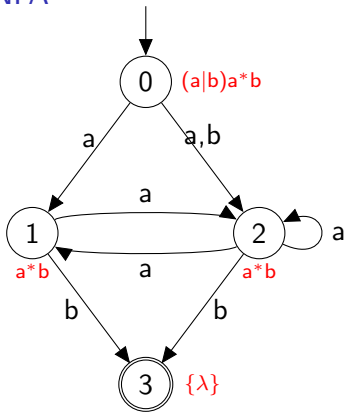


DFA

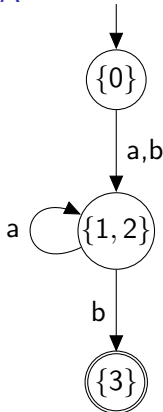


Example

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 $a \in \Sigma_{\mathcal{N}}$:
 - ▶ 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 $s R s'$.

Complexity issues

Closure

- ▶ Language inclusion is PSPACE-complete

Simulation semantics

- ▶ Simulation is a relation R on states satisfying, for $a \in \Sigma_{\mathcal{N}}$:
 - ▶ 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

	NFA		min. DFA	
	states	transitions	states	transitions
1	236	456	1.367	2.690
2	1.438	2.821	18.925	37.615

Preliminary results

	NFA		min. DFA	
	states	transitions	states	transitions
1	236	456	1.367	2.690
2	1.438	2.821	18.925	37.615

	DFA		memory	time
	states	transitions		
1 - normal	42.665	83.416	5,90 MB	0,200 sec
1 - solution 1	4.712	9.252	4,39 MB	0,844 sec
2 - normal	7.403.224	14.616.424	468,05 MB	72,468 sec
2 - solution 1	176.105	348.005	63,57 MB	561,022 sec

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: $\exists q \in P . p \neq q \wedge p \sqsubseteq_L q$

Solution 2

Idea

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

Solution 2

Idea

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

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: $\exists Q \subseteq P . p \notin Q \wedge p \sqsubseteq_L Q$

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: $\exists u \in T . t \neq u \wedge t \sqsubseteq_L u$

Definitions

Language semantics

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

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

Definitions

Language semantics

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

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

Definitions

Language semantics

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

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

Compression

For any set of transitions T define:

$$\downarrow T = \begin{cases} T & \text{if } \neg \exists t, u \in T. t \neq u \wedge t \sqsubseteq_L u \\ \downarrow(T - \{t\}) & \text{where } t \in T \text{ and } \exists u \in T. t \neq u \wedge t \sqsubseteq_L u, \\ & \text{otherwise} \end{cases}$$

Compression

Language equivalence

For a set of transitions T :

- ▶ 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 **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

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 \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

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