Automata Theory :: Minimal DFAs

Jörg Endrullis

Vrije Universiteit Amsterdam

Minimal DFAs (Hopcroft, 1971)

Goal

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$.

Construct the (unique) **minimal DFA** $\widehat{M}$ with $L(M) = L(\widehat{M})$.

(Here minimal is with respect to the number of states.)

Minimal DFAs (Hopcroft, 1971)

Goal

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$.

Construct the (unique) **minimal DFA** $\widehat{M}$ with $L(M) = L(\widehat{M})$.

(Here minimal is with respect to the number of states.)

Construction

Step 1: Remove all unreachable states from M.

Step 2: Partition Q in indistinguishable states.

Step 3: Read off the minimal DFA.

Minimal DFAs (Hopcroft, 1971)

Goal

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$.

Construct the (unique) **minimal DFA** $\widehat{M}$ with $L(M) = L(\widehat{M})$.

(Here minimal is with respect to the number of states.)

Construction

Step 1: Remove all unreachable states from M.

Step 2: Partition Q in indistinguishable states.

Step 3: Read off the minimal DFA.

Step 1: Remove all unreachable states from M.

Remove all states $q \in Q$ for which there is no path from q_0 to q.

Minimal DFAs (2)

States $q_1, q_2 \in Q$ are **distinguishable** if there exists $w \in \Sigma^*$ s.t.

$$q_1 \xrightarrow{w}\!\!\!\!\twoheadrightarrow q_1' \in F \qquad\qquad q_2 \xrightarrow{w}\!\!\!\!\twoheadrightarrow q_2' \not\in F \, ,$$

or vice versa.

Minimal DFAs (2)

States $q_1, q_2 \in Q$ are **distinguishable** if there exists $w \in \Sigma^*$ s.t.

$$q_1 \xrightarrow{w}\!\!\!\rightarrow q_1' \in F \qquad\qquad q_2 \xrightarrow{w}\!\!\!\rightarrow q_2' \notin F \ ,$$

or vice versa.

Step 2: Partition Q in indistinguishable states.

We construct the partitioning stepwise:

Minimal DFAs (2)

States $q_1, q_2 \in Q$ are **distinguishable** if there exists $w \in \Sigma^*$ s.t.

$$q_1 \xrightarrow{w}\!\!\!\twoheadrightarrow q_1' \in F \qquad\qquad q_2 \xrightarrow{w}\!\!\!\twoheadrightarrow q_2' \notin F \ ,$$

or vice versa.

Step 2: Partition Q in indistinguishable states.

We construct the partitioning stepwise:

- Initial partitioning is $\{\, Q \setminus F,\ F \,\}$.

Minimal DFAs (2)

States $q_1, q_2 \in Q$ are **distinguishable** if there exists $w \in \Sigma^*$ s.t.

$$q_1 \xrightarrow{w}\!\!\!\!\twoheadrightarrow q_1' \in F \qquad\qquad q_2 \xrightarrow{w}\!\!\!\!\twoheadrightarrow q_2' \notin F \ ,$$

or vice versa.

Step 2: Partition Q in indistinguishable states.

We construct the partitioning stepwise:

- Initial partitioning is $\{\, Q \setminus F,\ F \,\}$.
- If there are partitions R and S such that

$$\delta(q, a) \in S \qquad \text{and} \qquad \delta(q', a) \notin S,$$

  for some $a \in \Sigma$ and $q, q' \in R$, then we split R in

$$\{\, q \in R \mid \delta(q, a) \in S \,\} \qquad \{\, q \in R \mid \delta(q, a) \notin S \,\}$$

Minimal DFAs (2)

States $q_1, q_2 \in Q$ are **distinguishable** if there exists $w \in \Sigma^*$ s.t.

$$q_1 \xrightarrow{w}\!\!\!\!\rightarrow q_1' \in F \qquad\qquad q_2 \xrightarrow{w}\!\!\!\!\rightarrow q_2' \notin F \ ,$$

or vice versa.

Step 2: Partition Q in indistinguishable states.

We construct the partitioning stepwise:

- Initial partitioning is $\{ Q \setminus F,\ F \}$.
- If there are partitions R and S such that

  $$\delta(q, a) \in S \qquad \text{and} \qquad \delta(q', a) \notin S,$$

  for some $a \in \Sigma$ and $q, q' \in R$, then we split R in

  $$\{ q \in R \mid \delta(q, a) \in S \} \qquad \{ q \in R \mid \delta(q, a) \notin S \}$$

We keep splitting until no more split is possible.

Minimal DFAs (3)

Step 3: Read off the minimal DFA.

Minimal DFAs (3)

Step 3: Read off the minimal DFA.

Let $Q_1, \ldots, Q_n$ be the final partition of Q.

Minimal DFAs (3)

Step 3: Read off the minimal DFA.

Let $Q_1, \ldots, Q_n$ be the final partition of Q.

These are the states of the minimal DFA $\widehat{M}$.

Minimal DFAs (3)

Step 3: Read off the minimal DFA.

Let $Q_1, \ldots, Q_n$ be the final partition of Q.

These are the states of the minimal DFA $\widehat{M}$.

The transitions (arrows) of $\widehat{M}$ are:

$$Q_i \xrightarrow{a} Q_j \quad \iff \quad q \xrightarrow{a} q' \text{ for some } q \in Q_i,\ q' \in Q_j$$

Minimal DFAs (3)

Step 3: Read off the minimal DFA.

Let $Q_1, \ldots, Q_n$ be the final partition of Q.

These are the states of the minimal DFA $\widehat{M}$.

The transitions (arrows) of $\widehat{M}$ are:

$$Q_i \xrightarrow{a} Q_j \quad \iff \quad q \xrightarrow{a} q' \text{ for some } q \in Q_i,\ q' \in Q_j$$

The starting state is the set that contains q_0.

Minimal DFAs (3)

Step 3: Read off the minimal DFA.

Let $Q_1, \ldots, Q_n$ be the final partition of Q.

These are the states of the minimal DFA $\widehat{M}$.

The transitions (arrows) of $\widehat{M}$ are:

$$Q_i \xrightarrow{a} Q_j \quad\iff\quad q \xrightarrow{a} q' \text{ for some } q \in Q_i,\ q' \in Q_j$$

The starting state is the set that contains q_0.

The final states are the subsets of F.

Minimal DFAs (3)

Step 3: Read off the minimal DFA.

Let $Q_1, \ldots, Q_n$ be the final partition of Q.

These are the states of the minimal DFA $\widehat{M}$.

The transitions (arrows) of $\widehat{M}$ are:

$$Q_i \xrightarrow{a} Q_j \quad \Longleftrightarrow \quad q \xrightarrow{a} q' \text{ for some } q \in Q_i,\, q' \in Q_j$$

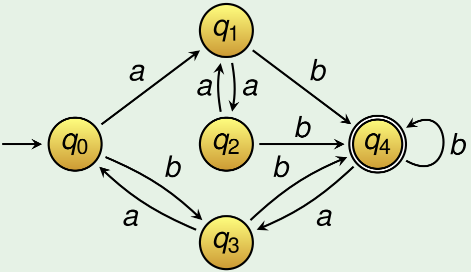
The starting state is the set that contains q_0.

The final states are the subsets of F.

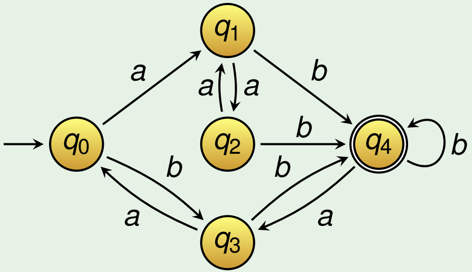
Worst-case time complexity: $O(|\Sigma| \cdot |Q|^2)$, since

- There are maximal $|Q| - 1$ splits.
- Every split costs maximal $O(|\Sigma| \cdot |Q|)$.

Exercise: DFA Minimisation

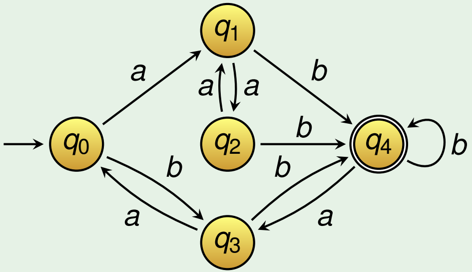


Exercise: DFA Minimisation



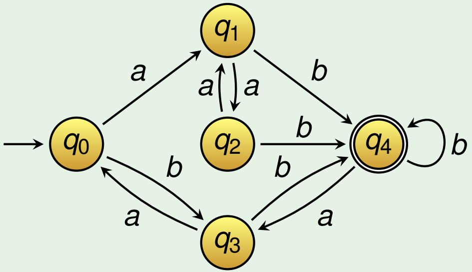
1. All states are reachable (nothing to remove).

Exercise: DFA Minimisation



1. All states are reachable (nothing to remove).
2. Initial partitioning: $\{Q \setminus F, F\} = \{\{q_0, q_1, q_2, q_3\}, \{q_4\}\}$

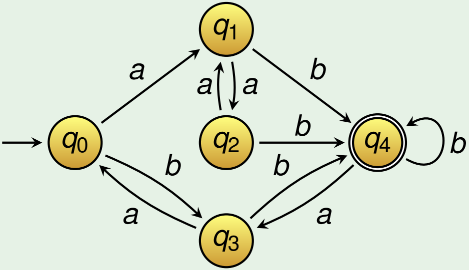
Exercise: DFA Minimisation



1. All states are reachable (nothing to remove).
2. Initial partitioning: $\{\, Q \setminus F, F \,\} = \{\{\, q_0, q_1, q_2, q_3 \,\}, \{\, q_4 \,\}\}$

   Splitting $R = \{\, q_0, q_1, q_2, q_3 \,\}$ with $S = \{\, q_4 \,\}$ and letter $b \in \Sigma$.

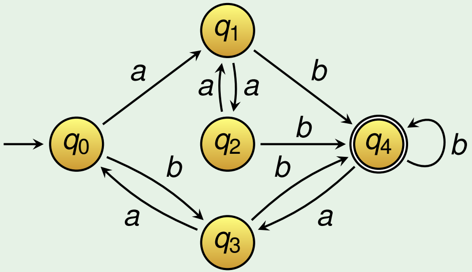
Exercise: DFA Minimisation



1. All states are reachable (nothing to remove).
2. Initial partitioning: $\{ Q \setminus F, F \} = \{\{ q_0, q_1, q_2, q_3 \}, \{ q_4 \}\}$

   Splitting $R = \{ q_0, q_1, q_2, q_3 \}$ with $S = \{ q_4 \}$ and letter $b \in \Sigma$.
   New partitioning: $\{\{ q_0 \}, \{ q_1, q_2, q_3 \}, \{ q_4 \}\}$.

Exercise: DFA Minimisation

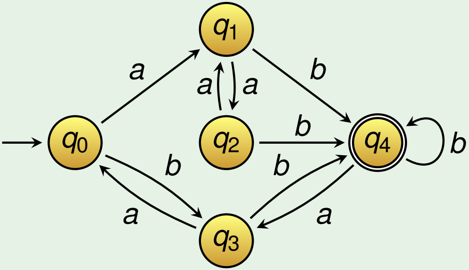


1. All states are reachable (nothing to remove).
2. Initial partitioning: $\{ Q \setminus F, F \} = \{\{ q_0, q_1, q_2, q_3 \},\{ q_4 \}\}$

   Splitting $R = \{ q_0, q_1, q_2, q_3 \}$ with $S = \{ q_4 \}$ and letter $b \in \Sigma$.
   New partitioning: $\{\{ q_0 \},\{ q_1, q_2, q_3 \},\{ q_4 \}\}$.

   Splitting $R = \{ q_1, q_2, q_3 \}$ with $S = \{ q_0 \}$ and letter $a \in \Sigma$.

Exercise: DFA Minimisation

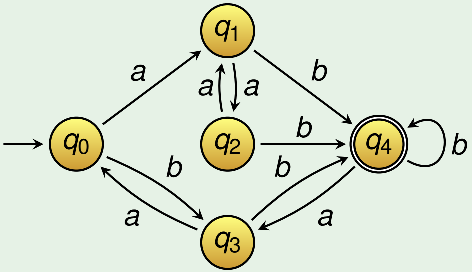


1. All states are reachable (nothing to remove).
2. Initial partitioning: $\{Q \setminus F, F\} = \{\{q_0, q_1, q_2, q_3\}, \{q_4\}\}$

   Splitting $R = \{q_0, q_1, q_2, q_3\}$ with $S = \{q_4\}$ and letter $b \in \Sigma$.
   New partitioning: $\{\{q_0\}, \{q_1, q_2, q_3\}, \{q_4\}\}$.

   Splitting $R = \{q_1, q_2, q_3\}$ with $S = \{q_0\}$ and letter $a \in \Sigma$.
   New partitioning: $\{\{q_0\}, \{q_1, q_2\}, \{q_3\}, \{q_4\}\}$.

Exercise: DFA Minimisation



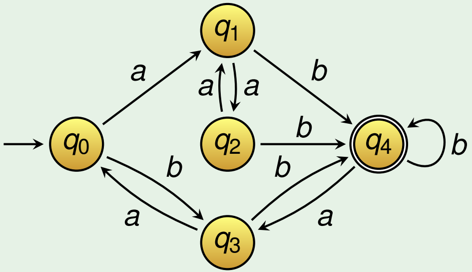
1. All states are reachable (nothing to remove).
2. Initial partitioning: $\{ Q \setminus F, F \} = \{\{ q_0, q_1, q_2, q_3 \},\{ q_4 \}\}$

   Splitting $R = \{ q_0, q_1, q_2, q_3 \}$ with $S = \{ q_4 \}$ and letter $b \in \Sigma$.
   New partitioning: $\{\{ q_0 \},\{ q_1, q_2, q_3 \},\{ q_4 \}\}$.

   Splitting $R = \{ q_1, q_2, q_3 \}$ with $S = \{ q_0 \}$ and letter $a \in \Sigma$.
   New partitioning: $\{\{ q_0 \},\{ q_1, q_2 \},\{ q_3 \},\{ q_4 \}\}$.

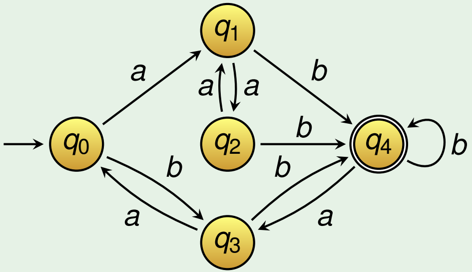
   Nothing more to split!

Exercise: DFA Minimisation

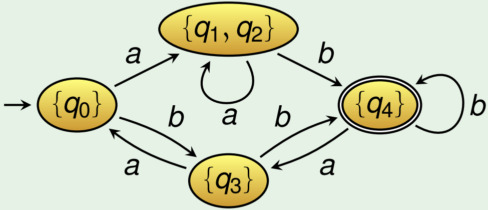


2. Final partitioning: $\{\{q_0\},\{q_1, q_2\},\{q_3\},\{q_4\}\}$.

Exercise: DFA Minimisation



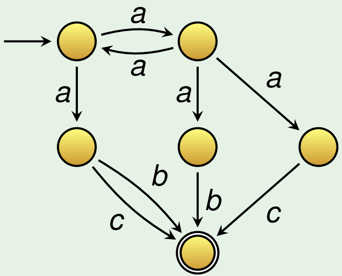
2. Final partitioning: $\{\{q_0\}, \{q_1, q_2\}, \{q_3\}, \{q_4\}\}$.
3. Reading off the minimal DFA:

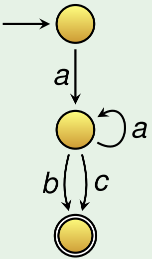


Minimising of NFAs

Minimising of NFAs is very difficult.

Example

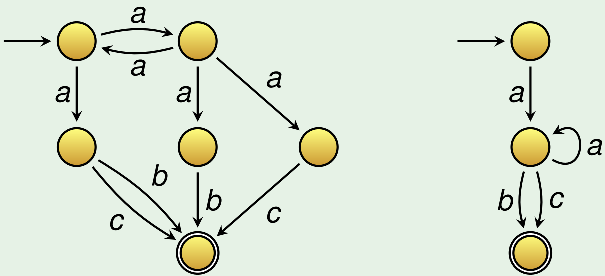




Minimising of NFAs

Minimising of NFAs is very difficult.

Example



Theorem

Minimising of NFAs is PSpace-complete.

The definition of PSpace-complete follows later.

Lexical Analysis

Lexical Analysis

Lexical analysis converts a sequence of **characters** into a sequence of **tokens**.

Programs that do lexical analysis are **lexers** or **tokenizers**.

Lexical Analysis

Lexical analysis converts a sequence of **characters** into a sequence of **tokens**.

Programs that do lexical analysis are **lexers** or **tokenizers**.

For example the expression

```
sum = 15 + 2
```

could be converted to the sequence of tokens

token	token category
`sum`	identifier
`=`	assignment
`15`	integer literal
`+`	operator
`2`	integer literal

Allows to write parsers on the more abstract level of tokens.

Lexical Analysis

How to get from characters to tokens?

Lexical Analysis

How to get from characters to tokens?

- Regular expressions $r_1, \ldots, r_n$ express the pattern.

  Every regular expression corresponds to a token.

Lexical Analysis

How to get from characters to tokens?

- Regular expressions $r_1, \ldots, r_n$ express the pattern.

  Every regular expression corresponds to a token.

- Lexical analysis repeatedly searches the **longest prefix** of the input that is matched by one of the regular expressions. This prefix is transformed into a token.

Lexical Analysis

How to get from characters to tokens?

- Regular expressions $r_1, \ldots, r_n$ express the pattern.

  Every regular expression corresponds to a token.
- Lexical analysis repeatedly searches the **longest prefix** of the input that is matched by one of the regular expressions. This prefix is transformed into a token.

For **improved performance**:

- Regular expressions are translated **minimal DFAs**.

Lexical Analysis

How to get from characters to tokens?

- Regular expressions $r_1, \dots, r_n$ express the pattern.

  Every regular expression corresponds to a token.

- Lexical analysis repeatedly searches the **longest prefix** of the input that is matched by one of the regular expressions. This prefix is transformed into a token.

For **improved performance**:

- Regular expressions are translated **minimal DFAs**.

Parser/lexer generators like

- JavaCC
- LEX

generate the lexer automatically. Thereby regular expressions or grammars are converted to minimal DFAs.