Languages and Machines (Module 7 TCS+IAM)

L&M 3: Regular Languages, Pumping Lemma, Minimisation

Ch 5:7, Ch 6:1,4-6.

Alexander Skopalik

Discrete Mathematics and Mathematical Programming, Applied Mathematics, University of Twente

Lecture 3

Contents

Contents

Loops in DFAs

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

- Suppose:
  - DFA M has 100 states
  - M accepts a word w with 100 symbols
- Then M accepts infinitely many words.

Loops in DFAs

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

- Suppose:
  - DFA M has 100 states
  - M accepts a word w with 100 symbols
- Then M accepts infinitely many words.

After all:

- 100 symbols pass through 101 states; there are only 100

Loops in DFAs

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

- Suppose:
  - DFA M has 100 states
  - M accepts a word w with 100 symbols
- Then M accepts infinitely many words.

After all:

- 100 symbols pass through 101 states; there are only 100
- Pigeon hole principle: at least 1 state is visited twice

Loops in DFAs

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

- Suppose:
  - DFA M has 100 states
  - M accepts a word w with 100 symbols
- Then M accepts infinitely many words.

After all:

- 100 symbols pass through 101 states; there are only 100
- Pigeon hole principle: at least 1 state is visited twice
- w visits some state q at least twice

Loops in DFAs

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

- Suppose:
  - DFA M has 100 states
  - M accepts a word w with 100 symbols
- Then M accepts infinitely many words.

After all:

- 100 symbols pass through 101 states; there are only 100
- Pigeon hole principle: at least 1 state is visited twice
- w visits some state q at least twice
- Then w can be split into $w = xyz$, $y \neq \lambda$, such that

Loops in DFAs

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

- Suppose:
  - DFA M has 100 states
  - M accepts a word w with 100 symbols
- Then M accepts infinitely many words.

After all:

- 100 symbols pass through 101 states; there are only 100
- Pigeon hole principle: at least 1 state is visited twice
- w visits some state q at least twice
- Then w can be split into $w = xyz$, $y \neq \lambda$, such that
- $\hat{\delta}(q_0, x) = q$, $\hat{\delta}(q, y) = q$, and $\hat{\delta}(q, z) \in F$

Loops in DFAs

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

- Suppose:
  - DFA M has 100 states
  - M accepts a word w with 100 symbols
- Then M accepts infinitely many words.

After all:

- 100 symbols pass through 101 states; there are only 100
- Pigeon hole principle: at least 1 state is visited twice
- w visits some state q at least twice
- Then w can be split into $w = xyz$, $y \neq \lambda$, such that
- $\hat{\delta}(q_0, x) = q$, $\hat{\delta}(q, y) = q$, and $\hat{\delta}(q, z) \in F$
- But then also $xy^i z \in \mathcal{L}(M)$, for every $i \geq 0$

Loops in DFAs

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA

- Suppose:
  - DFA M has 100 states
  - M accepts a word w with 100 symbols
- Then M accepts infinitely many words.

After all:

- 100 symbols pass through 101 states; there are only 100
- Pigeon hole principle: at least 1 state is visited twice
- w visits some state q at least twice
- Then w can be split into $w = xyz$, $y \neq \lambda$, such that
- $\hat{\delta}(q_0, x) = q$, $\hat{\delta}(q, y) = q$, and $\hat{\delta}(q, z) \in F$
- But then also $xy^i z \in \mathcal{L}(M)$, for every $i \geq 0$
- So $\mathcal{L}(M)$ is indeed infinite.

The Pumping lemma for Regular Languages

Pumping lemma

If L is a regular language, then
there exists a number $k > 0$, such that
for every word $z \in L$ with $|z| \geq k$:
there exist words u, v, w, such that

1. $z = uvw$; and
2. $|uv| \leq k$; and
3. $|v| > 0$; and
4. for every $i \geq 0$, $uv^i w \in L$,.

Proof of the Pumping lemma

$\forall L \in \mathbf{REG} : \exists k > 0 : \forall z \in L, |z| \geq k : \exists u, v, w : \forall i \cdots$

- Let L be an arbitrary regular language.

Proof of the Pumping lemma

$\forall L \in \mathbf{REG} : \exists k > 0 : \forall z \in L, |z| \geq k : \exists u, v, w : \forall i \cdots$

- Let L be an arbitrary regular language.
- Then a DFA $M = (Q, \Sigma, \delta, q_0, F)$ exists with $\mathcal{L}(M) = L$.

Proof of the Pumping lemma

$\forall L \in \mathbf{REG} : \exists k > 0 : \forall z \in L, |z| \geq k : \exists u, v, w : \forall i \cdots$

- Let L be an arbitrary regular language.
- Then a DFA $M = (Q, \Sigma, \delta, q_0, F)$ exists with $\mathcal{L}(M) = L$.
- Let $k := |Q|$, the number of elements Q (indeed $k > 0$)

Proof of the Pumping lemma

$\forall L \in \mathbf{REG} : \exists k > 0 : \forall z \in L, |z| \geq k : \exists u, v, w : \forall i \cdots$

- Let L be an arbitrary regular language.
- Then a DFA $M = (Q, \Sigma, \delta, q_0, F)$ exists with $\mathcal{L}(M) = L$.
- Let $k := |Q|$, the number of elements Q (indeed $k > 0$)
- Let an arbitrary $z \in L$ with $|z| \geq k$ be given.

Proof of the Pumping lemma

$\forall L \in \mathbf{REG} : \exists k > 0 : \forall z \in L, |z| \geq k : \exists u, v, w : \forall i \cdots$

- Let L be an arbitrary regular language.
- Then a DFA $M = (Q, \Sigma, \delta, q_0, F)$ exists with $\mathcal{L}(M) = L$.
- Let $k := |Q|$, the number of elements Q (indeed $k > 0$)
- Let an arbitrary $z \in L$ with $|z| \geq k$ be given.
- z follows a path in M of $k + 1$ states, so this paths visits a certain q at least twice (Pigeon Hole Principle)

Proof of the Pumping lemma

$\forall L \in$ REG : $\exists k > 0 : \forall z \in L, |z| \geq k : \exists u, v, w : \forall i \cdots$

- Let L be an arbitrary regular language.
- Then a DFA $M = (Q, \Sigma, \delta, q_0, F)$ exists with $\mathcal{L}(M) = L$.
- Let $k := |Q|$, the number of elements Q (indeed $k > 0$)
- Let an arbitrary $z \in L$ with $|z| \geq k$ be given.
- z follows a path in M of $k + 1$ states, so this paths visits a certain q at least twice (Pigeon Hole Principle)
- Then z can be split into uvw, with $|uv| \leq k$ and $|v| > 0$, and $\hat{\delta}(q_0, u) = q$ (1st time) $\quad \hat{\delta}(q, v) = q$ (2nd time) $\quad \hat{\delta}(q, w) \in F$

Proof of the Pumping lemma

$\forall L \in$ REG : $\exists k > 0 : \forall z \in L, |z| \geq k : \exists u, v, w : \forall i \cdots$

- Let L be an arbitrary regular language.
- Then a DFA $M = (Q, \Sigma, \delta, q_0, F)$ exists with $\mathcal{L}(M) = L$.
- Let $k := |Q|$, the number of elements Q (indeed $k > 0$)
- Let an arbitrary $z \in L$ with $|z| \geq k$ be given.
- z follows a path in M of $k + 1$ states, so this paths visits a certain q at least twice (Pigeon Hole Principle)
- Then z can be split into uvw, with $|uv| \leq k$ and $|v| > 0$, and $\hat{\delta}(q_0, u) = q$ (1st time) $\quad \hat{\delta}(q, v) = q$ (2nd time) $\quad \hat{\delta}(q, w) \in F$
- Then for every $i \geq 0$, $\hat{\delta}(q_0, uv^i w) \in F$ applies, so $uv^i w \in L$.

Using the Pumping lemma

Showing that a language is non-regular with the Pumping lemma

Using the Pumping lemma

Showing that a language is non-regular with the Pumping lemma

- Instead of:

  If L is a regular language, then
  there exists a number $k > 0$, such that
  for every word $z \in L$ with $|z| \geq k$:
  there exist words u, v, w, such that
  $z = uvw$ and $|uv| \leq k$ and $|v| > 0$ and
  for all $i \geq 0$, $uv^i w \in L$

- we will use the (logically equivalent!) contraposition:

Using the Pumping lemma

Showing that a language is non-regular with the Pumping lemma

- Instead of:

  If L is a regular language, then
  there exists a number $k > 0$, such that
  for every word $z \in L$ with $|z| \geq k$:
  there exist words u, v, w, such that
  $z = uvw$ and $|uv| \leq k$ and $|v| > 0$ and
  for all $i \geq 0$, $uv^i w \in L$

- we will use the (logically equivalent!) contraposition:

  If for every $k > 0$,
  there exists a word $z \in L$, with $|z| \geq k$, such that:
  for every u, v, w with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$,
  there exists some $i \geq 0$ such that $uv^i w \notin L$,
  then L is non-regular

Application 1: $L := \{a^i b^i \mid i \geq 0\}$ is non-regular

If for every $k > 0$,
 there exists a word $z \in L$, with $|z| \geq k$, such that:
  for every u, v, w with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$,
   there exists some $i \geq 0$ such that $uv^i w \notin L$,
then L is non-regular

Let $L := \{a^i b^i \mid i \geq 0\}$. To prove: L is non-regular.

Application 1: $L := \{a^i b^i \mid i \geq 0\}$ is non-regular

If for every $k > 0$,
 there exists a word $z \in L$, with $|z| \geq k$, such that:
  for every u, v, w with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$,
   there exists some $i \geq 0$ such that $uv^i w \notin L$,
then L is non-regular

Let $L := \{a^i b^i \mid i \geq 0\}$. To prove: L is non-regular.

- Let $k > 0$ be given

Application 1: $L := \{a^i b^i \mid i \geq 0\}$ is non-regular

If for every $k > 0$,
 there exists a word $z \in L$, with $|z| \geq k$, such that:
  for every u, v, w with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$,
   there exists some $i \geq 0$ such that $uv^i w \notin L$,
then L is non-regular

Let $L := \{a^i b^i \mid i \geq 0\}$. To prove: L is non-regular.

- Let $k > 0$ be given
- Choose $z := a^k b^k$, then $z \in L$ and $|z| \geq k$.

Application 1: $L := \{a^i b^i \mid i \geq 0\}$ is non-regular

If for every $k > 0$,
there exists a word $z \in L$, with $|z| \geq k$, such that:
for every u, v, w with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$,
there exists some $i \geq 0$ such that $uv^i w \notin L$,
then L is non-regular

Let $L := \{a^i b^i \mid i \geq 0\}$. To prove: L is non-regular.

- Let $k > 0$ be given
- Choose $z := a^k b^k$, then $z \in L$ and $|z| \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$

Application 1: $L := \{a^i b^i \mid i \geq 0\}$ is non-regular

If for every $k > 0$,
 there exists a word $z \in L$, with $|z| \geq k$, such that:
  for every u, v, w with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$,
   there exists some $i \geq 0$ such that $uv^i w \notin L$,
then L is non-regular

Let $L := \{a^i b^i \mid i \geq 0\}$. To prove: L is non-regular.

- Let $k > 0$ be given
- Choose $z := a^k b^k$, then $z \in L$ and $|z| \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$
- Then $u = a^p$ and $v = a^q$ and $w = a^{k-p-q} b^k$,
  for certain p, q with $p + q \leq k$ and $q > 0$.

Application 1: $L := \{a^i b^i \mid i \geq 0\}$ is non-regular

If for every $k > 0$,
 there exists a word $z \in L$, with $|z| \geq k$, such that:
  for every u, v, w with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$,
   there exists some $i \geq 0$ such that $uv^i w \notin L$,
then L is non-regular

Let $L := \{a^i b^i \mid i \geq 0\}$. To prove: L is non-regular.

- Let $k > 0$ be given
- Choose $z := a^k b^k$, then $z \in L$ and $|z| \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$
- Then $u = a^p$ and $v = a^q$ and $w = a^{k-p-q} b^k$,
  for certain p, q with $p + q \leq k$ and $q > 0$.
- Choose $i := 0$. Note that $uv^0 w = a^p a^{k-p-q} b^k = a^{k-q} b^k \notin L$

Application 1: $L := \{a^i b^i \mid i \geq 0\}$ is non-regular

If for every $k > 0$,
 there exists a word $z \in L$, with $|z| \geq k$, such that:
  for every u, v, w with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$,
   there exists some $i \geq 0$ such that $uv^i w \notin L$,
then L is non-regular

Let $L := \{a^i b^i \mid i \geq 0\}$. To prove: L is non-regular.

- Let $k > 0$ be given
- Choose $z := a^k b^k$, then $z \in L$ and $|z| \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$
- Then $u = a^p$ and $v = a^q$ and $w = a^{k-p-q} b^k$,
  for certain p, q with $p + q \leq k$ and $q > 0$.
- Choose $i := 0$. Note that $uv^0 w = a^p a^{k-p-q} b^k = a^{k-q} b^k \notin L$
- So: $L = \{a^i b^i \mid i \geq 0\}$ is non-regular

Application 2: $L := \{a^{m^2} \mid m \geq 0\}$ is non-regular

$L = \{\lambda, a, aaaa, aaaaaaaaa, \ldots\}$

- Let $k > 0$ be given

Application 2: $L := \{a^{m^2} \mid m \geq 0\}$ is non-regular

$L = \{\lambda, a, aaaa, aaaaaaaaa, \ldots\}$

- Let $k > 0$ be given
- Choose $z := a^{k^2}$, then $z \in L$ and $|z| = k^2 \geq k$.

Application 2: $L := \{a^{m^2} \mid m \geq 0\}$ is non-regular

$L = \{\lambda, a, aaaa, aaaaaaaaa, \ldots\}$

- Let $k > 0$ be given
- Choose $z := a^{k^2}$, then $z \in L$ and $|z| = k^2 \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$

Application 2: $L := \{a^{m^2} \mid m \geq 0\}$ is non-regular

$L = \{\lambda, a, aaaa, aaaaaaaaa, \ldots\}$

- Let $k > 0$ be given
- Choose $z := a^{k^2}$, then $z \in L$ and $|z| = k^2 \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$
- Then $u = a^p$ and $v = a^q$ and $w = a^{k^2-p-q}$,
  for certain p, q with $p + q \leq k$ and $q > 0$, so $1 \leq q \leq k$.

Application 2: $L := \{a^{m^2} \mid m \geq 0\}$ is non-regular

$L = \{\lambda, a, aaaa, aaaaaaaaa, \ldots\}$

- Let $k > 0$ be given
- Choose $z := a^{k^2}$, then $z \in L$ and $|z| = k^2 \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$
- Then $u = a^p$ and $v = a^q$ and $w = a^{k^2-p-q}$,
  for certain p, q with $p + q \leq k$ and $q > 0$, so $1 \leq q \leq k$.
- Choose $i := 2$. Note that
  $|uv^2w| = p + 2q + (k^2 - p - q) = k^2 + q$.

Application 2: $L := \{a^{m^2} \mid m \geq 0\}$ is non-regular

$L = \{\lambda, a, aaaa, aaaaaaaaa, \ldots\}$

- Let $k > 0$ be given
- Choose $z := a^{k^2}$, then $z \in L$ and $|z| = k^2 \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$
- Then $u = a^p$ and $v = a^q$ and $w = a^{k^2-p-q}$,
  for certain p, q with $p + q \leq k$ and $q > 0$, so $1 \leq q \leq k$.
- Choose $i := 2$. Note that
  $|uv^2w| = p + 2q + (k^2 - p - q) = k^2 + q$.
- Note that $k^2 < k^2 + q \leq k^2 + k < k^2 + 2k + 1 = (k + 1)^2$

Application 2: $L := \{a^{m^2} \mid m \geq 0\}$ is non-regular

$L = \{\lambda, a, aaaa, aaaaaaaaa, \ldots\}$

- Let $k > 0$ be given
- Choose $z := a^{k^2}$, then $z \in L$ and $|z| = k^2 \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$
- Then $u = a^p$ and $v = a^q$ and $w = a^{k^2-p-q}$,
  for certain p, q with $p + q \leq k$ and $q > 0$, so $1 \leq q \leq k$.
- Choose $i := 2$. Note that
  $|uv^2w| = p + 2q + (k^2 - p - q) = k^2 + q$.
- Note that $k^2 < k^2 + q \leq k^2 + k < k^2 + 2k + 1 = (k + 1)^2$
- $k^2 + q$ cannot be a square itself, so $uv^2w \notin L$.

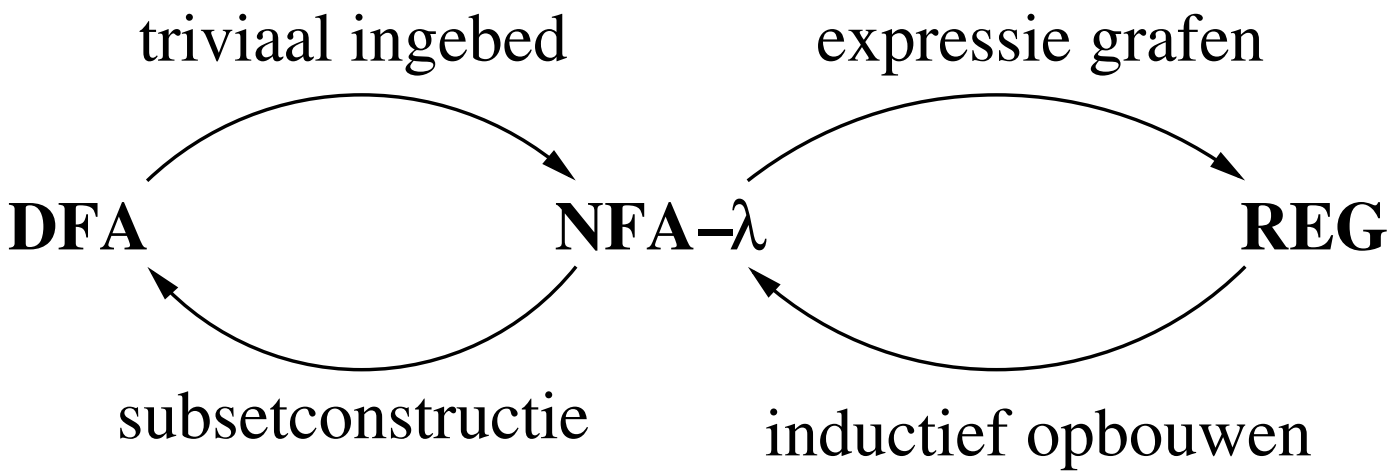
Application 2: $L := \{a^{m^2} \mid m \geq 0\}$ is non-regular

$L = \{\lambda, a, aaaa, aaaaaaaaa, \ldots\}$

- Let $k > 0$ be given
- Choose $z := a^{k^2}$, then $z \in L$ and $|z| = k^2 \geq k$.
- Let u, v, w be given, with $z = uvw$ and $|uv| \leq k$ and $|v| > 0$
- Then $u = a^p$ and $v = a^q$ and $w = a^{k^2-p-q}$,
  for certain p, q with $p + q \leq k$ and $q > 0$, so $1 \leq q \leq k$.
- Choose $i := 2$. Note that
  $|uv^2w| = p + 2q + (k^2 - p - q) = k^2 + q$.
- Note that $k^2 < k^2 + q \leq k^2 + k < k^2 + 2k + 1 = (k + 1)^2$
- $k^2 + q$ cannot be a square itself, so $uv^2w \notin L$.
- So: $L = \{a^{m^2} \mid m \geq 0\}$ is non-regular

Contents

Regular languages: the circle is now complete:



Three equivalent definitions for L is a regular language:

- A regular expression E exists, such that $\mathcal{L}(E) = L$;
- A DFA M exists, such that $\mathcal{L}(M) = L$;
- An NFA-λ M exists, such that $\mathcal{L}(M) = L$.

Let us study **REG**, the class of regular languages?

Exercise in abstract thinking: closed under ...

- Alphabet: Σ, example: symbol a
- Words: Σ^*, example: aba
- Languages: $\mathcal{P}(\Sigma^*)$, example: $\{(ab)^i \mid i \geq 0\}$
- Operations on languages: $\mathcal{P}(\Sigma^*) \times \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$, e.g.: $\cup$
- Classes of languages: $\mathcal{P}(\mathcal{P}(\Sigma^*))$, example: **REG**, **DFA**
- Properties of classes: $\mathcal{P}(\mathcal{P}(\mathcal{P}(\Sigma^*)))$, example: "is closed under union"

Closure property

A class of languages $\mathcal{K}$ is *closed* under binary operation $\otimes$, if for every language $\mathcal{L}_1 \in \mathcal{K}$ and $\mathcal{L}_2 \in \mathcal{K}$ the following also applies: the language $\mathcal{L}_1 \otimes \mathcal{L}_2 \in \mathcal{K}$.

Example: The class **REG** is closed under $\cup$ (union)
REG is also closed under complement (i.e.: $\mathcal{L} \in$ **REG** $\Rightarrow \overline{\mathcal{L}} \in$ **REG**)
(NB: complement is not a binary operation)

Closure properties

The class of regular languages is fairly robust.

If L_1 and L_2 are regular, then also:

- L_1^*, $L_1 L_2$ and $L_1 \cup L_2$ ..... (via regular expressions)

Closure properties

The class of regular languages is fairly robust.

If L_1 and L_2 are regular, then also:

- L_1^*, $L_1 L_2$ and $L_1 \cup L_2$ ..... (via regular expressions)
- $\overline{L_1}$, $L_1 \cap L_2$ and $L_1 - L_2$ .... (DFA-complement and De Morgan)

Closure properties

The class of regular languages is fairly robust.

If L_1 and L_2 are regular, then also:

- L_1^*, $L_1 L_2$ and $L_1 \cup L_2$ ..... (via regular expressions)
- $\overline{L_1}$, $L_1 \cap L_2$ and $L_1 - L_2$ .... (DFA-complement and De Morgan)
- L^R (reverse words) ..... (reverse arrows in NFA-λ)

Closure properties

The class of regular languages is fairly robust.

If L_1 and L_2 are regular, then also:

- L_1^*, $L_1 L_2$ and $L_1 \cup L_2$ ..... (via regular expressions)
- $\overline{L_1}$, $L_1 \cap L_2$ and $L_1 - L_2$ .... (DFA-complement and De Morgan)
- L^R (reverse words) ..... (reverse arrows in NFA-λ)

Adding / leaving out a finite sublanguage:

- Every finite language $X \subseteq_{fin} \Sigma^*$ is regular. ..... why?

Closure properties

The class of regular languages is fairly robust.

If L_1 and L_2 are regular, then also:

- L_1^*, $L_1 L_2$ and $L_1 \cup L_2$ ..... (via regular expressions)
- $\overline{L_1}$, $L_1 \cap L_2$ and $L_1 - L_2$ .... (DFA-complement and De Morgan)
- L^R (reverse words) ..... (reverse arrows in NFA-λ)

Adding / leaving out a finite sublanguage:

- Every finite language $X \subseteq_{fin} \Sigma^*$ is regular. ..... why?
- If L is regular, and X finite, then $L - X$ and $L \cup X$ also regular.

Closure properties

The class of regular languages is fairly robust.

If L_1 and L_2 are regular, then also:

- L_1^*, $L_1 L_2$ and $L_1 \cup L_2$ ..... (via regular expressions)
- $\overline{L_1}$, $L_1 \cap L_2$ and $L_1 - L_2$ .... (DFA-complement and De Morgan)
- L^R (reverse words) ..... (reverse arrows in NFA-λ)

Adding / leaving out a finite sublanguage:

- Every finite language $X \subseteq_{fin} \Sigma^*$ is regular. ..... why?
- If L is regular, and X finite, then $L - X$ and $L \cup X$ also regular.

The book has more examples

prefix(L), $h(L)$ with h homomorphism $\Sigma \to \Sigma^*$, ...

Pitfalls

The following doesn't hold

If L is regular, and $L' \subseteq L$, then L' is regular

Note that the largest (Σ^*) and smallest ($\emptyset$) languages are both regular

Pitfalls

The following doesn't hold

If L is regular, and $L' \subseteq L$, then L' is regular

Note that the largest (Σ^*) and smallest ($\emptyset$) languages are both regular

If L is regular, then so are the sets:

- $\{uv \mid u \in L \wedge v \notin L\}$ ..... but not: $\{uu \mid u \in L\}$

Pitfalls

The following doesn't hold

If L is regular, and $L' \subseteq L$, then L' is regular

Note that the largest (Σ^*) and smallest ($\emptyset$) languages are both regular

If L is regular, then so are the sets:

- $\{uv \mid u \in L \wedge v \notin L\}$ ..... but not: $\{uu \mid u \in L\}$
- $\{uv \mid u \in L \wedge v^R \in L\}$ ..... but not: $\{uu^R \mid u \in L\}$

Pitfalls

The following doesn't hold

If L is regular, and $L' \subseteq L$, then L' is regular

Note that the largest (Σ^*) and smallest ($\emptyset$) languages are both regular

If L is regular, then so are the sets:

- $\{uv \mid u \in L \wedge v \notin L\}$ ..... but not: $\{uu \mid u \in L\}$
- $\{uv \mid u \in L \wedge v^R \in L\}$ ..... but not: $\{uu^R \mid u \in L\}$
- The set of palindromes is not regular $\{w \mid w = w^R\}$.

Application of Closure Properties

We can also use the closure properties to prove that certain languages are not regular.
For instance, take the language with an equal number of a's and b's:

Claim: $L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$ is non-regular

Proof by contradiction:

Application of Closure Properties

We can also use the closure properties to prove that certain languages are not regular.
For instance, take the language with an equal number of a's and b's:

Claim: $L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$ is non-regular

Proof by contradiction:

Assume L is regular. Define $K = (a^*b^*)$, then K is regular by definition. Since REG is closed under intersection, $L \cap K$ is regular.

Application of Closure Properties

We can also use the closure properties to prove that certain languages are not regular.
For instance, take the language with an equal number of a's and b's:

Claim: $L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$ is non-regular

Proof by contradiction:
Assume L is regular. Define $K = (a^*b^*)$, then K is regular by definition. Since REG is closed under intersection, $L \cap K$ is regular. However, $L \cap K = \{a^n b^n \mid n \geq 0\}$, which is non-regular. Contradiction. Hence L is not regular.

Contents

Minimisation of DFAs: equivalent states

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

- Unreachable states can just be removed from Q
- Equivalent states could be combined into a single state

Minimisation of DFAs: equivalent states

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

- Unreachable states can just be removed from Q
- Equivalent states could be combined into a single state

Equivalent states

- When can two states q_1 and q_2 be considered equivalent?
  when they accept the same language:

$$\mathcal{L}((Q, \Sigma, \delta, q_1, F)) = \mathcal{L}((Q, \Sigma, \delta, q_2, F))$$

Minimisation of DFAs: equivalent states

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

- Unreachable states can just be removed from Q
- Equivalent states could be combined into a single state

Equivalent states

- When can two states q_1 and q_2 be considered equivalent?
  when they accept the same language:

$$\mathcal{L}((Q, \Sigma, \delta, q_1, F)) = \mathcal{L}((Q, \Sigma, \delta, q_2, F))$$

- We need a procedure to check that two states are equivalent.
  Note: checking that states q_i and q_j are distinguishable is easier

Minimisation of DFAs: equivalent states

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

- Unreachable states can just be removed from Q
- Equivalent states could be combined into a single state

Equivalent states

- When can two states q_1 and q_2 be considered equivalent?
  when they accept the same language:

$$\mathcal{L}((Q, \Sigma, \delta, q_1, F)) = \mathcal{L}((Q, \Sigma, \delta, q_2, F))$$

- We need a procedure to check that two states are equivalent.
  Note: checking that states q_i and q_j are distinguishable is easier
- Why? We only need one witness, a distinguishing word w:

$$\hat{\delta}(q_i, w) \in F \iff \hat{\delta}(q_j, w) \notin F$$

Minimisation of DFAs: equivalent states

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

- Unreachable states can just be removed from Q
- Equivalent states could be combined into a single state

Equivalent states

- When can two states q_1 and q_2 be considered equivalent?
  when they accept the same language:
  $$\mathcal{L}((Q, \Sigma, \delta, q_1, F)) = \mathcal{L}((Q, \Sigma, \delta, q_2, F))$$
- We need a procedure to check that two states are equivalent.
  Note: checking that states q_i and q_j are distinguishable is easier
- Why? We only need one witness, a distinguishing word w:
  $$\hat{\delta}(q_i, w) \in F \iff \hat{\delta}(q_j, w) \notin F$$
- Equivalent $\iff$ Undistinguishable

Minimisation of DFAs: distinguishability

Distinguishability relation: $\mathcal{D} \subseteq Q \times Q$

$\mathcal{D}$ is defined inductively, as the smallest relation,
such that for all $x, y \in Q$ and for all $a \in \Sigma$:

1. If ($x \in F$ and $y \notin F$) or ($x \notin F$ and $y \in F$), then $\mathcal{D}(x, y)$;
   (note: in this case λ distinguishes x and y)
2. If $\mathcal{D}(\delta(x, a), \delta(y, a))$, then $\mathcal{D}(x, y)$.
   (note: assume w distinguishes x' and y',
   then aw distinguishes x and y)

Note: x, y are already distinguishable if only one of their successors is.

Minimisation of DFAs: algorithm

Table filling algorithm - Algorithm 5.7.2 in Sudkamp (simplified)

- Array $D[i, j]$ computes $(q_i, q_j) \in \mathcal{D}$ iteratively; initially: $D[i, j] := 0$.
- Note: due to symmetry, we only consider the cases $i < j$.

Minimisation of DFAs: algorithm

Table filling algorithm - Algorithm 5.7.2 in Sudkamp (simplified)

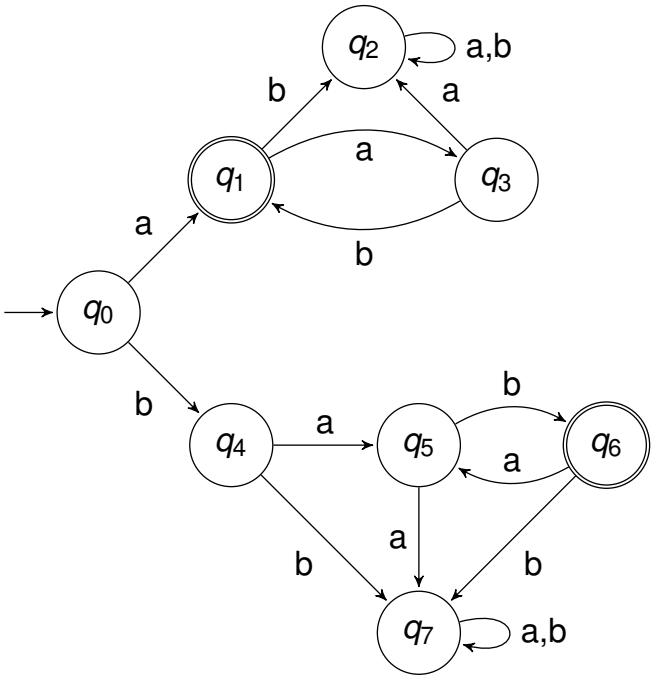
- Array $D[i, j]$ computes $(q_i, q_j) \in \mathcal{D}$ iteratively; initially: $D[i, j] := 0$.
- Note: due to symmetry, we only consider the cases $i < j$.
- Starting point:
  - F-states are distinguishable from $Q - F$-states:
  - So: for every $q_i \in F$ and $q_j \in Q - F$ (with $i < j$): D[i,j] := 1

Minimisation of DFAs: algorithm

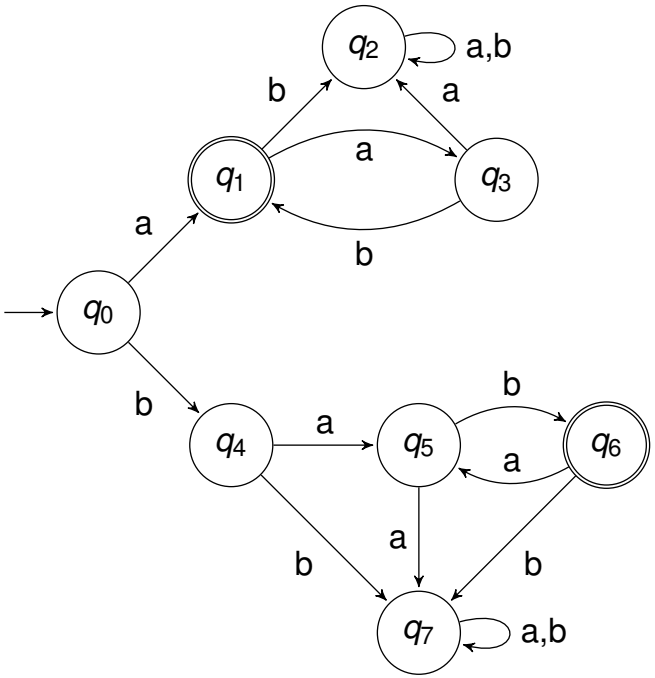
Table filling algorithm - Algorithm 5.7.2 in Sudkamp (simplified)

- Array $D[i,j]$ computes $(q_i, q_j) \in \mathcal{D}$ iteratively; initially: $D[i,j] := 0$.
- Note: due to symmetry, we only consider the cases $i < j$.
- Starting point:
  - F-states are distinguishable from $Q - F$-states:
  - So: for every $q_i \in F$ and $q_j \in Q - F$ (with $i < j$): D[i,j] := 1
- Repeat unless nothing changes:
  - If q_k and q_l are distinguishable, so are their a-predecessors:
  - So: if $D[k,l] = 1$, $\delta(q_i, a) = q_k$ and $\delta(q_j, a) = q_l$, then $D[i,j] := 1$.

Minimisation of DFAs: example (1)



Minimisation of DFAs: example (1)



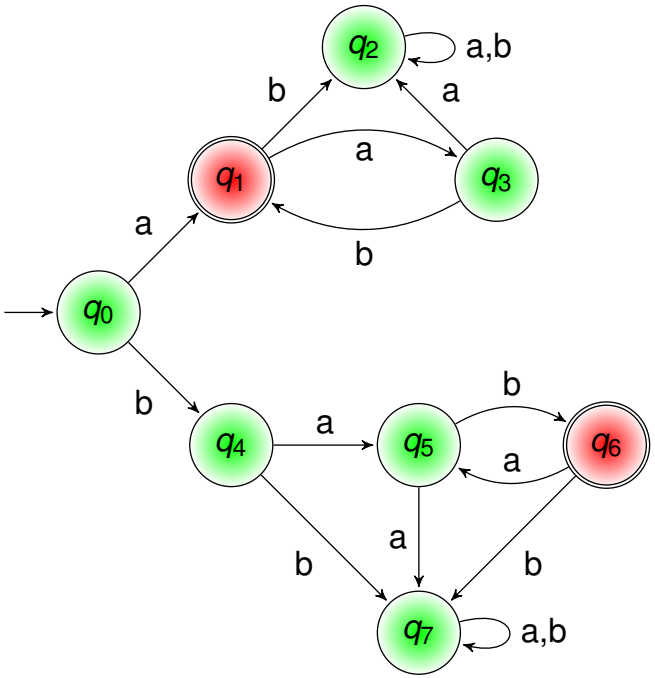
D	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0	—	1					1	
q_1	—	—	1	1	1	1		1
q_2	—	—	—				1	
q_3	—	—	—	—			1	
q_4	—	—	—	—	—		1	
q_5	—	—	—	—	—	—	1	
q_6	—	—	—	—	—	—	—	1
q_7	—	—	—	—	—	—	—	—

Distinguishable by the empty word:

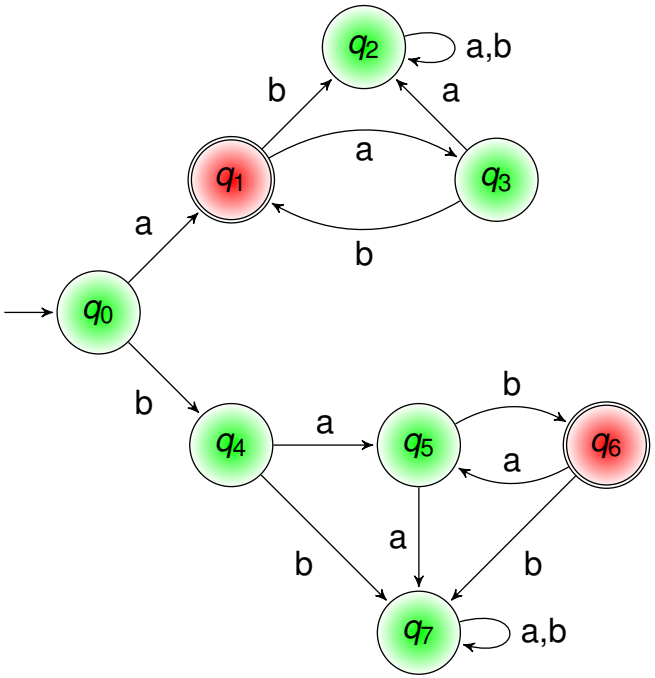
accepting or not, that's the question

$\{q_1, q_6\}$ versus $\{q_0, q_2, q_3, q_4, q_5, q_7\}$

Minimisation of DFAs: example (2)



Minimisation of DFAs: example (2)

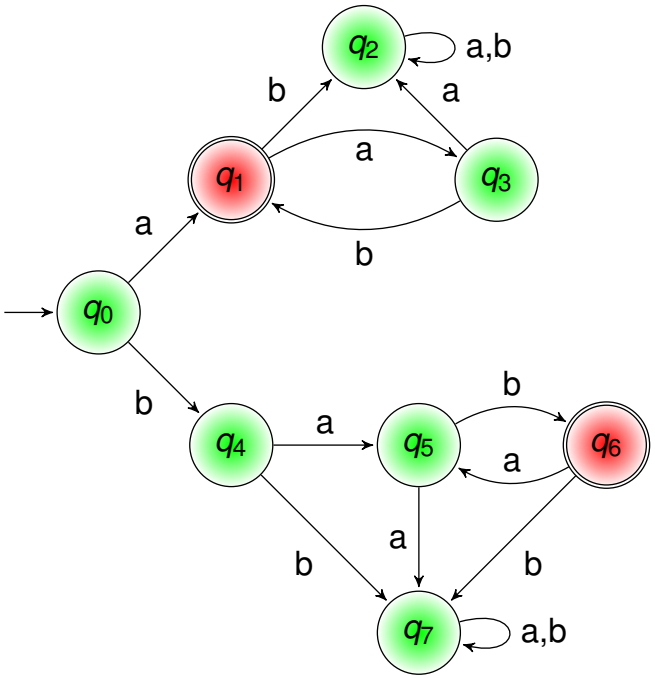


D	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0	–	1	**1**	**1**	**1**	**1**	1	**1**
q_1	–	–	1	1	1	1		1
q_2	–	–	–	**1**		**1**	1	
q_3	–	–	–	–	**1**		1	**1**
q_4	–	–	–	–	–	**1**	1	
q_5	–	–	–	–	–	–	1	**1**
q_6	–	–	–	–	–	–	–	1
q_7	–	–	–	–	–	–	–	–

Distinguishable by a 1-letter word:

$\delta(q_0, a) = q_1$ and $\delta(q_3, a) = q_2$ and $D[q_1, q_2] = 1$; so $D[q_0, q_3] := 1$

Minimisation of DFAs: example (2)



D	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0	–	1	**1**	**1**	**1**	**1**	1	**1**
q_1	–	–	1	1	1	1		1
q_2	–	–	–	**1**		**1**	1	
q_3	–	–	–	–	**1**		1	**1**
q_4	–	–	–	–	–	**1**	1	
q_5	–	–	–	–	–	–	1	**1**
q_6	–	–	–	–	–	–	–	1
q_7	–	–	–	–	–	–	–	–

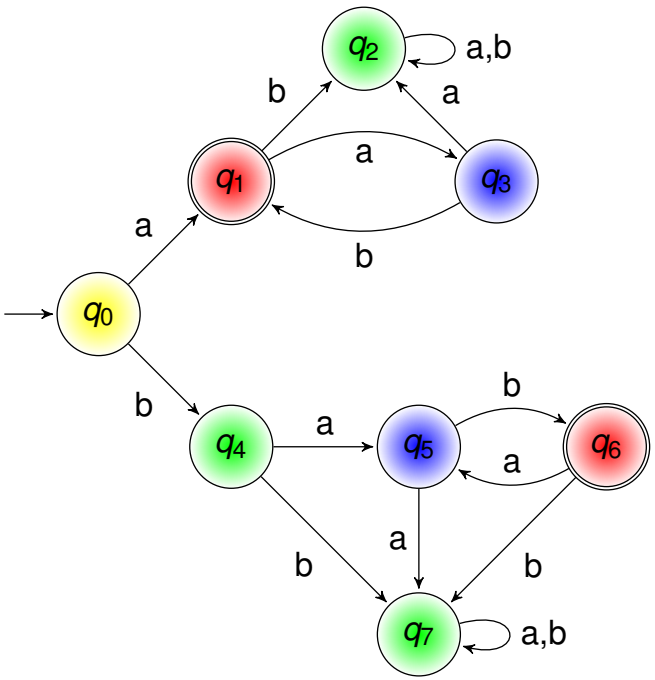
Distinguishable by a 1-letter word:

$\delta(q_0, a) = q_1$ and $\delta(q_3, a) = q_2$ and $D[q_1, q_2] = 1$; so $D[q_0, q_3] := 1$

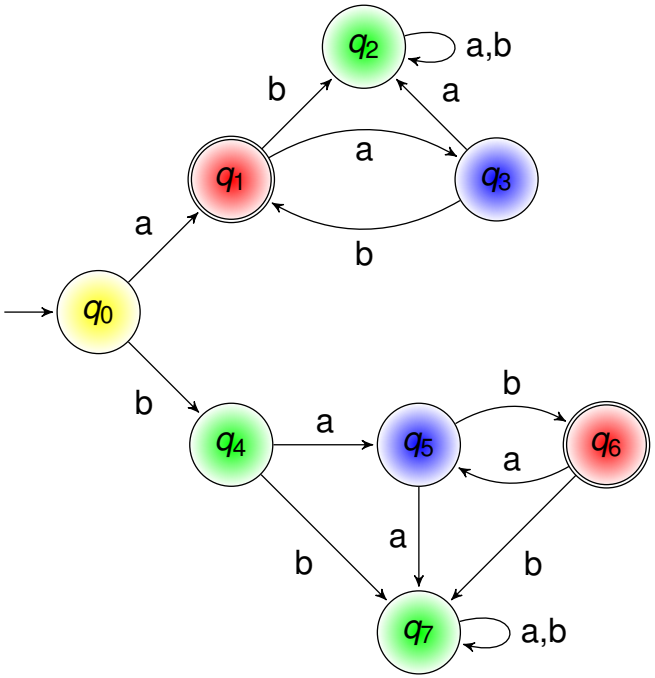
$\delta(q_0, b) = q_4$ and $\delta(q_5, b) = q_6$ and $D[q_4, q_6] = 1$; so $D[q_0, q_5] := 1$

Indeed, one cannot distinguish q_2, q_4 and q_7 with one symbol.

Minimisation of DFAs: example (3)



Minimisation of DFAs: example (3)



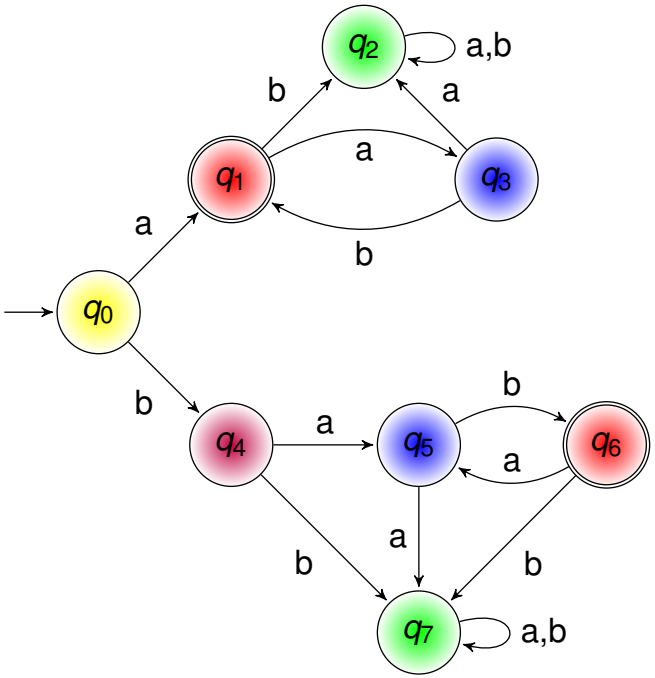
D	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0	–	1	1	1	1	1	1	1
q_1	–	–	1	1	1	1		1
q_2	–	–	–	1	**1**	1	1	
q_3	–	–	–	–	1		1	1
q_4	–	–	–	–	–	1	1	**1**
q_5	–	–	–	–	–	–	1	1
q_6	–	–	–	–	–	–	–	1
q_7	–	–	–	–	–	–	–	–

Next round (since we added new 1's in D):

$\delta(q_4, a) = q_5$ and $\delta(q_7, a) = q_7$ and $D[q_5, q_7] = 1$; so $D[q_4, q_7] := 1$

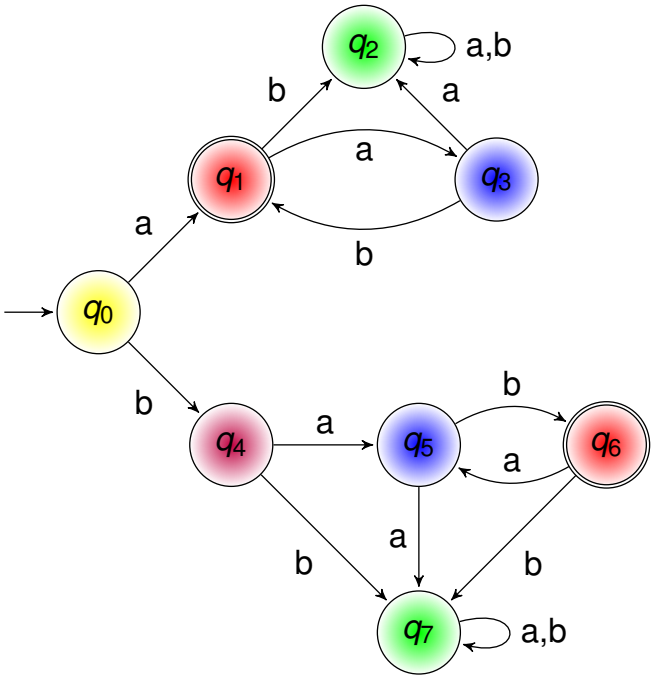
$\delta(q_2, a) = q_2$ and $\delta(q_4, a) = q_5$ and $D[q_2, q_5] = 1$; so $D[q_2, q_4] := 1$

Minimisation of DFAs: example (4)



D	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0	–	1	1	1	1	1	1	1
q_1	–	–	1	1	1	1	**0**	1
q_2	–	–	–	1	1	1	1	**0**
q_3	–	–	–	–	1	**0**	1	1
q_4	–	–	–	–	–	1	1	1
q_5	–	–	–	–	–	–	1	1
q_6	–	–	–	–	–	–	–	1
q_7	–	–	–	–	–	–	–	–

Minimisation of DFAs: example (4)



D	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0	–	1	1	1	1	1	1	1
q_1	–	–	1	1	1	1	**0**	1
q_2	–	–	–	1	1	1	1	**0**
q_3	–	–	–	–	1	**0**	1	1
q_4	–	–	–	–	–	1	1	1
q_5	–	–	–	–	–	–	1	1
q_6	–	–	–	–	–	–	–	1
q_7	–	–	–	–	–	–	–	–

Maximal distinctions

- We need one more round, but nothing will change.
- Conclusion: (q_1, q_6), (q_2, q_7) and (q_3, q_5) are equivalent!

Minimisation of DFAs: example (5)

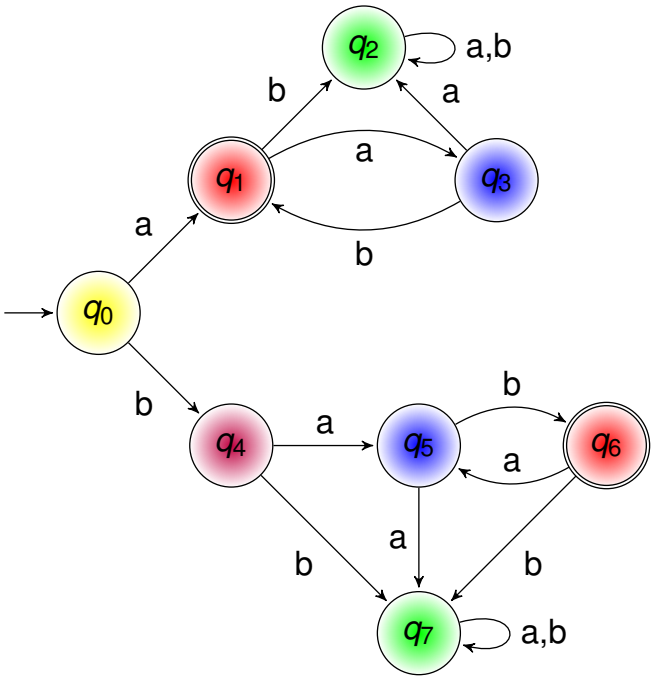
Minimal DFA: combine equivalent states

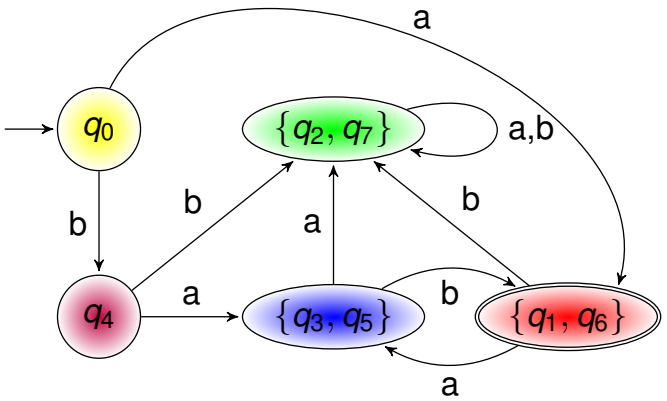
- The elements of an equivalence class accept the same language
- Make the equivalence classes states of the minimal DFA
- The transitions are derived from the class members

Minimisation of DFAs: example (5)

Minimal DFA: combine equivalent states

- The elements of an equivalence class accept the same language
- Make the equivalence classes states of the minimal DFA
- The transitions are derived from the class members





Minimisation: efficiency

table $S[i, j]$ in Algorithm 5.7.2

- Problem: for each change, we must recheck the whole D-table.
- Goal: Walk through the table only once: consider each pair (i, j) with $i < j$ just once.

Minimisation: efficiency

table $S[i, j]$ in Algorithm 5.7.2

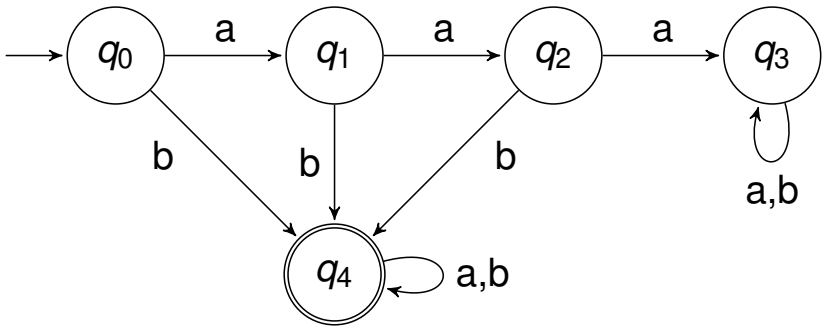
- Problem: for each change, we must recheck the whole D-table.
- Goal: Walk through the table only once: consider each pair (i, j) with $i < j$ just once.
- Idea:
  - Assume: we are considering pair (i, j)
  - Assume: $\delta(i, a) = k$, $\delta(j, a) = l$, but we have not yet considered (k, l).
  - We store (i, j) in a set to consider later: $S[k, l] = \{(i, j)\}$.

Minimisation: efficiency

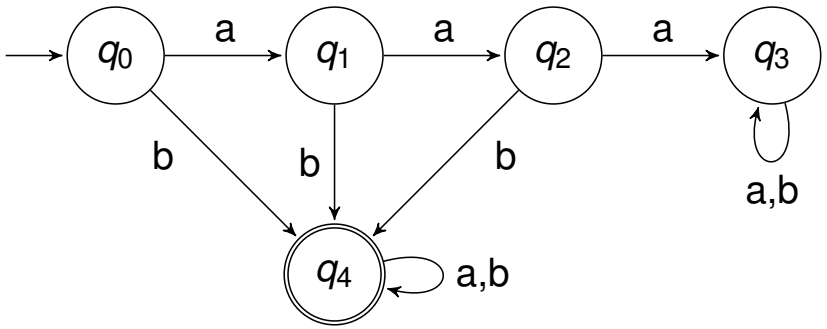
table $S[i, j]$ in Algorithm 5.7.2

- Problem: for each change, we must recheck the whole D-table.
- Goal: Walk through the table only once: consider each pair (i, j) with $i < j$ just once.
- Idea:
  - Assume: we are considering pair (i, j)
  - Assume: $\delta(i, a) = k$, $\delta(j, a) = l$, but we have not yet considered (k, l).
  - We store (i, j) in a set to consider later: $S[k, l] = \{(i, j)\}$.
  - When we (k, l) is treated later, and if $D[k, l] := 1$:
  - Re-consider every $(i, j) \in S[k, l]$, and set $D[i, j] := 1$
  - Note: in that case we must also reconsider pairs in $S[i, j]$

Efficient example (1)



Efficient example (1)

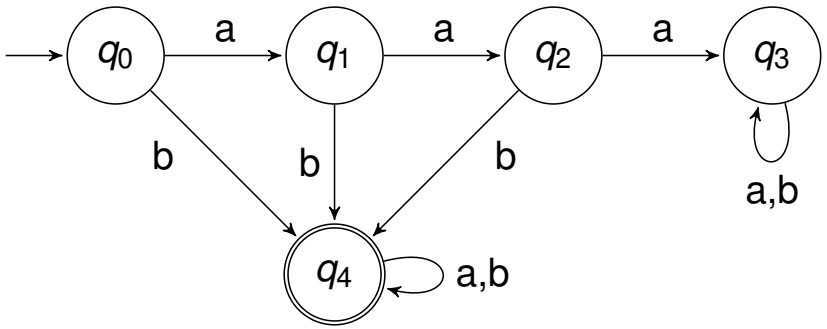


D/S	q_0	q_1	q_2	q_3	q_4
q_0	–	?	?	1	1
q_1	–	–	$(0,1)$	$(0,2)$	1
q_2	–	–	–	$(1,2)$	1
q_3	–	–	–	–	1
q_4	–	–	–	–	–

First steps:

- Initially, distinguish $\{4\}$ from $\{0,1,2,3\}$ (based on F)
- $(0,1)$ depends on $(1,2)$: $S[1,2] := \{(0,1)\}$
- $(0,2)$ depends $(1,3)$: $S[1,3] := \{(0,2)\}$

Efficient example (1)

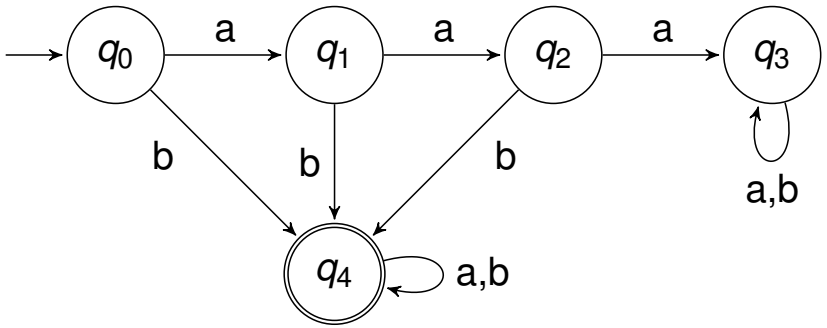


D/S	q_0	q_1	q_2	q_3	q_4
q_0	–	?	?	1	1
q_1	–	–	$(0,1)$	$(0,2)$	1
q_2	–	–	–	$(1,2)$	1
q_3	–	–	–	–	1
q_4	–	–	–	–	–

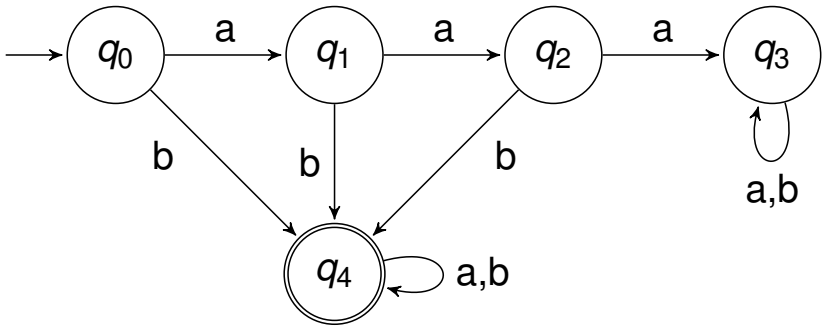
First steps:

- Initially, distinguish $\{4\}$ from $\{0,1,2,3\}$ (based on F)
- $(0,1)$ depends on $(1,2)$: $S[1,2] := \{(0,1)\}$
- $(0,2)$ depends $(1,3)$: $S[1,3] := \{(0,2)\}$
- $(0,3)$ proceeds with b to $(3,4)$; $D[3,4] = 1$. So: $D[0,3] := 1$
- $(1,2)$ depends on $(2,3)$, so we store $S[2,3] = \{(1,2)\}$.

Efficient example (2)



Efficient example (2)

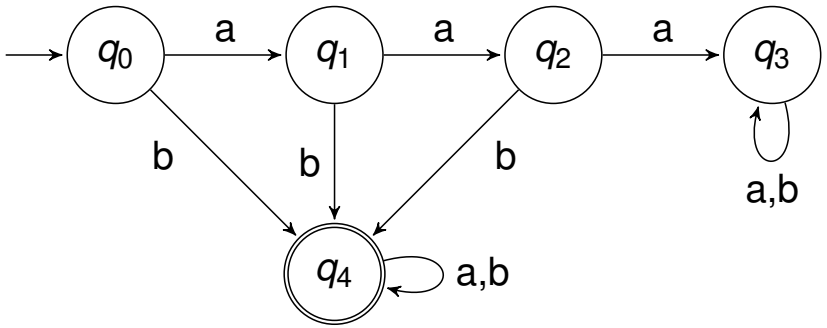


D/S	q_0	q_1	q_2	q_3	q_4
q_0	–	1	1	1	1
q_1	–	–	1	1	1
q_2	–	–	–	1	1
q_3	–	–	–	–	1
q_4	–	–	–	–	–

Last steps:

- Pair (1,3): *b*-steps to $(3,4)$: so $D[1,3] := 1$
- Now reconsider $S[1,3] = \{(0,2)\}$, so $D[0,2] := 1$
- Pair (2,3): *b*-steps to $(3,4)$: $D[2,3] := 1$
- Reconsider $S[2,3] = \{(1,2)\}$, so $D[1,2] := 1$
- Now reconsider $S[1,2] = \{(0,1)\}$, so $D[0,1] := 1$

Efficient example (2)



D/S	q_0	q_1	q_2	q_3	q_4
q_0	–	1	1	1	1
q_1	–	–	1	1	1
q_2	–	–	–	1	1
q_3	–	–	–	–	1
q_4	–	–	–	–	–

Last steps:

- Pair (1,3): *b*-steps to $(3,4)$: so $D[1,3] := 1$
- Now reconsider $S[1,3] = \{(0,2)\}$, so $D[0,2] := 1$
- Pair (2,3): *b*-steps to $(3,4)$: $D[2,3] := 1$
- Reconsider $S[2,3] = \{(1,2)\}$, so $D[1,2] := 1$
- Now reconsider $S[1,2] = \{(0,1)\}$, so $D[0,1] := 1$

Conclusion

All states are distinguishable, so the automaton was already minimal.

Minimalisation, final remarks

Minimisation

- Minimisation only works for determinisitc, complete DFA.
- The result is an equivalent DFA, which is the minimal DFA; The proof follows from the Theorem of Myhill-Nerode (Chapter 6.7).
- It is possible that there exists an even smaller NFA

Minimalisation, final remarks

Minimisation

- Minimisation only works for determinisitc, complete DFA.
- The result is an equivalent DFA, which is the minimal DFA; The proof follows from the Theorem of Myhill-Nerode (Chapter 6.7).
- It is possible that there exists an even smaller NFA

Checking equivalence of DFAs M_1 and M_2

- The minimal DFA unique
  - up to isomorphism (renaming of states)
- Method to check equivalence of DFAs:
  1. Minimise M_1 into M_1' and M_2 into M_2'
  2. Check if M_1' and M_2' are isomorphic

Real-Life Applications

Applications of regular languages:

1. Searching in text (pattern matching, cf. C 2.4)
2. Lexical analysis of software programs (keywords, identifiers)
3. Input validation for software security
4. System specification in software engineering

Real-Life Applications

Applications of regular languages:

1. Searching in text (pattern matching, cf. C 2.4)
2. Lexical analysis of software programs (keywords, identifiers)
3. Input validation for software security
4. System specification in software engineering

Typical approach:

1. Identify a number of patterns
2. Specify the language by a regular expression

Real-Life Applications

Applications of regular languages:

1. Searching in text (pattern matching, cf. C 2.4)
2. Lexical analysis of software programs (keywords, identifiers)
3. Input validation for software security
4. System specification in software engineering

Typical approach:

1. Identify a number of patterns
2. Specify the language by a regular expression
3. Transform the expression to an NFA-λ

Real-Life Applications

Applications of regular languages:

1. Searching in text (pattern matching, cf. C 2.4)
2. Lexical analysis of software programs (keywords, identifiers)
3. Input validation for software security
4. System specification in software engineering

Typical approach:

1. Identify a number of patterns
2. Specify the language by a regular expression
3. Transform the expression to an NFA-λ
4. Transform the NFA-λ into a DFA, using the subset construction

Real-Life Applications

Applications of regular languages:

1. Searching in text (pattern matching, cf. C 2.4)
2. Lexical analysis of software programs (keywords, identifiers)
3. Input validation for software security
4. System specification in software engineering

Typical approach:

1. Identify a number of patterns
2. Specify the language by a regular expression
3. Transform the expression to an NFA-λ
4. Transform the NFA-λ into a DFA, using the subset construction
5. Minimise the DFA

Real-Life Applications

Applications of regular languages:

1. Searching in text (pattern matching, cf. C 2.4)
2. Lexical analysis of software programs (keywords, identifiers)
3. Input validation for software security
4. System specification in software engineering

Typical approach:

1. Identify a number of patterns
2. Specify the language by a regular expression
3. Transform the expression to an NFA-λ
4. Transform the NFA-λ into a DFA, using the subset construction
5. Minimise the DFA
6. Translate the minimal DFA directly into code (C, Java, Python)

A regular expression for E-mail addresses

according to standard RFC 5322, cf. http://emailregex.com

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-
\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-
9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|
[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])
```

Note: this regexp uses extended notation, which you find in many practical examples, like ? for optional, [a-z] for ranges of symbols.