Leksiline analüüs

Leksiline analüüs

- Leksiline analüüs kontrollib programmi sõnade (literaalsümbolite) vastavust leksilistele reeglitele ning teisendab programmi sümbolite (tokens) jadaks:
  - eemaldab tühisümbolid ja kommentaarid;
  - identifitseerib võtmesõnad, identifikaatorid ja konstandid;
  - konstrueerib sümbolite tabeli;
  - leiab sümbolite rea/veeru numbrid;
  - teavitab vajadusel leksiliste vigadest.
- Leksilist analüüsi kutsutakse skaneerimiseks (scanning) ning vastavat analüsaatorit nimetatakse skanneriks (scanner).

Regulaaravaldised

- Regulaaravaldised üle (lõpliku) tähestiku Σ

$$E ::= \emptyset \mid \varepsilon \mid a \mid (E\ E) \mid (E \mid E) \mid E^\star$$

kus $a \in \Sigma$.

- Regulaaravaldis E defineerib keele $L(E) \subseteq \Sigma^\star$

$$\begin{array}{lcllcl} L(\emptyset) & = & \emptyset & L(E_1\ E_2) & = & \{uv \mid u \in L(E_1),\ v \in L(E_2)\} \\ L(\varepsilon) & = & \{\varepsilon\} & L(E_1 \mid E_2) & = & L(E_1) \cup L(E_2) \\ L(a) & = & \{a\} & L(E^\star) & = & \{w^i \mid w \in L(E),\ i \geq 0\} \end{array}$$

kus $w^0 = \varepsilon$ ja $w^{n+1} = ww^n$.

Regulaaravaldised

- Näiteid:

Regulaaravaldis	Defineeritav keel
$a \mid b$	$\{a, b\}$
$abba$	$\{abba\}$
$ab^\star a$	$\{aa, aba, abba, abbba, \ldots\}$
$(ab)^\star$	$\{\varepsilon, ab, abab, ababab, \ldots\}$

- Regulaaravaldistes esinevate sulgude vähendamiseks on operaatoritele määratud prioriteedid:
  - sulundioperaator $(\cdot)^\star$ seob kõige tugevamalt;
  - valikuoperaator $(\cdot \mid \cdot)$ seob kõige nõrgemalt.

Regulaaravaldised

- Regulaarne kirjeldus tähestikus Σ on reeglite hulk

$$\begin{array}{lcl} d_1 & \to & E_1 \\ d_2 & \to & E_2 \\ & \ldots & \\ d_n & \to & E_n \end{array}$$

  kus d_i on (unikaalne) nimi ja E_i on regulaaravaldis tähestikus $\Sigma \cup \{d_1, \ldots, d_{i-1}\}$.

- Lühendavaid tähistusi regulaaravaldiste esitamiseks:
  - *mittetühi sulund*: $E^+ = EE^\star$;
  - *optsioon*: $E? = \varepsilon \mid E$;
  - *märgiklassid*: näit. $[a, b, c] = a \mid b \mid c$ või $[a - z] = a \mid \ldots \mid z$.

Regulaaravaldised

Näiteid regulaarsetest kirjeldustest:

Identifikaatorid:

$$
\begin{array}{lcl}
\text{Letter} & \longrightarrow & [a-z,\ A-Z] \\
\text{Digit} & \longrightarrow & [0-9] \\
\text{Identifier} & \longrightarrow & \text{Letter}\,(\text{Letter} \mid \text{Digit})^{\star}
\end{array}
$$

Arvkonstandid:

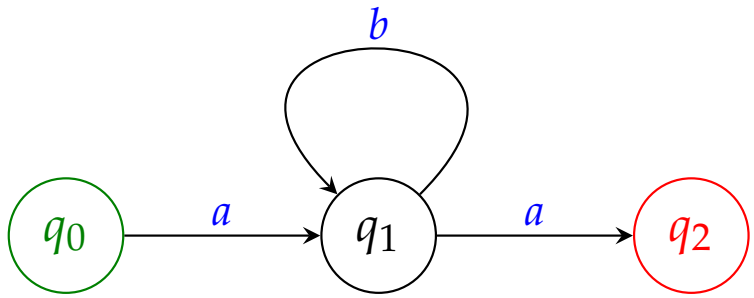
$$
\begin{array}{lcl}
\text{Sign} & \longrightarrow & (+ \mid -)? \\
\text{Integer} & \longrightarrow & 0 \mid \text{Sign}\,[1-9]\,\text{Digit}^{\star} \\
\text{Decimal} & \longrightarrow & \text{Integer}\,.\,\text{Digit}^{+} \\
\text{Real} & \longrightarrow & (\text{Integer} \mid \text{Decimal})\,E\,\text{Integer}
\end{array}
$$

Lõplikud automaadid

- Lõplik automaat on viisik $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, kus
  - Q on lõplik olekute hulk;
  - Σ on lõplik tähestik;
  - $\delta \subseteq Q \times (\Sigma \cup \varepsilon) \times Q$ on üleminekurelatsioon;
  - $q_0 \in Q$ on algolek;
  - $F \subseteq Q$ on lõppolekute hulk.
- Lõplik automaat on determineeritud (DFA), kui üleminekurelatsioon on funktsioon $\delta : Q \times \Sigma \to Q$.
- Vastasel korral on lõplik automaat mittedetermineeritud (NFA).

Lõplikud automaadid

- Lõplike automaate esitatakse tihti üleminekudiagrammidena:



- Lõplik automaat $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ aktsepteerib keele

$$L(A) = \{w \in \Sigma^\star \mid (q_0, w, q_f) \in \delta^\star,\ q_f \in F\}$$

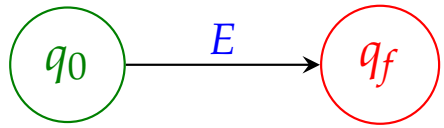
kus $\delta^\star \subseteq Q \times \Sigma^\star \times Q$ on üleminekurelatsiooni δ refleksiivne transitiivne sulund.

- Teoreem: Lõplike automaatide poolt aktsepteeritavate keelte klass langeb kokku regulaarsete keeltega.

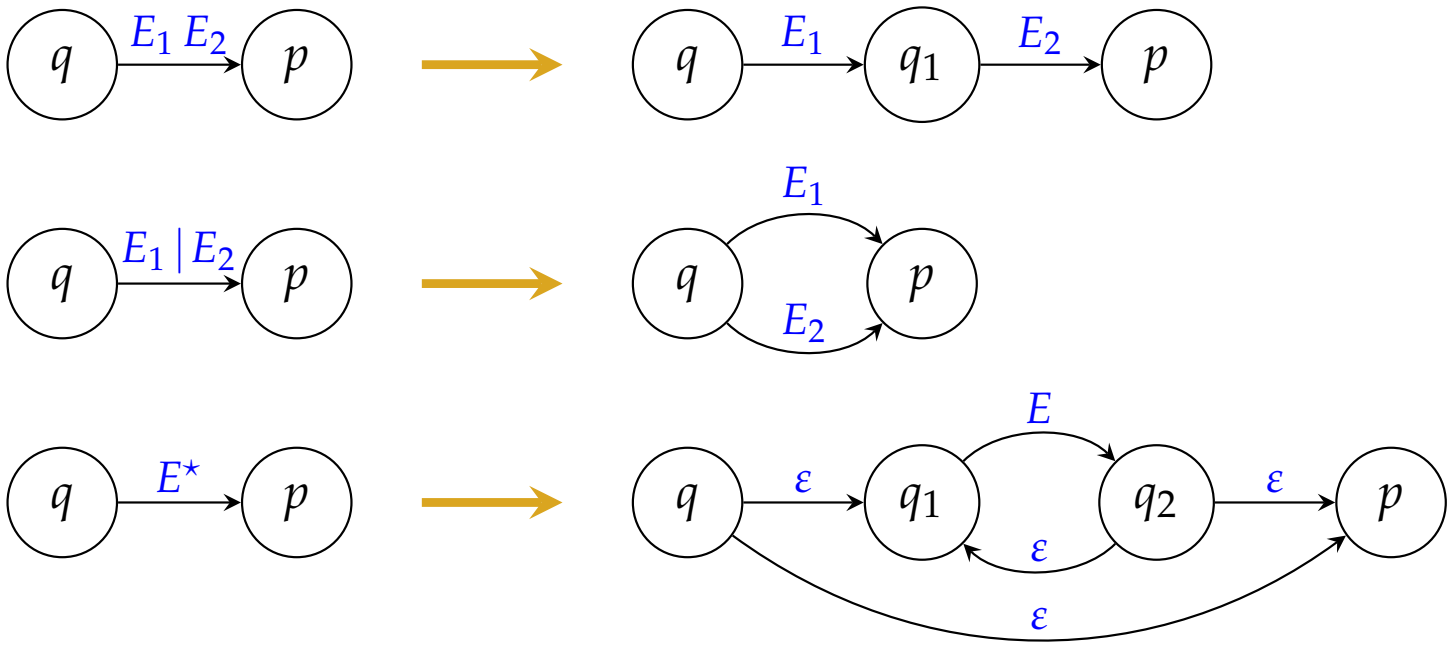
Regulaaravaldise teisendamine automaadiks

Thompsoni konstruktsioon regulaaravaldise teisendamiseks (mittedetermineeritud) lõplikuks automaadiks:

- regulaaravaldisele E seame vastavusse "automaadi":

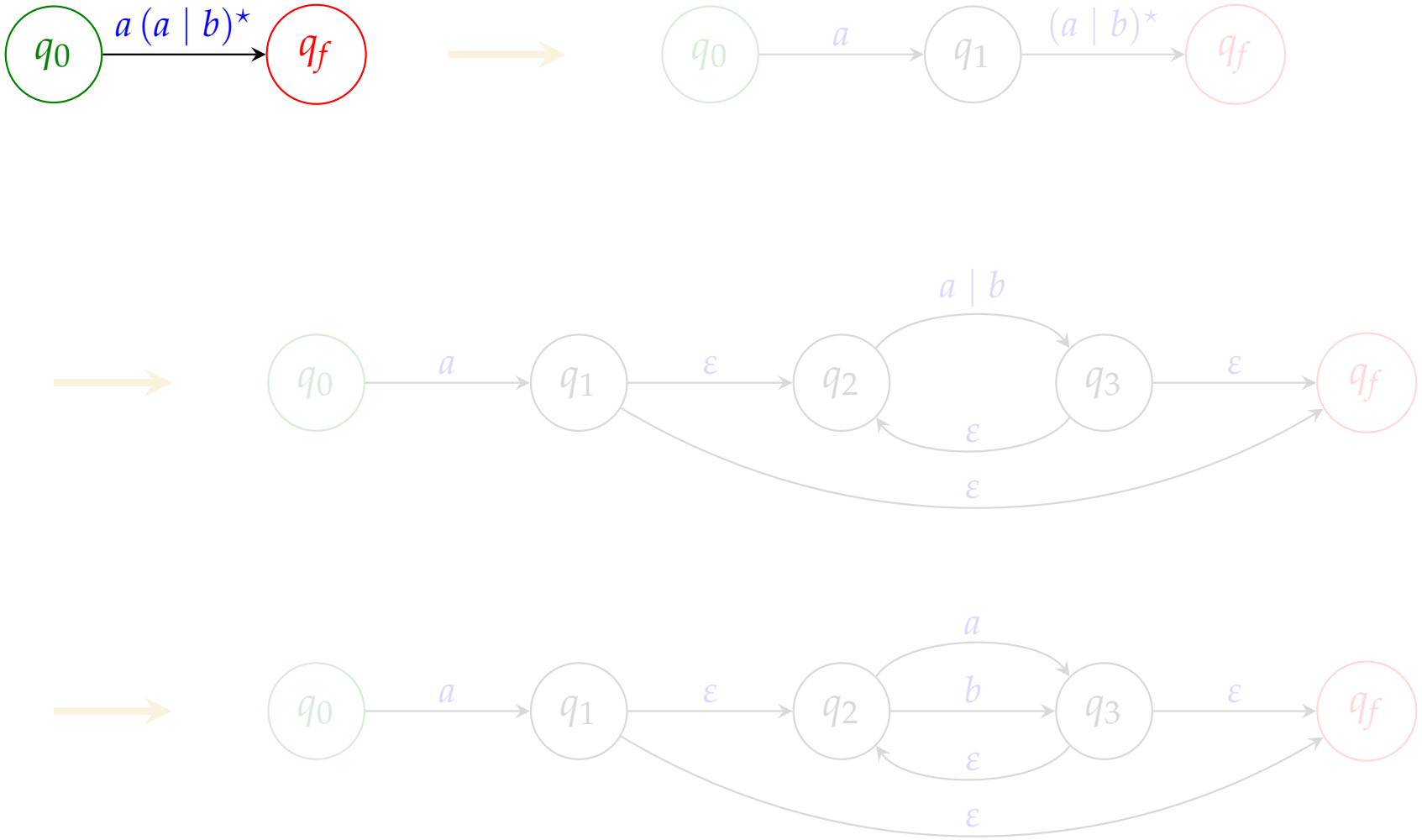


- teisendame "automaati" järgmiste reeglite abil, kuni kõik üleminekud on kas ε või üksikud tähed:



Regulaaravaldise teisendamine automaadiks

Näide:

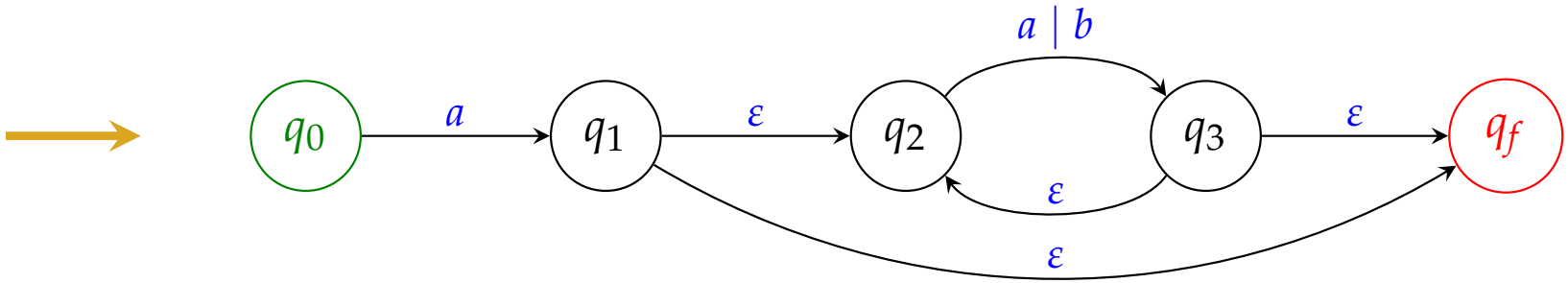


Regulaaravaldise teisendamine automaadiks

Näide:

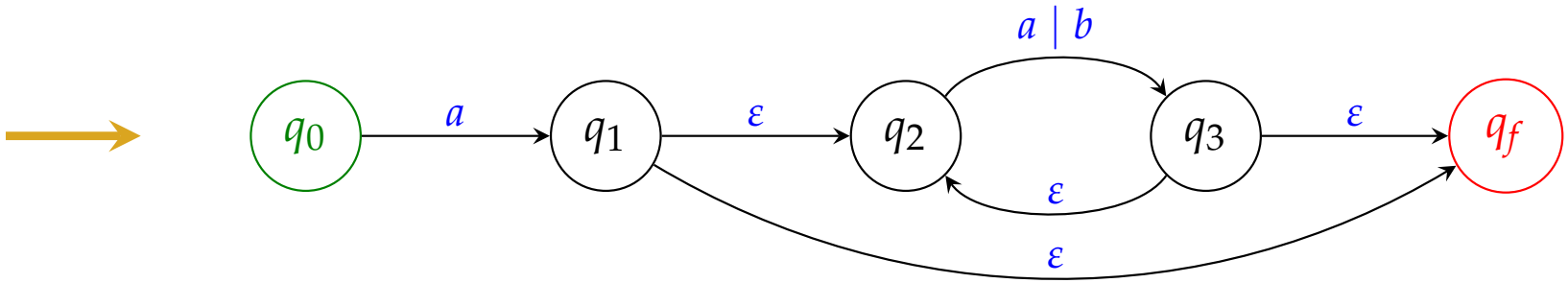
Regulaaravaldise teisendamine automaadiks

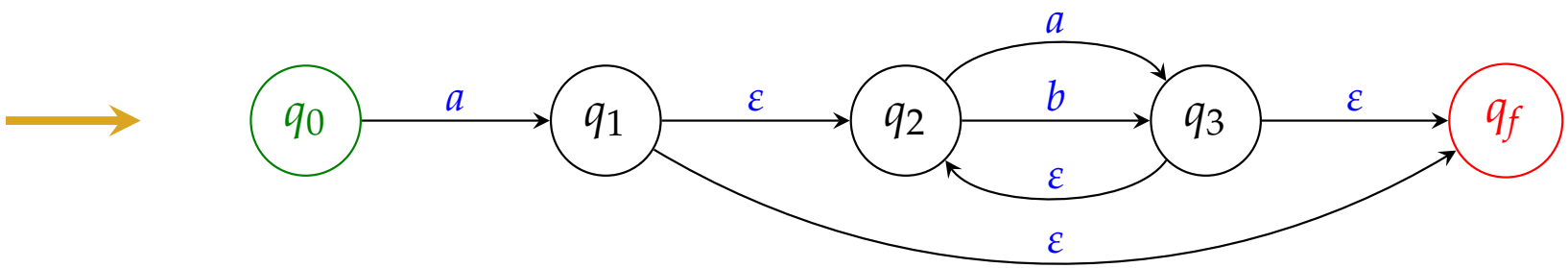
Näide:



Regulaaravaldise teisendamine automaadiks

Näide:





Determineeritud lõpliku automaadi koostamine

- Mittedeterminееritud lõpliku automaadiga $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ ekvivalentse determineeritud lõpliku automaadi $A' = \langle Q', \Sigma, \delta', q_0', F' \rangle$ konstrueerimine osahulkade moodustamise abil.
- Abifuntsioonid:
  - tühikäigusulundi funktsioon $\varepsilon\text{-}closure : 2^Q \to 2^Q$

$$\varepsilon\text{-}closure(S) = \{p \mid q \in S,\ (q, \varepsilon, p) \in \delta^\star\}$$

  - ühe sammu funktsioon $move : 2^Q \times \Sigma \to 2^Q$

$$move(S, a) = \{p \mid q \in S,\ (q, a, p) \in \delta\}$$

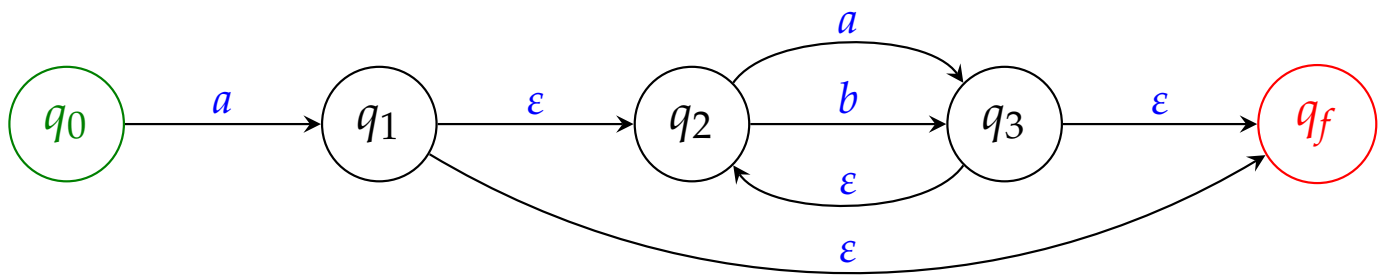
Determineeritud lõpliku automaadi koostamine

Algoritm:

$Q' := \emptyset;\ F' := \emptyset;\ \delta' := \emptyset;$
$q'_0 := \varepsilon\text{-}closure(\{q_0\});\ U := \{q'_0\};$
while $\exists S \in U$ **do**
  $U := U \setminus S;\ Q' := Q' \cup \{S\};$
  foreach $a \in \Sigma$ **do**
    $T := \varepsilon\text{-}closure(move(S, a));$
    if $T \notin U \cup Q'$ **then** $U := U \cup \{T\};$
    $\delta' := \delta' \cup \{(S, a) \mapsto T\};$
  end
end
$F' := \{S \in Q' \mid S \cap F \neq \emptyset\};$

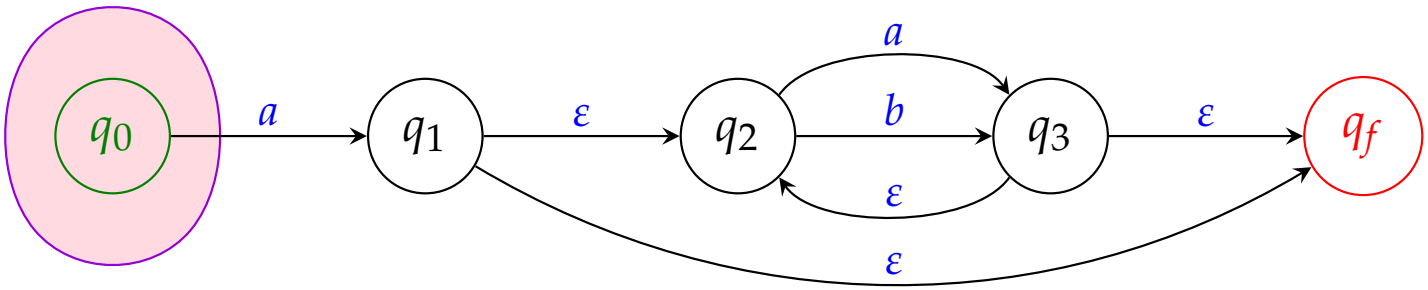
Determineeritud lõpliku automaadi koostamine

Näide:



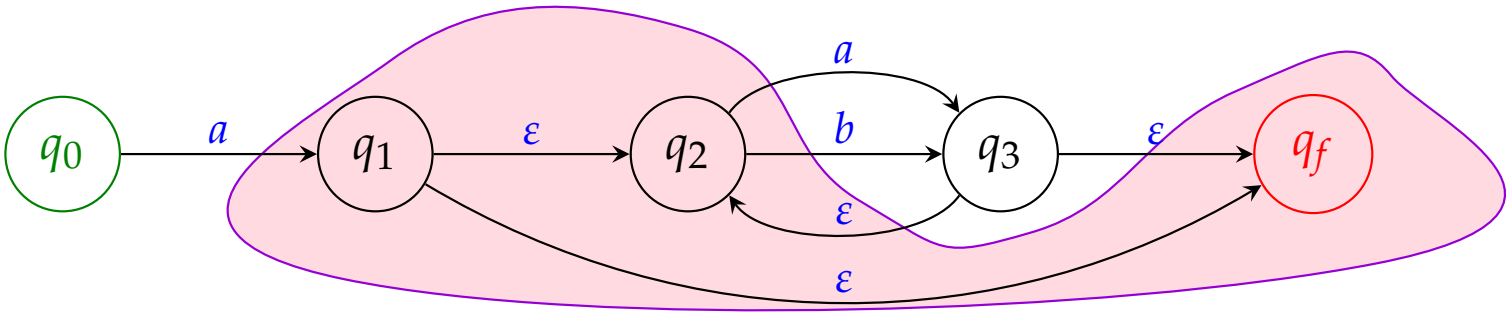
Determineeritud lõpliku automaadi koostamine

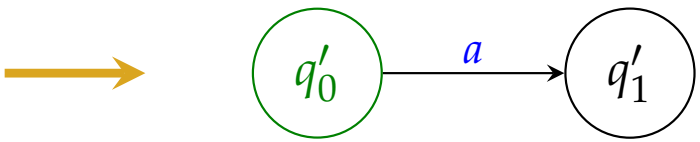
Näide:



Determineeritud lõpliku automaadi koostamine

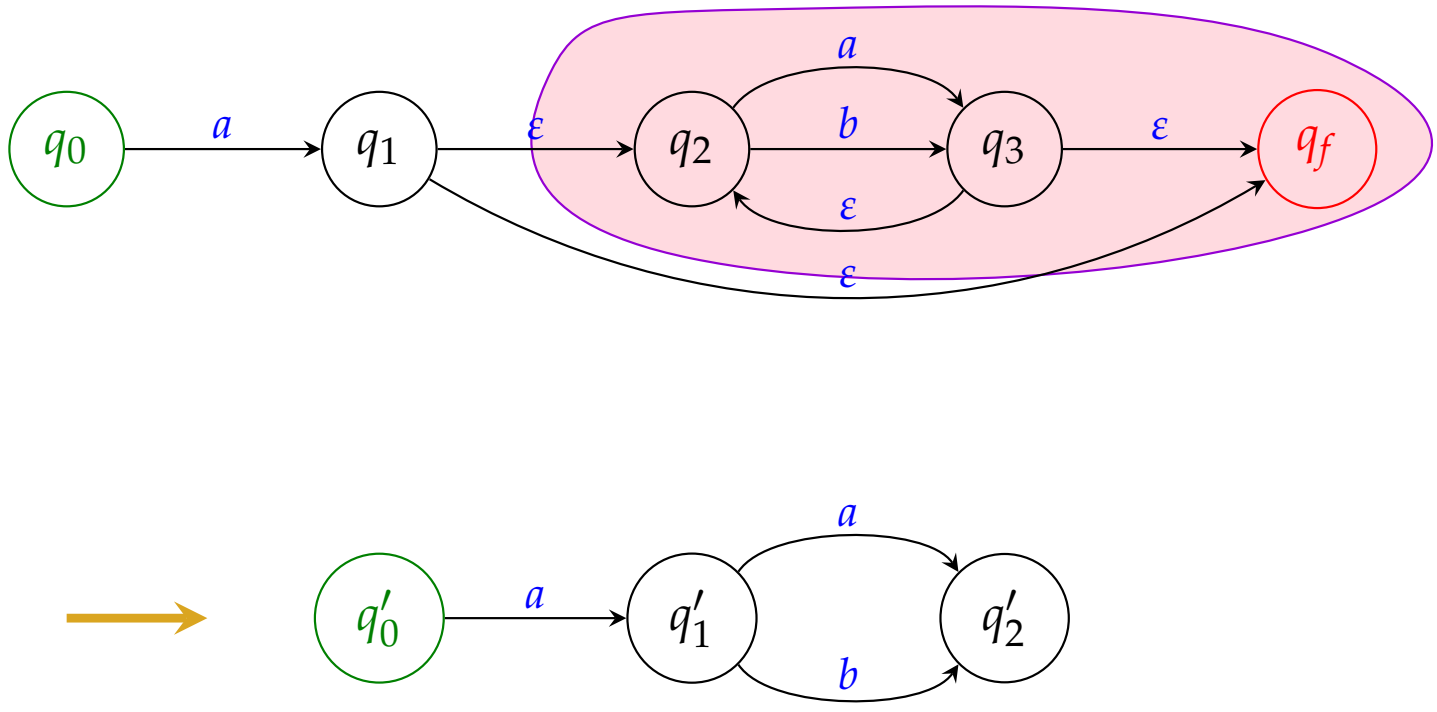
Näide:





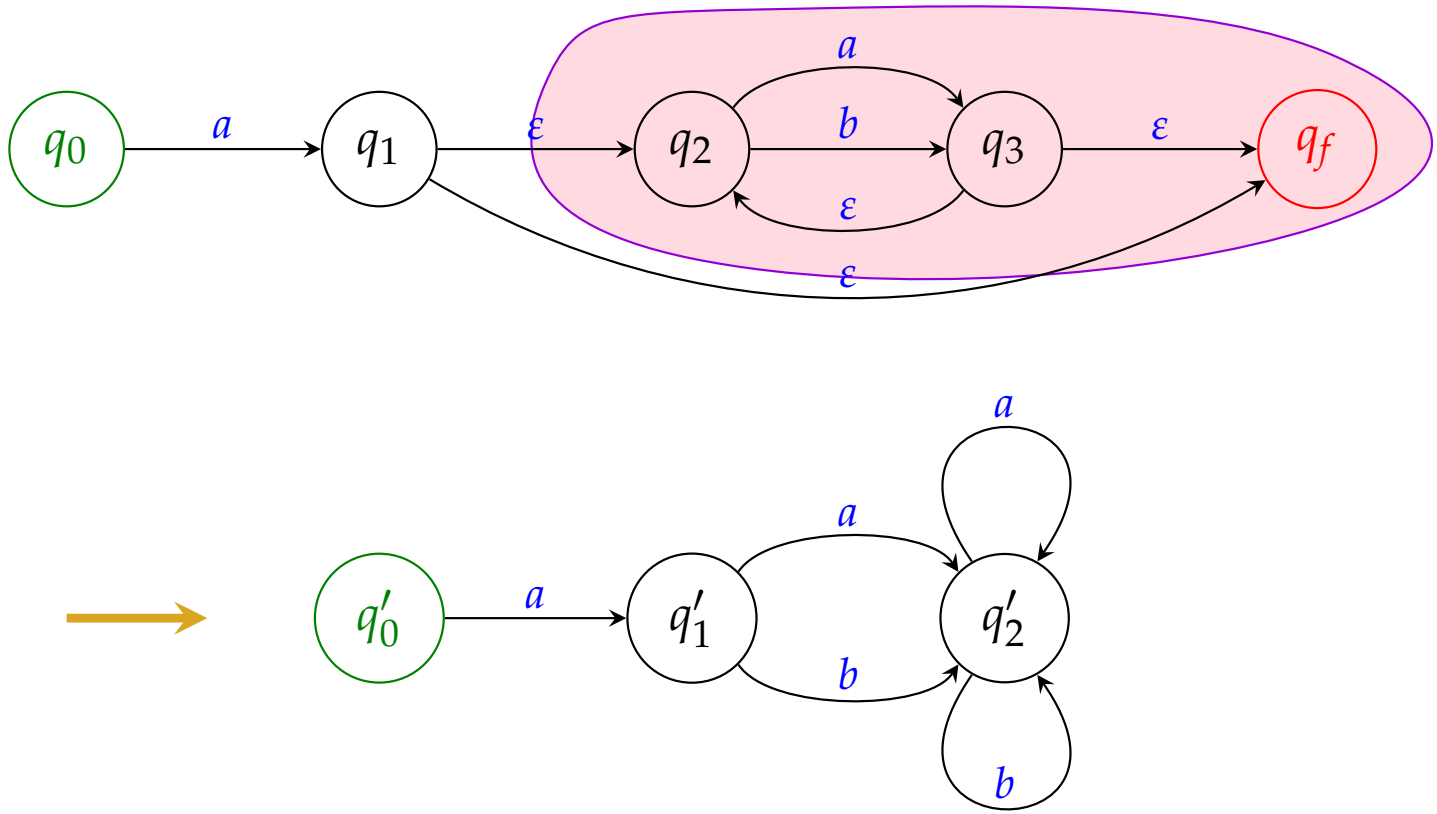
Determineeritud lõpliku automaadi koostamine

Näide:



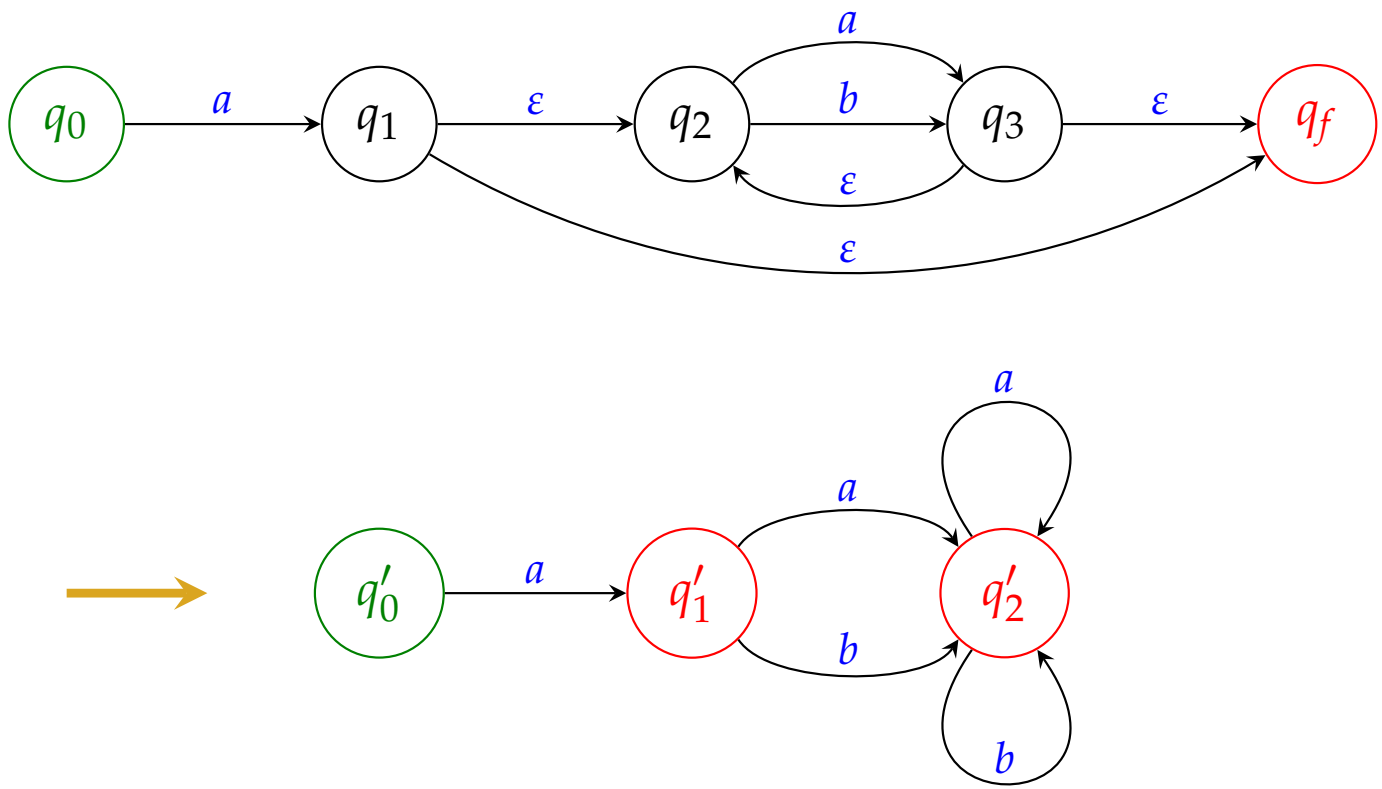
Determineeritud lõpliku automaadi koostamine

Näide:



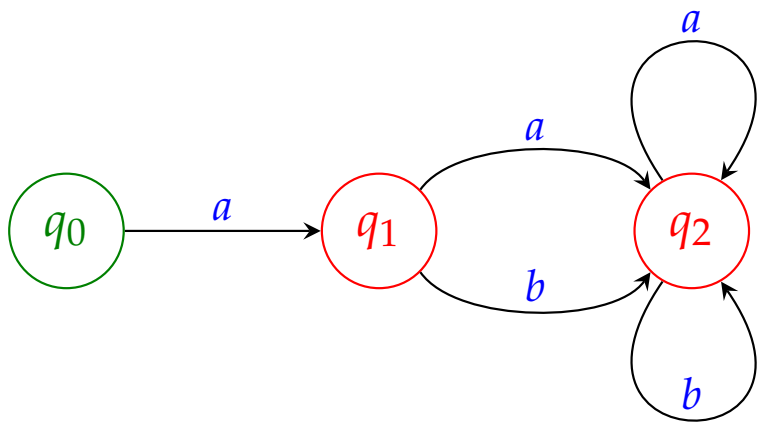
Determineeritud lõpliku automaadi koostamine

Näide:

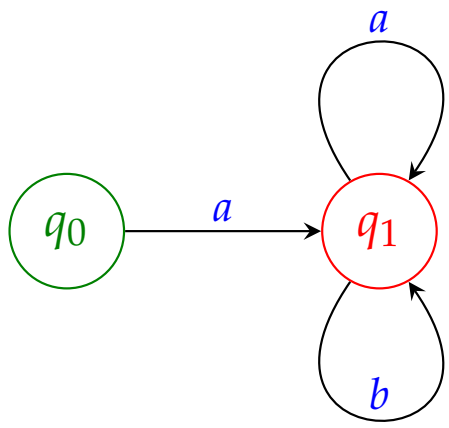


Determineeritud lõpliku automaadi minimiseerimine

- Regulaaravaldisest $a(a \mid b)^\star$ konstrueeritud determineeritud lõplik automaat:



- Temaga ekvivalentne, vähema olekute arvuga, automaat:



Determineeritud lõpliku automaadi minimiseerimine

- Determineeritud lõplik automaat on minimaalne, kui ei leidu temaga ekvivalentset, vähemate olekute arvuga, determineeritud lõplikku automaati.
- Iga determineeritud lõpliku automaadi $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ korral leidub (unikaalne) temaga ekvivalentne minimaalne determineeritud lõplik automaat $A' = \langle Q', \Sigma, \delta', q_0', F' \rangle$.
- Idee: tükeldame olekute hulga ekvivalentsiklassideks.
  - Olekud $p, q \in Q$ on ekvivalentsed ehk eristamatud, kui iga sõna $w \in \Sigma^\star$ korral automaat, alustades neist olekutest, mõlemal juhul kas õnnestub või ebaõnnestub.
  - Iga tähega üleminek viib ekvivalentsed olekud ekvivalentseteks olekuteks.

Determineeritud lõpliku automaadi minimiseerimine

Minimiseerimise algoritm:

- Eemadame kõik algolekust q_0 kättesaamatud olekud.
- Järelejäänud olekute hulgal leiame suurima tükelduse Π ekvivalentsiklassideks.
- Konstrueerime uue automaadi $A' = \langle Q', \Sigma, \delta', q_0', F' \rangle$, kus
  - olekutehulk $Q' = \Pi$;
  - algolek $q_0' = P_0$, kus $P_0 \in \Pi$ ja $q_0 \in P_0$;
  - lõppolekute hulk $F' = \{P \in \Pi \mid P \cap F \neq \emptyset\}$;
  - üleminekufunktsioon
    $\delta' = \{(P_i, a) \mapsto P_j \mid P_j \in move(P_i, a)\}$.

Determineeritud lõpliku automaadi minimiseerimine

Naiivne algoritm tükelduse leidmiseks:

$P := \{F,\ Q \setminus F\};$
do $\Pi := P;\ P := \emptyset;$
  foreach $S \in \Pi$ **do**
    foreach $a \in \Sigma$ **do**
      $U := \{T \in \Pi \mid T \cap move(S, a) \neq \emptyset\};$
      $V := \{S \cap move_a^{-1}(T) \mid T \in U\};$
      $P := P \cup V;$
    end
  end
until $\Pi = P;$

Determineeritud lõpliku automaadi minimiseerimine

- Toodud algoritm proovib igas iteratsioonis ”peenendada” kõiki tükke.
  - Halvimal juhul ruutkeerukusega.
  - Piisab, kui vaadelda ainult neid tükke, millest on võimalik ”liikuda” mõnda ”lõhenenud” tükki.
- Hopcroft’i algoritm tükelduse leidmiseks:
  - kasutab ”töölisti” veel läbi vaatamata ”peenenenud” tükkide hoidmiseks;
  - kui mõni väljaspool ”töölisti” olev tükk lõheneb, siis paigutatakse ainult üks (väiksem) alamtükk ”töölisti”.

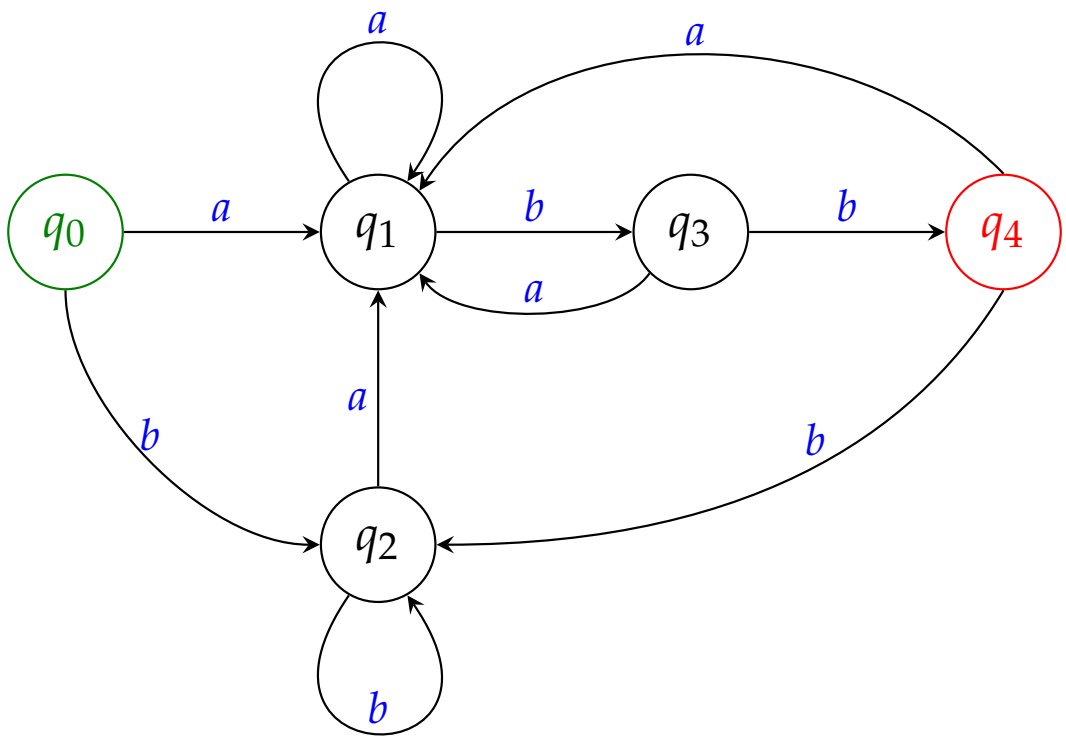
Determineeritud lõpliku automaadi minimiseerimine

Hopcroft'i algoritm:

$\Pi := \{F,\ Q \setminus F\};\ W := \Pi;$
while $\exists S \in W$ **do**
  $W := W \setminus S;$
  foreach $a \in \Sigma$ **do**
    $P := move_a^{-1}(S);$
    foreach $R \in \{T \in \Pi \mid T \cap P \neq \emptyset,\ T \not\subseteq P\}$ **do**
      $R_1 := R \cap P;\ R_2 := R \setminus R_1;$
      $\Pi := (\Pi \setminus R) \cup \{R_1, R_2\};$
      if $R \in W$ **then** $W := (W \setminus R) \cup \{R_1, R_2\};$
      else if $|R_1| \leq |R_2|$ **then** $W := W \cup \{R_1\};$
      else $W := W \cup \{R_2\};$
    end
  end
end

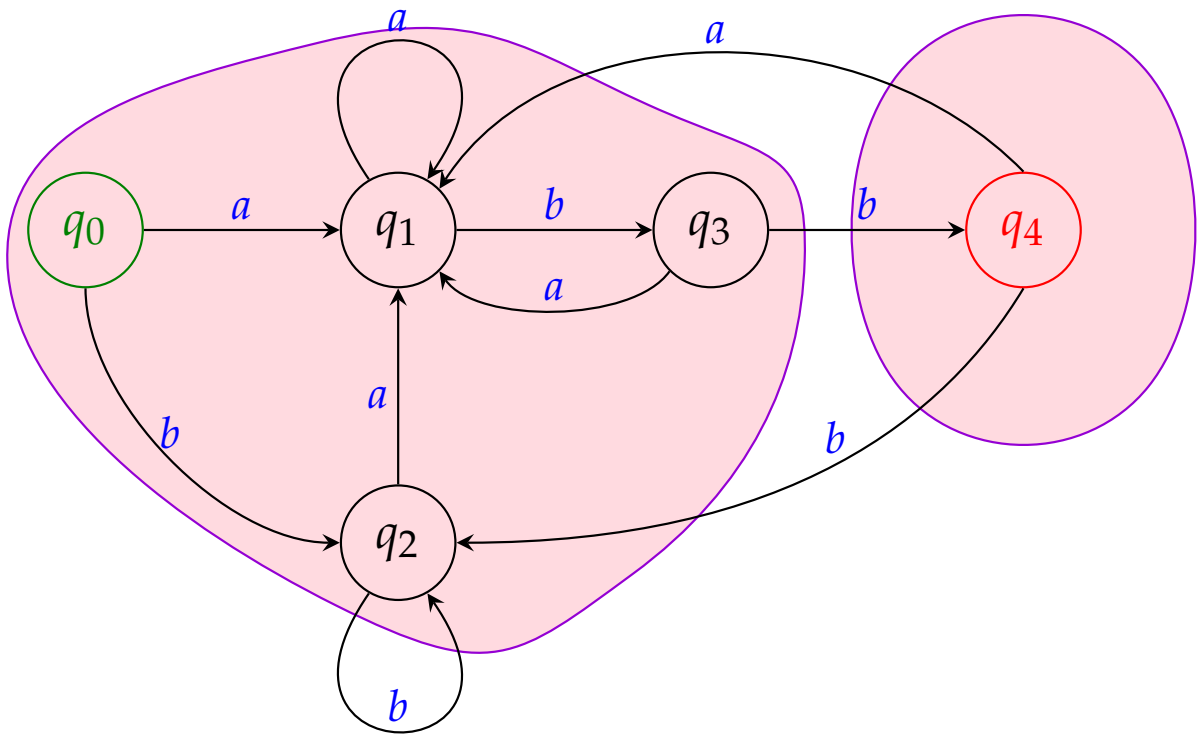
Determineeritud lõpliku automaadi minimiseerimine

Näide – regulaaravaldisele $(a \mid b)^\star abb$ vastava DFA minimiseerimine:



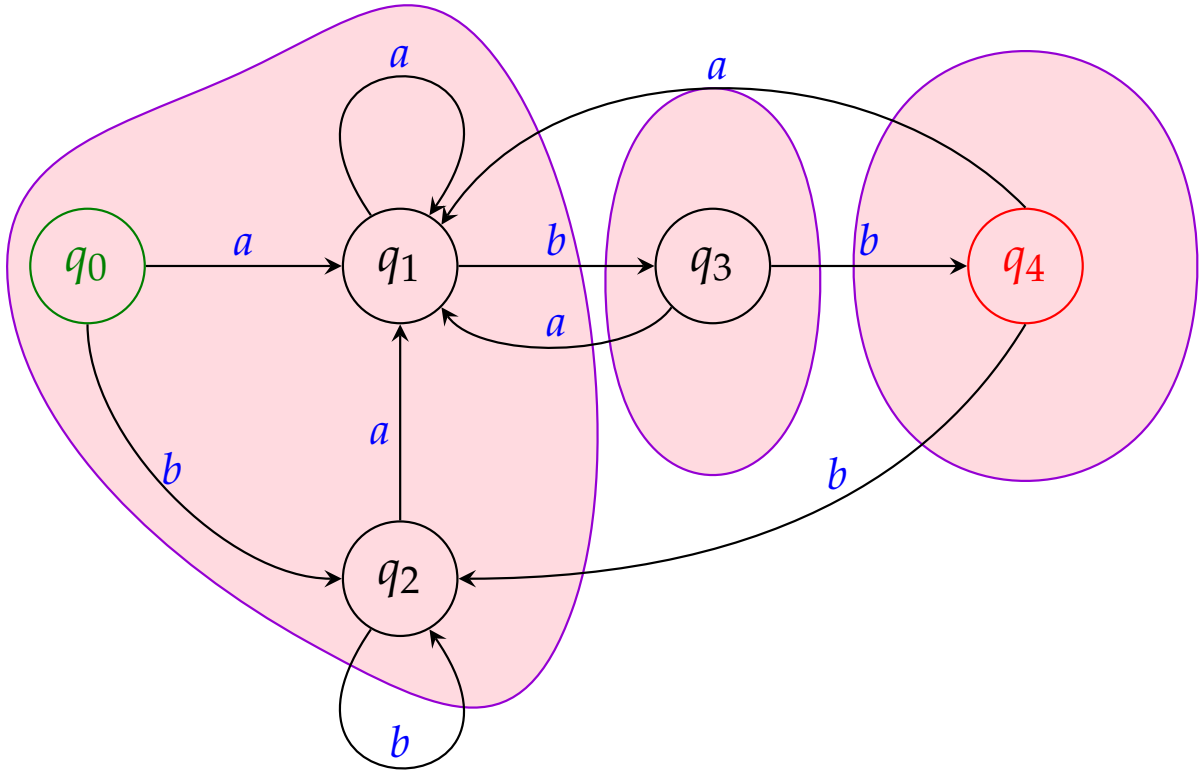
Determineeritud lõpliku automaadi minimiseerimine

Näide – regulaaravaldisele $(a \mid b)^\star abb$ vastava DFA minimiseerimine:



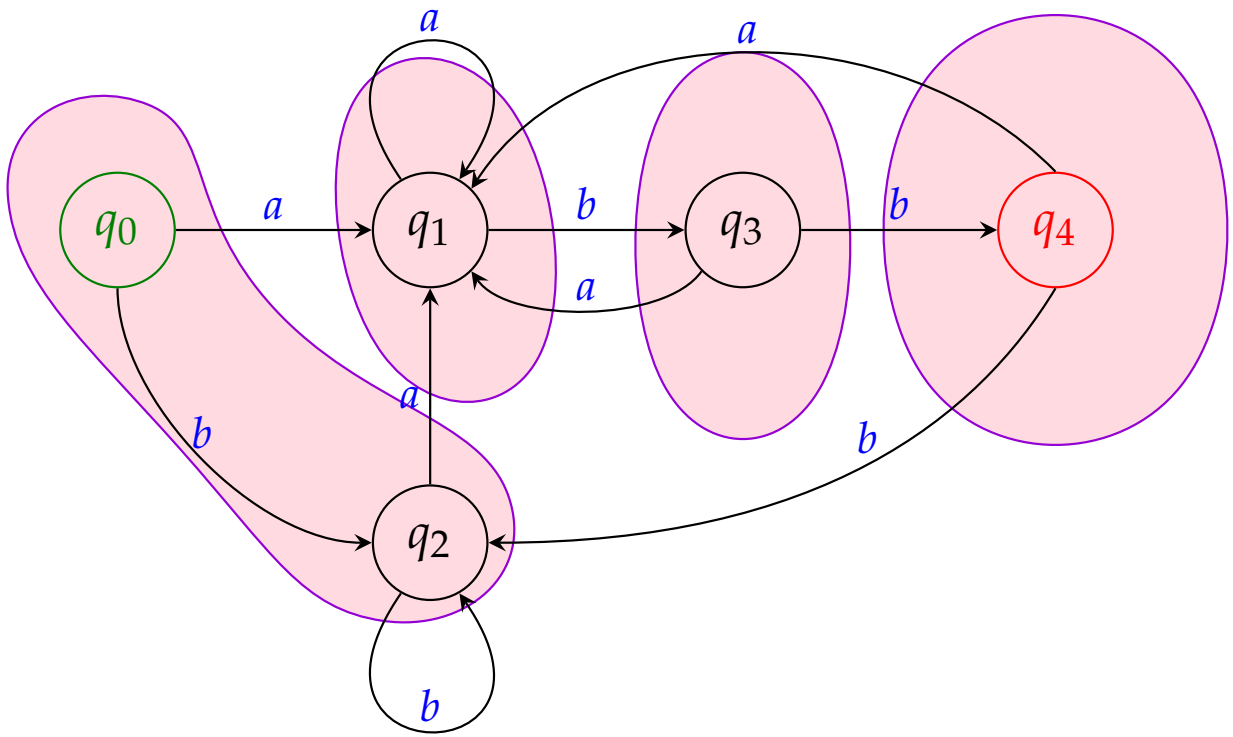
Determineeritud lõpliku automaadi minimiseerimine

Näide – regulaaravaldisele $(a \mid b)^\star abb$ vastava DFA minimiseerimine:



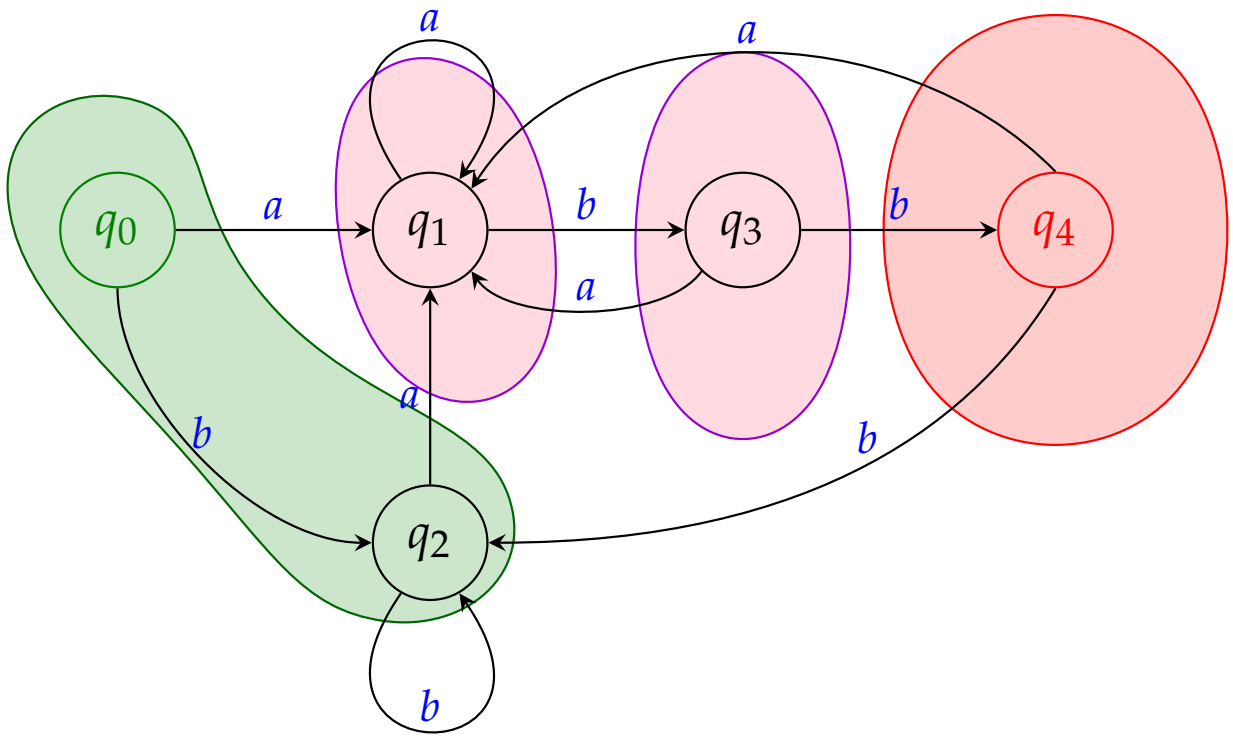
Determineeritud lõpliku automaadi minimiseerimine

Näide – regulaaravaldisele $(a \mid b)^\star abb$ vastava DFA minimiseerimine:



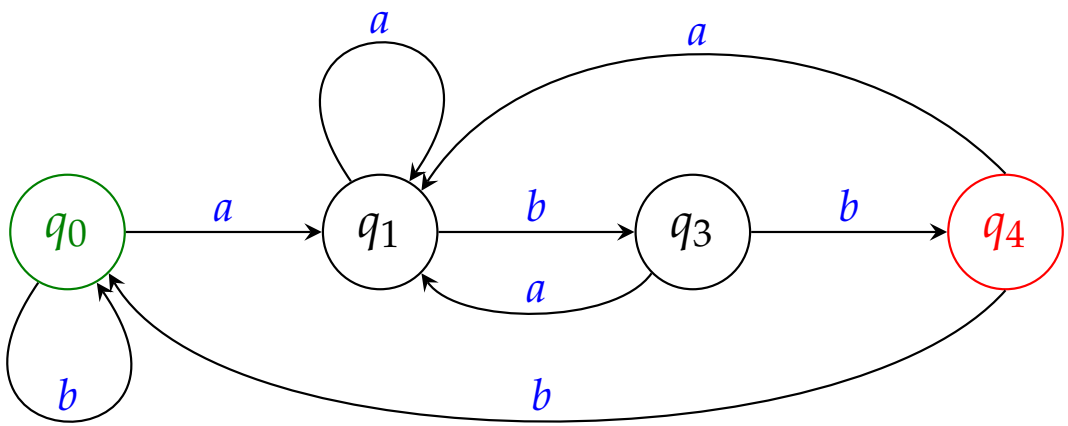
Determineeritud lõpliku automaadi minimiseerimine

Näide – regulaaravaldisele $(a \mid b)^\star abb$ vastava DFA minimiseerimine:

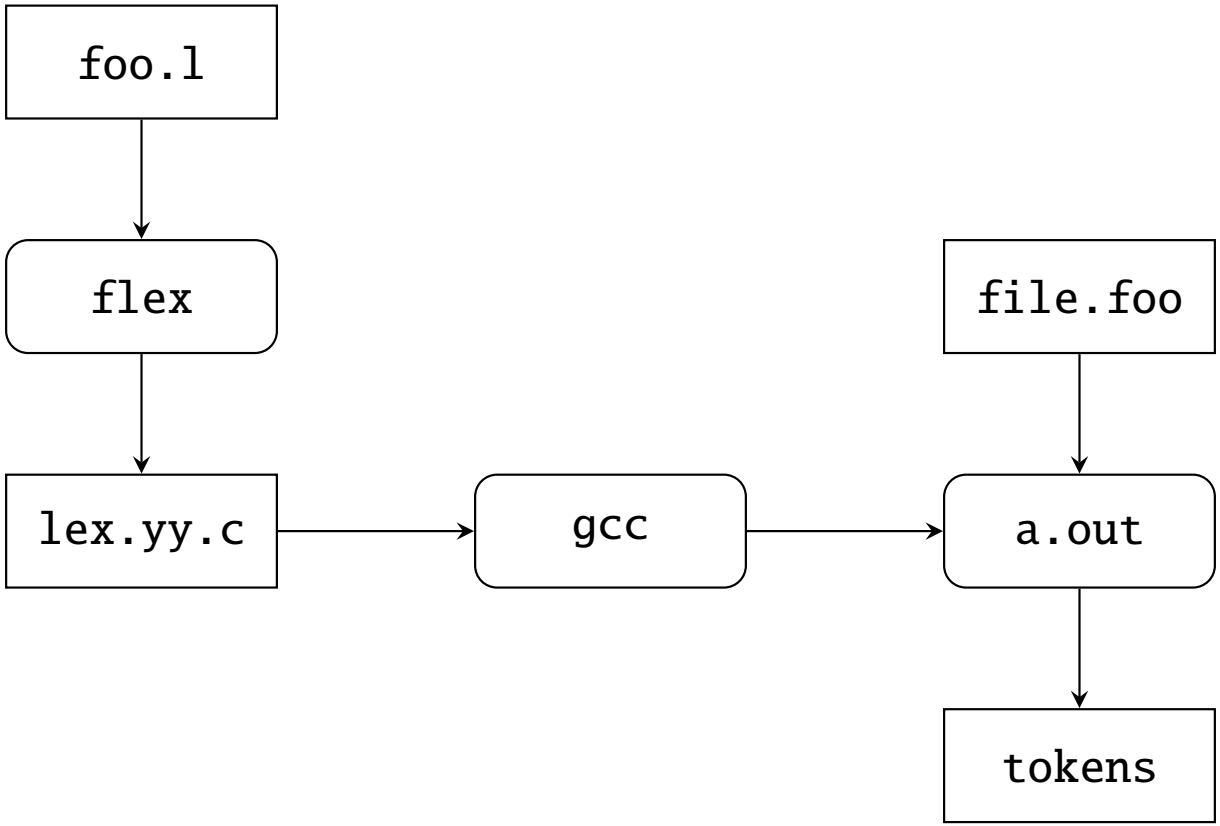


Determineeritud lõpliku automaadi minimiseerimine

Näide – regulaaravaldisele $(a \mid b)^\star abb$ vastava DFA minimiseerimine:



Skannerite generaator Flex



Skannerite generaator Flex

Sisendfaili formaat:

- Flex-i sisendfail koosneb kolmest osast:

```
definitions
%%
rules
%%
user code
```

- Definitsioonide osa koosneb:
  - C kood (kaasatavad päisfailid ja globaalsete muutujate definitsioonid);
  - regulaarsed kirjeldused;
  - algtingimuste definitsioonid.

Skannerite generaator Flex

- Reeglite osa koosneb paaride jadast kujul:

  `pattern  action`

  kus näidis peab algama ilma taandeta ning lõpeb esimese tühisümboliga; aktsioon peab algama näidisega samalt realt.
- Näidis on (laiendatud) regulaaravaldis; aktsioon on suvaline C lause.
  - Kui aktsioon on tühi, siis näidisele vastav sisend eemaldatakse.
  - Kui sisend ei sobi ühegi näidisega, siis ta kopeeritakse.
- Sisendfaili kolmas osa koosneb C koodist, mis kopeeritakse loodavasse faili `lex.yy.c` ilma ühegi muutuseta.
  - Võib puududa, millisel juhul võib ka teise eraldusrea ära jätta.

Skannerite generaator Flex

Liides parseriga suhtlemiseks:

`int yylex(void)`	peafunktsioon; väljastab leitud sõna klassi; kui faililõpp, siis 0
`char *yytext`	viit viimati skaneeritud sõnale
`int yyleng`	viimati skaneeritud sõna pikkus
`FILE *yyin`	vaikimisi loetav sisendfail
`FILE *yyout`	vaikimisi kasutatav väljundfail
`int yywrap(void)`	peaks olema defineeritud kolmandas osas; kui ei ole, siis linkimisel kasutada '`-lfl`'; reeglina väljastab lihtsalt 1
`YYSTYPE yylval`	sümboli väärtust sisaldav struktuur; defineeritud parseris (kaasata päisfail `parser.tab.h`)