Teil VI

Anwendungen, Teil 2: Pattern-Matching

Das Pattern-Matching-Problem

Pattern-Matching-Problem

Eingabe: Wörter p und t.

Gesucht: Alle Vorkommen von p in t.

Das Pattern-Matching-Problem

Pattern-Matching-Problem

Eingabe: Wörter p und t.
Gesucht: Alle Vorkommen von p in t.

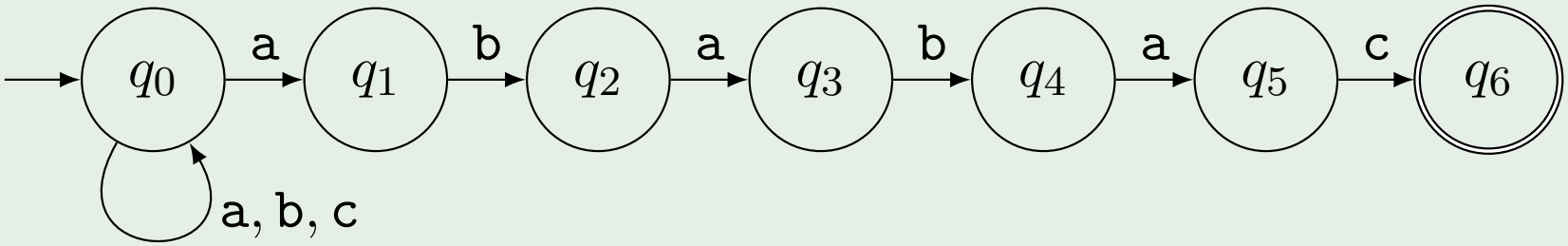
Möglicher Lösungsansatz

Verwende endliche Automaten!

Pattern-Match-NFA

Beispiel

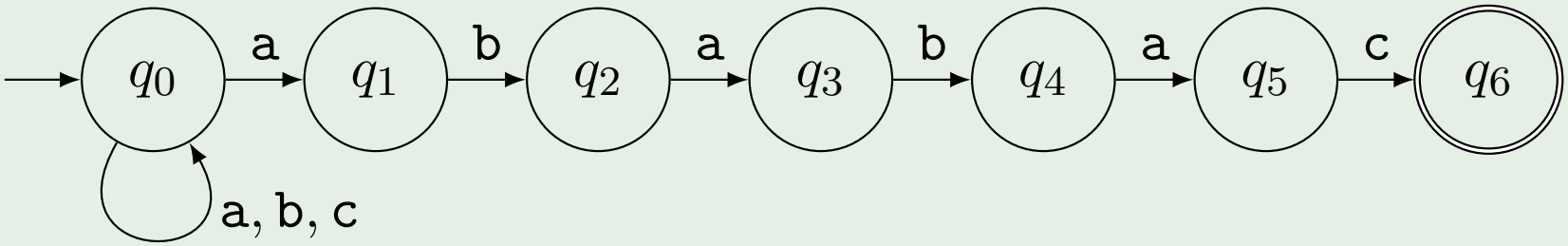
Sei $p :=$ `ababac`.



Pattern-Match-NFA

Beispiel

Sei $p := \texttt{ababac}$.



Definition

Für alle $p \in \Sigma^+$ sei

- $L_p := \Sigma^*\{p\}$
- $A_{N,p}$ kanonischer NFA für L_p.

Pattern-Match-DFA

Definition

Für alle $p \in \Sigma^+$ sei

- $A_{D,p} := \mathtt{pot}(A_{N,p})$.

Pattern-Match-DFA

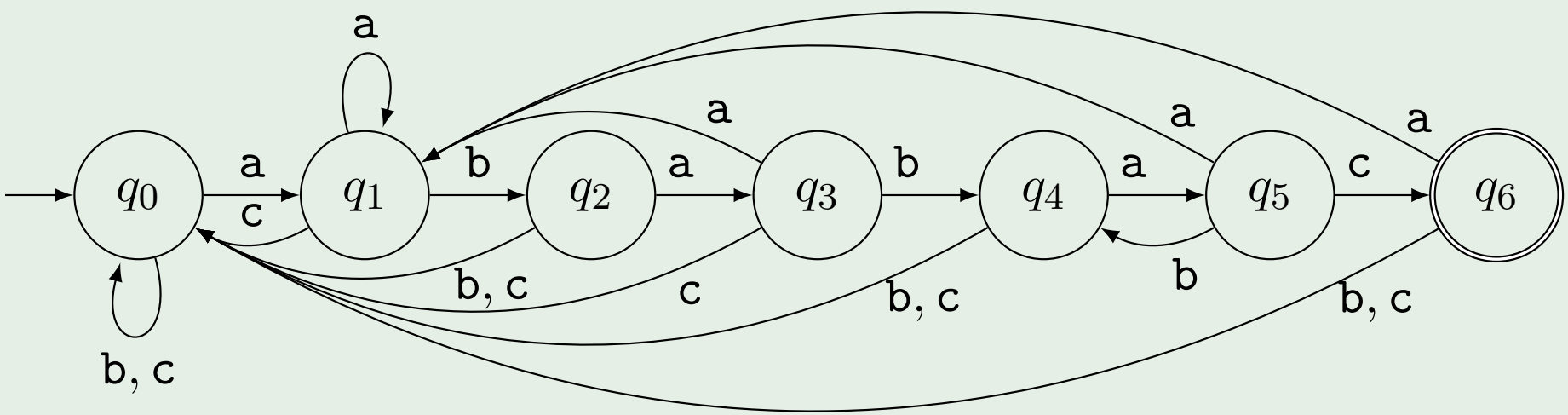
Definition

Für alle $p \in \Sigma^+$ sei

- $A_{D,p} := \texttt{pot}(A_{N,p})$.

Beispiel

Sei $p := \texttt{ababac}$.



Pattern-Match-DFA

Definition

Für alle $p \in \Sigma^+$ sei

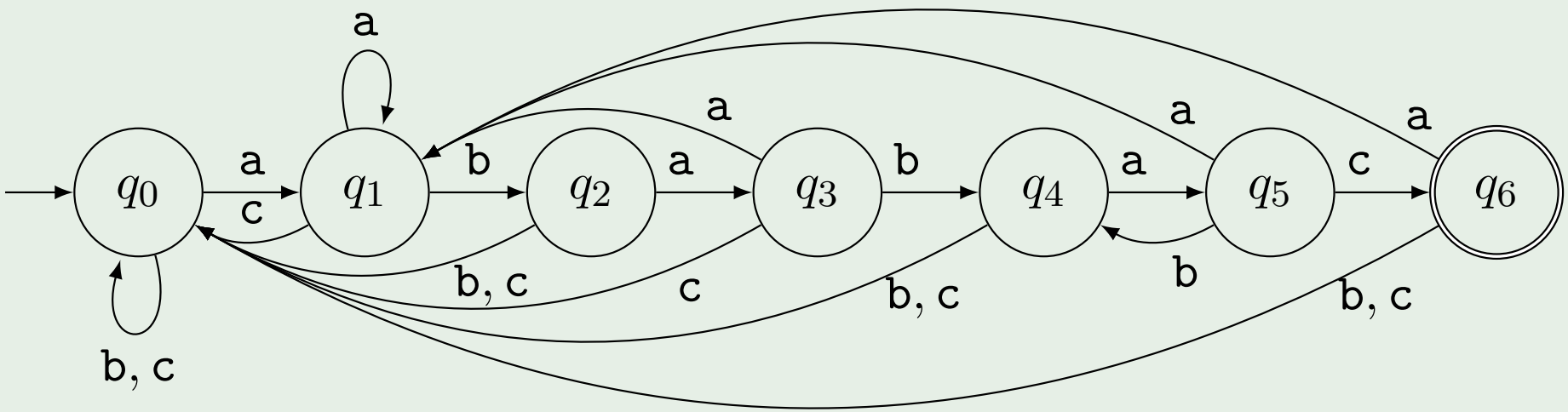
- $A_{D,p} := \texttt{pot}(A_{N,p})$.

Frage

Ist $A_{D,p}$ minimal?

Beispiel

Sei $p := \texttt{ababac}$.



Mehr zu Pattern-Match-DFAs

Lemma

Für alle $p \in \Sigma^+$ ist $A_{D,p}$ minimaler DFA für L_p.

Mehr zu Pattern-Match-DFAs

Lemma

Für alle $p \in \Sigma^+$ ist $A_{D,p}$ minimaler DFA für L_p.

Frage

Wie können wir $A_{D,p}$ direkt berechnen?

Mehr zu Pattern-Match-DFAs

Lemma

Für alle $p \in \Sigma^+$ ist $A_{D,p}$ minimaler DFA für L_p.

Frage

Wie können wir $A_{D,p}$ direkt berechnen?

Hilfsmittel

- Seien $x, y \in \Sigma^*$.
- **Maximale Überlappung von x und y:**
  Das längste Wort aus $\text{suffix}(x) \cap \text{prefix}(y)$.
- $\text{mov}(x, y)$

Mehr zu Pattern-Match-DFAs

Lemma

Für alle $p \in \Sigma^+$ ist $A_{D,p}$ minimaler DFA für L_p.

Frage

Wie können wir $A_{D,p}$ direkt berechnen?

Hilfsmittel

- Seien $x, y \in \Sigma^*$.
- **Maximale Überlappung von x und y:**
  Das längste Wort aus $\text{suffix}(x) \cap \text{prefix}(y)$.
- $\text{mov}(x, y)$

Lemma

$x \equiv_{L_p} y$ genau dann, wenn $\text{mov}(x, p) = \text{mov}(y, p)$

Pattern-Match-DFA (alternative Definitionen)

Äquivalenzklassenautomat

$$
\begin{aligned}
A_{D,p} &:= (\Sigma, Q, \delta, 0, F), \\
Q &:= \{i \in \mathbb{N} \mid 0 \leq i \leq m\}, \\
F &:= \{m\}, \\
\delta(i, a) &:= |\operatorname{mov}(p_1 \cdots p_i \cdot a, p)|
\end{aligned}
$$

Pattern-Match-DFA (alternative Definitionen)

Äquivalenzklassenautomat

$$
\begin{aligned}
A_{D,p} &:= (\Sigma, Q, \delta, 0, F), \\
Q &:= \{i \in \mathbb{N} \mid 0 \leq i \leq m\}, \\
F &:= \{m\}, \\
\delta(i, a) &:= |\operatorname{mov}(p_1 \cdots p_i \cdot a, p)|
\end{aligned}
$$

Übergangsfunktion

$$
\delta_F(i, a) := \begin{cases} i+1 & \text{falls } a = p_{i+1}, \\ 0 & \text{falls } a \neq p_{i+1} \text{ und } i = 0, \\ \delta_F(\text{fail}(i), a) & \text{falls } a \neq p_{i+1} \text{ und } i \neq 0 \end{cases}
$$

Pattern-Match-DFA (alternative Definitionen)

Äquivalenzklassenautomat

$$
\begin{aligned}
A_{D,p} &:= (\Sigma, Q, \delta, 0, F), \\
Q &:= \{i \in \mathbb{N} \mid 0 \leq i \leq m\}, \\
F &:= \{m\}, \\
\delta(i,a) &:= |\operatorname{mov}(p_1 \cdots p_i \cdot a, p)|
\end{aligned}
$$

Übergangsfunktion

$$
\delta_F(i,a) := \begin{cases} i+1 & \text{falls } a = p_{i+1}, \\ 0 & \text{falls } a \neq p_{i+1} \text{ und } i = 0, \\ \delta_F(\operatorname{fail}(i), a) & \text{falls } a \neq p_{i+1} \text{ und } i \neq 0 \end{cases}
$$

Fehler-Funktion

$$
\operatorname{fail}(i) := |\operatorname{mov}(p_2 \cdots p_i, p_1 \cdots p_{i-1})|
$$

Der Knuth-Morris-Pratt-Algorithmus

KMP

Eingabe : Wörter $p = p_1 \cdots p_m$ und $t = t_1 \cdots t_n$, $0 < m \leq n$
Ausgabe : Alle Stellen j mit $p = t_{j-m+1} \cdots t_j$

```
fail ← KMPfail(p);
i ← 0;
j ← 1;
while j ≤ n do
    while (i > 0 and p_{i+1} ≠ t_j) do i ← fail(i);
    if p_{i+1} = t_j then i ← i + 1;
    if i > m then
        output j;
        i ← fail(i);
    j ← j + 1;
```

Berechnen der fail-Funktion

KMPfail

Eingabe : Ein Wort $p = p_1 \cdots p_m$, $m \geq 1$
Ausgabe : Die Funktion fail für p

$i \leftarrow 0$;
$j \leftarrow 2$;
while $j \leq m$ **do**
    while $(i > 0$ **and** $p_{i+1} \neq p_j)$ **do** $i \leftarrow \text{fail}(i)$;
    if $p_{i+1} = p_j$ **then**
        $i \leftarrow i + 1$;
        $\text{fail}(j) \leftarrow i$
    else $\text{fail}(j) \leftarrow \text{fail}(i)$;
    $j \leftarrow j + 1$;
return fail;