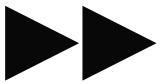
Theoretische Informatik
Einschub 1: String-Matching-Verfahren

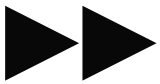


Problemstellung

Suchen von Mustern (/* Pattern */) in einem Text

Dieses Kapitel hat als Ziel, die Funktionalität, den Aufbau und die Implementierung moderner Texteditoren vorzustellen und die historische Entwicklung der Texteditoren zu beschreiben. An der historischen Entwicklung interessiert vor allem, auf welchen Wegen (und Umwegen) sich das heutige konzeptuelle Editormodell entwickelt hat. Die ersten Generationen von Texteditoren gehorchten nämlich anderen Modellen, oft mangels der Einsicht der Entwickler in die Möglichkeiten der rechnergestützten Textverarbeitung. Da wurden eher Schreibmaschinen,

Theoretische Informatik
Einschub 1: String-Matching-Verfahren



Problemstellung (/ formal */)*

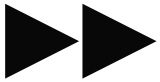
Notationen

- zugrunde liegendes Alphabet Σ
- Text T = T[1] ... T[n]
- Muster (/* Pattern */) P = P[1] ... P[m]

Aufgabe

- gegeben:
  - Text T und Pattern P
- gesucht:
  - alle $s \leq n-m$ mit:
    $T[s] = P[1], T[s+1] = P[2], ..., T[s+m-1] = P[m]$

Theoretische Informatik
Einschub 1: String-Matching-Verfahren



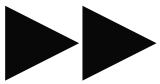
Naiver Algorithmus

```
for ( s = 1; s <= n; ++s )
        if ( test(s) == true ) return(s);

bool test(s) {
        result = true;
        for ( i = 0; i <= m-1; ++ i )
                if ( T[s+i] != P[i+1] ) result = false;
         return(result);
}
```

im „worst case“ geht das mit O(nm)-vielen Vergleichen

Theoretische Informatik
Einschub 1: String-Matching-Verfahren



String-Matching mit endlichen Automaten

Ansatzmöglichkeit für eine Verbesserung

das Pattern um möglichst mehr als eine Position nach rechts verschieben, falls Pattern und Text nicht übereinstimmen

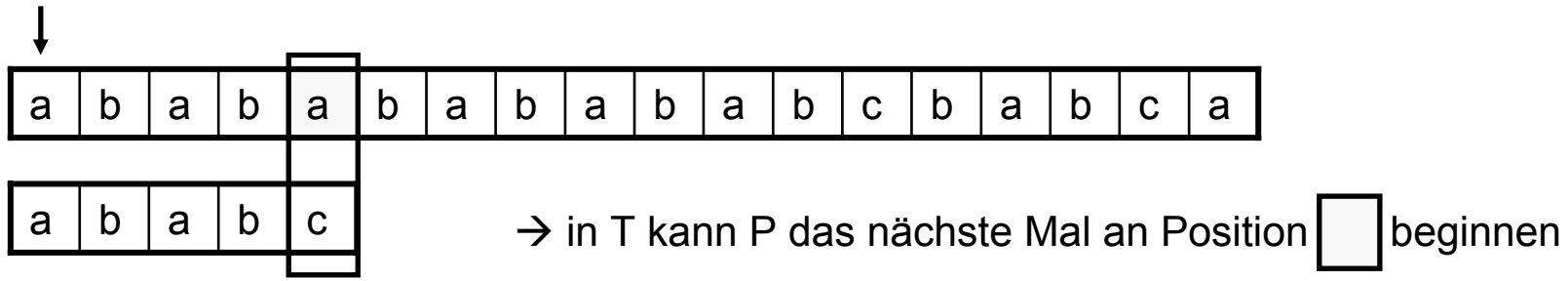
- um wie viele Positionen verschoben wird, hängt ab
  - von der Position im Pattern, bis zu welcher Text und Pattern übereinstimmen
  - vom Zeichen im Text, das als nächstes gelesen wird

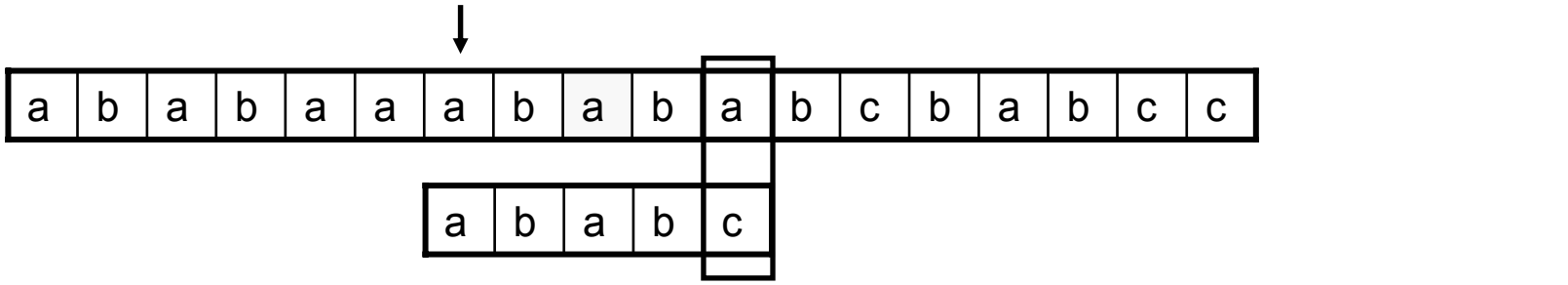
Theoretische Informatik
Einschub 1: String-Matching-Verfahren

String-Matching mit endlichen Automaten

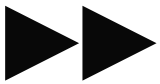
Ansatzmöglichkeiten für eine Verbesserung

das Pattern um möglichst mehr als eine Position nach rechts verschieben, falls Pattern und Text nicht übereinstimmen





Theoretische Informatik
Einschub 1: String-Matching-Verfahren



String-Matching mit endlichen Automaten

Realisierung der Idee

es seien das Pattern P = P[1] ... P[m] und der Text T = T[1] ... T[n] gegeben

suchen einen deterministischen endlichen Automat A, der folgende Eigenschaften hat

- A bekommt den Text T[1] ... T[n] als Eingabe
- A hat die Zustände 0, ..., m
- wenn sich A nach Lesen des Zeichens T[s] im Zustand z ($z \geq 1$) befindet, soll gelten: T[s-z+1] T[s-z+2] ... T[s] = P[1] ... P[z]

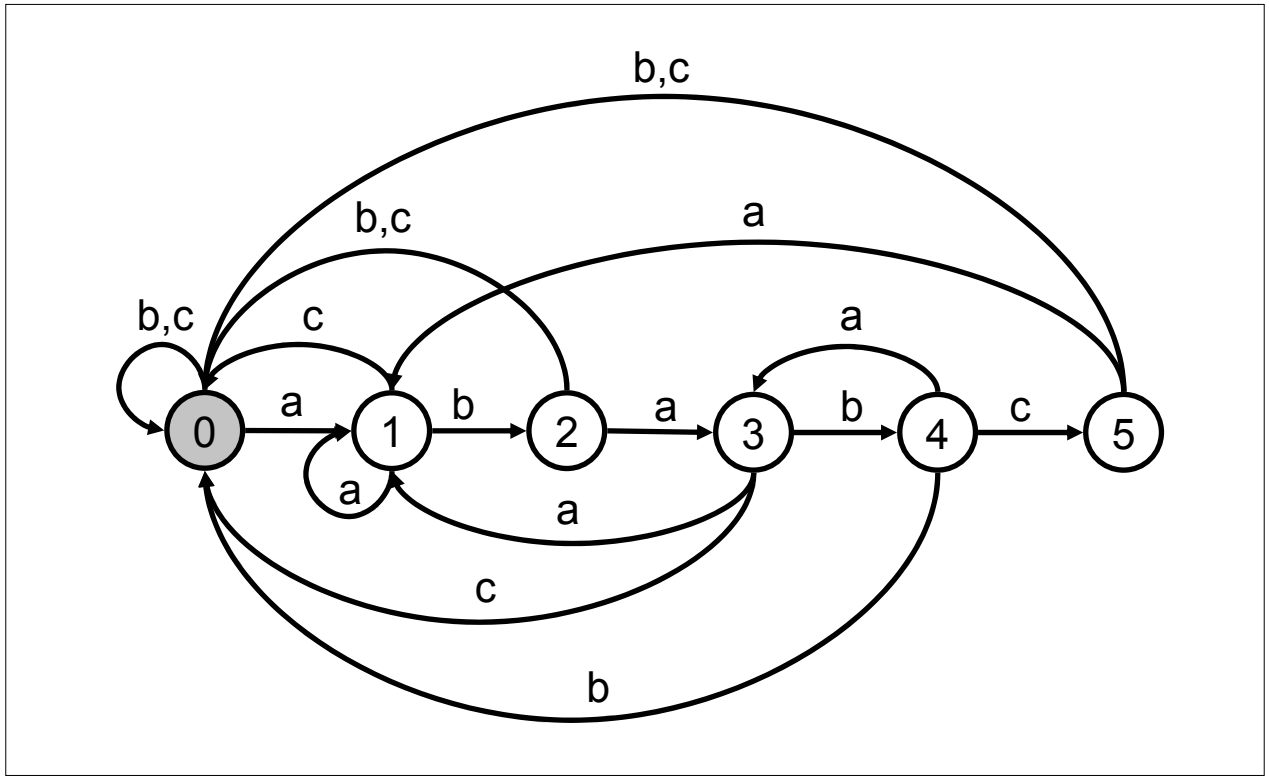
... wenn sich also A im Zustand m befindet, so beginnt an der der Stelle T[s-m+1] das Pattern P im Text T

Theoretische Informatik
Einschub 1: String-Matching-Verfahren

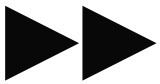
String-Matching mit endlichen Automaten

Automat für das Pattern

a	b	a	b	c



Theoretische Informatik
Einschub 1: String-Matching-Verfahren



String-Matching mit endlichen Automaten

```
/* berechne den Automaten A */
j = 0;
for ( s = 1; s <= n-m; ++ s ) {
      x = T[s];
      j = δ(j,x);
      if ( j == m ) return( s - m + 1 );
}
```

im „worst case“ geht das mit O(n) + V(m,card(Σ))) vielen Vergleichen

Anzahl der Vergleiche zur Konstruktion von A

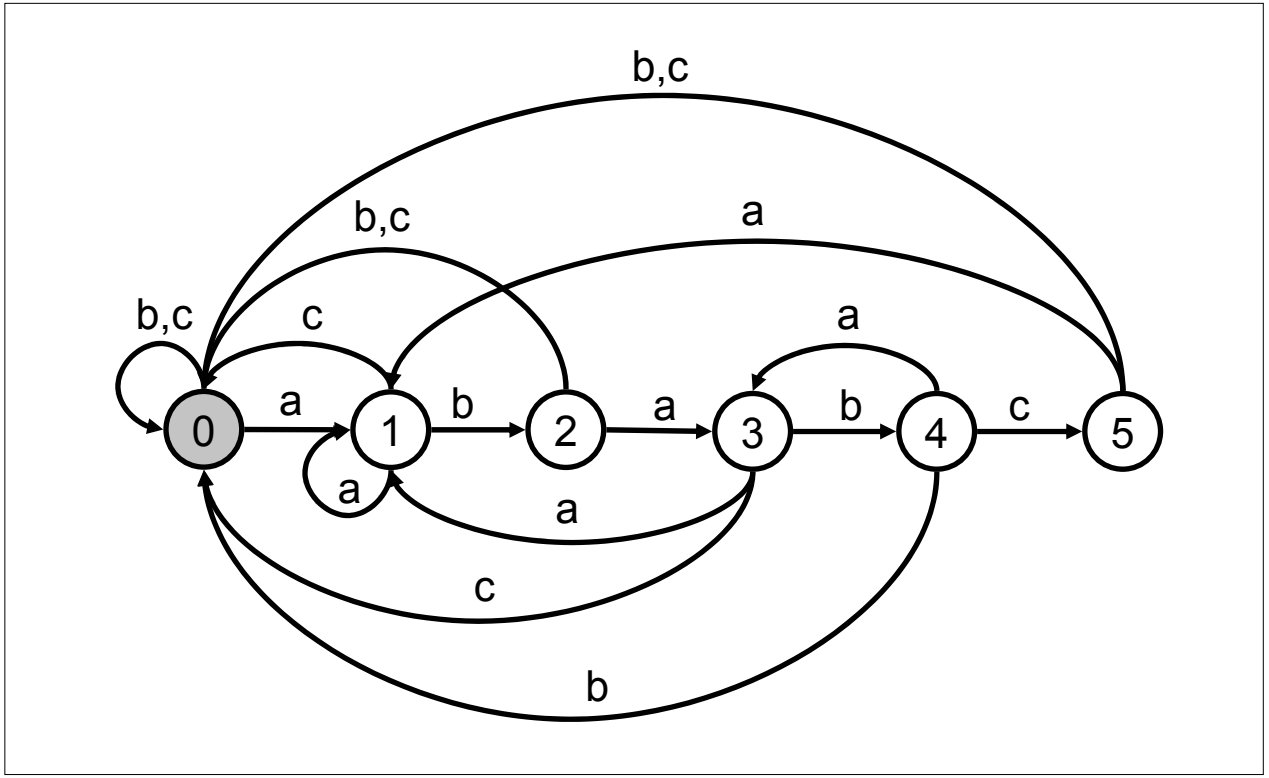
Theoretische Informatik
Einschub 1: String-Matching-Verfahren

String-Matching mit endlichen Automaten

Konstruktion des Automaten für das Pattern

a	b	a	b	c

einfache Fälle

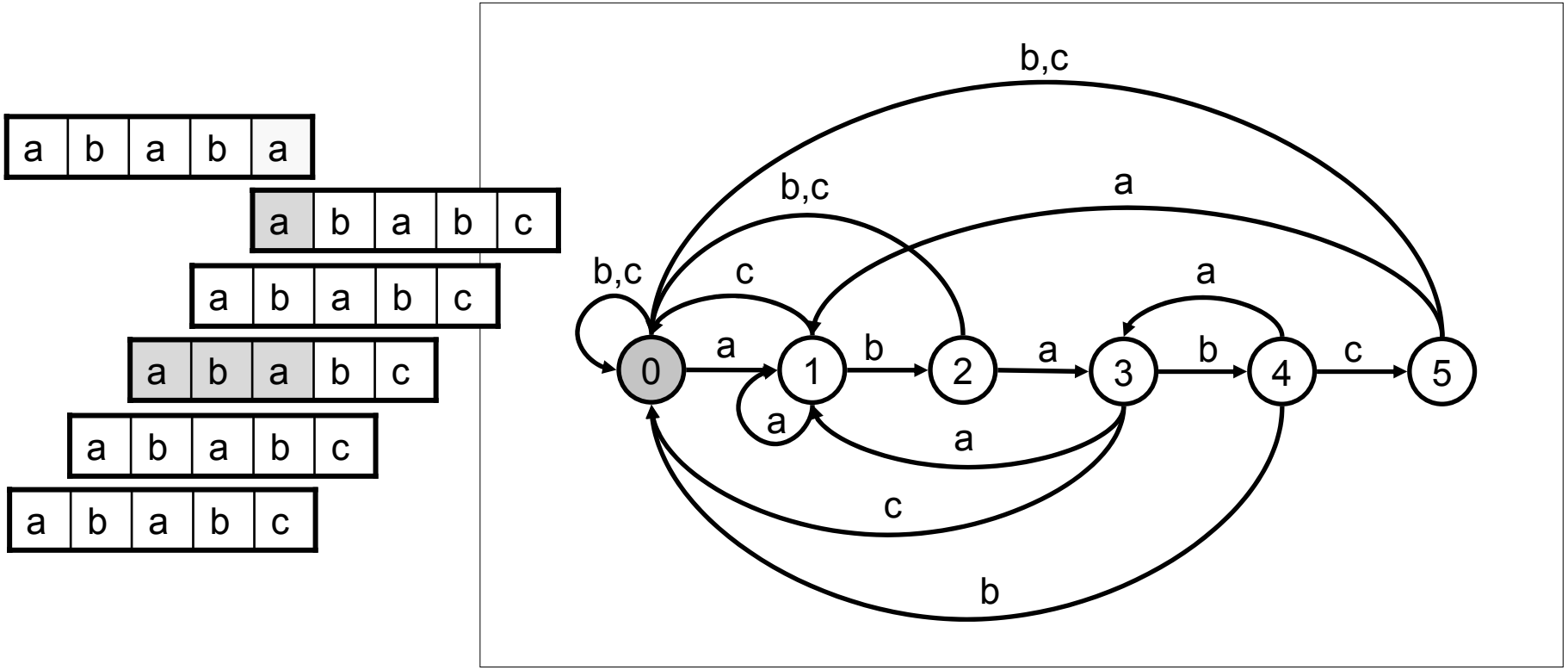


Theoretische Informatik
Einschub 1: String-Matching-Verfahren

String-Matching mit endlichen Automaten

Konstruktion des Automaten für das Pattern

a	b	a	b	c

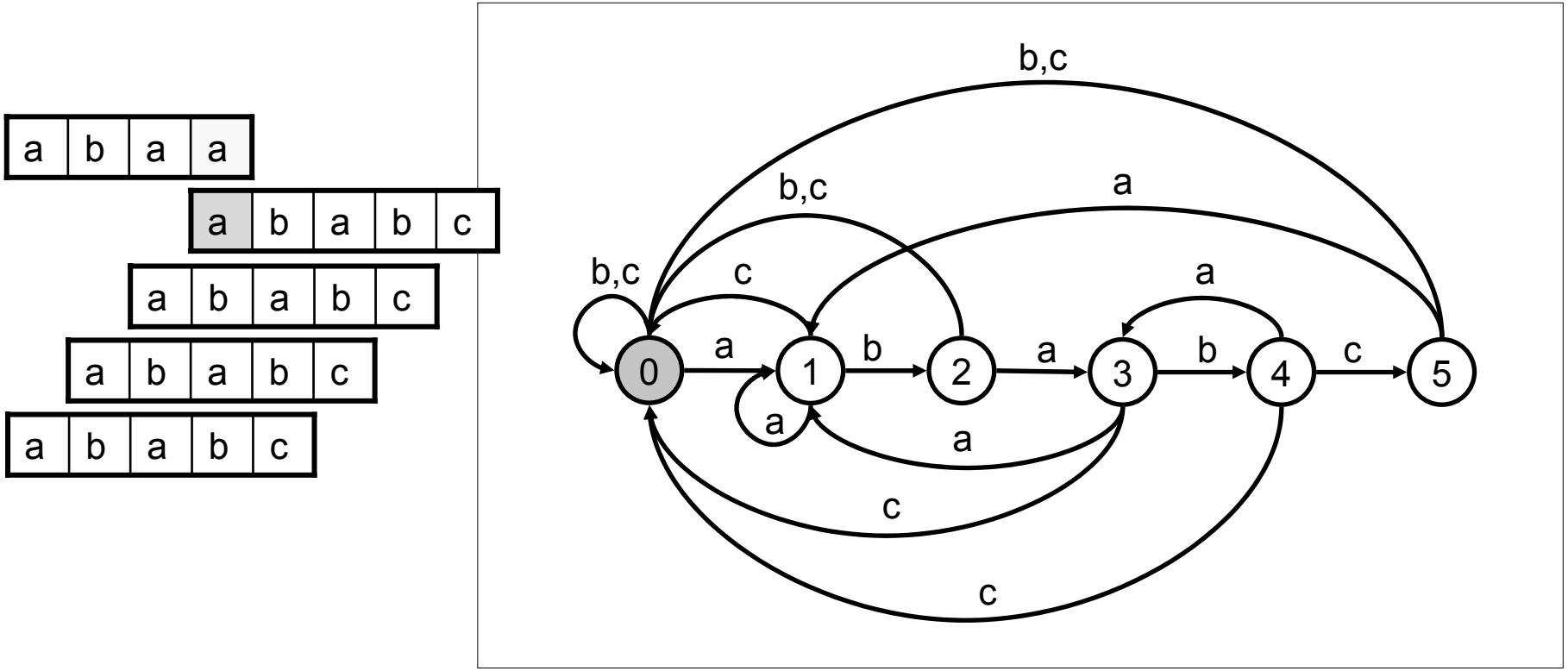


Theoretische Informatik
Einschub 1: String-Matching-Verfahren

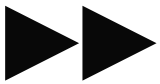
String-Matching mit endlichen Automaten

Konstruktion des Automaten für das Pattern

a	b	a	b	c



Theoretische Informatik
Einschub 1: String-Matching-Verfahren



Grundlegende Algorithmen

Konstruktion des Automaten (/* allgemein */)

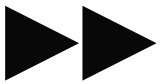
es sei das Pattern P = P[1] ... P[m] gegeben
es seien $z \in \{ 0, ..., m \}$ und $x \in \Sigma$

dann bestimmt sich $\delta(z,x)$ wie folgt:

Fall 1: z < m und x = P[z+1]

setze $\delta(z,x) = z+1$

Theoretische Informatik
Einschub 1: String-Matching-Verfahren



Grundlegende Algorithmen

Konstruktion des Automaten (/* allgemein */)

es sei das Pattern P = P[1] ... P[m] gegeben
es seien $z \in \{ 0, ..., m \}$ und $x \in \Sigma$

dann bestimmt sich $\delta(z,x)$ wie folgt:

Fall 2: sonst

- bestimme das maximale $k \geq 1$, so daß P[1] ... P[k] Suffix von P[1] ... P[z]x
- Falls k gefunden wurde, so setze $\delta(z,x) = k$
- sonst setze $\delta(z,x) = 0$

Hinweis: für z = 0 gilt P[1] ... P[z]x = x

Konstruktion des Automaten (allgemein)

wenn man den Automaten wie erläutert konstruiert, so geht daß mit höchstens $O(m^3 \text{ card}(\Sigma))$ vielen Vergleichen

... wenn man den Automaten „cleverer“ konstruiert, so geht daß sogar mit höchsten $O(m \text{ card}(\Sigma))$ vielen Vergleichen