

# Algorithmen auf Sequenzen

## Erweiterte Muster

Sven Rahmann

Genominformatik  
Universitätsklinikum Essen  
Universität Duisburg-Essen  
Universitätsallianz Ruhr

Bisher wurden nur einfache Muster (Strings) behandelt.

In vielen Anwendungen ist es interessant, Muster zu betrachten, bei denen an bestimmten Stellen

- verschiedene Zeichen (oder beliebige Zeichen) stehen dürfen,
- oder Folgen beliebiger Zeichen mit variabler (beschränkter) Länge,
- oder Muster mit optionalen Zeichen.

Bisher wurden nur einfache Muster (Strings) behandelt.

In vielen Anwendungen ist es interessant, Muster zu betrachten, bei denen an bestimmten Stellen

- verschiedene Zeichen (oder beliebige Zeichen) stehen dürfen,
- oder Folgen beliebiger Zeichen mit variabler (beschränkter) Länge,
- oder Muster mit optionalen Zeichen.

Diese Muster sind eine Teilmenge von regulären Ausdrücken, so dass man aufbauend effiziente Algorithmen zum Suchen nach regulären Ausdrücken (grep-Tool) entwickeln kann.

Bisher wurden nur einfache Muster (Strings) behandelt.

In vielen Anwendungen ist es interessant, Muster zu betrachten, bei denen an bestimmten Stellen

- verschiedene Zeichen (oder beliebige Zeichen) stehen dürfen,
- oder Folgen beliebiger Zeichen mit variabler (beschränkter) Länge,
- oder Muster mit optionalen Zeichen.

Diese Muster sind eine Teilmenge von regulären Ausdrücken, so dass man aufbauend effiziente Algorithmen zum Suchen nach regulären Ausdrücken (grep-Tool) entwickeln kann.

Alle hier genannten Erweiterungen lassen sich durch Modifikationen des Shift-And-Algorithmus realisieren.

# Verallgemeinerte Strings

- Verallgemeinerte Strings über  $\Sigma$  sind Strings, die *Teilmengen* von  $\Sigma$  als Zeichen besitzen, also Strings über  $2^\Sigma$ .
- Mustermenge  $\{\text{Meier}, \text{Meyer}\} \mapsto \text{Me}[\text{iy}]\text{er}$ ;  
M Abkürzung für Menge  $\{M\}$ ;  $[\text{iy}]$  für  $\{i, y\}$
- Wir schreiben  $\#$  für  $\Sigma \in 2^\Sigma$  (beliebiges Zeichen aus  $\Sigma$ )

# Verallgemeinerte Strings

- Verallgemeinerte Strings über  $\Sigma$  sind Strings, die *Teilmengen* von  $\Sigma$  als Zeichen besitzen, also Strings über  $2^\Sigma$ .
- Mustermenge  $\{\text{Meier}, \text{Meyer}\} \mapsto \text{Me}[\text{iy}]\text{er}$ ;  
M Abkürzung für Menge  $\{M\}$ ;  $[\text{iy}]$  für  $\{i, y\}$
- Wir schreiben  $\#$  für  $\Sigma \in 2^\Sigma$  (beliebiges Zeichen aus  $\Sigma$ )
- Erweiterung des Shift-And-Algorithmus durch Anpassung der Bitmasken

# Verallgemeinerte Strings

- Verallgemeinerte Strings über  $\Sigma$  sind Strings, die *Teilmengen* von  $\Sigma$  als Zeichen besitzen, also Strings über  $2^\Sigma$ .
- Mustermenge  $\{\text{Meier}, \text{Meyer}\} \mapsto \text{Me}[\text{iy}]\text{er}$ ;  
M Abkürzung für Menge  $\{M\}$ ;  $[\text{iy}]$  für  $\{i, y\}$
- Wir schreiben  $\#$  für  $\Sigma \in 2^\Sigma$  (beliebiges Zeichen aus  $\Sigma$ )
- Erweiterung des Shift-And-Algorithmus durch Anpassung der Bitmasken
- Beispiel:  $P = \text{abba}\#b$  mit  $\Sigma = \{a, b\}$ .

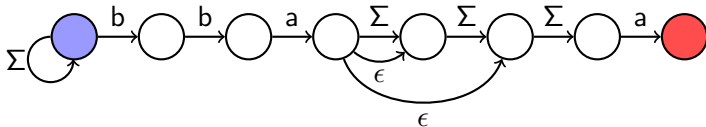
	$b\#abba$
$mask^a$	011001
$mask^b$	110110

- Unter beliebigen Zeichenfolgen beschränkter Länge verstehen wir Folgen von konsekutiven  $\Sigma$  (als  $\#$  geschrieben) mit fest definierter unterer und oberer Schranke für die Länge.
- $P = \text{bba}\#(1, 3) a$ :  
1 bis 3 beliebige Zeichen; 1 notwendig, 2 optional



# Beliebige Zeichenfolgen beschränkter Länge

- Unter beliebigen Zeichenfolgen beschränkter Länge verstehen wir Folgen von konsekutiven  $\Sigma$  (als  $\#$  geschrieben) mit fest definierter unterer und oberer Schranke für die Länge.
- $P = \text{bba}\#(1, 3)\text{a}$ :  
1 bis 3 beliebige Zeichen; 1 notwendig, 2 optional
- Realisierung mit  $\epsilon$ -Übergängen im NFA,  
 $\epsilon$ -Kanten gehen vom **Beginn** des Elements aus!



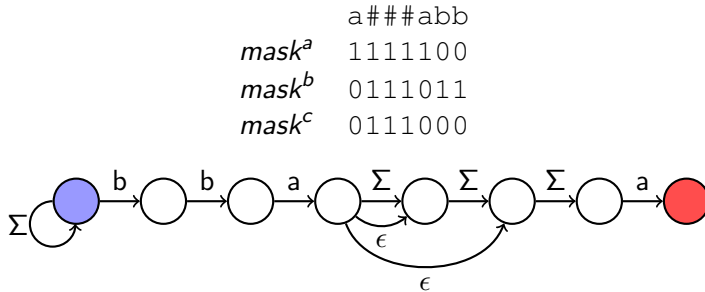
Einschränkungen:

- 1 Elemente  $\#(,)$  nicht an erster oder letzter Stelle im Muster
- 2 Nicht zwei solche Elemente hintereinander  $[\#(u, v)\#(u', v') \hat{=} \#(u + u', v + v')]$ .
- 3 Für  $\#(u, v)$  muss gelten:  $1 \leq u \leq v$  (Fall  $u = 0$  aufwändiger!)

# Implementierung

Für die Zeichenmasken gilt: Für das Element  $\#(u, v)$  werden für jedes Zeichen aus dem Alphabet genau  $v$  1en hinzugefügt.

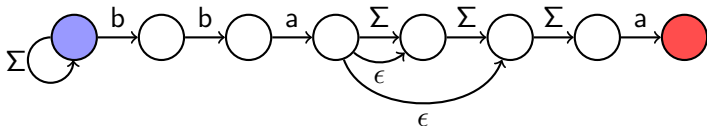
Beispiel mit  $P = bba\#(1, 3)a$  und  $\Sigma = \{a, b, c\}$ :



- Bitmaske  $I$  enthält Zustände, von denen  $\epsilon$ -Kanten ausgehen.
- Bitmaske  $F$  enthält Zustände **hinter** dem Ziel der letzten  $\epsilon$ -Kante jedes Elements

	a###abb
$F$	0100000
$I$	0000100

	a###abb
$mask^a$	1111100
$mask^b$	0111011
$mask^c$	0111000



# Implementierung

- Um festzustellen, ob ein  $I$ -Zustand aktiv ist, wird  $I$  mit der aktiven Zustandsmenge verundet:  $A \& I$ .
- Durch Subtraktion  $F - (A \& I)$  werden alle Zustände, die mit  $\epsilon$ -Kanten von aktiven  $I$ -Zuständen erreicht werden, aktiviert:

$$\begin{array}{r}
 F \quad \quad 0100000 \\
 A \& I \quad 0000100 \\
 \hline
 - \quad \quad 0011100
 \end{array}$$

# Implementierung

- Um festzustellen, ob ein  $I$ -Zustand aktiv ist, wird  $I$  mit der aktiven Zustandsmenge verundet:  $A \& I$ .
- Durch Subtraktion  $F - (A \& I)$  werden alle Zustände, die mit  $\epsilon$ -Kanten von aktiven  $I$ -Zuständen erreicht werden, aktiviert:

$$\begin{array}{r} F \quad \quad 0100000 \\ A \& I \quad 0000100 \\ \hline - \quad \quad 0011100 \end{array}$$

- Problem, wenn ein  $I$ -Zustand nicht aktiv ist:  
Der zugehörige  $F$ -Zustand bleibt gesetzt.

$$\begin{array}{r} F \quad \quad 010000100000 \\ A \& I \quad 000000000100 \\ \hline - \quad \quad 010000011100 \end{array}$$

- Lösung: Durch Negation von  $F$  wird eine Maske erzeugt, mit der wir unerwünschte  $F$ -Bits löschen:

$F$	010000100000
$A \& I$	000000000100
<hr/>	
$F - (A \& I)$	010000011100

- Lösung: Durch Negation von  $F$  wird eine Maske erzeugt, mit der wir unerwünschte  $F$ -Bits löschen:

$F$	010000100000
$A \& I$	000000000100
<hr/>	
$F - (A \& I)$	010000011100
$\sim F$	101111011111
<hr/>	
$(F - (A \& I)) \& \sim F$	000000011100



- Lösung: Durch Negation von  $F$  wird eine Maske erzeugt, mit der wir unerwünschte  $F$ -Bits löschen:

$F$	010000100000
$A \& I$	000000000100
<hr/>	
$F - (A \& I)$	010000011100
$\sim F$	101111011111
<hr/>	
$(F - (A \& I)) \& \sim F$	000000011100

- Wegen der Einschränkung, dass zwei Elemente nicht unmittelbar aufeinander folgen, ist kein  $F$ -Zustand gleichzeitig  $I$ -Zustand des nächsten Elements.

Erweiterte Zustandsaktualisierung des Shift-And-Algorithmus:

- 1 Herkömmliche Shift-And-Update-Funktion anwenden:

$$A = ((A \ll 1) \mid 1) \& \text{mask}[c]$$

- 2 Zustandsvektor um neue aktive Zustände erweitern:

$$A = A \mid ((F - (A \& I)) \& \sim F)$$

# Beispiel

Sei  $P = \text{bba\#(1, 3) a}$  und  $T = \text{bbacca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

$$A = 0000000$$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{b} \text{ (red) } \text{bacca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 1 mit b:  $A = 0000001$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{b}^{\text{red}}\text{acca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 2:  $A = 0000001$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{b}\text{b}\text{acca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 1 mit b:  $A = 0000011$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{b}\text{b}\text{acca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 2:  $A = 0000011$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{bb}\textcolor{red}{a}\text{cca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 1 mit a:  $A = 0000100$



# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{bb}\textcolor{red}{a}\text{cca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000100
$\text{mask}^c$	0111000	—	0011100

Update 2:             $A = 0011100$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{bba}\text{c}\text{ca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 1 mit  $c$ :  $A = 0111000$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{bba}\text{c}\text{ca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \ \& \ I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 2:             $A = 0111000$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{bbac}\text{ca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 1 mit  $c$ :  $A = 0110000$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{bbac}\text{ca}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 2:  $A = 0110000$

# Beispiel

Sei  $P = \text{bba\#(1, 3) a}$  und  $T = \text{bbacc\textcolor{red}{a}}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \ \& \ I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 1 mit a:      $A = 1100000$

# Beispiel

Sei  $P = \text{bba}\#(1, 3)\text{a}$  und  $T = \text{bbacc}\text{a}$ :

	a###abb		
$\text{mask}^a$	1111100	$F$	0100000
$\text{mask}^b$	0111011	$(A \& I)$	0000000
$\text{mask}^c$	0111000	—	0000000

Update 2:                     $A = 1100000$                     Treffer

- Mit einer zusätzlichen Erweiterung können optionale Zeichen verarbeitet werden, entspricht  $\#(0, 1)$ .  
(bisher waren 0 Vorkommen nicht möglich!)
- Notation: ? hinter dem optionalen Zeichen
- $P = \{\text{color, colour}\} \mapsto P = \text{colou?r}$ .
- Konsekutive  $\epsilon$ -Transitionen (Blöcke) sind erlaubt.
- Auch diese Erweiterung lässt sich mit dem Shift-And-Algorithmus realisieren.

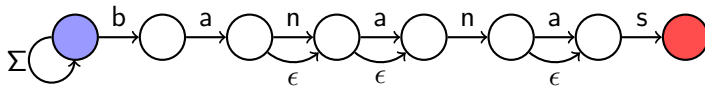


# Optionale Zeichen

Implementierung: drei zusätzliche Bitmasken

$I$ : Blockbeginn;  $F$ : Blockende;  $O$ : Ziele von  $\epsilon$ -Kanten

Beispiel:  $P = \text{ban?}a?na?s$



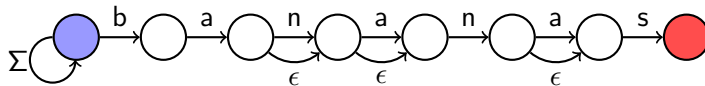
$I$ :	0	1	0	0	1	0	0
$F$ :	0	0	0	1	0	1	0
$O$ :	0	0	1	1	0	1	0

# Optionale Zeichen

Implementierung: drei zusätzliche Bitmasken

$I$ : Blockbeginn;  $F$ : Blockende;  $O$ : Ziele von  $\epsilon$ -Kanten

Beispiel:  $P = \text{ban? a? na? s}$



$I$ :	0	1	0	0	1	0	0
$F$ :	0	0	0	1	0	1	0
$O$ :	0	0	1	1	0	1	0

Sobald ein Zustand innerhalb eines Blocks aktiv ist,  
müssen seine Folgezustände im Block aktiviert werden.

# Implementierung optionaler Zeichen

## ■ Propagierung mit Subtraktion

$$\begin{array}{r}
 1101010000 \\
 - \phantom{00000000} 1 \\
 \hline
 1101001111
 \end{array}$$

$$\begin{array}{r}
 1101011000 \\
 - \phantom{00000000} 100 \\
 \hline
 1101010100
 \end{array}$$

# Implementierung optionaler Zeichen

## ■ Propagierung mit Subtraktion

$$\begin{array}{r} 1101010000 \\ - \phantom{00000000} 1 \\ \hline 1101001111 \end{array}$$

$$\begin{array}{r} 1101011000 \\ - \phantom{00000000} 100 \\ \hline 1101010100 \end{array}$$

## ■ Propagierung vom niederwertigsten aktiven Bit innerhalb eines Blocks bis zum $F$ -Bit

A	.0010100.	A	.0010100.
I	.0000001.	A   F	.1010100.
O	.1111110.	(A   F) - I	.1010011.
F	.1000000.	(A   F - I) = (A   F)	.1111000.
		O & (A   F - I) = (A   F)	.1111000.
>>	.1111100.	A   (O & (A   F - I) = (A   F))	.1111100.

# Implementierung optionaler Zeichen

## ■ Propagierung mit Subtraktion

$$\begin{array}{r} 1101010000 \\ - \phantom{00000000} 1 \\ \hline 1101001111 \end{array}$$

$$\begin{array}{r} 1101011000 \\ - \phantom{00000000} 100 \\ \hline 1101010100 \end{array}$$

## ■ Propagierung vom niederwertigsten aktiven Bit innerhalb eines Blocks bis zum $F$ -Bit

A	.0010100.	A	.0010100.
I	.0000001.	A F	.1010100.
O	.1111110.	(A F) - I	.1010011.
F	.1000000.	(A F - I) = (A F)	.1111000.
		O & (A F - I) = (A F)	.1111000.
>>	.1111100.	A   (O & (A F - I) = (A F))	.1111100.

## ■ Bitweise Gleichheit $X = Y$ implementiert als $\sim(X \oplus Y)$ .

# Implementierung optionaler Zeichen

Implementierung:

- 1 Masken für alle Zeichen aus  $\Sigma$  erstellen;  
dabei die optionalen Zeichen wie reguläre Zeichen behandeln.

- 2 Herkömmliche Shift-And-Update-Funktion anwenden:

$$A = ((A \ll 1) \mid 1) \& \text{mask}[c]$$

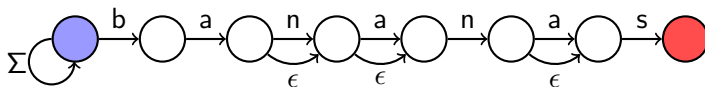
- 3 Zustandsvektor um neue aktive Zustände erweitern:

$$A\_f = A \mid F$$

$$A = A \mid (0 \& (\sim(A\_f - I) \wedge A\_f))$$

(Hier ist  $\wedge$  die xor-Operation.)

# Beispiel

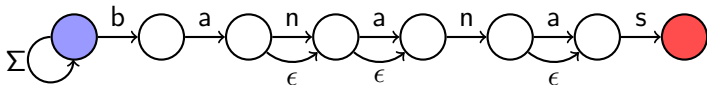


Sei  $P = \text{ban?a?na?s}$  und  $T = \text{banns}$ :

	sanab		
$\text{mask}^a$	0101010	$I$	0010010
$\text{mask}^b$	0000001	$F$	0101000
$\text{mask}^n$	0010100	$O$	0101100
$\text{mask}^s$	1000000	$O \& (\sim(A_f - I) \oplus A_f)$	0000000

$$A = 0000000$$

# Beispiel



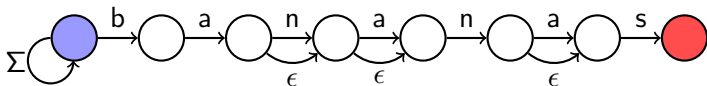
Sei  $P = \text{ban?a?na?s}$  und  $T = \text{bannns}$ :

	sanab		
$mask^a$	0101010	$I$	0010010
$mask^b$	0000001	$F$	0101000
$mask^n$	0010100	$O$	0101100
$mask^s$	1000000	$O \& (\sim(A_f - I) \oplus A_f)$	0000000

Update 1 mit b:  $A = 0000001$



# Beispiel

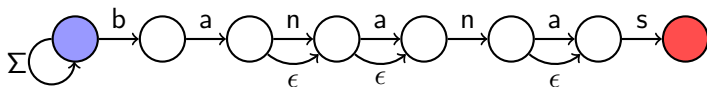


Sei  $P = \text{ban?a?na?s}$  und  $T = \text{banns}$ :

	sanab		
$\text{mask}^a$	0101010	$I$	0010010
$\text{mask}^b$	0000001	$F$	0101000
$\text{mask}^n$	0010100	$O$	0101100
$\text{mask}^s$	1000000	<hr/>	
		$O \& (\sim(A_f - I) \oplus A_f)$	0000000

Update 2:  $A = 0000001$

# Beispiel

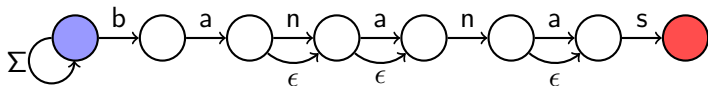


Sei  $P = \text{ban?}a?na?s$  und  $T = \text{b}a\text{nn}s$ :

	sanab		
$mask^a$	0101010	$I$	0010010
$mask^b$	0000001	$F$	0101000
$mask^n$	0010100	$O$	0101100
$mask^s$	1000000	<hr/>	
		$O \& (\sim(A_f - I) \oplus A_f)$	0000000

Update 1 mit a:  $A = 0000010$

# Beispiel

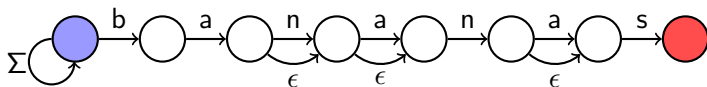


Sei  $P = \text{ban?}a?na?s$  und  $T = \text{b}\textcolor{red}{a}\text{nnns}$ :

	sananab		
$mask^a$	0101010	$I$	0010010
$mask^b$	0000001	$F$	0101000
$mask^n$	0010100	$O$	0101100
$mask^s$	1000000		
		$O \& (\sim(A_f - I) \oplus A_f)$	0001100

Update 2:  $A = 0001110$

# Beispiel

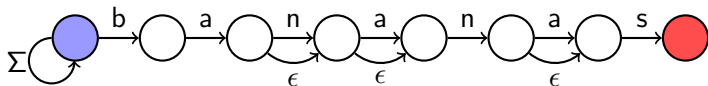


Sei  $P = \text{ban?a?na?s}$  und  $T = \text{ban}\textcolor{red}{n}\text{s}$ :

	sananab		
$\text{mask}^a$	0101010	$I$	0010010
$\text{mask}^b$	0000001	$F$	0101000
$\text{mask}^n$	0010100	$O$	0101100
$\text{mask}^s$	1000000	<hr/>	
		$O \& (\sim(A_f - I) \oplus A_f)$	0000000

Update 1 mit n:  $A = 0010100$

# Beispiel

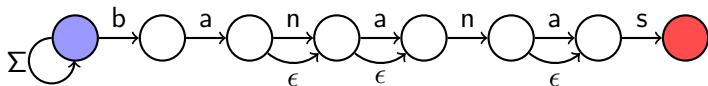


Sei  $P = \text{ban?a?na?s}$  und  $T = \text{ban}\textcolor{red}{n}\text{s}$ :

	sananab		
$\text{mask}^a$	0101010	$I$	0010010
$\text{mask}^b$	0000001	$F$	0101000
$\text{mask}^n$	0010100	$O$	0101100
$\text{mask}^s$	1000000	$O \& (\sim(A_f - I) \oplus A_f)$	0101100

Update 2:  $A = 0111100$

# Beispiel

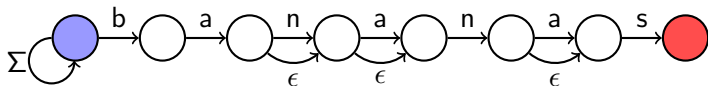


Sei  $P = \text{ban?a?na?s}$  und  $T = \text{ban}\textcolor{red}{n}\text{s}$ :

	sanab		
$\text{mask}^a$	0101010	$I$	0010010
$\text{mask}^b$	0000001	$F$	0101000
$\text{mask}^n$	0010100	$O$	0101100
$\text{mask}^s$	1000000	$O \& (\sim(A_f - I) \oplus A_f)$	0000000

Update 1 mit n:  $A = 0010000$

# Beispiel

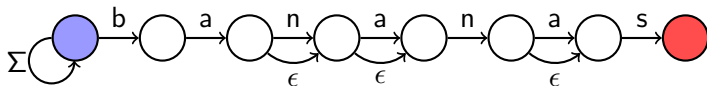


Sei  $P = \text{ban?a?na?s}$  und  $T = \text{ban}\textcolor{red}{n}\text{s}$ :

	sananab		
$\text{mask}^a$	0101010	$I$	0010010
$\text{mask}^b$	0000001	$F$	0101000
$\text{mask}^n$	0010100	$O$	0101100
$\text{mask}^s$	1000000		
		$O \& (\sim(A_f - I) \oplus A_f)$	0100000

Update 2:  $A = 0110000$

# Beispiel



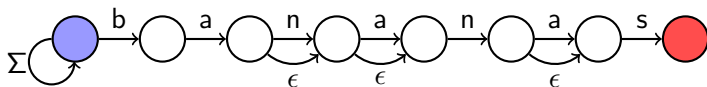
Sei  $P = \text{ban?a?na?s}$  und  $T = \text{bann}\textcolor{red}{s}$ :

	sanab		
$\text{mask}^a$	0101010	$I$	0010010
$\text{mask}^b$	0000001	$F$	0101000
$\text{mask}^n$	0010100	$O$	0101100
$\text{mask}^s$	1000000	<hr/>	
		$O \& (\sim(A_f - I) \oplus A_f)$	0000000

Update 1 mit  $s$ :  $A = 1000000$



# Beispiel



Sei  $P = \text{ban?a?na?s}$  und  $T = \text{bann}\textcolor{red}{s}$ :

	sananab		
$\text{mask}^a$	0101010	$I$	0010010
$\text{mask}^b$	0000001	$F$	0101000
$\text{mask}^n$	0010100	$O$	0101100
$\text{mask}^s$	1000000	$O \ \& \ (\sim(A_f - I) \oplus A_f)$	0000000

Update 2:                       $A = 1000000$                       Treffer

- Durch Modifikationen lässt sich der Shift-And-Algorithmus auf flexiblere Muster erweitern.
- Verallgemeinerte Strings, beliebige Zeichenfolgen beschränkter Länge und optionale Zeichen sind möglich.
- Laufzeiten ändern sich nicht.