

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Beispiel: Match

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$.

	b		b		a		a		a		b		b		b	
0	1	2	3	4	5	6	7	8								

Beispiel: Match

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$.

	b		b		a		a		a		b		b		b	
0	1	2	3	4	5	6	7	8								

Bspw. ist $\{(0,0), (0,8), (3,7)\} \in \mathcal{M}$ eine Menge von Matches mit

$(0,0) \approx ||bbaaabb b \approx \varepsilon$

$(0,8) \approx |bbaaabb b| \approx w$

$(3,7) \approx bba|aabb|b \approx aabb$

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $Parser := \mathcal{M}^{\mathcal{M}}$. Für $f \in Parser$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $Parser := \mathcal{M}^{\mathcal{M}}$. Für $f \in Parser$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

$readChar_w : \Sigma \rightarrow Parser$

$readChar_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $Parser := \mathcal{M}^{\mathcal{M}}$. Für $f \in Parser$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

$readChar_w : \Sigma \rightarrow Parser$

$$readChar_w(\sigma) = \underbrace{M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}}_{\in Parser}$$

Beispiel: readChar

$\text{readChar}_w : \Sigma \rightarrow \text{Parser}$

$\text{readChar}_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$.

	b		b		a		a		a		b		b		b	
0	1	2	3	4	5	6	7	8								

Beispiel: readChar

$\text{readChar}_w : \Sigma \rightarrow \text{Parser}$

$\text{readChar}_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$.

	b		b		a		a		a		b		b		b	
0	1	2	3	4	5	6	7	8								

$$f_a(\{(0, 0), (2, 2), (2, 4)\}) = \{(2, 3), (2, 5)\}$$

Beispiel: readChar

$\text{readChar}_w : \Sigma \rightarrow \text{Parser}$

$\text{readChar}_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$.

	b		b		a		a		a		b		b		b	
0	1	2	3	4	5	6	7	8								

$$f_a(\{(0, 0), (2, 2), (2, 4)\}) = \{(2, 3), (2, 5)\}$$

$$\begin{aligned} &\approx f_a(\{||bbaaabb b, bb||aaabbb, bb|aa|abbb\}) \\ &= \{bb|a|aabbb, bb|aaa|bbb\} \end{aligned}$$

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $Parser := \mathcal{M}^{\mathcal{M}}$. Für $f \in Parser$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

$readChar_w : \Sigma \rightarrow Parser$

$readChar_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

$concatenate_w : Parser \times Parser \rightarrow Parser$

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $\text{Parser} := \mathcal{M}^{\mathcal{M}}$. Für $f \in \text{Parser}$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

$\text{readChar}_w : \Sigma \rightarrow \text{Parser}$

$\text{readChar}_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

$\text{concatenate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{concatenate}_w(f_1, f_2) = f_2 \circ f_1$

Beispiel: concatenate

$\text{concatenate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{concatenate}_w(f_1, f_2) = f_2 \circ f_1$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$
 $f_b := \text{readChar}_w(b)$, $f_{ab} := \text{concatenate}_w(f_a, f_b)$.

	b		b		a		a		a		b		b		b	
0	1	2	3	4	5	6	7	8								

Beispiel: concatenate

$\text{concatenate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{concatenate}_w(f_1, f_2) = f_2 \circ f_1$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$
 $f_b := \text{readChar}_w(b)$, $f_{ab} := \text{concatenate}_w(f_a, f_b)$.

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

$$\begin{aligned} f_{ab}(\{(0,0), (2,2), (2,4)\}) &= f_b(f_a(\{(0,0), (2,2), (2,4)\})) \\ &= f_b(\{(2,3), (2,5)\}) = \{(2,6)\} \end{aligned}$$

Beispiel: concatenate

$\text{concatenate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{concatenate}_w(f_1, f_2) = f_2 \circ f_1$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$
 $f_b := \text{readChar}_w(b)$, $f_{ab} := \text{concatenate}_w(f_a, f_b)$.

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

$$\begin{aligned} f_{ab}(\{(0,0), (2,2), (2,4)\}) &= f_b(f_a(\{(0,0), (2,2), (2,4)\})) \\ &= f_b(\{(2,3), (2,5)\}) = \{(2,6)\} \end{aligned}$$

$$\begin{aligned} &\approx f_{ab}(\{|bbaaabb b, bb|aaabbb, bb|aa|abbb\}) \\ &= f_b(\{bb|a|aabbb, bb|aaa|bbb\}) = \{bb|aaab|bb\} \end{aligned}$$

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $\text{Parser} := \mathcal{M}^{\mathcal{M}}$. Für $f \in \text{Parser}$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

$\text{readChar}_w : \Sigma \rightarrow \text{Parser}$

$\text{readChar}_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

$\text{concatenate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{concatenate}_w(f_1, f_2) = f_2 \circ f_1$

$\text{alternate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $\text{Parser} := \mathcal{M}^{\mathcal{M}}$. Für $f \in \text{Parser}$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

$\text{readChar}_w : \Sigma \rightarrow \text{Parser}$

$\text{readChar}_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

$\text{concatenate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{concatenate}_w(f_1, f_2) = f_2 \circ f_1$

$\text{alternate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{alternate}_w(f_1, f_2) = M \mapsto f_1(M) \cup f_2(M)$

Beispiel: alternate

$\text{alternate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{alternate}_w(f_1, f_2) = M \mapsto f_1(M) \cup f_2(M)$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$
 $f_b := \text{readChar}_w(b)$, $f_{a|b} := \text{alternate}_w(f_a, f_b)$.

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

Beispiel: alternate

$\text{alternate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{alternate}_w(f_1, f_2) = M \mapsto f_1(M) \cup f_2(M)$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$
 $f_b := \text{readChar}_w(b)$, $f_{a|b} := \text{alternate}_w(f_a, f_b)$.

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

$$\begin{aligned} f_{ab}(\{(0,0), (2,2), (2,4)\}) &= f_a(\{\cdots\}) \cup f_b(\{\cdots\}) \\ &= \{(2,3), (2,5)\} \cup \{(0,1)\} = \{(0,1), (2,3), (2,5)\} \end{aligned}$$

Beispiel: alternate

$\text{alternate}_w : \text{Parser} \times \text{Parser} \rightarrow \text{Parser}$

$\text{alternate}_w(f_1, f_2) = M \mapsto f_1(M) \cup f_2(M)$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a := \text{readChar}_w(a)$
 $f_b := \text{readChar}_w(b)$, $f_{a|b} := \text{alternate}_w(f_a, f_b)$.

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

$$\begin{aligned} f_{ab}(\{(0,0), (2,2), (2,4)\}) &= f_a(\{\dots\}) \cup f_b(\{\dots\}) \\ &= \{(2,3), (2,5)\} \cup \{(0,1)\} = \{(0,1), (2,3), (2,5)\} \end{aligned}$$

$$\begin{aligned} f_{a|b}(\{||bbaaabb b, bb||aaabb b, bb|aa|abb b\}) &= \\ = \{|b|baaabb b, bb|a|aabb b, bb|aaa|bbb\} \end{aligned}$$

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $Parser := \mathcal{M}^{\mathcal{M}}$. Für $f \in Parser$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

$readChar_w : \Sigma \rightarrow Parser$

$readChar_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

$concatenate_w : Parser \times Parser \rightarrow Parser$

$concatenate_w(f_1, f_2) = f_2 \circ f_1$

$alternate_w : Parser \times Parser \rightarrow Parser$

$alternate_w(f_1, f_2) = M \mapsto f_1(M) \cup f_2(M)$

$iterate_w : Parser \rightarrow Parser$

$iterate_w(f) = M \mapsto \bigcup_{i \in \mathbb{N}_0} f^i(M) = M \mapsto M \cup f(M) \cup f(f(M)) \dots$

Beispiel: iterate

$\text{iterate}_w : \text{Parser} \rightarrow \text{Parser}$

$$\text{iterate}_w(f) = M \mapsto \bigcup_{i \in \mathbb{N}_0} f^i(M)$$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_a^* := \text{iterate}_w(f_a)$

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

Beispiel: iterate

$\text{iterate}_w : \text{Parser} \rightarrow \text{Parser}$

$$\text{iterate}_w(f) = M \mapsto \bigcup_{i \in \mathbb{N}_0} f^i(M)$$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_{a^*} := \text{iterate}_w(f_a)$

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

$$f_{a^*}(\{(0,0), (2,2), (2,4)\}) = \{\dots\} \cup f_a(\{\dots\}) \cup f_a(f_a(\{\dots\})) \cup \dots$$

Beispiel: iterate

$\text{iterate}_w : \text{Parser} \rightarrow \text{Parser}$

$$\text{iterate}_w(f) = M \mapsto \bigcup_{i \in \mathbb{N}_0} f^i(M)$$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_{a^*} := \text{iterate}_w(f_a)$

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

$$\begin{aligned} f_{a^*}(\{(0,0), (2,2), (2,4)\}) &= \{\dots\} \cup f_a(\{\dots\}) \cup f_a(f_a(\{\dots\})) \cup \dots \\ &= \{(0,0), (2,2), (2,4)\} \cup \{(2,3), (2,5)\} \cup \{(2,4)\} \cup \{(2,5)\} \cup \emptyset \dots \end{aligned}$$

Beispiel: iterate

$\text{iterate}_w : \text{Parser} \rightarrow \text{Parser}$

$$\text{iterate}_w(f) = M \mapsto \bigcup_{i \in \mathbb{N}_0} f^i(M)$$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb b \in \Sigma^*$, $f_{a^*} := \text{iterate}_w(f_a)$

	b		b		a		a		a		b		b		b	
0		1		2		3		4		5		6		7		8

$$\begin{aligned} f_{a^*}(\{(0,0), (2,2), (2,4)\}) &= \{\dots\} \cup f_a(\{\dots\}) \cup f_a(f_a(\{\dots\})) \cup \dots \\ &= \{(0,0), (2,2), (2,4)\} \cup \{(2,3), (2,5)\} \cup \{(2,4)\} \cup \{(2,5)\} \cup \emptyset \dots \\ &= \{(0,0), (2,2), (2,3), (2,4), (2,5)\} \end{aligned}$$

Beispiel: iterate

$\text{iterate}_w : \text{Parser} \rightarrow \text{Parser}$

$$\text{iterate}_w(f) = M \mapsto \bigcup_{i \in \mathbb{N}_0} f^i(M)$$

Sei $\Sigma = \{a, b\}$, $w = bbaaabb \in \Sigma^*$, $f_a^* := \text{iterate}_w(f_a)$

$$\begin{array}{cccccccc} | & \mathbf{b} & | & \mathbf{b} & | & \mathbf{a} & | & \mathbf{a} & | & \mathbf{a} & | & \mathbf{b} & | & \mathbf{b} & | & \mathbf{b} & | \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}$$

$$\begin{aligned} f_a^*(\{(0,0), (2,2), (2,4)\}) &= \{\dots\} \cup f_a(\{\dots\}) \cup f_a(f_a(\{\dots\})) \cup \dots \\ &= \{(0,0), (2,2), (2,4)\} \cup \{(2,3), (2,5)\} \cup \{(2,4)\} \cup \{(2,5)\} \cup \emptyset \dots \\ &= \{(0,0), (2,2), (2,3), (2,4), (2,5)\} \end{aligned}$$

$$\begin{aligned} f_a^*(\{||bbaaabb, bb||aaabbb, bb|aa|abbb\}) \\ = \{||bbaaabb, bb||aaabbb, bb|a|aabbb, bb|aa|abbb, bb|aaa|bbb\} \end{aligned}$$

Definitionen

Gegeben: Eingabealphabet Σ , Eingabestring $w \in \Sigma^*$.

Bezeichne $m \in \mathbb{N}_0 \times \mathbb{N}_0$ als Match. (startIndex, endIndex)

Definiere $\mathcal{M} := \mathcal{P}(\mathbb{N}_0 \times \mathbb{N}_0)$ als Menge der Mengen von Matches.

Sei nun $Parser := \mathcal{M}^{\mathcal{M}}$. Für $f \in Parser$ ist also $f : \mathcal{M} \rightarrow \mathcal{M}$.

$readChar_w : \Sigma \rightarrow Parser$

$readChar_w(\sigma) = M \mapsto \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\}$

$concatenate_w : Parser \times Parser \rightarrow Parser$

$concatenate_w(f_1, f_2) = f_2 \circ f_1$

$alternate_w : Parser \times Parser \rightarrow Parser$

$alternate_w(f_1, f_2) = M \mapsto f_1(M) \cup f_2(M)$

$iterate_w : Parser \rightarrow Parser$

$iterate_w(f) = M \mapsto \bigcup_{i \in \mathbb{N}_0} f^i(M)$

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

```
f := concatenatew(  
  alternatew(readCharw(a), readCharw(b)),  
  iteratew(readCharw(c)))
```

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

$f := \text{concatenate}_w(\text{alternate}_w(\text{readChar}_w(a), \text{readChar}_w(b)), \text{iterate}_w(\text{readChar}_w(c)))$

Definiere Startmenge $S_w \in \mathcal{M}$ durch $S_w := \{(i, i) \mid i \in [0, |w| - 1]\}$.

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

$f := \text{concatenate}_w(\text{alternate}_w(\text{readChar}_w(a), \text{readChar}_w(b)), \text{iterate}_w(\text{readChar}_w(c)))$

Definiere Startmenge $S_w \in \mathcal{M}$ durch $S_w := \{(i, i) \mid i \in [0, |w| - 1]\}$.

$$S_w = \{(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\}$$
$$\approx \{ \textcolor{red}{|}xabccx, x\textcolor{red}{|}abccx, xa\textcolor{red}{|}bccx, xab\textcolor{red}{|}ccx, xabc\textcolor{red}{|}cx, xabcc\textcolor{red}{|}x \}$$

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

```
f := concatenatew(  
  alternatew(readCharw(a), readCharw(b)),  
  iteratew(readCharw(c)))
```

Definiere Startmenge $S_w \in \mathcal{M}$ durch $S_w := \{(i, i) \mid i \in [0, |w| - 1]\}$.

	x		a		b		c		c		x	
0		1		2		3		4		5		6

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

$f := \text{concatenate}_w(\text{alternate}_w(\text{readChar}_w(a), \text{readChar}_w(b)), \text{iterate}_w(\text{readChar}_w(c)))$

Definiere Startmenge $S_w \in \mathcal{M}$ durch $S_w := \{(i, i) \mid i \in [0, |w| - 1]\}$.

	x		a		b		c		c		x	
0		1		2		3		4		5		6

$$f(S_w) = f_{c*}(f_{a|b}(S_w)) = f_{c*}(f_a(S_w) \cup f_b(S_w))$$

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

$f := \text{concatenate}_w(\text{alternate}_w(\text{readChar}_w(a), \text{readChar}_w(b)), \text{iterate}_w(\text{readChar}_w(c)))$

Definiere Startmenge $S_w \in \mathcal{M}$ durch $S_w := \{(i, i) \mid i \in [0, |w| - 1]\}$.

$\begin{array}{ccccccccc} & | & x & | & a & | & b & | & c & | & c & | & x & | \\ & 0 & & 1 & & 2 & & 3 & & 4 & & 5 & & 6 \end{array}$

$$\begin{aligned} f(S_w) &= f_{c*}(f_{a|b}(S_w)) = f_{c*}(f_a(S_w) \cup f_b(S_w)) \\ &= f_{c*}(\{(1, 2)\} \cup f_b(S_w)) = f_{c*}(\{(1, 2), (2, 3)\}) \end{aligned}$$

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

$f := \text{concatenate}_w(\text{alternate}_w(\text{readChar}_w(a), \text{readChar}_w(b)), \text{iterate}_w(\text{readChar}_w(c)))$

Definiere Startmenge $S_w \in \mathcal{M}$ durch $S_w := \{(i, i) \mid i \in [0, |w| - 1]\}$.

$\begin{array}{ccccccccc} & | & \mathbf{x} & | & \mathbf{a} & | & \mathbf{b} & | & \mathbf{c} & | & \mathbf{c} & | & \mathbf{x} & | \\ & 0 & & 1 & & 2 & & 3 & & 4 & & 5 & & 6 \end{array}$

$$f(S_w) = f_{c*}(f_{a|b}(S_w)) = f_{c*}(f_a(S_w) \cup f_b(S_w))$$

$$= f_{c*}(\{(1, 2)\} \cup f_b(S_w)) = f_{c*}(\{(1, 2), (2, 3)\})$$

$$= \bigcup_{i \in \mathbb{N}_0} f_c^i(\{(1, 2), (2, 3)\}) = \{(1, 2), (2, 3)\} \cup \{(2, 4)\} \cup \{(2, 5)\} \cup \emptyset \dots$$

Komplettbeispiel

Sei $\Sigma = \{a, b, c, x\}$, $w = xabccx$. Regex $(a \mid b)c^*$.

$f := \text{concatenate}_w(\text{alternate}_w(\text{readChar}_w(a), \text{readChar}_w(b)), \text{iterate}_w(\text{readChar}_w(c)))$

Definiere Startmenge $S_w \in \mathcal{M}$ durch $S_w := \{(i, i) \mid i \in [0, |w| - 1]\}$.

$\begin{array}{ccccccccc} & | & x & | & a & | & b & | & c & | & c & | & x & | \\ & 0 & & 1 & & 2 & & 3 & & 4 & & 5 & & 6 \end{array}$

$$f(S_w) = f_{c*}(f_{a|b}(S_w)) = f_{c*}(f_a(S_w) \cup f_b(S_w))$$

$$= f_{c*}(\{(1, 2)\} \cup f_b(S_w)) = f_{c*}(\{(1, 2), (2, 3)\})$$

$$= \bigcup_{i \in \mathbb{N}_0} f_c^i(\{(1, 2), (2, 3)\}) = \{(1, 2), (2, 3)\} \cup \{(2, 4)\} \cup \{(2, 5)\} \cup \emptyset \dots$$

$$= \{(1, 2), (2, 3), (2, 4), (2, 5)\}$$

$$\approx \{x|a|bccx, xa|b|ccx, xa|bc|cx, xa|bcc|x\}$$

Rekursiv gedacht

Σ Alphabet, \mathcal{R}_Σ als Menge der Regexe über Σ . Weiter $w \in \Sigma^*$.

$\text{ext}_w : \mathcal{R}_\Sigma \times \mathcal{M} \rightarrow \mathcal{M}$

$$\text{ext}_w(r, M) := \left\{ \begin{array}{l} \end{array} \right.$$

Rekursiv gedacht

Σ Alphabet, \mathcal{R}_Σ als Menge der Regexe über Σ . Weiter $w \in \Sigma^*$.

$\text{ext}_w : \mathcal{R}_\Sigma \times \mathcal{M} \rightarrow \mathcal{M}$

$$\text{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\} & , r = \sigma \in \Sigma \end{cases}$$

Rekursiv gedacht

Σ Alphabet, \mathcal{R}_Σ als Menge der Regexe über Σ . Weiter $w \in \Sigma^*$.

$\text{ext}_w : \mathcal{R}_\Sigma \times \mathcal{M} \rightarrow \mathcal{M}$

$$\text{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\} & , r = \sigma \in \Sigma \\ \text{ext}_w(r_2, \text{ext}_w(r_1, M)) & , r = (r_1 r_2), r_1, r_2 \in \mathcal{R}_\Sigma \end{cases}$$

Rekursiv gedacht

Σ Alphabet, \mathcal{R}_Σ als Menge der Regexe über Σ . Weiter $w \in \Sigma^*$.

$\text{ext}_w : \mathcal{R}_\Sigma \times \mathcal{M} \rightarrow \mathcal{M}$

$$\text{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\} & , r = \sigma \in \Sigma \\ \text{ext}_w(r_2, \text{ext}_w(r_1, M)) & , r = (r_1 r_2), r_1, r_2 \in \mathcal{R}_\Sigma \\ \text{ext}_w(r_1, M) \cup \text{ext}_w(r_2, M) & , r = (r_1 \mid r_2), r_1, r_2 \in \mathcal{R}_\Sigma \end{cases}$$

Rekursiv gedacht

Σ Alphabet, \mathcal{R}_Σ als Menge der Regexe über Σ . Weiter $w \in \Sigma^*$.

$\text{ext}_w : \mathcal{R}_\Sigma \times \mathcal{M} \rightarrow \mathcal{M}$

$$\text{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\} & , r = \sigma \in \Sigma \\ \text{ext}_w(r_2, \text{ext}_w(r_1, M)) & , r = (r_1 r_2), r_1, r_2 \in \mathcal{R}_\Sigma \\ \text{ext}_w(r_1, M) \cup \text{ext}_w(r_2, M) & , r = (r_1 \mid r_2), r_1, r_2 \in \mathcal{R}_\Sigma \\ \text{iter}_w(t, M) & , r = t^*, t \in \mathcal{R}_\Sigma \end{cases}$$

$$\text{iter}_w(t, M) := \begin{cases} \emptyset & \text{falls } M = \emptyset \\ M \cup \text{iter}_w(t, \text{ext}_w(t, M)) & \text{sonst} \end{cases}$$

Rekursiv gedacht

Σ Alphabet, \mathcal{R}_Σ als Menge der Regexe über Σ . Weiter $w \in \Sigma^*$.

$\text{ext}_w : \mathcal{R}_\Sigma \times \mathcal{M} \rightarrow \mathcal{M}$

$$\text{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i, j) \in M \wedge w_j = \sigma\} & , r = \sigma \in \Sigma \\ \text{ext}_w(r_2, \text{ext}_w(r_1, M)) & , r = (r_1 r_2), r_1, r_2 \in \mathcal{R}_\Sigma \\ \text{ext}_w(r_1, M) \cup \text{ext}_w(r_2, M) & , r = (r_1 \mid r_2), r_1, r_2 \in \mathcal{R}_\Sigma \\ \text{iter}_w(t, M) & , r = t^*, t \in \mathcal{R}_\Sigma \end{cases}$$

$$\text{iter}_w(t, M) := \begin{cases} \emptyset & \text{falls } M = \emptyset \\ M \cup \text{iter}_w(t, \text{ext}_w(t, M)) & \text{sonst} \end{cases}$$

Aufruf wäre dann:

$$\text{ext}_w((a \mid b)c^*, S_w)$$

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

Match Element von $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$. Entsprechend \mathcal{M} anpassen.

$$S_w := \{(i, i, 0) \mid i \in [0, |w| - 1]\}$$

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

Match Element von $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$. Entsprechend \mathcal{M} anpassen.

$$S_w := \{(i, i, 0) \mid i \in [0, |w| - 1]\}$$

$$\text{readChar}_w(\sigma) = M \mapsto \{(i, j + 1, p) \mid (i, j, p) \in M \wedge w_j = \sigma\}$$

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

Match Element von $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$. Entsprechend \mathcal{M} anpassen.

$$S_w := \{(i, i, 0) \mid i \in [0, |w| - 1]\}$$

$$\text{readChar}_w(\sigma) = M \mapsto \{(i, j + 1, p) \mid (i, j, p) \in M \wedge w_j = \sigma\}$$

$$\text{decrPrio}: \mathcal{M} \rightarrow \mathcal{M}$$

$$\text{decrPrio}(M) = \{(i, j, p + 1) \mid (i, j, p) \in M\}$$

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

Match Element von $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$. Entsprechend \mathcal{M} anpassen.

$$S_w := \{(i, i, 0) \mid i \in [0, |w| - 1]\}$$

$$\text{readChar}_w(\sigma) = M \mapsto \{(i, j + 1, p) \mid (i, j, p) \in M \wedge w_j = \sigma\}$$

$$\text{decrPrio}: \mathcal{M} \rightarrow \mathcal{M}$$

$$\text{decrPrio}(M) = \{(i, j, p + 1) \mid (i, j, p) \in M\}$$

$$\text{alternate}_w(f_1, f_2) = M \mapsto f_1(M) \cup \text{decrPrio}(f_2(M))$$

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

Match Element von $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$. Entsprechend \mathcal{M} anpassen.

$$S_w := \{(i, i, 0) \mid i \in [0, |w| - 1]\}$$

$$\text{readChar}_w(\sigma) = M \mapsto \{(i, j + 1, p) \mid (i, j, p) \in M \wedge w_j = \sigma\}$$

$$\text{decrPrio}: \mathcal{M} \rightarrow \mathcal{M}$$

$$\text{decrPrio}(M) = \{(i, j, p + 1) \mid (i, j, p) \in M\}$$

$$\text{alternate}_w(f_1, f_2) = M \mapsto f_1(M) \cup \text{decrPrio}(f_2(M))$$

nicht leer $>$ startIndex $>$ Priorität $>$ Länge

$$(0, 3, 0) > (1, 5, 0) > (1, 8, 1) > (1, 5, 1) > (0, 0, 0)$$

Erweiterung: Priorität bei Alternation (cont.)

Unerwartetes Verhalten bei Iteration:

$$\Sigma = \{a, b\}, \quad w = ab. \quad f \approx (a \mid b)^*.$$

Erweiterung: Priorität bei Alternation (cont.)

Unerwartetes Verhalten bei Iteration:

$\Sigma = \{a, b\}$, $w = ab$. $f \approx (a \mid b)^*$.

$$\begin{aligned} f(S_w) &= f(\{(0, 0, 0), (1, 1, 0)\}) \\ &= \{(0, 0, 0), (1, 1, 0)\} \cup f_{a|b}(\{\dots\}) \cup f_{a|b}(f_{a|b}(\{\dots\})) \dots \\ &= \{(0, 0, 0), (1, 1, 0)\} \cup \underbrace{\{(0, 1, 0)\}}_{f_a(\{\dots\})} \cup \underbrace{\{(1, 2, 1)\}}_{f_b(\{\dots\})} \cup \underbrace{\{(0, 2, 1)\}}_{f_b(f_{a|b}(\{\dots\}))} \end{aligned}$$

Erweiterung: Priorität bei Alternation (cont.)

Unerwartetes Verhalten bei Iteration:

$$\Sigma = \{a, b\}, \quad w = ab. \quad f \approx (a \mid b)^*.$$

$$\begin{aligned} f(S_w) &= f(\{(0, 0, 0), (1, 1, 0)\}) \\ &= \{(0, 0, 0), (1, 1, 0)\} \cup f_{a|b}(\{\dots\}) \cup f_{a|b}(f_{a|b}(\{\dots\})) \dots \\ &= \{(0, 0, 0), (1, 1, 0)\} \cup \underbrace{\{(0, 1, 0)\}}_{f_a(\{\dots\})} \cup \underbrace{\{(1, 2, 1)\}}_{f_b(\{\dots\})} \cup \underbrace{\{(0, 2, 1)\}}_{f_b(f_{a|b}(\{\dots\}))} \\ &\implies \underbrace{(0, 1, 0) > (0, 2, 1)}_{|a|b > |ab|} > (1, 1, 0) > (1, 2, 1) > (0, 0, 0) \end{aligned}$$

Implementierung in Java

```
java.util.function.Function<A,B>
```

```
Function<A,B>.apply :  $A \rightarrow B$ 
```

```
Function<A,B>.compose : Function<B,C>  $\rightarrow$  Function<A,C>
```

Implementierung in Java

```
java.util.function.Function<A,B>  
Function<A,B>.apply : A → B  
Function<A,B>.compose : Function<B,C> → Function<A,C>
```

Syntax mit Java-Lambdas, ähnlich zum Syntax hier:

```
Function<String, String> f = s -> s+"!";  
System.out.println(f.apply("ok"));
```

Ausgabe: ok!

Implementierung in Java

```
java.util.function.Function<A,B>  
Function<A,B>.apply : A → B  
Function<A,B>.compose : Function<B,C> → Function<A,C>
```

Syntax mit Java-Lambdas, ähnlich zum Syntax hier:

```
Function<String, String> f = s -> s+"!";  
System.out.println(f.apply("ok"));
```

Ausgabe: ok!

```
java.util.stream.Stream<A>  
Stream<A>.map : (A → B) → Stream<B>
```

Implementierung in Java

```
java.util.function.Function<A,B>  
Function<A,B>.apply : A → B  
Function<A,B>.compose : Function<B,C> → Function<A,C>
```

Syntax mit Java-Lambdas, ähnlich zum Syntax hier:

```
Function<String, String> f = s -> s+"!";  
System.out.println(f.apply("ok"));
```

Ausgabe: ok!

```
java.util.stream.Stream<A>  
Stream<A>.map : (A → B) → Stream<B>  
Stream.iterate : (A, A → A) → Stream<A>
```

Implementierung in Java

```
java.util.function.Function<A,B>  
Function<A,B>.apply :  $A \rightarrow B$   
Function<A,B>.compose :  $\text{Function}\langle B,C \rangle \rightarrow \text{Function}\langle A,C \rangle$ 
```

Syntax mit Java-Lambdas, ähnlich zum Syntax hier:

```
Function<String, String> f = s -> s+"!";  
System.out.println(f.apply("ok"));
```

Ausgabe: ok!

```
java.util.stream.Stream<A>  
Stream<A>.map :  $(A \rightarrow B) \rightarrow \text{Stream}\langle B \rangle$   
Stream.iterate :  $(A, A \rightarrow A) \rightarrow \text{Stream}\langle A \rangle$ 
```

```
Stream<Double> s = Stream.iterate(0, i -> i+1); //  $\mathbb{N}_0$   
s.map(i -> Math.pow(2,3*i)).limit(8).forEach(System.out::println);
```

Ausgabe: 1, 8, 64, 512, 4096, 32768, 262144, 2097152.

Dies entspricht den ersten 8 Werten der Folge $(2^{3i})_{i \in \mathbb{N}_0}$.