Definitionen

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

Definitionen

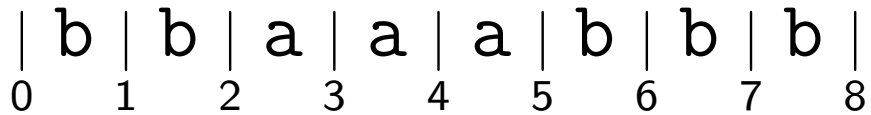
Gegeben: Eingabealphabet Σ, Eingabestring $w \in \Sigma^*$.

```
type Match = (Int, Int)
```
(startIndex, endIndex)

Beispiel: Match

`type Match = (Int, Int)` (startIndex, endIndex)

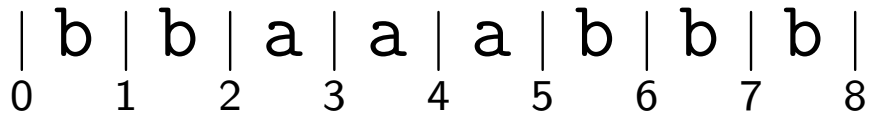
Sei $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, $w = \mathtt{bbaaabbb} \in \Sigma^*$.



Beispiel: Match

`type Match = (Int, Int)` (startIndex, endIndex)

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



$(0, 0) \approx \varepsilon$
$(0, 8) \approx w$
$(3, 7) \approx \mathtt{aabb}$

Definitionen

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

```
type Match = (Int, Int)                    (startIndex, endIndex)
type Parser = {Match} → {Match}
```

Definitionen

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

```
type Match = (Int, Int)                    (startIndex, endIndex)
type Parser = {Match} → {Match}
```

$$\texttt{readChar}_w \texttt{ :: Char} \rightarrow \texttt{Parser}$$
$$\texttt{readChar}_w\texttt{(c)} = M \mapsto \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\}$$

Definitionen

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

```
type Match = (Int, Int)                    (startIndex, endIndex)
type Parser = {Match} → {Match}
```

$$\texttt{readChar}_w \texttt{ :: Char} \rightarrow \texttt{Parser}$$

$$\texttt{readChar}_w\texttt{(c) =} \underbrace{M \mapsto \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\}}_{::\{Match\}\rightarrow\{Match\}}$$

Beispiel: `readChar`

$\texttt{readChar}_w$ :: Char $\to$ Parser

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

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

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

Beispiel: `readChar`

$\texttt{readChar}_w$ :: Char $\to$ Parser

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

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

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`

$$\texttt{readChar}_w :: \texttt{Char} \to \texttt{Parser}$$
$$\texttt{readChar}_w(\texttt{c}) = M \mapsto \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\}$$

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

$$\underset{0}{|}\,\texttt{b}\,\underset{1}{|}\,\texttt{b}\,\underset{2}{|}\,\texttt{a}\,\underset{3}{|}\,\texttt{a}\,\underset{4}{|}\,\texttt{a}\,\underset{5}{|}\,\texttt{b}\,\underset{6}{|}\,\texttt{b}\,\underset{7}{|}\,\texttt{b}\,\underset{8}{|}$$

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

$$f_a(\{\texttt{||bbaaabbb}, \texttt{bb||aaabbb}, \texttt{bb|aa|abbb}\})$$
$$= \{\texttt{bb|a|aabbb}, \texttt{bb|aaa|bbb}\}$$

Definitionen

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

```
type Match = (Int, Int)                    (startIndex, endIndex)
type Parser = {Match} → {Match}
```

$\texttt{readChar}_w$ `:: Char → Parser`

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

```
concatenate :: (Parser, Parser) → Parser
```

Definitionen

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

`type Match = (Int, Int)` (startIndex, endIndex)
`type Parser = {Match}` $\rightarrow$ `{Match}`

$\texttt{readChar}_w$ `:: Char` $\rightarrow$ `Parser`
$\texttt{readChar}_w\texttt{(c)} = \; M \mapsto \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\}$

```
concatenate :: (Parser, Parser) → Parser
concatenate(p1,p2) = p2 ∘ p1
```

Beispiel: `concatenate`

```
concatenate :: (Parser, Parser) → Parser
concatenate(p1,p2) = p2 ∘ p1
```

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

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

Beispiel: `concatenate`

```
concatenate :: (Parser, Parser) → Parser
concatenate(p1,p2) = p2 ∘ p1
```

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

$$\underset{0}{|}\,\texttt{b}\,\underset{1}{|}\,\texttt{b}\,\underset{2}{|}\,\texttt{a}\,\underset{3}{|}\,\texttt{a}\,\underset{4}{|}\,\texttt{a}\,\underset{5}{|}\,\texttt{b}\,\underset{6}{|}\,\texttt{b}\,\underset{7}{|}\,\texttt{b}\,\underset{8}{|}$$

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

Beispiel: `concatenate`

```
concatenate :: (Parser, Parser) → Parser
concatenate(p1,p2) = p2 ∘ p1
```

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

$$\underset{0}{|}\,\texttt{b}\,\underset{1}{|}\,\texttt{b}\,\underset{2}{|}\,\texttt{a}\,\underset{3}{|}\,\texttt{a}\,\underset{4}{|}\,\texttt{a}\,\underset{5}{|}\,\texttt{b}\,\underset{6}{|}\,\texttt{b}\,\underset{7}{|}\,\texttt{b}\,\underset{8}{|}$$

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

$$f_{ab}(\{\texttt{||bbaaabbb}, \texttt{bb||aaabbb}, \texttt{bb|aa|abbb}\})$$
$$= \{\texttt{bb|aaab|bb}\}$$

Definitionen

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

```
type Match = (Int, Int)                    (startIndex, endIndex)
type Parser = {Match} → {Match}
```

$\texttt{readChar}_w$ `:: Char → Parser`

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

```
concatenate :: (Parser, Parser) → Parser
concatenate(p1,p2) = p2 ∘ p1
```

```
alternate :: (Parser, Parser) → Parser
```

Definitionen

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

`type Match = (Int, Int)` (startIndex, endIndex)

`type Parser =` $\{$`Match`$\}$ $\rightarrow$ $\{$`Match`$\}$

$\texttt{readChar}_w$ `::` `Char` $\rightarrow$ `Parser`

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

`concatenate :: (Parser, Parser)` $\rightarrow$ `Parser`

`concatenate(p1,p2) = p2` $\circ$ `p1`

`alternate :: (Parser, Parser)` $\rightarrow$ `Parser`

`alternate(p1,p2)` $= M \mapsto$ `p1`$(M) \cup$ `p2`(M)

Beispiel: `alternate`

`alternate :: (Parser, Parser)` $\rightarrow$ `Parser`

`alternate(p1,p2)` $= M \mapsto$ `p1`$(M) \cup$ `p2`(M)

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

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

Beispiel: `alternate`

```
alternate :: (Parser, Parser) → Parser
```
$\texttt{alternate(p1,p2)} = M \mapsto \texttt{p1}(M) \cup \texttt{p2}(M)$

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

$$\underset{0}{|}\,\texttt{b}\,\underset{1}{|}\,\texttt{b}\,\underset{2}{|}\,\texttt{a}\,\underset{3}{|}\,\texttt{a}\,\underset{4}{|}\,\texttt{a}\,\underset{5}{|}\,\texttt{b}\,\underset{6}{|}\,\texttt{b}\,\underset{7}{|}\,\texttt{b}\,\underset{8}{|}$$

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

Beispiel: `alternate`

`alternate :: (Parser, Parser)` $\rightarrow$ `Parser`
`alternate(p1,p2)` $= M \mapsto$ `p1`$(M) \cup$ `p2`(M)

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

$$\underset{0}{|}\,\texttt{b}\,\underset{1}{|}\,\texttt{b}\,\underset{2}{|}\,\texttt{a}\,\underset{3}{|}\,\texttt{a}\,\underset{4}{|}\,\texttt{a}\,\underset{5}{|}\,\texttt{b}\,\underset{6}{|}\,\texttt{b}\,\underset{7}{|}\,\texttt{b}\,\underset{8}{|}$$

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

$$f_{a|b}(\{\texttt{||bbaaabbb},\ \texttt{bb||aaabbb},\ \texttt{bb|aa|abbb}\}) =$$
$$= \{\texttt{|b|baaabbb},\ \texttt{bb|a|aabbb},\ \texttt{bb|aaa|bbb}\}$$

Definitionen

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

`type Match = (Int, Int)` (startIndex, endIndex)
`type Parser = {Match}` $\rightarrow$ `{Match}`

$\texttt{readChar}_w$ `:: Char` $\rightarrow$ `Parser`
$\texttt{readChar}_w\texttt{(c)} = M \mapsto \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\}$

`concatenate :: (Parser, Parser)` $\rightarrow$ `Parser`
`concatenate(p1,p2) = p2` $\circ$ `p1`

`alternate :: (Parser, Parser)` $\rightarrow$ `Parser`
`alternate(p1,p2)` $= M \mapsto \texttt{p1}(M) \cup \texttt{p2}(M)$

`iterate :: Parser` $\rightarrow$ `Parser`
`iterate(p)` $= M \mapsto \bigcup_{i \in \mathbb{N}_0} \texttt{p}^i(M) = M \mapsto M \cup \texttt{p}(M) \cup \texttt{p}(\texttt{p}(M)) \cdots$

Beispiel: `iterate`

$$\texttt{iterate} :: \texttt{Parser} \to \texttt{Parser}$$

$$\texttt{iterate(p)} = M \mapsto \bigcup_{i \in \mathbb{N}_0} \texttt{p}^i(M)$$

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

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

Beispiel: `iterate`

$$\texttt{iterate} :: \texttt{Parser} \to \texttt{Parser}$$
$$\texttt{iterate(p)} = M \mapsto \bigcup_{i \in \mathbb{N}_0} \texttt{p}^i(M)$$

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

$$\underset{0}{|}\,\texttt{b}\,\underset{1}{|}\,\texttt{b}\,\underset{2}{|}\,\texttt{a}\,\underset{3}{|}\,\texttt{a}\,\underset{4}{|}\,\texttt{a}\,\underset{5}{|}\,\texttt{b}\,\underset{6}{|}\,\texttt{b}\,\underset{7}{|}\,\texttt{b}\,\underset{8}{|}$$

$$f_{a^*}(\{(0,0),(2,2),(2,4)\}) = \{\cdots\} \cup f_a(\{\cdots\}) \cup f_a(f_a(\{\cdots\}) \cup \cdots$$
$$= \{(0,0),(2,2),(2,4)\} \cup \{(2,3),(2,5)\} \cup \{(2,4)\} \cup \{(2,5)\} \cup \varnothing \cdots$$
$$= \{(0,0),(2,2),(2,3),(2,4),(2,5)\}$$

Beispiel: `iterate`

$$\texttt{iterate} :: \texttt{Parser} \to \texttt{Parser}$$
$$\texttt{iterate(p)} = M \mapsto \bigcup_{i \in \mathbb{N}_0} \texttt{p}^i(M)$$

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

$$\underset{0}{|}\,\texttt{b}\,\underset{1}{|}\,\texttt{b}\,\underset{2}{|}\,\texttt{a}\,\underset{3}{|}\,\texttt{a}\,\underset{4}{|}\,\texttt{a}\,\underset{5}{|}\,\texttt{b}\,\underset{6}{|}\,\texttt{b}\,\underset{7}{|}\,\texttt{b}\,\underset{8}{|}$$

$$f_{a^*}(\{(0,0),(2,2),(2,4)\}) = \{\cdots\} \cup f_a(\{\cdots\}) \cup f_a(f_a(\{\cdots\}) \cup \cdots$$
$$= \{(0,0),(2,2),(2,4)\} \cup \{(2,3),(2,5)\} \cup \{(2,4)\} \cup \{(2,5)\} \cup \varnothing \cdots$$
$$= \{(0,0),(2,2),(2,3),(2,4),(2,5)\}$$

$$f_{a^*}(\{\texttt{||bbaaabb},\ \texttt{bb||aaabbb},\ \texttt{bb|aa|abbb}\})$$
$$= \{\texttt{||bbaaabb},\ \texttt{bb||aaabbb},$$
$$\texttt{bb|a|aabbb},\ \texttt{bb|aa|abbb},\ \texttt{bb|aaa|bbb}\}$$

Definitionen

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

`type Match = (Int, Int)` (startIndex, endIndex)
`type Parser =` $\{$`Match`$\} \to \{$`Match`$\}$

$\texttt{readChar}_w$ `::` `Char` $\to$ `Parser`
$\texttt{readChar}_w$`(c) =` $M \mapsto \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\}$

`concatenate :: (Parser, Parser)` $\to$ `Parser`
`concatenate(p1,p2) = p2` $\circ$ `p1`

`alternate :: (Parser, Parser)` $\to$ `Parser`
`alternate(p1,p2) =` $M \mapsto$ `p1`$(M) \cup$ `p2`(M)

`iterate :: Parser` $\to$ `Parser`
`iterate(p) =` $M \mapsto \bigcup_{i \in \mathbb{N}_0}$ `p`$^i(M)$

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $\texttt{(a|b)c}^*$ auf w matchen.

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $\texttt{(a|b)c}^*$ auf w matchen.

$f :=$ `concatenate(`
`    alternate(readChar`$_w$`('a'), readChar`$_w$`('b')),`
`    iterate(readChar`$_w$`('c'))`
`)`

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $(\texttt{a}|\texttt{b})\texttt{c}^*$ auf w matchen.

$f_x := \texttt{readChar}_w(x)$ für $x \in \{\texttt{a},\texttt{b},\texttt{c}\}$
$f_{c*} := \texttt{iterate}(f_c)$

$f_{a|b} := \texttt{alternate}(f_a, f_b)$,
$f := \texttt{concatenate}(f_{a|b},\ f_{c*})$.

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $(\texttt{a}|\texttt{b})\texttt{c}^*$ auf w matchen.

$f_x := \texttt{readChar}_w(x)$ für $x \in \{\texttt{a},\texttt{b},\texttt{c}\}$
$f_{c*} := \texttt{iterate}(f_c)$
$f_{a|b} := \texttt{alternate}(f_a, f_b)$,
$f := \texttt{concatenate}(f_{a|b},\ f_{c*})$.

$\texttt{initParser} :: \Sigma^* \to \{\texttt{Match}\}$
$\texttt{initParser(w)} = \{(i,i) \mid i \in [0, |w| - 1]\}$

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $(\texttt{a}|\texttt{b})\texttt{c}^*$ auf w matchen.

$f_x := \texttt{readChar}_w(x)$ für $x \in \{\texttt{a},\texttt{b},\texttt{c}\}$
$f_{c*} := \texttt{iterate}(f_c)$
$\texttt{initParser} :: \Sigma^* \to \{\texttt{Match}\}$
$\texttt{initParser(w)} = \{(i,i) \mid i \in [0, |w| - 1]\}$

$f_{a|b} := \texttt{alternate}(f_a, f_b)$,
$f := \texttt{concatenate}(f_{a|b},\ f_{c*})$.

$$M := \texttt{initParser}(w) = \{(0,0),(1,1),(2,2),(3,3),(4,4),(5,5)\}$$

$$\approx \{\texttt{||xabccx},\ \texttt{x||abccx},\ \texttt{xa||bccx},\ \texttt{xab||ccx},\ \texttt{xabc||cx},\ \texttt{xabcc||x}\}$$

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $(\texttt{a}|\texttt{b})\texttt{c}^*$ auf w matchen.

$f_x := \texttt{readChar}_w(x)$ für $x \in \{\texttt{a},\texttt{b},\texttt{c}\}$
$f_{c*} := \texttt{iterate}(f_c)$
$\texttt{initParser} :: \Sigma^* \to \{\texttt{Match}\}$
$\texttt{initParser(w)} = \{(i,i) \mid i \in [0, |w| - 1]\}$

$f_{a|b} := \texttt{alternate}(f_a, f_b)$,
$f := \texttt{concatenate}(f_{a|b},\ f_{c*})$.

$$\underset{0}{|}\ \texttt{x}\ \underset{1}{|}\ \texttt{a}\ \underset{2}{|}\ \texttt{b}\ \underset{3}{|}\ \texttt{c}\ \underset{4}{|}\ \texttt{c}\ \underset{5}{|}\ \texttt{x}\ \underset{6}{|}$$

$$f(\texttt{initParser}(w)) = f(M) = f_{c*}(f_{a|b}(M)) = f_{c*}(f_a(M) \cup f_b(M))$$

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $\texttt{(a|b)c}^*$ auf w matchen.

$f_x := \texttt{readChar}_w(x)$ für $x \in \{\texttt{a},\texttt{b},\texttt{c}\}$
$f_{c*} := \texttt{iterate}(f_c)$
$\texttt{initParser} :: \Sigma^* \to \{\texttt{Match}\}$
$\texttt{initParser(w)} = \{(i,i) \mid i \in [0, |w| - 1]\}$

$f_{a|b} := \texttt{alternate}(f_a, f_b)$,
$f := \texttt{concatenate}(f_{a|b},\ f_{c*})$.

$$\underset{0}{|}\ \texttt{x}\ \underset{1}{|}\ \texttt{a}\ \underset{2}{|}\ \texttt{b}\ \underset{3}{|}\ \texttt{c}\ \underset{4}{|}\ \texttt{c}\ \underset{5}{|}\ \texttt{x}\ \underset{6}{|}$$

$$f(\texttt{initParser}(w)) = f(M) = f_{c*}(f_{a|b}(M)) = f_{c*}(f_a(M) \cup f_b(M))$$

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

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $\texttt{(a|b)c}^*$ auf w matchen.

$f_x := \texttt{readChar}_w(x)$ für $x \in \{\texttt{a},\texttt{b},\texttt{c}\}$
$f_{c*} := \texttt{iterate}(f_c)$
$\texttt{initParser} :: \Sigma^* \to \{\texttt{Match}\}$
$\texttt{initParser(w)} = \{(i,i) \mid i \in [0, |w| - 1]\}$

$f_{a|b} := \texttt{alternate}(f_a, f_b)$,
$f := \texttt{concatenate}(f_{a|b},\ f_{c*})$.

$$\underset{0}{|}\ \texttt{x}\ \underset{1}{|}\ \texttt{a}\ \underset{2}{|}\ \texttt{b}\ \underset{3}{|}\ \texttt{c}\ \underset{4}{|}\ \texttt{c}\ \underset{5}{|}\ \texttt{x}\ \underset{6}{|}$$

$$f(\texttt{initParser}(w)) = f(M) = f_{c*}(f_{a|b}(M)) = f_{c*}(f_a(M) \cup f_b(M))$$

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

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

Komplettbeispiel

Sei $\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}$, $w = \texttt{xabccx}$. Wir wollen die Regex $(\texttt{a}|\texttt{b})\texttt{c}^*$ auf w matchen.

$f_x := \texttt{readChar}_w(x)$ für $x \in \{\texttt{a},\texttt{b},\texttt{c}\}$
$f_{c*} := \texttt{iterate}(f_c)$
$\texttt{initParser} :: \Sigma^* \to \{\texttt{Match}\}$
$\texttt{initParser(w)} = \{(i,i) \mid i \in [0, |w| - 1]\}$

$f_{a|b} := \texttt{alternate}(f_a, f_b)$,
$f := \texttt{concatenate}(f_{a|b},\ f_{c*})$.

$$\underset{0}{|}\ \texttt{x}\ \underset{1}{|}\ \texttt{a}\ \underset{2}{|}\ \texttt{b}\ \underset{3}{|}\ \texttt{c}\ \underset{4}{|}\ \texttt{c}\ \underset{5}{|}\ \texttt{x}\ \underset{6}{|}$$

$$f(\texttt{initParser}(w)) = f(M) = f_{c*}(f_{a|b}(M)) = f_{c*}(f_a(M) \cup f_b(M))$$

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

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

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

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

Rekursiv gedacht

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

$\texttt{ext}_w$ :: $(\mathcal{R}_\Sigma, \{\texttt{Match}\}) \to \{\texttt{Match}\}$

$$\texttt{ext}_w(r, M) := \begin{cases} \end{cases}$$

Rekursiv gedacht

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

$\texttt{ext}_w$ :: $(\mathcal{R}_\Sigma, \{\texttt{Match}\}) \to \{\texttt{Match}\}$

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

Rekursiv gedacht

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

$\texttt{ext}_w$ :: $(\mathcal{R}_\Sigma, \{\texttt{Match}\}) \to \{\texttt{Match}\}$

$$\texttt{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\} & \text{falls } r = c \in \Sigma \\ \texttt{ext}_w(r_2, \texttt{ext}_w(r_1, M)) & \text{falls } r = (r_1 r_2) \text{ für } r1, r2 \in \mathcal{R}_\Sigma \end{cases}$$

Rekursiv gedacht

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

$\texttt{ext}_w$ :: $(\mathcal{R}_\Sigma, \{\texttt{Match}\}) \to \{\texttt{Match}\}$

$$\texttt{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\} & \text{falls } r = c \in \Sigma \\ \texttt{ext}_w(r_2, \texttt{ext}_w(r_1, M)) & \text{falls } r = (r_1 r_2) \text{ für } r1, r2 \in \mathcal{R}_\Sigma \\ \texttt{ext}_w(r1, M) \cup \texttt{ext}_w(r2, M) & \text{falls } r = (r1 \mid r2) \text{ für } r1, r2 \in \mathcal{R}_\Sigma \end{cases}$$

Rekursiv gedacht

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

$\texttt{ext}_w$ :: $(\mathcal{R}_\Sigma, \{\texttt{Match}\}) \to \{\texttt{Match}\}$

$$\texttt{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\} & \text{falls } r = c \in \Sigma \\ \texttt{ext}_w(r_2, \texttt{ext}_w(r_1, M)) & \text{falls } r = (r_1 r_2) \text{ für } r1, r2 \in \mathcal{R}_\Sigma \\ \texttt{ext}_w(r1, M) \cup \texttt{ext}_w(r2, M) & \text{falls } r = (r1 \mid r2) \text{ für } r1, r2 \in \mathcal{R}_\Sigma \\ \texttt{iter}_w(t, M) & \text{falls } r = t^* \text{ für } t \in \mathcal{R}_\Sigma \end{cases}$$

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

Rekursiv gedacht

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

$\texttt{ext}_w$ :: $(\mathcal{R}_\Sigma, \{\texttt{Match}\}) \to \{\texttt{Match}\}$

$$\texttt{ext}_w(r, M) := \begin{cases} \{(i, j+1) \mid (i,j) \in M \wedge w_j = c\} & \text{falls } r = c \in \Sigma \\ \texttt{ext}_w(r_2, \texttt{ext}_w(r_1, M)) & \text{falls } r = (r_1 r_2) \text{ für } r1, r2 \in \mathcal{R}_\Sigma \\ \texttt{ext}_w(r1, M) \cup \texttt{ext}_w(r2, M) & \text{falls } r = (r1 \mid r2) \text{ für } r1, r2 \in \mathcal{R}_\Sigma \\ \texttt{iter}_w(t, M) & \text{falls } r = t^* \text{ für } t \in \mathcal{R}_\Sigma \end{cases}$$

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

$\Sigma = \{\texttt{a},\texttt{b},\texttt{c},\texttt{x}\}, w = \texttt{xabccx}$:

$$\texttt{ext}_w((\texttt{a}|\texttt{b})\texttt{c}^*, \texttt{initParser}(w))$$

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

`type Match = (Int, Int, Int)` (startIndex, endIndex, priority)

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

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

`type Match = (Int, Int, Int)` (startIndex, endIndex, priority)

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

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

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

`type Match = (Int, Int, Int)` (startIndex, endIndex, priority)

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

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

`decrPrio :: {Match}` $\rightarrow$ `{Match}`
`decrPrio(M)` $= \{(i, j, p + 1) \mid (i, j, p) \in M\}$

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

`type Match = (Int, Int, Int)` (startIndex, endIndex, priority)

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

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

`decrPrio :: {Match}` $\rightarrow$ `{Match}`
`decrPrio(M)` $= \{(i, j, p + 1) \mid (i, j, p) \in M\}$

`alternate(p1,p2)` $= M \mapsto$ `p1`$(M) \cup$ `decrPrio(p2(`M`))`

Erweiterung: Priorität bei Alternation

0 höchste, ∞ niedrigste Priorität.

`type Match = (Int, Int, Int)` (startIndex, endIndex, priority)

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

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

```
decrPrio :: {Match} → {Match}
decrPrio(M) = {(i, j, p + 1) | (i, j, p) ∈ M}
```

`alternate(p1,p2)` $= M \mapsto$ `p1`$(M) \cup$ `decrPrio(p2(`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:
$\Sigma = \{a, b\},\ w = ab.\ f \approx (a|b)^*$.

Erweiterung: Priorität bei Alternation (cont.)

Unerwartetes Verhalten bei Iteration:
$\Sigma = \{a, b\},\ w = ab.\ f \approx (a|b)^*$.

$$f(\texttt{initParser}(w)) = f(\{(0,0,0),(1,1,0)\})$$

$$= \{(0,0,0),(1,1,0)\} \cup f_{a|b}(\{\cdots\}) \cup f_{a|b}(f_{a|b}(\{\cdots\}))\cdots$$

$$= \{(0,0,0),(1,1,0)\} \cup \underbrace{\{(0,1,0)\}}_{f_a(\{\cdots\})} \cup \underbrace{\{(1,2,1)\}}_{f_b(\{\cdots\})} \cup \underbrace{\{(0,2,1)\}}_{f_b(f_{a|b}(\{\cdots\}))}$$

Erweiterung: Priorität bei Alternation (cont.)

Unerwartetes Verhalten bei Iteration:
$\Sigma = \{a, b\},\ w = ab.\ f \approx (a|b)^*$.

$$f(\texttt{initParser}(w)) = f(\{(0,0,0),(1,1,0)\})$$

$$= \{(0,0,0),(1,1,0)\} \cup f_{a|b}(\{\cdots\}) \cup f_{a|b}(f_{a|b}(\{\cdots\}))\cdots$$

$$= \{(0,0,0),(1,1,0)\} \cup \underbrace{\{(0,1,0)\}}_{f_a(\{\cdots\})} \cup \underbrace{\{(1,2,1)\}}_{f_b(\{\cdots\})} \cup \underbrace{\{(0,2,1)\}}_{f_b(f_{a|b}(\{\cdots\}))}$$

$$\Longrightarrow \underbrace{(0,1,0) > (0,2,1)}_{|a|b \quad > \quad |ab|} > (1,1,0) > (1,2,1) > (0,0,0)$$

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>
```

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<A>.filter :: (A → Bool) → Stream<A>
```

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<A>.filter :: (A → Bool) → Stream<A>
Stream.iterate :: (A, A → A) → Stream<A>
```

```
Stream<Double> s = Stream.iterate(0, i -> i+1); // ℕ0
s = s.filter(i -> i % 3 == 0).map(i -> Math.pow(2,i));
s.limit(10).forEach(System.out::println);
```

Ausgabe: $1, 8, 64, 512, 4096$.