

Aufgabe H7 Der Aufruf der Methode `check` gibt für einen regulären Ausdruck (RA) R zurück, ob dieser nur eine endliche Anzahl an Wörtern beschreibt. Folgender Algorithmus kann dies mit einer Worst-Case Laufzeit von $O(n)$ entscheiden, wobei n die Anzahl der Teilausdrücke, die R aufbauen, ist:

```
bool check(RA R) {
    if (R ∈ Σ) return true;
    if (R is A + B || R is AB) return check(A) && check(B);
    if (R is A* || R is A+) return isEmpty(A);
}

bool isEmpty(RA R) {
    if (R is ε) return true;
    if (R ∈ Σ \ {ε}) return false;
    if (R is A + B || R is AB) return isEmpty(A) && isEmpty(B);
}
```

A und B sind hier reguläre Ausdrücke, die hier R darstellen können.

Aufgabe H8

- a) Die Äquivalenzklassen sind $[a]_{\equiv_L} = \{a, b\}^*$, $[c]_{\equiv_L} = \{a, b\}^*c$ und $[ca]_{\equiv_L} = \{a, b\}^*c\{a, b, c\}^*$. Erstere beschreibt alle Worte (keines aus L), bei denen man nur Worte der Form $\{a, b\}^*c$ hinten anhängen kann, sodass sie immer noch in der Sprache L sind. Die zweite Äquivalenzklasse beinhaltet alle Worte der Sprache L . Bei denen kann man nur ϵ anfügen, sodass sie in der Sprache sind. Letztere Äquivalenzklasse beschreibt alle Worte (auch alle aus L), die mindestens ein c haben, wodurch man nichts an das Wort anhängen kann, sodass es in der Sprache ist. Alle drei Äquivalenzklassen vereinigt bilden Σ^* :

$$[a]_{\equiv_L} \cup [c]_{\equiv_L} \cup [ca]_{\equiv_L} = \{a, b\}^* \cup \{a, b\}^*c \cup \{a, b\}^*c\{a, b, c\}^* = \{a, b, c\}^* = \Sigma^*$$

Das leere Wort ist in $[a]_{\equiv_L}$ beinhaltet sowie alle Wörter, die kein c enthalten. $[c]_{\equiv_L}$ beinhaltet alle Wörter, die mind. ein c enthalten, an egal welcher Stelle. Die Vereinigung der drei Äquivalenzklassen stellt also Σ^* dar.

Damit sind diese Äquivalenzklassen die einzigen bzgl. der Myhill-Nerode Relation.

- b)

Aufgabe H9 Der Code ist beigefügt. Für die Aufgabe $\hat{\delta}(7, \text{abababbaa})$ fand das Programm nach ca. 289200ns das Ergebnis

[32, 1, 2, 34, 3, 5, 6, 8, 9, 10, 11, 17, 22, 23, 25, 27, 28, 29, 31].

Für die 4 langen Wörter kam das Programm zu den Endzuständen:

[32, 1, 34, 2, 3, 5, 6, 8, 9, 10, 11, 17, 20, 21, 25, 27, 28, 29, 31]
(6336664800ns)

[32, 1, 34, 3, 4, 5, 6, 8, 9, 12, 13, 17, 18, 19, 25, 27, 29, 30, 31]
(408921700ns)

[32, 1, 34, 3, 4, 5, 6, 8, 9, 17, 18, 19, 25, 27, 29, 30, 31] (2034853500ns)

[32, 1, 34, 3, 4, 5, 6, 8, 9, 17, 18, 19, 25, 27, 29, 30, 31] (21452262300ns)