Prof. Dr. Ralf Hinze
Alexander Dinges, M.Sc.
Sebastian Schloßer, M.Sc.

RPTU
Fachbereich Informatik
AG Programmiersprachen

Übungsblatt 8: Grundlagen der Programmierung (WS 2023/24)

Ausgabe: 19. Dezember 2023
Abgabe: 08./09./10. Januar 2024, siehe Homepage

Sprechstunden zu den Übungen Sie haben Schwierigkeiten mit den Übungsaufgaben und machen sich Sorgen, dass es Ihnen nicht gelingen wird die zur Klausurzulassung nötigen 60% der erreichbaren Punkte zu erlangen?

Dann besuchen Sie unsere Sprechstunden zu den Übungen! Dort erhalten Sie Tipps und Lösungshinweise, wenn Sie mit einer Aufgabe nicht weiterkommen. Sie können dort auch zu früheren Aufgaben Fragen stellen. Alle Informationen zu den Übungssprechstunden finden Sie auf unserer Homepage.

Aufgabe 1 Reguläre Ausdrücke (Präsenzaufgabe)

Motivation: In dieser Aufgabe sollen Sie sich mit regulären Ausdrücken beschäftigen. Die Aufgabe soll Ihnen dabei helfen den Weg von einem regulären Ausdruck schrittweise bis zu einer Akzeptorfunktion nachzuvollziehen. Sie können sich an den Vorlesungsfolien 553 bis 623 sowie am Skript Kapitel 6.1 und 6.2 orientieren.

Wir betrachten den regulären Ausdruck `b*a` über dem Alphabet Alphabet $A = \{\mathsf{a}, \mathsf{b}\}$.

a) Bestimmen Sie **alle** Rechtsfaktoren (inkl. Rechtsfaktoren der Ergebnisse). Geben Sie dabei in der Rechnung jeweils den ersten Schritt explizit an, nachfolgende Zwischenschritte dürfen Sie zusammenfassen.

b) Zeichnen Sie den Aufrufgraphen für den Akzeptor (wie auf Vorlesungsfolie 600).

   Umranden Sie Ausdrücke, die nullable sind, doppelt. Wenn wir beim Einlesen eines Wortes an einem solchen nullable Ausdruck landen und keine weitere Eingabe mehr folgt, gehört das eingelesene Wort zur durch den regulären Ausdruck beschriebenen Sprache. Durch die doppelte Umrandung können wir einfacher ablesen, dass wir an einem möglichen Ende angekommen sind (daher werden solche Knoten auch „Endzustände“ genannt).

c) Implementieren Sie die Akzeptorfunktionen. Gehen Sie dabei **streng nach dem Verfahren aus der Vorlesung** vor (Folie 601). Nutzen Sie für das Alphabet den Typ `type Alphabet = | A | B`.

   Hinweis: Wir empfehlen die einzelnen Akzeptorfunktionen als verschränkt rekursive Hilfsfunktionen innerhlab von `accept` *zu definieren und am Ende die Start-Akzeptorfunktion mit der Eingabe aufzurufen.*

Aufgabe 2 Datentypen (Einreichaufgabe, 5 Punkte)

Motivation: In dieser Aufgabe sollen Sie üben, selber geeignete Datentypen zu definieren. Sie können sich an den Vorlesungsfolien 260 bis 373 und 379 bis 386 sowie am Skript Kapitel 4 orientieren.

Schreiben Sie Ihre Lösungen in die Datei `Datatypes.fs` *aus der Vorlage* `Aufgabe-8-2.zip`.

Hinweis: Zu dieser Aufgabe gibt es keine Tests!

a) Schreiben Sie einen Datentypen, der Mahlzeiten modelliert. Eine Mahlzeit besteht aus einem Namen, einem Preis (in Cent, also einer natürlichen Zahl) und aus einer unbestimmten Anzahl an Zutaten.

b) Schreiben Sie einen Datentypen, der „magische Sequenzen“ modelliert. Eine magische Sequenz ist entweder ein einzelnes „A“ oder ein „B“ gefolgt von zwei magischen Sequenzen

c) Schreiben Sie einen Datentypen, der parametrische Listen modelliert, die immer eine gerade Anzahl an Elementen haben.

d) Schreiben Sie einen Datentypen, der parametrische Bäume modelliert, deren Knoten Werte von einem Typ `'a` enthalten und deren Blätter Werte von einem Typ `'b` enthalten.

e) Schreiben Sie einen Datentypen, der parametrische Multiway-Bäume modelliert. Multiway-Bäume sind entweder Blätter, die einen Wert vom Typ `'a` enthalten, oder Knoten, die eine Liste aus Multiway-Bäumen enthalten.

Aufgabe 3 Reguläre Ausdrücke (Einreichaufgabe, 10 Punkte)

Motivation: In dieser Aufgabe sollen Sie sich mit regulären Ausdrücken beschäftigen. Die Aufgabe soll Ihnen dabei helfen den Weg von einem regulären Ausdruck schrittweise bis zu einer Akzeptorfunktion nachzuvollziehen. Sie können sich an den Vorlesungsfolien 553 bis 623 sowie am Skript Kapitel 6.1 und 6.2 orientieren.

Praxistipp: Das UNIX- bzw. Linuxprogramm `grep`[1] erlaubt die Suche in Dateien und Datenströmen anhand von regulären Ausdrücken. Unter Windows stellt das PowerShell Kommando `Select-String -Pattern` eine ähnliche Funktionalität zur Verfügung.

Hinweis: Die Präzedenz der Operatoren ist wie folgt definiert: Die Alternative (`|`) bindet am schwächsten, dann folgt die Konkatenation ($\cdot$) und zuletzt der Stern (`*`).

Schreiben Sie Ihre Lösungen in die Datei `RegExp.fs` *aus der Vorlage* `Aufgabe-8-3.zip`.

Wir betrachten den regulären Ausdruck `ab(ab)*|ba(ba)*` über dem Alphabet $A = \{\mathtt{a}, \mathtt{b}\}$.

a) Bestimmen Sie **alle** Rechtsfaktoren (inkl. Rechtsfaktoren der Ergebnisse). Geben Sie dabei in der Rechnung jeweils den ersten Schritt explizit an, nachfolgende Zwischenschritte dürfen Sie zusammenfassen.

b) Zeichnen Sie den Aufrufgraphen für den Akzeptor (wie auf Vorlesungsfolie 600).

   Umranden Sie Ausdrücke, die nullable sind, doppelt. Wenn wir beim Einlesen eines Wortes an einem solchen nullable Ausdruck landen und keine weitere Eingabe mehr folgt, gehört das eingelesene Wort zur durch den regulären Ausdruck beschriebenen Sprache. Durch die doppelte Umrandung können wir einfacher ablesen, dass wir an einem möglichen Ende angekommen sind (daher werden solche Knoten auch „Endzustände“ genannt).

c) Implementieren Sie die Akzeptorfunktionen. Gehen Sie dabei **streng nach dem Verfahren aus der Vorlesung** vor (Folie 601). Nutzen Sie für das Alphabet den Typ `type Alphabet = | A | B`.

   Hinweis: Wir empfehlen die einzelnen Akzeptorfunktionen als verschränkt rekursive Hilfsfunktionen innerhlab von `accept` *zu definieren und am Ende die Start-Akzeptorfunktion mit der Eingabe aufzurufen.*

[1] https://de.wikipedia.org/wiki/Grep