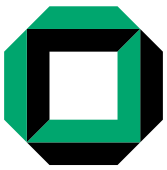

Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen
Sprachtechnologie und Compiler WS 2008/2009 http://pp.info.uni-karlsruhe.de/
Dozent: Prof. Dr.-Ing. G. Snelting snelting@ipd.info.uni-karlsruhe.de
Übungsleiter: Matthias Braun braun@ipd.info.uni-karlsruhe.de

Übungsblatt 1 | Ausgabe: 23.10.2008 | Besprechung: 30.10.2008

Aufgabe 1: *Korrektheit von Übersetzungen*

Gegeben sei folgendes (Basic-) Programm (alle Zahlen sind vorzeichenbehaftet):

```
dim i as integer
for i = 1 to n step k
    print i
next
```

1.1 *Erster Versuch*

Angenommen `k = 1`. Wie würden Sie die Schleife übersetzen, wenn Ihre Zielsprache[1] bedingte Sprünge und Vergleiche zulässt?

1.2 *Zweiter Versuch*

Verhält sich Ihre Implementierung korrekt, wenn `n=maxint`?

1.3 *Dritter Versuch*

Verhält sich Ihre Implementierung korrekt, wenn `k` keine Konstante aber positv (vorzeichenlos) ist? Wie müssen Sie sie ändern?

Aufgabe 2: *Korrektheit von Übersetzungen*

Gegeben sei folgendes Programm:

```
while(bedingung) {
    i = i+1;
    a = a*i;
    b = (x1 * x2) / (x1 + x2);
}
```

Alle Variable sind primitive Typen.

2.1 *Optimierung I*

Darf ein Übersetzer die Berechnung von `b` aus der Schleife herausziehen?

2.2 *Optimierung II*

Ersetzen Sie *bedingung* durch `i<100 && (x1 + x2) != 0` und überdenken Sie Ihre Antwort aus 1!

[1] Die Zielsprache ist beliebig, man sollte nur keine Schleifenkonstrukte verwenden.

Aufgabe 3: *Korrektheit von Übersetzungen*

Gegeben sei folgendes Programm:

```
#include <stdio.h>

int main(void)
{
    float r = 0.7f * 100000.0f + 0.3f;
    r = r * 1000000.0f;
    r = (r - 300000.0f)/10000000000.0f;
    if(r == 7.0f) {
        printf("Ok");
    } else {
        printf("Not␣OK,␣r␣is␣not␣7,␣but␣%f\n", r);
    }
    return 0;
}
```

3.1 *Semantik & Pragmatik*

Überraschender Weise gibt das Programm `Not OK, r is not 7, but 7.000000` aus. Was kann dazu geführt haben?

3.2 *Schuldfrage*

Ist dem Übersetzerbauer ein Fehler unterlaufen?

Aufgabe 4: *Reguläre Ausdrücke*

4.1 *Regulärer Ausdruck → Sprache*

Beschreiben Sie die Sprachen, die mit den folgenden regulären Ausdrücken bezeichnet werden:

1. $\mathtt{a}(\mathtt{a}+\mathtt{b})^*a$
2. $((\epsilon+\mathtt{a})\mathtt{b}^*)^*$
3. $(\mathtt{a}+\mathtt{b})^*\mathtt{a}(\mathtt{a}+\mathtt{b})(\mathtt{a}+\mathtt{b})$
4. $\mathtt{a}^*\mathtt{b}\mathtt{a}^*\mathtt{b}\mathtt{a}^*\mathtt{b}\mathtt{a}^*$

4.2 *endliche Automaten*

Entwerfen Sie (deterministische oder nichtdeterministische) endliche Automaten für alle Sprachen aus der letzten Teilaufgabe.

4.3 *Sprache → Regulärer Ausdruck*

Schreiben Sie reguläre Definitionen für folgende Sprachen. Für die Konstruktion $(\mathtt{a}+\ldots+\mathtt{z})$ können sie die Kurzschreibweise $[a-z]$ verwenden.

1. Alle Strings aus Kleinbuchstaben, die die fünf Vokale in ihrer Reihenfolge enthalten.
2. Alle Strings aus Kleinbuchstaben, in der die Buchstaben in absteigender alphabetischer Reihenfolge erscheinen
3. Kommentare, die aus einem in `/*` und `*/` eingeschlossenen String bestehen, ohne `*/` dazwischen - es sei denn, innerhalb von doppelten Anführungszeichen "
4. Die Menge aller Schachzüge in informeller Schreibweise, zum Beispiel `p-k4` oder `kbp x qn`
5. Alle Strings aus `a` und `b` mit einer geraden Anzahl von `a` und einer ungeraden Anzahl von `b`

Aufgabe 5: *Reguläre Ausdrücke*

E bezeichne die leere Sprache, *eps* die Sprache, die nur die leere Zeichenkette ϵ enthält. Was ist dann

1. EX
2. $epsX$
3. $E + X$
4. $eps + X$

für eine beliebige Menge X von Zeichenketten?