Karlsruher Institut für Technologie

Lehrstuhl für Programmierparadigmen

Joachim Breitner
breitner@kit.edu

Denis Lohner
denis.lohner@kit.edu

Semantik von Programmiersprachen – SS 2015

http://pp.ipd.kit.edu/lehre/SS2015/semantik

Blatt 5: Compiler Besprechung: 11.05.2012

1. Welche der folgenden Aussagen sind richtig, welche falsch? (H)

(a) [`ASSN x 5`, `JMPF` -1 `true`, `JMP 2`, `ASSN y (x + 7)`] ist ein ASM-Programm.

(b) $[\texttt{JMP } 0] \vdash \langle 0,\ \sigma\rangle \overset{\infty}{\to}$

(c) comp(`if (x <= 0) then x := 0 - x; y := 0 else skip`) =
[`JMPF 3 (x <= 0)`, `ASSN x (0 - x)`, `ASSN y 0`]

(d) [`JMP 3`, `ASSN y 0`, `JMPF` -3 $(y$ `== 1)`, `ASSN y 1`] ist abgeschlossen.

(e) Wenn $P +\!+ P' \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |P| + |P'|,\ \sigma'\rangle$, dann gibt es ein σ^* mit $P \vdash \langle 0,\ \sigma\rangle \overset{*}{\to} \langle |P|,\ \sigma^*\rangle$ und $P' \vdash \langle 0,\ \sigma^*\rangle \overset{*}{\to} \langle |P'|,\ \sigma'\rangle$.

(f) Zu jedem While-Programm gibt es ein semantisch äquivalentes ASM-Programm.

2. `repeat` c `until` b-Schleife (H)

Neben `while`-Schleifen sind auch `repeat`-Schleifen in vielen Programmiersprachen verbreitet. In dieser Aufgabe sei Com um die Produktion `repeat` c `until` b erweitert. Die Small-Step-Semantik für `repeat` c `until` b sei durch folgende Regel gegeben:

$$\textsc{Repeat}: \langle \texttt{repeat } c \texttt{ until } b,\ \sigma\rangle \to_1 \langle c\texttt{; if (}b\texttt{) then skip else repeat } c \texttt{ until } b,\ \sigma\rangle$$

(a) Geben Sie Regeln in der Big-Step-Semantik an, die `repeat` c `until` b das gleiche Verhalten geben wie Repeat in der Small-Step-Semantik. Welche Schritte wären für einen Äquivalenzbeweis nötig?

(b) Erweitern Sie die Definition des Compilers comp auf Repeat-Schleifen.

(c) Erweitern Sie den Korrektheitsbeweis des Compilers (Thm. 12 und 17) um Repeat-Schleifen.

3. Compiler für arithmetische Ausdrücke (Ü)

Für die arithmetischen Ausdrücke Aexp der Sprache While soll ein Compiler angegeben werden. Der Compiler nimmt einen Ausdruck und übersetzt ihn in eine Liste von Instruktionen einer Stackmaschine. Auf der Stackmaschine stehen folgende Instruktionen zur Verfügung:

`CONST` n Lege die Konstante n auf den Stack ($n \in \mathbb{Z}$).

`LOAD` x Lade den Wert der Variablen x auf den Stack.

`APPLY` f Wende die binäre Funktion f auf die beiden obersten Stackelemente an und ersetze sie durch das Ergebnis.

Die Maschine, auf der die übersetzten Ausdrücke abgearbeitet werden sollen, nimmt eine Liste von Instruktionen, einen (anfangs leeren) Stack und einen Zustand mit der aktuellen Variablenbelegung als Argumente. Auf die Variablenbelegung kann mit Hilfe von `LOAD` _ zugegriffen werden. Die Stackmaschine liefert einen neuen Stack als Ergebnis, der am Ende der Berechnung nur noch deren Ergebnis enthalten soll.

(a) Geben Sie eine formale Semantik für die Stack-Maschine an.

(b) Schreiben Sie einen Compiler für arithmetische Ausdrücke.

(c) Formulieren Sie die Aussage, dass der Compiler korrekt ist, formal.

(d) Beweisen Sie die Korrektheitsaussage.