
Semantik von Programmiersprachen – SS 2019

<http://pp.ipd.kit.edu/lehre/SS2019/semantik>

Blatt 4: Big-Step- und Small-Step-Semantik

Besprechung: 20.05.2019

1. Semantik mit Ausführungszeiten (H)

Die Big-Step-Semantik für `While` berücksichtigt nicht, wie viele Schritte die Ausführung eines Programms benötigt. Dies sollen Sie in dieser Aufgabe modellieren:

- Definieren Sie, auf Papier oder in Prolog, eine Auswertungsrelation $\langle c, \sigma \rangle \Downarrow_t \sigma'$ mit der Bedeutung, dass die Ausführung von c im Anfangszustand σ im Endzustand σ' endet und dafür t Schritte benötigt. Finden Sie dafür eine geeignete Definition, was ein Schritt sein soll.
- Beschreiben Sie formal, in welcher Beziehung $\langle c, \sigma \rangle \Downarrow \sigma'$ und $\langle c, \sigma \rangle \Downarrow_t \sigma'$ stehen. Wie würde man diese Beziehung beweisen?
- Überprüfen Sie, welche der folgenden Eigenschaften der Big-Step-Semantik $\langle c, \sigma \rangle \Downarrow \sigma'$ sich auf $\langle c, \sigma \rangle \Downarrow_t \sigma'$ übertragen lassen. Formulieren Sie die Aussagen entsprechend. Wie müssten die Beweise angepasst werden?
 - Schleifenabwicklungslemma (Lem. 9)
 - Determinismus (Thm. 10)
 - Äquivalenz zwischen Big-Step- und Small-Step-Semantik (Kor. 23)

2. Semantische Äquivalenz bei Small-Step (Ü)

Zwei Programme c und c' sind *äquivalent* bezüglich der Small-Step-Semantik, falls für alle Zustände σ, σ' gilt:

- $\langle c, \sigma \rangle \xrightarrow{*}_1 \langle \text{skip}, \sigma' \rangle$ genau dann, wenn auch $\langle c', \sigma \rangle \xrightarrow{*}_1 \langle \text{skip}, \sigma' \rangle$ und
- $\langle c, \sigma \rangle \xrightarrow{\infty}_1$ genau dann, wenn auch $\langle c', \sigma \rangle \xrightarrow{\infty}_1$.

Vergleichen Sie diesen Äquivalenzbegriff mit dem semantischen Äquivalenzbegriff für die Big-Step-Semantik.

Beweisen Sie, dass die folgenden Programme im Small-Step-Sinn äquivalent sind:

- `while (b) do c` und `if (b) then c; while (b) do c else skip`
- `while (true) do c'` und `c; while (true) do c''`
- Wenn c und c' äquivalent sind, dann auch `while (b) do c` und `while (b) do c'`.

Überlegen Sie sich einen sinnvollen Äquivalenzbegriff, für den (c) nicht gilt.