Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Clemens Ballarin

Vorlesung Semantik
Wintersemester 2002/03
Übungsblatt 6
Besprechung am 28. November 2002

Aufgabe 1 (H) *(Korrektheit des WHILE-Compilers)*

In der Vorlesung wurde eine Richtung der Korrektheit des Compilers für WHILE mit struktureller Induktion über s und Induktion über die Länge der Ableitung $\rightarrow^*$ gezeigt.

$$\text{comp } s \vdash \langle 0, \sigma\rangle \rightarrow^* \langle |\text{comp } s|, \sigma'\rangle \;\Rightarrow\; \langle s, \sigma\rangle \rightarrow \sigma'$$

a) An welcher Stelle und aus welchem Grund scheitert der Beweis, wenn man nur mit Induktion über die Länge der Ableitung arbeitet?

b) Wie müsste man die Funktion `comp` und die abstrakte Maschine ändern, damit der Beweis nicht scheitert?

Aufgabe 2 (H) (*Definite Assignment*)

Wird in einer Programmiersprache *Definite Assignment* verlangt, bedeutet dies die Forderung, dass alle Variablen vor ihrer ersten Benutzung auch initialisiert werden. Das Statement $x := 1$ ist zum Beispiel also auf jeden Fall zulässig, $x := y$ dagegen darf nur zugelassen werden, wenn feststeht, dass der Variablen y vorher ein Wert zugewiesen wurde. Definite Assignment lässt sich mit einem System von Typregeln beschreiben.

a) Geben Sie Typregeln für Definite Assignment in der Sprache WHILE an. Benutzen Sie dazu die Schreibweise $\{V\}\ s\ \{W\}$, die bedeutet: „Wenn alle Variablen in V initialisiert sind, dann werden nach der Ausführung von s alle Variablen in W initialisiert sein“. Hierbei bezeichnen V und W Mengen von Variablen.

   Die Typregeln sollen überprüfen, ob nur initialisierte Variablen benutzt werden, und gleichzeitig die Menge der initialisierten Variablen berechnen.

b) Modifizieren Sie die Big-Step-Semantik so, dass beim Benutzen einer nicht initialisierten Variable ein Fehler auftritt.

c) Formulieren Sie formal die Korrektheitsaussage: „In typkorrekten Programmen tritt kein Fehler wegen unitialisierten Variablen auf.“

Aufgabe 3 (Ü) (*Statische Modifikation von Typen*)

Das Typsystem der Sprache WHILE$_T$ soll derartig erweitert werden, dass Typen von Variablen im Programmverlauf verändert werden können. Beispielsweise soll folgendes Programm als typkorrekt angesehen werden: $X := 5; X := \mathit{true}$.

a) Entwickeln Sie Typisierungsregeln für ein solches Typsystem. Beachten Sie dabei, dass sich der Deklarationskontext Γ während des Programmverlaufs verändern kann. Deshalb sollen Typisierungsregeln die Form $\{\Gamma\}\ s\ \{\Gamma'\}$ besitzen mit der Bedeutung, dass s typkorrekt ist, falls Γ vor der Ausführung und Γ' nach der Ausführung gilt.

b) Geben Sie eine Big-Step-Semantik für die erweiterte Sprache an.

c) Beweisen Sie die Typsicherheit.

Aufgabe 4 (P) (*Typchecking und Typinferenz*)

In der Vorlesung wurde die um Typen erweiterte Sprache WHILE_T vorgestellt.

a) Programmieren Sie den Typüberprüfungsalgorithmus dieser Sprache in der Programmiersprache Prolog.

b) Testen Sie Ihr Programm als Typinferenzverfahren, indem Sie für ein geeignetes Beispiel die Typen ausrechnen lassen.

Zu dieser Aufgabe gibt es kein Rahmenprogramm!