
Semantik von Programmiersprachen
<http://www-pu.informatik.uni-tuebingen.de/semantik-2002/>

Blatt 9

Abgabe: 20.1.2003

1. [30 Punkte] Betrachten Sie folgende Erweiterungen der imperativen Sprache aus der Vorlesung:

- (a) Betrachten Sie die Erweiterung der Sprache mit **fail** um den **catchin**-Befehl:

$$\langle \text{comm} \rangle ::= \mathbf{catchin} \langle \text{comm} \rangle \mathbf{with} \langle \text{comm} \rangle$$

Dieser Befehl sorgt dafür, daß es nach einem Fehlschlag im Programm weitergehen kann: **catchin** c_0 **with** c_1 führt zunächst zur Ausführung von c_0 . Falls c_0 normal terminiert (also ohne einen Fehlschlag, der nicht durch einen weiter innen sitzendes **catchin**-Befehl abgefangen wird), dann terminiert auch **catchin** c_0 **with** c_1 normal mit dem Resultat von c_0 . Falls jedoch c_0 durch Fehlschlag terminiert, dann wird c_1 in dem Zustand ausgeführt, in dem c_0 fehlgeschlagen ist. **catchin** c_0 **with** c_1 terminiert dann (normal oder durch Fehlschlag) mit dem Ergebnis von c_1 .

Erweitern Sie die Semantik aus der Vorlesung um das neue Konstrukt. Dabei soll der Typ der semantischen Funktionen unverändert bleiben.

- (b) Betrachten Sie die Erweiterung um markierte Fehlschläge:

$$\langle \text{comm} \rangle ::= \mathbf{fail} \langle \text{label} \rangle \mid \mathbf{catch} \langle \text{label} \rangle \mathbf{in} \langle \text{comm} \rangle \mathbf{with} \langle \text{comm} \rangle$$

Dabei stammt $\langle \text{label} \rangle$ aus einer abzählbaren Menge von Markierungen. Der neue Befehl **fail** ℓ verursacht nun einen Fehlschlag mit Markierung ℓ . Der Befehl **catch** ℓ **in** c_0 **with** c_1 führt zur Ausführung von c_0 . Falls c_0 normal terminiert oder mit einem Fehlschlag mit einer anderen Markierung als ℓ , dann terminiert **catch** ℓ **in** c_0 **with** c_1 ebenso mit dem gleichen Ergebnis. Falls c_0 mit einem Fehlschlag mit Markierung ℓ terminiert (der nicht durch einen weiter innen sitzenden **catch**-Befehl abgefangen wird), dann wird c_1 in dem Zustand ausgeführt, in dem c_0 terminiert ist, und **catch** ℓ **in** c_0 **with** c_1 terminiert mit dem Ergebnis von c_1 .

Hier ein Beispiel:

$$\mathbf{catch} \text{ cold} \mathbf{in} \mathbf{catch} \text{ cold} \mathbf{in} (x := 0; \mathbf{fail} \text{ cold}) \mathbf{with} y := 0 \mathbf{with} z := 0$$

ist äquivalent zu $x := 0; y := 0$. Allerdings ist

$$\mathbf{catch} \text{ cold} \mathbf{in} \mathbf{catch} \text{ flu} \mathbf{in} (x := 0; \mathbf{fail} \text{ cold}) \mathbf{with} y := 0 \mathbf{with} z := 0$$

äquivalent zu $x := 0; z := 0$.

Erweitern Sie die Semantik aus der Vorlesung um die neuen Konstrukte. Der Typ der Bedeutungsfunktion für Befehle soll unverändert bleiben, allerdings ändert sich die Bedeutung von $\hat{\Sigma}$:

$$\hat{\Sigma} = \Sigma \cup (\langle \text{label} \rangle \times \Sigma)$$

Dabei ist $\langle \ell, \sigma \rangle$ das Resultat eines Befehls, der mit Markierung ℓ fehlschlägt.