Vorlesung vom 13.06.06: Ausnahmebehandlung (in Java und IMP)

Wo sind wir?

- Hoare-Kalkül
- Jenseits von Hoare:
  - Einführung
  - *Reasoning in JML — Ausnahmen*
  - Werkzeugunterstützung
- Korrekte Programmentwicklung

Eine Semantik für JML

- Problem: *Semantik* für *JML* braucht *Semantik* für *Java*.
- Verschiedene Ansätze:
  - Koalgebraisch (Huisman, Jacobs *et al.*)
  - Denotational (Leino *et al.*)
  - Axiomatisch (Poetzsch-Hefter; KeY)
- Plan heute:
  - Behandlung von *Ausnahmen* in **IMP**
  - Verallgemeinerung für Java: *abrupte Termination*

Ausnahmen

- Erweiterung von **IMP** um *Ausnahmen*: **Com**

$$c ::= \ldots \mid \texttt{throw}\ a \mid \texttt{try}\ c_0\ \texttt{catch}\ X.c_1$$

  Menge **Exn** von *Ausnahmeausdrücken*
- *Semantik*: Menge E von *Ausnahmen*

$$\begin{aligned}\mathcal{D} &: \textbf{Com} \to (\Sigma \to (E * \Sigma) + \Sigma_\perp\\ \mathcal{E}x &: \textbf{Exn} \to (\Sigma \to E)\end{aligned}$$

  - Entweder normaler *Folgezustand* ($inr(\sigma)$)
  - . . . oder *Ausnahme* und Folgezustand ($inl(e, \sigma)$)
- Auch Ausnahmen haben *Folgezustand*!

Semantik mit Ausnahmen

$$\begin{aligned}
\mathcal{C}[\![\texttt{throw}\ a]\!] &\stackrel{def}{=} \lambda\sigma.inl(\mathcal{E}x[\![a]\!]\sigma, \sigma)\\
\mathcal{C}[\![c_0; c_1]\!] &\stackrel{def}{=} \lambda\sigma.cases\ \mathcal{C}[\![c_0]\!]\sigma\ of\\
&\qquad\qquad inr(\sigma') \to \mathcal{C}[\![\sigma_1]\!]\sigma'\\
&\qquad\qquad inl(e, \sigma') \to inl(e, \sigma')\\
\mathcal{C}[\![\texttt{try}\ c_0\ \texttt{catch}\ X.c_1]\!] &\stackrel{def}{=} \lambda\sigma.cases\ \mathcal{C}[\![c_0]\!]\sigma\ of\\
&\qquad\qquad inr(\sigma') \to inr(\sigma')\\
&\qquad\qquad inl(e, \sigma') \to \mathcal{C}[\![c_1]\!](\sigma'[\mathcal{E}x[\![a]\!]\sigma/X])
\end{aligned}$$

Entsprechende Änderungen auch in $\mathcal{E}, \mathcal{A}$ (Ausnahmen in *Ausdrücken*!)

Hoare-Kalkül mit Ausnahmen

- Hoare-Tripel mit *Ausnahmeverhalten*
- $\{A\}c\{B \parallel X.S\}$
  - B *Normale* Nachbedingung
  - S *Ausnahemverhalten*, X kann Wert der Ausnahme binden
- Erfüllbarkeit:

$$\begin{aligned}&\sigma \models^I \{A\}c\{B \parallel X.S\} \Longleftrightarrow\\ &\qquad \sigma \models^I A \Longrightarrow cases\ \mathcal{C}[\![c]\!]\sigma\ of\ \ inr(\sigma') \to (\sigma' \models^I B)\\ &\qquad\qquad\qquad\qquad\qquad\qquad\quad inl(e, \sigma') \to (\sigma'[e/X] \models^I S)\end{aligned}$$

Hoare-Kalkül mit Ausnahmen

- *Anpassung* einiger Regeln:

$$\frac{\{A\}c_0\{C \parallel X.S\} \quad \{C\}c_1\{B \parallel X.S\}}{\{A\}c_0; c_1\{B \parallel X.S\}}Seq$$

- *Neue Regeln*:

$$\frac{}{\{A\}\texttt{throw}\ a\{\perp \parallel X.X = a \land A\}}Throw$$

  wobei X nicht in A auftritt.

$$\frac{\{A\}c_0\{B \parallel X.S\} \quad \{A\}c_1\{S[X/a] \parallel T\}}{\{A\}\texttt{try}\ c_0\ \texttt{catch}\ X.c_1\{B \lor S[X/a] \parallel T\}}Catch$$

Ausnahmen in Java

- Java nutzt Ausnahmen für
  - normale Ausnahmen (`Exception`, `try`, `catch`)
  - Rückgabewerte (`return`)
  - Abbruch einer bestimmten Schleife (`break`)
  - Nichtabbruch einer bestimmten Iteration (`continue`)
- Viele *verschiedene Fälle* in *Regeln*!

Eine Java-Semantik

- Für **IMP**: *Zustandstransformer* $\Sigma \to \Sigma_\perp$ mit $\Sigma_\perp = \Sigma + 1$. Hier:

$$\begin{aligned} & \Sigma \to \Sigma_\perp + \mathbf{StatAbn} \\ \mathbf{StatAbn} &= \mathbf{Exn} + \mathbf{Return} + \mathbf{Break} + \mathbf{Cont} \\ \mathbf{Return} &= \Sigma * F \\ \mathbf{Break} &= \Sigma * L \\ \mathbf{Cont} &= \Sigma * L \end{aligned}$$

  mit F Menge von *Funktionsnamen*, L Menge von *Labeln*
- Für Ausdrücke vom Typ α:

$$\Sigma \to (\Sigma * \alpha)_\perp + \mathbf{Exn}$$

Java-Semantik: Weitere Feinheiten

- *Speicherverwaltung* in der *JVM*
  - Felder (Arrays)
  - Referenzen (z.B. auf Ausnahmen)
- *Dynamische* Bindung: *Methodenaufruf*
- *Klassendefinitionen*

Lokale Variablen

- Erweiterung von **Com** um *Blöcke*:

$$c ::= \ldots \mid \texttt{begin var } v_1, \ldots v_n;\ c \texttt{ end}$$

- Neue Regel:

$$\frac{\{A\} c \{B\}}{\{A\}\texttt{begin var } v_1, \ldots v_n;\ c \texttt{ end}\{B\}} \mathit{Block}$$

  wobei $v_1, \ldots, v_n$ *nicht* in A oder B *auftreten*.

Zusammenfassung

- Java-Semantik (Ausblick)
- Besonderheiten von Java: Ausnahmeverhalten, lokale Variablen
- Erweiterung von **IMP** um *einfache Ausnahmen*
  - . . . aber Java hat viele *Ausnahmen*
- *Blockregel* für *lokale* Variablen