Syntaktische Analyse eines Termsystems für nicht-wachsende Ptime Berechnungen

Klaus Aehlig

Ptime

- Alle Funktionen die auf Turingmaschinen berechenbar sind, wobei die Zahl der Schritte beschränkt ist durch ein Polynom in der Länge der Eingabe

- Kurz: berechenbar durch effiziente Algorithmen ("feasible computation")

- Insbesondere ist das Wachstum solcher Funktionen polynomial beschränkt

$\texttt{insertion sort} \quad \& \quad x \mapsto 2^x$

$$
\begin{array}{lcl}
\texttt{sort}([]) & = & [] \\
\texttt{sort}([a \mid l]) & = & \texttt{insert}(a, \texttt{sort}(l)) \\
\\
\texttt{insert}(a, []) & = & [a] \\
\texttt{insert}(a, [b \mid \ell]) & = & [b' \mid \texttt{insert}(a', \ell)]
\end{array}
\qquad
\begin{array}{lcl}
\texttt{exp}(0) & = & S0 \\
\texttt{exp}(Sx) & = & \texttt{double}(\texttt{exp}(x)) \\
\\
\texttt{double}(0) & = & 0 \\
\texttt{double}(Sx) & = & S(S(\texttt{double}(x)))
\end{array}
$$

a', b' Permutation von a, b mit $b' \leq a'$

Größe des Rekursionsarguments?

- Ansatz:

  "klein, da alles (von diesem Typ) klein ist"

- Also nur:

  nicht-wachsende `Ptime` Funktionen

- Nicht wachsende Maßzahl?!

$$t \to t' \Longrightarrow \mathrm{FV}\,(t) \supset \mathrm{FV}\,(t')$$

Daraus ein Termsystem?

- **1. Schritt:** Freie Variablen erzwingen
  - Neuer Typ $\diamond$ mit
$$t^\diamond \Longrightarrow \mathrm{FV}\,(t) \neq \emptyset$$
  - Nachfolger vom Typ
$$\diamond \multimap \mathbf{N} \multimap \mathbf{N}$$

- **Einwand:**
$$\mathbf{S}_1 d^\diamond (\mathbf{S}_1 d^\diamond (\ldots (\mathbf{S}_1 d^\diamond 0)))$$

- **2. Schritt:** Lineare Typisierung fordern. Also (st) nur, falls
$$\mathrm{FV}\,(s) \cap \mathrm{FV}\,(t) = \emptyset$$

Überblick über das Termsystem

- Eliminationen als Applikationen geschrieben ("Vektorschreibweise"):
  - Applikation: $t^{\tau \multimap \rho} s$
  - Iteration: $t^{\mathbf{L}(\tau)} \{s\}\, r$
    $t^{\mathbf{N}} \{s_0, s_1\}\, r$
  - Projektion: $t^{\tau \times \rho}\mathbf{t}$ und $t^{\tau \times \rho}\mathbf{f}$
  - Tensor-Elimination: $t^{\tau \otimes \rho}(\lambda x^{\tau}, y^{\rho}\, s)$
  - Fallunterscheidung: $t^{\mathrm{bool}} \langle s_0, s_1 \rangle$

- Abstraktionen
  - λ-Abstraktion: $\lambda x\, t$
  - Paar: $\langle s, t \rangle$

Überblick über das Termsystem (cont'd)

- Datentypen (durch Konstanten eingeführt):
  - binäre Numerale: 0, $\mathbf{S}_0$, $\mathbf{S}_1$
  - Listen: $\mathbf{nil}_\tau$, $\mathbf{cons}_\tau$
  - Tensor: $\otimes_{\tau,\rho}$
  - boolesche Werte: $\mathbf{t}$, $\mathbf{f}$

- Typ $\diamond$: $x^\diamond$

. . . **and now in all glory details** . . .

$\texttt{insertion sort}$ again

Sei τ ein Typ, auf dem eine lineare Ordnung definiert ist. Diese möge durch einen Term $\lhd_\tau$ vom Typ $\tau \multimap \tau \multimap \mathrm{bool} \otimes (\tau \otimes \tau)$ berechnet werden, das heißt $\lhd_\tau ts \to^* \otimes_{\mathrm{bool},\tau\otimes\tau}\mathbf{t}(\otimes_{\tau,\tau}ts)$, falls s "kleiner" als t und $\lhd_\tau ts \to^* \otimes_{\mathrm{bool},\tau\otimes\tau}\mathbf{f}(\otimes_{\tau,\tau}ts)$ sonst.

Man definiert zunächst eine "elementare Sortierfunktion" $\lhd'_\tau$ vom Typ $\tau \multimap \tau \multimap \tau \otimes \tau$, die zwei Werte nimmt und in sortierter Reihenfolge wieder ausgibt.

$$\lhd'_\tau = \lambda p_1^\tau, p_2^\tau \;\lhd_\tau p_1 p_2 (\lambda y^{\mathrm{bool}}, p^{\tau\otimes\tau}\,(p(\lambda p_1^\tau, p_2^\tau\, y\,\langle \otimes_{\tau,\tau} p_1 p_2, \otimes_{\tau,\tau} p_2 p_1\rangle)))$$

Dann:

$$\texttt{insert} = \lambda l\, l \left\{\lambda x_1^\diamond, y_1^\tau, p^{\diamond\multimap\tau\multimap\mathbf{L}(\tau)}, x_2^\diamond, y_2^\tau \;\lhd'_\tau y_1 y_2 (\lambda z_1^\tau, z_2^\tau\, \mathbf{cons}_\tau x_1 z_1 (p x_2 z_2))\right\}$$
$$(\lambda x^\diamond, y^\tau\, \mathbf{cons}_\tau xy\mathbf{nil}_\tau)$$

Und schließlich: $\texttt{insertionsort} = \lambda l^\tau\, l\,\{\lambda x^\diamond, y^\tau, l^{\mathbf{L}(\tau)}\, \texttt{insert} lxy\}\,\mathbf{nil}_\tau$