Berechenbarkeit und λ–Definierbarkeit

Sei $C*$ eine Menge konstanter Objekte, $U,\ V \subseteq C*$; eine partielle Funktion $f \subseteq U \times V$ heisst intuitiv berechenbar falls es eine mechanisierbare Methode gibt, mithilfe deren zu jedem Argument $e \in D \subseteq U$ nach endlich vielen Transformationsschritten ein Funktionswert $f(\ e\ ) \in V$ hergeleitet werden kann, und zu jedem Argument $e \in U \setminus D$ entweder keine Transformationsfolge terminiert (ist nicht entscheidbar), oder aber nach endlicher Folge der Wert $\perp \notin V$ hergeleitet wird. $\|$

Eine partielle Funktion ist intuitiv berechenbar genau dann wenn sie λ–definierbar ist. $\|$

Eine partielle Funktion $f \subseteq {N_0}^n \times N_0$ heisst λ–definierbar g.d.w. es einen λ–term e gibt, so dass

$$(\ e\ x_1 \ldots\ x_n\ )\ =\ f(\ x_1,\ \ldots\ ,\ x_n\ ) \mid x_1, \ldots\ ,\ x_n \in N_0 \quad \|$$

Abstrakte λ–Calculus Maschine

$\rightarrow$ basiert auf einem von Landin 1964 entwickelten Maschinenmodell $\rightarrow$ sog. SECD–machine $\rightarrow$ benannt nach den Bezeichnern fuer verwendete Laufzeitstrukturen

$\rightarrow$ bildet Basis fuer daraus abgeleitete aufwendigere Maschinenmodelle zur Ausfuehrung von Programmen funktionaler | funktionsbasierter | imperativer Sprachen

$\rightarrow$ schraenkt den λ–Kalkuel soweit ein, dass er mit einfachen Mechanismen realisierbar ist $\rightarrow$ keine vollstaendige β–Reduktion $\rightarrow$ Vermeidung potentieller Namenskonflikte statt korrekter Aufloesung

$\rightarrow$ benutzt ausschliesslich Stapel | Stack Strukturen als Laufzeitumgebung

$\rightarrow$ folgt der i.a. rekursiven Auffaltung | Schachtelung (und dem nachfolgenden Kollabieren) von Termen

Laufzeitumgebung der SE(M)CD–machine

$\rightarrow$ Code Struktur C $\rightarrow$ stellt den auszufuehrenden Programmtext als Folge von Termen (oder Instruktionen) bereit;

$\rightarrow$ Arbeitsstack S $\rightarrow$ speichert Zwischenwerte von λ–Termen (deren Normalformen) und enthaelt das Resultat der Programmausfuehrung (Normalform des gesamtem Programm–Terms);

$\rightarrow$ Environment Stack E $\rightarrow$ nimmt Variablen | Werte Paare auf, die durch Instantiierungen von λ–gebundenen Variablen im Zuge von β–Reduktionen entstehen;

$\rightarrow$ Dump D $\rightarrow$ speichert relevante Komponenten des Maschinenzustandes vor Ausfuehrung von β–Reduktionen ab, von dem sie nach Beendigung der Auswertung der jeweiligen Abstraktionsruempfe wieder hergestellt werden;

$\rightarrow$ Zwischenstack M $\rightarrow$ dient der temporaeren Zwischenspeicherung von Konstruktorsymbolen (Applikatoren) beim Traversieren von Termen