Was bisher geschah

Turing-Maschinen $(X, Q, \Gamma, \delta, q_0, \square)$

- Simulation nichtdet. durch deterministische TM (DTM)
- Mehr-Band-TM, Simulation durch Ein-Band-TM
- DTM berechnet (partielle) Funktion $f_M : X^* \to \Gamma^*$
- Turing-berechenbare Funktionen
- Turing =
  Menge aller durch eine DTM berechenbaren (partiellen) Funktionen
- Existenz nicht Turing-berechenbarer Funktionen
  (indirekter Nachweis, Kardinalität)
- TM für Basisfunktionen:
  Konstanten, Nachfolger, Vorgänger, Test auf Bandinhalt $= 0$,
  Identität (Kopie), Projektionen
- Komposition von TM:
  Nacheinanderausführung, Einsetzen, Verzweigung

Codierungen von

- strukturierten Daten als Wörter, Zahlen (Gödelnummer)
- Problemen als Sprachen, Mengen natürlicher Zahlen

Strukturierte Programmierung

Formalisierung des imperativen Programmierens

- Programm: Folge von Befehlen
- Programmausführung: Folge von Zustandsänderungen
- Zustand: Speicherbelegung und Befehlsnummer

Eigenschaften dieses Modells:

- ist einfach in Hardware realisierbar
  (seit Jahrzehnten werden Rechner so gebaut)
- ist softwaretechnisch unzweckmäßig
  (der Beweis für die Korrektheit eines Programms sieht ganz anders aus als das Programm selbst)

Semantik: Speicher

- Speicher der Maschine besteht aus Registern (Zellen),
- Registerinhalte $\in \mathbb{N}$,
- Register numeriert durch $\mathbb{N}$,
- nur endlich viele Register verwendet (Rest bleibt 0).

Menge der möglichen Speicherbelegungen:

$$S := \left\{ s \in \mathbb{N}^{\mathbb{N}} \mid \{x \mid s(x) \neq 0\} \text{ ist endlich} \right\}$$

Beispiel: $s(0) = 42, s(1) = 10, \forall x \in \mathbb{N} : x \geqslant 2 \rightarrow s(x) = 0$

Notation für Speicher-Änderungen: $s[x := y]$
Funktion $z \mapsto (\text{if } z = x \text{ then } y \text{ else } s(z))$.
Beispiel: $s[1 := 8](1) = \ldots, s[1 := 8](2) = \ldots$

ÜA: Gilt $s[a := b][c := d] = s[c := d][a := b]$?

Syntax: Befehle und Programme

Menge B der Befehle:

- arithmetische Befehle:
  - Inc $\mathbb{N}$
  - Dec $\mathbb{N}$
- Sprungbefehle:
  - Goto $\mathbb{N}$
  - GotoZ $\mathbb{N} \times \mathbb{N}$
- Stop

Menge P der goto-Programme $= B^*$ (Folgen von Befehlen)

Beispiel für ein goto-Programm:

```
[ GotoZ 1 5, Dec 1, Inc 0, Inc 0, Goto 0, Stop ]
```

Semantik: Befehle

Menge der Konfigurationen $C \subseteq \mathbb{N} \times S$ mit

- Befehlszähler $c \in \mathbb{N}$
  (enthält die Nr. des nächsten auszuführenden Befehls)
- Speicherbelegung $s \in S$

Übergangsrelation des goto-Programms p:
$\text{step}_p \subseteq C \times C$ mit $((c, s), (c', s')) \in \text{step}_p$,
falls $c < |p|$ und ... und $p_c =$

- Inc i, dann $c' = c + 1, s' = s[i := s(i) + 1]$
  $(\text{step}_{\text{Inc i}} = \{((c, s), (c + 1, s[i := s(i) + 1])) \mid c \in \mathbb{N}, s \in S, i \in \mathbb{N}\})$
- Dec i, dann ...
- Goto k, dann ...
- GotoZ i k, dann: wenn $s(i) = 0$, dann ... sonst ...

Satz: Die Relation step_p ist eine partielle Funktion. (ÜA)

Semantik: Programme

- initiale Konfiguration $I(x)$ mit Eingabe $x = (x_1, \ldots, x_n) \in \mathbb{N}^n$: $(0, s)$ mit $s(1) = x_1, \ldots, s(n) = x_n, s(i) = 0$ sonst
- finale Konfiguration (c, s), wobei $c < |p|$ und $p_c = \text{Stop}$
- die Ausgabe $O(c, s)$ einer Konfiguration ist $s(0)$

Jedes goto-Programm p berechnet eine partielle Funktion
$f_p : \mathbb{N}^n \to \mathbb{N}$ mit
$\forall (x, y) \in \mathbb{N}^n \times \mathbb{N}$:
$(x, y) \in f_p$ gdw. $(I(x), F) \in \text{step}_p^*$ mit F final und $y = O(F)$

Beispiel: $p =$

```
[GotoZ 1 4, Dec 1, Goto 0, Inc 1, Inc 3, Inc 0, Stop]
```

berechnet $f_p : \mathbb{N}^2 \to \mathbb{N}$ mit $f_p(2, 1) = \ldots$

Goto-berechenbare Funktionen

Partielle Funktion $f : \mathbb{N}^n \to \mathbb{N}$ heißt goto-berechenbar gdw.
ein goto-Programm p mit $f = f_p$ existiert.

GOTO = Menge aller goto-berechenbaren Funktionen

Beispiel: Programm p, welches die Funktion $f_p(x_1) = 3$ berechnet?

Semantik des leeren Programms (mit $|p| = 0$) ?

goto-Programm für $x \mapsto 2x$

```
p = [ GotoZ 1 5 , Dec 1 , Inc 0 , Inc 0 , Goto 0 , Stop ]
```

Nachweis für $\forall x \in \mathbb{N} : f_P(x) = 2x$ in zwei Teilen

- das goto-Programm hält für jede Eingabe (vgl. Def. step_p^*)
- die Ausgabe ist korrekt

jeder Konfiguration (c, s) wird zugeordnet:

Invariante $s(0) + 2 \cdot s(1)$

Schranke $s(1)$

und gezeigt (für die Teilfolge aller Konfigurationen mit $c = 0$):

- die Invariante ist
  1. anfangs wahr,
  2. invariant
- die Schranke nimmt ab (um wieviel?) und bleibt $\geqslant 0$.

Elementare goto-berechenbare Funktionen

goto-berechenbar sind z.B. die folgenden Funktionen:

- Konstante 0
- jede konstante Funktion
- identische Funktion
- jede Projektion $(x_1, \ldots, x_n) \mapsto x_k$
- Addition
- schwache Subtraktion $(x_1, x_2) \mapsto x_1 \dot{-} x_2 (= \max(0, x_1 - x_2))$

Abschluss-Eigenschaften

Nacheinanderausführung: Sind $f : \mathbb{N} \to \mathbb{N}$ und $g : \mathbb{N} \to \mathbb{N}$ goto-berechenbar,
dann ist auch $x \mapsto f(g(x))$ goto-berechenbar.
Beweis:
[Prog. für g ; $s(1) := s(0); s(0) := 0$; Prog. für f]
Warum $s(0) := 0$?

Einsetzen: Sind $f : \mathbb{N}^2 \to \mathbb{N}, g_1, g_2 : \mathbb{N} \to \mathbb{N}$ goto-berechenbar,
dann auch $x \mapsto f(g_1(x), g_2(x))$.
einfach [Programm für g_1, Programm für g_2, ...] ?

GOTO $\subseteq$ Turing

Satz: $\forall f : \mathbb{N}^k \to \mathbb{N}$: f goto-berechenbar $\to$ f Turing-berechenbar.

Beweis (Idee):
Ausführung des goto-Programmes p zur Berechnung von $f = f_p$ auf einer Mehrband-TM M

- in p werden k Register verwendet – M hat k Arbeits-Bänder
- Register i mit Inhalt x – Arbeits-Band i hat Inhalt 1^x
- Ausführung von `Inc i`, `Dec i` durch M auf Band i (Unterprogramme)
- Programmablaufsteuerung:
  Befehlsnummer (PC) wird im Zustand der TM verwaltet
- für jede Programmzeile eine Zustandsmenge zur Ausführung des entsprechenden Unterprogrammes der TM
- Realisierung der Sprungbefehle durch Zustandsübergänge

ÜA: Ergänzung der Details zur Herstellung der Initialkonfiguration, Ablesen des Resultates aus Finalkonfiguration

Turing $\subseteq$ GOTO

Satz: $\forall f : \mathbb{N}^k \to \mathbb{N}$: f TM-berechenbar $\to$ f goto-berechenbar.
Beweis (Idee):

- Jedes Band der k-Band-TM wird simuliert
  durch zwei Register $(l, r) \in \mathbb{N}^2$
- Jede Zahl $n \in \mathbb{N}$ ist Wort zur Basis $b = 1 + |\Gamma|$ (oder $= |\Gamma|$.).
- l enthält den Bandinhalt links vom Kopf,
  r enthält den Bandinhalt rechts vom Kopf (gespiegelt).
  Kopfposition = erste Stelle in r^R = letzte Stelle in r)
  Beispiel: Für $\underbrace{1010}_{l}\, p\, \underbrace{011}_{r^R}$ (mit $b = 2$) ist $l = 10, r = 6$
- Zeichen am Kopf lesen: $r \mod b$, Beispiel: $r \mod b = 0$
- Zeichen x schreiben und Kopf nach rechts bewegen $((a, p, x, q, R))$:
  $l' := b \cdot l + x$, $r' = \lfloor r/b \rfloor$
  Beispiel: $(0, p, 1, q, R)$ $1010p011 \vdash 10101q11$ mit
  $l' := b \cdot l + x = 2 \cdot 10 + 1 = 21$ (10101) , $r' = \lfloor r/b \rfloor = \lfloor 6/2 \rfloor = 3$ (11)
  analog für Übergänge mit Kopfbewegung L, N (ÜA)
- diese Rechnungen werden in einem zusätzlichen Register ausgeführt
  insgesamt: k Bänder $\mapsto$ $2k + 1$ Register

Zusammenfassung GOTO

- formalisiert maschinennahe imperative Programmierung (Programmausführung als Folge von Speicherzustandsänderungen)
- der Befehlssatz ist klein, die Ausdruckskraft scheint gering, aber
- einfache Basis-Funktionen sind in GOTO (Konstanten, Verschieben von Registerinhalten, arithmetische Operationen)
- GOTO ist abgeschlossen bzgl.
  - Nacheinanderausführung,
  - Einsetzen
- GOTO = Turing (beide Inklusionen, jeweils Beweisidee)