Formale Modellierung
Vorlesung 9 vom 15.06.15: Berechnungsmodelle

Christoph Lüth

Universität Bremen

Sommersemester 2015

Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit rekursiven Definitionen
  - Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - Berechungsmodelle (Models of Computation)
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

Fahrplan für heute

- Eine Behauptung von Church
- Registermaschinen
- Rekursiv definierbare Funktionen
- Universelle Berechenbarkeit

Die Churchsche Behauptung

Church's Thesis (1936)

Intuitiv berechenbare Funktionen sind genau die durch Rekursion definierbaren.

- Nicht beweisbar: das Konzept "intuitiv berechenbar" ist nicht formalisierbar.
- Aber: es gibt sehr viele, unterschiedliche (und meist voneinandar unabhängig gefundene) formale Definition von Berechenbarkeit, die alle gleich mächtig sind.
- Es spricht alles für die Churchsche Behauptung.

Registermaschinen

- Variation von Turing-Maschinen, konzeptionell etwas einfacher.

Registermaschinen (Minsky)

- Sequenz von Registern $R_1, R_2, \ldots$, die natürliche Zahlen enthalten
- Programm besteht aus:
  - Endliche Anzahl von Zuständen $S_0, S_1, S_2, \ldots$, und
  - für $i > 0$, Anweisung (S_0: terminaler Zustand)

- Anweisungen:

$(j, +, k)$

$$S_i \xrightarrow{R_j + 1} S_k$$

Erhöhe R_j um 1 und gehe zu S_k

$(j, -, k, l)$

$$\begin{tikzcd} S_i \arrow[r, "R_j - 1"] \arrow[dr, dashed] & S_k \\ & S_l \end{tikzcd}$$

Wenn $R_j = 0$, gehe zu S_l; ansonsten verringere R_j um 1 und gehe zu S_k.

Beispielprogramme

Programm 1:

1: $(1, -, 1, 2)$
2: $(2, -, 3, 0)$
3: $(1, +, 2)$

$R_1' := R_2;$
$R_2' := 0$

Programm 2:

1: $(3, -, 1, 2)$
2: $(2, -, 3, 6)$
3: $(3, +, 4)$
4: $(1, +, 5)$
5: $(1, +, 2)$
6: $(3, -, 7, 0)$
7: $(2, +, 6)$

$R_3' := 0;$
$R_1' := R_1 + 2 * R_2;$
$R_2' := R_2$

Berechenbarkeit durch Registermaschine

- NB: Jede Registermaschine hat endlich viele Register, aber Gesamtanzahl unbeschränkt.
- Eine Registermaschine R berechnet eine Funktion $f : \mathbb{N}^k \to \mathbb{N}$ (mit einem Programm P), wenn beim Start (von S_1) mit Registerinhalt $(n_1, n_2, \ldots, n_k, 0, \ldots)$ die Maschine schließlich mit Registerinhalt $f(n_1, \ldots, n_k)$ in R_1 anhält.
- Aber welche Funktionen sind berechenbar?

Theorem 1: Berechenbare Funktionen

(i) Die Projektionen $\pi_i(n_1, \ldots, n_k) = n_i$ ist berechenbar.

(ii) $const_0 : \mathbb{N} \to \mathbb{N}$ und $succ : \mathbb{N} \to \mathbb{N}$ sind berechenbar.

(iii) Komposition: Sei f mit Arität k berechenbar, und $g_1, \ldots, g_k$ mit Arität l berechenbar, dann ist h berechenbar:

$$h \stackrel{def}{=} f(g_1(n_1, \ldots, n_l), \ldots, g_k(n_1, \ldots, n_l))$$

(iv) Rekursion: Seien f und g berechenbar mit Arität k bzw. $k+2$, dann ist h berechenbar:

$$h(n_1, \ldots, n_k, 0) \stackrel{def}{=} f(n_1, \ldots, n_k)$$
$$h(n_1, \ldots, n_k, n_{k+1} + 1) \stackrel{def}{=} g(n_1, \ldots, n_k, n_{k+1}, h(n_1, \ldots, n_k, n_{k+1}))$$

(v) Minimalisierung (μ-Operator): Sei f berechenbar mit Arität $k+1$, dann ist g berechenbar:

$$g(n_1, \ldots, n_k) = n \text{ mit } \begin{array}{l} f(n_1, \ldots, n_k, n) = 0 \text{ und} \\ \forall m.\, m < n \longrightarrow f(n_1, \ldots, n_k, m) > 0 \end{array}$$

Berechenbarkeit, Rekursion, Primitive Rekursion

- Primitiv rekursive Funktionen: kleinste unter (i) – (iv) abgeschlosene Klasse von Funktionen
- Rekursive Funktionen: kleinste unter (i) – (v) abgeschlossene Klasse von Funktionen
- Primitiv rekursive Funktionen sind total, aber nicht jede totale rekursive Funktion ist primitiv rekursiv. (Gegenbeispiel: Ackermann)
- Es gilt der Umkehrschluss:

Theorem 2
Jede rekursive Funktion ist berechenbar.

Kodierung von Programmen

- Programme können durch Zahlen kodiert werden (Gödel-Kodierung):

$$(j, +, k) \mapsto 2^j \cdot 5^k$$
$$(j, -, k, l) \mapsto 2^j \cdot 3 \cdot 5^k \cdot 7^l$$
$$i_1, \ldots, i_n \mapsto 2^{i_1} \cdot 3^{i_2} \cdot \ldots p_{n-1}^{i_n}$$

- Notation: $f_{n,k}$ ist die durch n kodierte Funktion der Arität k.

Theorem 3 (Universelle Funktionen)
Es gibt universelle rekursive Funktion u so dass
- *$u(n, k, m) = r$ wenn n ein Programm kodiert, k ein k-Tupel $(m_1, \ldots, m_k)$ und $f_{n,k}(m_1, \ldots, m_k) = r$,*
- *und ansonsten $u(n, k, m)$ undefiniert.*

Rekursive Funktionen und Peano-Arithmetik

Theorem 4 (Rekursion ist PA)
Jede rekursive Funktion ist in PA definierbar.

- Ohne Multiplikation funktioniert der Beweis nicht.
- Deshalb: nicht jede rekursive Funktion in Presburger-Arithmetik definierbar.

Weitere Modelle von Berechenbarkeit

- Turing-Maschinen (remember those?)
- Chomsky-Grammatiken (Typ 0)
- λ-Kalkül
- Kombinatorlogik
- π-Kalkül
- Neuronale Netze
- Gegenbeispiel: Aktoren (Carl Hewitt)
  - Literaturlage nicht eindeutig ...

Zusammenfassung

- Behauptung von Church: intuitiv berechenbare Funktionen sind genau durch Turing-Maschinen berechenbare.
- Vielzahl von formalen Berechnungsmodellen, welche die gleiche Mächtigkeit haben.
- Wir haben gesehen:
  - Registermaschinen
  - Rekursive Funktionen
  - In Peano-Arithmetik definierbare Funktionen
- Beweis der Mächtigkeit durch Kodierung:
  - Berechnungen können in Zahlen kodiert werden
  - Damit universelle Berechnungen (Compiler!) möglich
- Nächste Woche: Grenzen der Beweisbarkeit — die Gödelschen Unvollständigkeitssätze