Unentscheidbarkeit des Gültigkeitsproblems

Wir gehen in zwei Schritten vor:

- Das Terminierungsproblem ist unentscheidbar.
  Es gibt kein Programm, welches als Eingabe ein Programm P und eine Belegung β der Variablen von P akzeptiert und entscheidet, ob P mit β als Anfangsbelegung terminiert.
- Wenn das Gültigkeitsproblem entscheidbar wäre, dann wäre auch das Terminierungsproblem entscheidbar.

Kodierungen

Fakt: Sowohl Programme als auch Anfangsbelegungen können als Integer kodiert werden.

Wir beschränken uns auf Programme, deren Eingabe aus einem Tupel von Integern besteht. Wir nehmen an, dass Variablen $x_1, \ldots, x_n$ mit dieser Eingabe initialisiert werden.
Einige Notationen:

- $P(a_1, \ldots, a_i)$ bezeichnet das Programm P mit der Anfangsbelegung $(a_1, \ldots, a_i, 0, \ldots, 0)$.
  D.h., die Variablen $x_1, \ldots, x_i$ bekommen Anfangswerte $a_1, \ldots, a_i$ und die Variablen $x_{i+1}, \ldots, x_n$ den Anfangswert 0.
- Π_n bezeichet das Programm mit dem Code n (wenn es ein solches Programm gibt).

Berechenbare Kodierungen

Fakt: Es gibt berechenbare Kodierungen, d.h., Kodierungen, für die die folgenden zwei Programme existieren:

- Der Kodierer.
  Eingabe: ein Programm P.
  Ausgabe: der Code von P, d.h., die Zahl n mit $P = \Pi_n$.
- Der Dekodierer.
  Eingabe: eine Zahl n.
  Ausgabe: das Programm Π_n falls n ein Programm kodiert, sonst 'KP' (Kein Programm).

Annahme: Es gibt ein Programm T, so dass für jedes Paar $n, m \in \mathbb{N}$ das initialisierte Programm $T(n, m)$ terminiert und zwar mit

KP	falls n kein Programm kodiert
JA	falls n ein Programm kodiert und $\Pi_n(m)$ terminiert
NEIN	falls n ein Programm kodiert und $\Pi_n(m)$ nicht terminiert

Wir zeigen, dass diese Annahme zu einem Widerspruch führt.

Der Widerspruch

Fakt: Aus der Annahme folgt, dass es ein Programm T' gibt, so dass für jedes $n \in \mathbb{N}$

$$\begin{array}{ll} T'(n) \text{ terminiert} & \text{falls } n \text{ ein Programm kodiert und} \\ & \Pi_n(n) \text{ nicht terminiert} \\ T'(n) \text{ terminiert nicht} & \text{falls } n \text{ kein Programm kodiert oder} \\ & \Pi_n(n) \text{ terminiert} \end{array}$$

Sei k der Code des Programms T', d.h. $\Pi_k = T'$. Das initialisierte Programm $T'(k)$ terminiert oder terminiert nicht. Wir haben jedoch:

$$\begin{array}{lll} & T'(k) \text{ terminiert} & \\ \Longrightarrow & k \text{ kodiert ein Programm und} & \\ & \Pi_k(k) \text{ terminiert nicht} & (\text{Def. von } T') \\ \Longrightarrow & T'(k) \text{ terminiert nicht} & (\Pi_k = T') \end{array}$$

$$\begin{array}{lll} & T'(k) \text{ terminiert nicht} & \\ \Longrightarrow & \Pi_k(k) \text{ terminiert} & (\text{Def. von } T'\text{, denn } k \text{ ist Code}) \\ \Longrightarrow & T'(k) \text{ terminiert} & (\Pi_k = T') \end{array}$$

Damit ist die Annahme falsch.

Unentscheidbarkeit des Gültigkeitsproblems

Wir ordnen jedem Programm P und Variablenbelegung β eine Formel $\phi_{P,\beta}$ der Prädikatenlogik zu mit

$\phi_{P,\beta}$ ist gültig

genau dann, wenn

das Programm P mit der Anfangsbelegung β terminiert

Die Formel $\phi_{P,\beta}$ kann von einem Programm konstruiert werden.

Daraus folgt, dass kein Programm das Gültigkeitsproblem der Prädikatenlogik lösen kann!

if-goto-Programme

$$\begin{array}{lll} \mathit{Prog} & ::= & \ell : \mathit{Zuw} & \text{(Zuweisung)} \\ & & \ell : \ \mathbf{goto}\ \ell' & \text{(unbedingter Sprung)} \\ & & \ell : \mathbf{if}\ x_i \neq 0\ \mathbf{then\ goto}\ \ell' & \text{(bedingter Sprung)} \\ & & \ell : \mathbf{halt} & \text{(Terminierung)} \\ & & \mathit{Prog}\ ;\ \mathit{Prog} & \text{(Hintereinanderausführung)} \end{array}$$

$$\begin{array}{lll} \mathit{Zuw} & ::= & x_i := 0 \ \mid \ x_i := x_j \\ & & x_i := x_j + 1 \ \mid \ x_i := x_j - 1 \\ \ell & ::= & 1 \mid 2 \mid 3 \mid \ldots \end{array}$$

Beispiel

1: **if** $x_1 = 0$ **then goto** 4;

2: $x_1 := x_1 - 1$;

3: **goto** 1;

4: **halt**

Behauptung: **if-goto**-Programme können alle anderen Programme simulieren.

Konsequenz: Es gibt kein Programm, egal in welcher Sprache, für die Terminierung von **if-goto**-Programmen

Notationen und Definitionen

Mit k bezeichnen wir die Anzahl der Anweisungen von P
(Die letzte Anweisung ist immer **halt**)

Mit n bezeichnen wir die Anzahl der Variablen von P
(D.h. die Variablen von P sind $x_1, \ldots, x_n$)

Eine Konfiguration von P ist eine Tupel $(Z, m_1, \ldots, m_n) \in \mathbb{N}^{n+1}$.
Z bezeichnet die aktuelle Anweisung und $m_1, \ldots m_n$ die aktuelle Belegung der Variablen

Konvention: die Nachfolgekonfiguration einer Konfiguration der Gestalt $(\mathbf{k}, m_1, \ldots, m_n)$ ist wieder $(\mathbf{k}, m_1, \ldots, m_n)$

Symbole der Formel $\phi_{P,\beta}$

- R, Prädikatensymbol, $(n+2)$-stellig;
- $<$, Prädikatensymbol, 2-stellig;
- f, Funktionssymbol, 1-stellig;
- $\mathbf{0}$, Konstante.

Eine Interpretation $\mathcal{A}$

- Universum: $\mathbb{N}$
- $<^{\mathcal{A}}$ ist die gewöhnliche Ordnung auf $\mathbb{N}$
- $\mathbf{0}^{\mathcal{A}} = 0$
- $f^{\mathcal{A}}$ ist die Nachfolgerfunktion, i.e., $f^{\mathcal{A}}(n) = n + 1$
- $R^{\mathcal{A}}(s, Z, m_1, \ldots, m_n) = 1$ wenn $(Z, m_1, \ldots, m_n)$ die Konfiguration von P nach s Schritten ist (für die Anfangsbelegung β).

Die Hilfsformel $\psi_{P,\beta}$

$$\psi_{P,\beta} = \quad \psi_0 \quad \wedge \quad R(\mathbf{0}, f(\mathbf{0}), \overline{\beta}) \quad \wedge \quad \psi_1 \quad \wedge \quad \ldots \quad \wedge \quad \psi_{k-1}$$

$\overline{\beta}$ bezeichnet die Tupel von Termen, die unter der Interpretation $\mathcal{A}$ die Anfangsbelegung β darstellen. Z.B. wenn P zwei Variablen hat und $\beta = (2,1)$ dann gilt $\overline{\beta} = (f(f(\mathbf{0})), f(\mathbf{0}))$

Unter der Interpretation $\mathcal{A}$ besagt $R(\mathbf{0}, f(\mathbf{0}), \beta)$, dass $(1, \beta)$ die Anfangskonfiguration des Programms P ist

Unter der Interpretation $\mathcal{A}$ beschreibt ψ_i mit $\mathbf{i} \in \{1\ldots, k-1\}$ die Wirkungsweise der $\mathbf{i}$-ten Zeile von P. Zum Beispiel:

- Wenn die $\mathbf{i}$-te Zeile der Gestalt $\mathbf{i}$: $x_j := x_j + 1$ ist, dann

$$\begin{aligned} \psi_i \quad = \quad & \forall x \forall y_1 \ldots \forall y_n \ (\\ & R(x, \mathbf{i}, y_1, \ldots y_n) \quad \rightarrow \\ & R(f(x), f(\mathbf{i}), y_1, \ldots y_{j-1}, f(y_j), y_{j+1}, \ldots, y_n) \\ &) \end{aligned}$$

- Wenn die $\mathbf{i}$-te Zeile der Gestalt $\mathbf{i}$: $\mathbf{if}\ x_j = 0\ \mathbf{then\ goto\ j}$ ist, dann

$$\begin{aligned} \psi_i \quad = \quad & \forall x \forall y_1 \ldots \forall y_n \ (\\ & R(x, \mathbf{i}, y_1, \ldots y_n) \rightarrow \\ & \quad (\quad y_j = \mathbf{0} \quad \wedge \quad R(f(x), \mathbf{j}, y_1, \ldots, y_n) \\ & \qquad\qquad \vee \\ & \qquad \neg(y_j = \mathbf{0}) \quad \wedge \quad R(f(x), f(\mathbf{i}), y_1, \ldots, y_n) \\ & \quad) \\ &) \end{aligned}$$

ψ_0 garantiert, dass unter allen Interpretationen $<$ eine Ordnung ist mit $\mathbf{0}$ als kleinstem Element, dass stets $x \leq f(x)$ gilt, und dass $f(x)$ der unmittelbare $\leq$-Nachfolger von x ist

$$\begin{aligned} \psi_0 \quad = \quad & \forall x \forall y (x < y \rightarrow \neg(y < x)) \quad \wedge \\ & \forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z) \quad \wedge \\ & \forall x (\mathbf{0} < x \vee \mathbf{0} = x) \quad \wedge \\ & \forall x (x < f(x)) \quad \wedge \\ & \forall x \forall z (x < z \rightarrow (f(x) < z \vee f(x) = z) \end{aligned}$$

Die Formel $\phi_{P,\beta}$

Wir setzen

$$\phi_{P,\beta} \quad = \quad \psi_{P,\beta} \quad \to \quad \exists x \exists y_1 \ldots \exists y_n \; R(x, \mathbf{k}, y_1, \ldots, y_n)$$

Wir haben

$\phi_{P,\beta}$ ist gültig

genau dann, wenn

das Programm P mit der Anfangsbelegung β terminiert