Aufgaben zum Tut am 05.12.2005

Thomas Pajor

4. Dezember 2005

Aufgabe 1)

Das *Halteproblem* $\mathcal{H}$ sei wie folgt definiert:

$$\mathcal{H} := \{wv \mid \mathcal{T}_w \text{ hält auf Eingabe } v\}$$

Zeigen Sie dass das Halteproblem nicht entscheidbar ist.

Hinweis: Es reicht zu zeigen, dass das *spezielle Halteproblem*

$$\mathcal{H}_{\text{spez}} := \{w \mid \mathcal{T}_w \text{ hält auf Eingabe } w\}$$

nicht entscheidbar ist.

Lösung

In der Vorlesung wurde der Beweis (hoffentlich) durch Reduktion auf die Diagonalsprache durchgeführt. Wir wissen aber, dass es auch andere gleichmächtige Maschinenmodelle gibt, und führen den Widerspruch „algorithmisch“ durch Pseudocode. Dabei betrachten wir das spezielle Halteproblem.

Sei das Halteproblem $\mathcal{H}_{\text{spez}}$ entscheidbar, dann existieren folgende beiden Funktionen in Pseudocode:

```
function HalteTest(Programm, Eingabe) {
    falls Programm(Eingabe) terminiert
        return JA;
    sonst
        return NEIN;
}
```

```
function Test(Programm) {
    while (HalteTest(Programm, Programm) == JA) {
       // Terminiere nie, falls Programm(Eingabe)
       // terminiert
    }
    // Terminiere hier falls Programm(Eingabe)
    // nicht terminiert
}
```

Wir machen mit den obigen Funktionen nun folgendes Szenario: `HalteTest(Test, Test)`. Wir prüfen also mit Hilfe von `HalteTest` ob `Test(Test)` terminiert. Es gibt nun folgende Möglichkeiten

- `HalteTest(Test,Test)` liefert JA. Dann ist laut dem Code von `Test` die `while`-Schleife abgebrochen, was aber heißt, dass `Test(Test)` nicht terminiert. Dies ist ein Widerspruch.
- `HalteTest(Test,Test)` liefert NEIN. Dann bricht laut dem Code von `Test` die `wihle`-Schleife nie ab. Das heißt aber dass `Test(Test)` terminiert, was wiederum ein Widerspruch ist.

$\Rightarrow$ Das (spezielle) Halteproblem ist nicht entscheidbar. $\square$

Aufgabe 2)

Zeigen Sie dass die Sprache

$$L_{\text{äquiv}} := \{u\#v \mid \mathcal{L}(\mathcal{T}_u) = \mathcal{L}(\mathcal{T}_v)\}$$

nicht entscheidbar ist.

Lösung

Wir führen einen Widerspruchsbeweis. Sei also $L_{\text{äquiv}}$ entscheidbar. Dann gibt es eine Turingmaschine $\mathcal{T}$ die $L_{\text{äquiv}}$ entscheidet. Das heißt sie entscheidet für alle $w \in \Sigma^*$ ob $\chi_u(w) = \chi_v(w)$, wobei χ_u und χ_v die charakteristischen Funktionen von $\mathcal{T}_u$ und $\mathcal{T}_v$ sind. Das heißt χ_u und χ_v sind total berechenbar.

Betrachte nun $w := u$. Da χ_u berechenbar ist, kann $\mathcal{T}$ also insbesondere entscheiden ob $\mathcal{T}_u$ mit Eingabe u hält. Dies ist ein Widerspruch zur Unentscheidbarkeit des speziellen Halteproblems.

$\Rightarrow L_{\text{äquiv}}$ ist nicht entscheidbar. $\square$

Aufgabe 3)

Eine Turingmaschine heißt *platzbeschränkt*, falls der Schreib/Lesekopf niemals den Bereich der Eingabe verlässt.

Geben Sie einen Algorithmus (in Pseudo–Code) an, der das Halteproblem auf Eingaben $\langle \mathcal{T} \rangle w$ korrekt berechnet, wobei $w \in \{0,1\}^*$ und $\langle \mathcal{T} \rangle$ Gödelnummer einer platzbeschränkten Turingmaschine ist.

Lösung

Für jede Turingmaschine $\mathcal{T}$ gilt: $|Q| < \infty$, $|\delta| < \infty$ $|\Sigma| < \infty$ und $|\Gamma| < \infty$. Da $\mathcal{T}$ zusätzlich platzbeschränkt ist, gibt es jedoch bloß eine endliche Anzahl an möglichen Konfigurationen. Die TM terminiere o.B.d.A. immer in einem Endzustand aus F. Der folgende Algorithmus leistet dann das Gewünschte:

Algorithm 1 HALTETEST($\mathcal{T}, w$)

$K := \emptyset$;
$q := s$;
while $q \notin F$ **do**
    if $x(q)ay \in K$ **then**
        Terminiere(„$\mathcal{T}$ hält nicht auf w“);
    end if
    $K := K \cup \{x(q)ay\}$;
    $q := \delta(q, a)$;
end while
Terminiere(„$\mathcal{T}$ hält auf w“);

Ist eine Konfiguration k schon einmal dran gewesen, und kommt sie ein zweites Mal vor, so befindet sich $\mathcal{T}$ offenbar in einer Endloschleife und terminiert nie.