Aufgaben zum Tut am 30.01.2007

Thomas Pajor

30. Januar 2007

Aufgabe 1.

Beweisen Sie: Es kann keinen (Sprach–)Compiler $\mathcal{C}$ geben, der bezüglich der erzeugten Maschinencodegröße optimal ist, also immer den kürzest möglichen Code erzeugt.

Lösung.

Wir führen Beweis durch Widerspruch. Angenommen, es gibt einen Compiler $\mathcal{C}$, der bezüglich der erzeugten Codelänge optimal ist. Wir zeigen, dass dann auch das Halteproblem (auf leerer Eingabe) entscheidbar wäre.

Sei P ein Programm, von dem wir testen möchten ob P terminiert. Wir können o.B.d.A. annehmen, dass P keine Eingabe liest, andernfalls modifizieren wir P so, dass im ersten Schritt von P die Eingabe verworfen wird. Wir werden nun P mit $\mathcal{C}$ kompilieren, und das Kompilat $\mathcal{C}(P)$ untersuchen. Terminiert das Programm P nicht, also enthält es eine Endlosschleife, und der kürzeste Code hat offenbar die Form „`A: GOTO A`“. Damit können wir entscheiden, nämlich durch Prüfung ob $\mathcal{C}(P) =$ `A: GOTO A`, ob P mit leerer Eingabe terminiert. Dies ist ein Widerpsruch zur Unentscheidbarkeit des Halteproblems.

Die Behauptung ist somit nicht haltbar, und es kann keinen optimalen Compiler $\mathcal{C}$ geben. □

Aufgabe 2.

Zum Problem SAT aus der Vorlesung, sei das Problem 3SAT wie folgt definiert:

Sei U eine Menge von aussagenlogischen Variablen und C eine Menge von Klauseln über U, wobei jede Klausel aus C genau die Länge 3 habe. Eine Klausel aus C ist ein Ausdruck der

Form

$$y_1 \vee y_2 \vee y_3 \text{ mit } y_i \in \{u_1, \ldots, u_m\} \cup \{\neg u_1, \ldots, \neg u_m\} \cup \{\texttt{true}, \texttt{false}\}$$

Gesucht: Eine Belegung der Variablen $u_i \in U$ mit `true` oder `false`, so dass alle Klauseln erfüllt werden.

Zeigen Sie: 3SAT $\in \mathcal{NP}$.

Lösung.

Was müssen wir zeigen? Wenn wir eine nichtdeterministische Turingmaschine $\mathcal{T}$ angeben können, die zu jeder Probleminstanz w von 3SAT eine erfüllende Belegung in $\mathrm{ntime}(w) \leq p(n)$ für ein Polynom p berechnen kann, so haben wir die Aussage bewiesen.

Betrachte eine Orakel–Turingmaschine $\mathcal{T}$. Wir wissen, dass diese (vom Zeitaufwand) äquivalent zu einer NTM ist. Benutze das Orakel, um in der Orakelphase eine Lösung zu „raten“. Der deterministische Teil der Turingmaschine muss jetzt nurnoch in polynomieller Zeit überprüfen ob diese Lösung auch korrekt ist.

Folgender Algorithmus in Pseudo–Code realisiert dies (o.B.d.A. soll das Orakel nur strukturell gültige Lösungen raten):

Algorithmus 1 : TESTE–3SAT

Eingabe : Eine vom Orakel geratene Variablenbelegung

```
1 für alle c ∈ C tue
2     Evaluiere c mit gegebener Variablenbelegung
3     wenn Evaluation fehlgeschlagen dann
4         Lehne Orakelvorschlag ab
5 Akzeptiere Orakelvorschlag
```

Es folgt also 3SAT $\in \mathcal{NP}$. □

Aufgabe 3.

Eine Relation $\rho \subseteq M \times M$ heißt *transitiv*, falls für alle $a, b, c \in M$ gilt:

$$a\rho b \wedge b\rho c \Rightarrow a\rho c$$

Sei $\leq_p$ die Relation „polynomiell reduzierbar“ wie in der Vorlesung definiert.

Zeigen Sie: $\leq_p$ ist transitiv.

Lösung

Seien A, B, C beliebige Sprachen, und es gelte $A \leq_p B$ und $B \leq_p C$. Nach Definition von $\leq_p$ existieren somit zwei berechenbare Funktionen f und g, wobei:

$$\forall w \in \Sigma^* : w \in A \Leftrightarrow f(w) \in B \qquad \text{und} \qquad \forall w \in \Sigma^* : w \in B \Leftrightarrow g(w) \in C$$

Weiterhin existieren zwei Polynome p und q, so dass gilt:

$$f \in \mathrm{TIME}(p(n)) \qquad \text{und} \qquad g \in \mathrm{TIME}(q(n))$$

Betrachte nun die Abbildung $\Phi := g \circ f$. Nach Definition von f und g gilt für Φ:

$$\forall w \in \Sigma^* : w \in A \Leftrightarrow f(w) \in B \Leftrightarrow \underbrace{g(f(w))}_{=\Phi(w)} \in C$$

Also insbesondere $\forall w \in \Sigma^* : w \in A \Leftrightarrow \Phi(w) \in C$.

Da die Hintereinanderausführung der Polynome p und q selbst auch wieder ein Polynom ist, gibt es ein Polynom r, so dass

$$\Phi \in \mathrm{TIME}(r(n))$$

Also ist $A \leq_p C$ und somit $\leq_p$ transitiv. $\square$