Entscheidbarkeit der Presburger-Arithmetik

Freie Universität Berlin
Proseminar Theoretische Informatik
Tim Pauli

20.01.2015

1 Motivation

Das Entscheiden, ob Aussagen wahr oder falsch sind, ist eine Kernaufgabe der Mathematik und Informatik. Das Automatisieren dieses Prozesses ist in vielerlei Hinsicht interessant. Sei es zur Verifikation von Beweisen oder um die Grenzen eben dieser Automatisierung aufzuzeigen. Es ist bewiesen, dass es nicht für alle Mengen von Aussagen funktioniert. Im Folgenden soll eine Menge von Aussagen vorgestellt werden, für die es noch möglich ist.

2 Problemdefinition

2.1 Syntax

Definition 1. *$\wedge$, $\vee$ und $\neg$ nennen wir **boolsche Operationssymbole**. $"($ und $")"$ nennen wir **Klammersymbole**. $\forall$ und $\exists$ nennen wir **Quantorensymbole**. x benutzen wir um **Variablen** auszudrücken. $R_1, \ldots, R_k$ nennen wir **Relationssymbole**.*

Definition 2. *Wir nennen $R_i(x_1, \ldots, x_j)$ eine **atomare Formel**. Den Wert j nennen wir **Stelligkeit**. Alle Vorkommen der gleichen Relation in einer Formel sollen die gleiche Stelligkeit besitzen.*

Definition 3. *Über dem Alphabet $\{\wedge, \vee, \neg, (,), \forall, x, \exists, R_1, \ldots, R_k\}$ nennen wir eine Zeichenkette Φ **Formel**, wenn*

1. *Φ eine atomare Formel ist.*
2. *Φ die Form $\Phi_1 \wedge \Phi_2$ oder $\Phi_1 \vee \Phi_2$ oder $\neg\Phi_1$ hat, wo Φ_1 und Φ_2 kleinere Formeln sind.*
3. *Φ die Form $\forall x_i(\Phi_1)$ oder $\exists x_i(\Phi_1)$ hat, wo Φ_1 eine kleinere Formel ist.*

Definition 4. *Wir nennen den Bereich zwischen zwei zusammengehörigen Klammern, welche einer quantifizierten Variable folgen, **Geltungsbereich**. Eine Variable, die nicht gebunden ist, im Geltungsbereich eines Quantoren nennen wir*

***freie Variable**. Eine Formel mit keinen freien Variablen nennen wir **Satz**.*

Definition 5. *Eine Formel liegt in **Pränexform** vor, wenn sich alle Quantorensymbole am Anfang der Formel befinden.*

Wir nehmen an, dass alle Formeln, die wir betrachten, in Pränexform vorliegen.

2.2 Semantik

Die boolschen Operationssymbole, die Klammersymbole und die Quantorensymbole haben ihre in der Mathematik üblichen Bedeutungen.

Definition 6. *Wir nennen die Menge, aus welcher die Variablen Werte annehmen können, **Universum**.*

Definition 7. ***Relation** nennen wir eine Funktion, die j-Tupel auf {TRUE, FALSE} abbildet.*

Definition 8. *Wir nennen das Tupel $(U, P_1, \ldots, P_k)$, wo U ein Universum ist und $P_1, \ldots, P_k$ die Relationen sind, welche den Symbolen $R_1, \ldots, R_k$ zugewiesen sind, **Modell**. Die Stelligkeit eines Relationssymbols soll dem der zugewiesenen Relation entsprechen. **Sprache eines Modells** nennen wir die Menge von Formeln, die nur die vom Modell zugewiesen Relations-Symbole nutzen. Der Satz Φ ist im Modell M entweder wahr oder falsch. Wenn der Satz Φ im Modell M wahr ist, nennen wir M **ein Modell von** Φ.*

Definition 9. *Wir nennen die Menge der wahren Sätze in der Sprache eines Modells M **Theorie von M**, geschrieben $Th(M)$. $Th(M)$ nennen wir **entscheidbar**, wenn ein Algorithmus existiert, der für jeden Satz in der Sprache von M entscheiden kann, ob er wahr ist.*

Definition 10. *Sei die Relation $+(a, b, c) = TRUE$, wenn $a+b = c$, und FALSE sonst.*

3 Beweisidee

Im Folgenden soll ein Algorithmus angegeben werden, der entscheidet, ob ein Satz Φ in der Sprache von $(\mathbb{N}, +)$ wahr ist.

Definition 11. *Sei $\Phi = Q_1 x_1 Q_2 x_2 \ldots Q_l x_l(\Psi)$, wo $Q_1 \ldots Q_l$ entweder ein $\exists$ oder ein $\forall$ ist und Ψ eine Formel ohne Quantoren mit den Variablen $x_1 \ldots x_l$ ist. Wir definieren für jedes i von 0 bis l $\Phi_i = Q_{i+1} x_{i+1} Q_{i+2} x_{i+2} \ldots Q_l x_l(\Psi)$.*

Beobachtung 1. *$\Phi_0 = \Phi$ und $\Phi_l = \Psi$. Φ_i hat i freie Variablen.*

Definition 12. *Wir definieren für $i \in \mathbb{N}^+$ das Alphabet:*

$$\Sigma_i = \{ \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \}$$

Jedes Zeichen besteht aus i Bits. Eine Zeichenkette über Σ_i^* soll i binäre Zahlen darstellen (reihenweise).

Algorithmus 1. *Eingabe: Satz Φ in der Sprache von $(\mathbb{N}, +)$.*

1. *Konstruiere alle definierten Φ_i.*
2. *Für jedes i von 0 bis l konstruiere einen endlichen Automaten A_i, der die Menge von Tupeln $(a_1, \ldots, a_i) \in \mathbb{N}^i$ (Belegungen für die freien Variablen), welche Φ_i wahr machen, erkennt.*
3. *Teste, ob A_0 die leere Zeichenkette erkennt. Wenn A_0 akzeptiert, gebe TRUE zurück.*

4 Beweis

Satz 1. *Th$(\mathbb{N}, +)$ ist entscheidbar.*

Proof.

1. Der erste Schritt kann durch eine einfache Turing-Maschine realisiert werden.
2. Eine endliche Kontrolle für einen endlichen Automaten kann für eine Turing-Maschine kodiert werden. Folglich kann so eine endliche Kontrolle von einer Turing-Maschine auch konstruiert werden.
   (a) Konstruktion von A_l:
   Φ_l enthält nur boolsche Operationen und Additionen mit Gleichheitsüberprüfung von natürlichen Zahlen. A_l wird als deterministischer, endlicher Automat angelegt. Konstruiere für die Additionen mit Gleichheitsüberprüfung die endlichen Automaten $B_1, \ldots, B_m$. Addition mit Gleichheitsüberprüfung kann bitweise überprüft werden durch unser gewähltes Eingabeformat für Zahlen. Konstruiere A_l durch Anwendung der Konstruktionen für Schnitt, Vereinigung und Komplementierung für endliche Automaten auf $B_1, \ldots, B_m$ den boolschen Operationen entsprechend.
   (b) Konstruktion von A_i aus A_{i+1} durch Fallunterscheidung:
      i. $\Phi_i = \exists x_{i+1} \Phi_{i+1}$:
      A_i wird als nichtdeterministischer, endlicher Automat angelegt. A_{i+1} hat als Eingabe eine Zeichenkette über Σ_{i+1}^* und A_i hat als Eingabe eine Zeichenkette über Σ_i^*. Konstruiere einen neuen Startzustand und einen Zustand für jeden Zustand von A_{i+1}.

Für jedes Symbol

$$\begin{bmatrix} b_1 \\ \vdots \\ b_{i-1} \\ b_i \end{bmatrix} \text{ mit } b_i \in 0, 1,$$

das gelesen wird, gibt es in jedem Zustand zwei mögliche Übergänge ($b_{i+1} = 0$ oder $b_{i+1} = 1$). a_{i+1} wird sozusagen simuliert. Da a_{i+1} durch Übertrag bei einer Addition um ein Bit länger sein kann als alle a_j mit $j < i+1$, gibt es vom Startzustand einen möglichen Übergang zu einem ehemaligen Startzustand für den Fall, dass a_{i+1} um ein Bit länger ist, und einen möglichen Übergang für den Fall, dass a_{i+1} nicht um ein Bit länger ist. Verbildlicht betrachtet, wird die simulierte Eingabe an

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \text{ oder } \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

angehängt. A_i akzeptiert $(a_1, \ldots, a_i)$, falls ein a_{i+1} existiert, sodass A_{i+1} $(a_1, \ldots, a_{i+1})$ akzeptiert.

ii. $\Phi_i = \forall x_{i+1} \Phi_{i+1}$:
$\forall x_{i+1} \Phi_{i+1} = \neg \exists x_{i+1} \neg \Phi_{i+1}$, d.h. wir können A_i konstruieren durch Konstruktion des Komplements von A_{i+1}, Anwendung der Konstruktion für $\exists$ und nochmaliger Konstruktion des Komplements.

3. Da A_0 konstruiert wurde, können wir den Test einfach durchführen.

□

Literatur

[1] Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, 1997.