ZUR ENTSCHEIDBARKEIT DER FUNKTIONALEN ÄQUIVALENZ

H. Beck
Institut für Informatik I, Universität Karlsruhe
D-7500 Karlsruhe 1, Postfach 6380

Abstract: In this paper we consider the equivalence problem for several classes of programs. We show that it is unsolvable for all "acceptable" program classes of the Grzegorczyk class $\mathcal{E}_0$ and of the polynominals over the integers formed with $\dot{-}$ instead of $-$. Furthermore we show that the equivalence problem is solvable for the class $L_1(\dot{-}1)$ of loop programs with the additional instruction $X := X\dot{-}1$, but again unsolvable for the class $L_1(\dot{-}1,+)$ and the Cleave-Ritchie-classes $L_2^2 \subsetneq L_2$ and $L_2^1(\dot{-}1)$.

1. Einführung

Will man Programme verifizieren oder optimieren, so stellt sich oft die Frage: "gibt es einen Algorithmus, der für zwei beliebige Programme einer Programmklasse Γ entscheidet, ob diese Programme dieselbe Funktion berechnen ?"; mit anderen Worten: "ist die (funktionale) Äquivalenz entscheidbar ?"

Tsichritzis (1970) hat für die Klasse L_1 der Loophierarchie einen solchen Algorithmus angegeben. Weiterhin zeigt er mit Resultaten von Meyer und Ritchie (1967), daß für die Klasse L_2 (diese Klasse berechnet gerade die elementaren Funktionen $\mathcal{E} = \mathcal{E}_3$ der Grzegorczykhierarchie) kein solcher Algorithmus existieren kann. Dabei lautet Lemma 1: "The equivalence problem is recursively unsolvable for primitive recursive functions". Man kann aber Programmklassen für die primitiv rekursiven Funktionen mit entscheidbarem Äquivalenzproblem angeben (siehe 2.6). Um diesen Widerspruch zu lösen, definieren wir in Abschnitt 2 in Analogie zu den Gödelnumerierungen (vgl. Rogers (1958)) die "akzeptablen" Programmklassen für subrekursive Funktionenklassen und übertragen dann das Äquivalenzproblem auf Klassen von Funktionen.

Im 3. Abschnitt zeigen wir mit Hilfe von Ergebnissen von Grzegorczyk (1953) und Matijasevič (1970), daß für alle Klassen der Grzegorczykhierarchie und für die mit $+,\dot{-},\cdot$ gebildeten Polynome über den natürlichen Zahlen die Äquivalenz unentscheidbar ist. Im 4. Abschnitt erweitern wir die Klasse L_1 der Loophierarchie durch Hinzunahme der Befehle $X := X\dot{-}1$ und $X := Y+Z$ zu den Klassen $L_1(\dot{-}1)$ und $L_1(\dot{-}1,+)$. Für die zugehörigen Funktionenklassen gilt: $\mathcal{L}_1(\dot{-}1,+) \supsetneq \mathcal{L}_1(\dot{-}1) \supsetneq \mathcal{L}_1$ und in $\mathcal{L}_1(\dot{-}1)$ ist die Äquivalenz entscheidbar, während sie in $\mathcal{L}_1(\dot{-}1,+)$ nicht entscheidbar ist.

Im letzten Abschnitt untersuchen wir die "Cleave-Ritchie-Hierarchien"

zwischen L_1 und L_2 bzw. $L_1(\dot{-}1)$ und $L_2(\dot{-}1)$ und zeigen, daß für L_2^2 und $L_2^1(\dot{-}1)$ die Äquivalenz nicht entscheidbar ist.

2. Definitionen und Schreibweisen

Sei $PR^{(n)}$ die Menge der n-stelligen partiell rekursiven Funktionen und sei $PR = \bigcup_n PR^{(n)}$. Seien i,j,k,n,m immer aus $\mathbb{N} = \{0,1,\ldots\}$. Seien C_k^n, U_i^n, s die Funktionen mit $\forall x_1 \ldots x_n (C_k^n(x_1,\ldots,x_n) = k,\ U_i^n(x_1,\ldots,x_n) = x_i)$ und $\forall x\ s(x) = x+1$. Sei $\mathcal{K} = \{x \mid$ x-te Turingmaschine angesetzt auf x hält$\}$.

Def. 2.1: A sei ein endliches Alphabet. Ist $\mathbb{P} \subset A^*$ mit $\mathbb{P} = \bigcup_n \mathbb{P}^{(n)}$ und $\mathbb{P}^{(n)} \cap \mathbb{P}^{(m)} = \emptyset$ für $n \neq m$ und ist φ eine partiell rekursive Abbildung $A^* \times \mathbb{N}^* \to \mathbb{N}$, dann heiße $(\mathbb{P},\varphi)$ Programmklasse mit dem Sprachschatz $\mathbb{P}$ und dem Interpreter φ. Die Menge $\mathcal{F} = \bigcup_n \mathcal{F}^{(n)}$, wobei

$$\mathcal{F}^{(n)} = \{\varphi_P \mid \exists P (P \in \mathbb{P}^{(n)} \wedge \varphi_P : \mathbb{N}^n \to \mathbb{N} \wedge \forall \underline{x} \in \mathbb{N}^n (\varphi_P(\underline{x}) = \varphi(P,\underline{x}))\}$$

heiße die zugehörige Funktionenklasse.

Def. 2.2: Eine Programmklasse $(\mathbb{P},\varphi)$ heiße akzeptabel, falls sie das uniforme S_n^m-Theorem erfüllt, das heißt, falls es eine partiell rekursive Abbildung $S: A^* \times \mathbb{N}^* \to A^*$ gibt, so daß $\forall n \forall m \forall P \in \mathbb{P}^{(n+m)} \forall \underline{x} \in \mathbb{N}^m$

$$\forall \underline{y} \in \mathbb{N}^n : S(P,\underline{x}) \in \mathbb{P}^{(n)} \wedge \varphi_P(\underline{x},\underline{y}) = \varphi_{S(P,\underline{x})}(\underline{y}).$$

Def. 2.3: Eine Programmklasse $(\mathbb{P},\varphi)$ heiße entscheidbar, falls es eine totale berechenbare Abbildung $t: A^* \to \mathbb{N}$ gibt, so daß

$$\forall \omega \in A^* : (t(\omega) = k+1 \text{ falls } \omega \in \mathbb{P}^{(k)}) \wedge (t(\omega) = 0 \text{ sonst})).$$

Durch Gödelisierungen lassen sich (akzeptable, entscheidbare) Programmklassen rekursiv auf (akzeptable, entscheidbare) Programmklassen mit $A^* = \mathbb{N}$ und zurück übersetzen. Ist $(\mathbb{P},\varphi)$ eine akzeptable entscheidbare Programmklasse mit $A^* = \mathbb{N}$ für PR, dann bilden die Listen $\{\varphi_P\}$ mit $P \in \mathbb{P}^{(n)}$ Gödelnumerierungen für $PR^{(n)}$; weiterhin läßt sich jede Gödelnumerierung mit geeignetem $(\mathbb{P},\varphi)$ so gewinnen.

Def. 2.4: Die Äquivalenz ist für eine Programmklasse $(\mathbb{P},\varphi)$ entscheidbar genau dann, wenn es eine partiell rekursive Abbildung $E: A^* \times A^* \to \mathbb{N}$ gibt, so daß $\forall n \forall P,Q \in \mathbb{P}^{(n)} : (E(P,Q) = 0 \text{ falls } \varphi_P \equiv \varphi_Q) \wedge (E(P,Q) = 1 \text{ sonst})$.

Lemma 2.5: Sind $(\mathbb{P},\varphi)$ und $(\mathbb{P}',\varphi')$ akzeptable Programmklassen und ist τ eine rekursive Übersetzung von $(\mathbb{P},\varphi)$ nach $(\mathbb{P}',\varphi')$, dann überträgt sich die Entscheidbarkeit der Äquivalenz von $(\mathbb{P}',\varphi')$ auf $(\mathbb{P},\varphi)$.

Lemma 2.6: Die Äquivalenz ist für eine entscheidbare Programmklasse der primitiv rekursiven Funktionen entscheidbar.

Beweisskizze: Man wähle die üblichen Definitionsschemen für die primitiv

rekursiven Funktion. Es gibt einen Algorithmus, der die kleinsten Definitionsschemen und somit für jede Funktion genau ein Schema aufzählt. Wählt man die Urbilder einer solchen Aufzählung als Programme, so hat man eine entscheidbare Programmklasse, die das Äquivalenzproblem löst. Um solche etwas pathologischen Programmklassen auszuschließen definieren wir:

Def. 2.7: Die Äquivalenz ist genau dann für eine Funktionenklasse entscheidbar, wenn sie für eine akzeptable Programmklasse einer Obermenge $\mathcal{F}'$ von $\mathcal{F}$ entscheidbar ist.

Folgerung 2.8: Nach Def. 2.7 überträgt sich die Entscheidbarkeit der Äquivalenz auf die Teilmengen und die Unentscheidbarkeit auf Obermengen einer Funktionenklasse. Außerdem kann man sich wegen Lemma 2.5 beim Führen der Entscheidbarkeits- oder Unentscheidbarkeitsbeweise auf Programmklassen mit $A^* = \mathbb{N}$ beschränken.

Lemma 2.9: Ist für eine Funktionenklasse $\mathcal{F}$ $f \leq g$ entscheidbar, so ist auch die funktionale Äquivalenz entscheidbar.

Beweis: $f \equiv g \nLeftrightarrow f \leq g \wedge g \leq f$.

3. Die Grzegorczykhierarchie und modifizierte Polynome

In Grzegorczyk (1953) werden die Klassen $\mathcal{E}_i$ eingeführt, welche bezüglich der Substitution und der beschränkten Rekursion abgeschlossen sind.

Satz 3.1: Die Äquivalenz ist für keine Grzegorczykklasse $\mathcal{E}_i$ entscheidbar.

Beweis: Wegen Folgerung 2.8 genügt es zu zeigen, daß es für kein $\mathcal{F} \supset \mathcal{E}_0$ eine akzeptable Programmklasse $(\mathbb{P},\varphi)$ mit $A^* = \mathbb{N}$ und entscheidbarer Äquivalenz geben kann. Sei $(\mathbb{P},\varphi)$ eine Programmklasse für $\mathcal{F} \supset \mathcal{E}_0$ mit $A^* = \mathbb{N}$. Nach Grzegorczyk (1953) (theorem 5.3) gibt es für jede rekursiv aufzählbare Menge also auch für $\mathcal{K}$ ein $f \in \mathcal{E}_0$ mit $\mathcal{K} = \{x \mid \exists y f(x,y) = 0\}$. Da die Signumfunktion sig aus $\mathcal{E}_0$ ist, ist auch $sig(f) \in \mathcal{E}_0$. Sei $k \in \mathbb{P}^{(2)}$ ein Programm für sig(f) und $j \in \mathbb{P}^{(1)}$ für C_1^1. Dann gilt:

$$\forall x(x \notin \mathcal{K} \nLeftrightarrow \neg(\exists y \varphi_k(x,y) = 0) \nLeftrightarrow \forall y(\varphi_k(x,y) = 1)$$

$$\nLeftrightarrow \forall y(\varphi_{S(k,x)}(y) = \varphi_j(y)) \nLeftrightarrow \varphi_{S(k,x)} \equiv \varphi_j \nLeftrightarrow E(S(k,x),j) = 0$$

Da $k,j \in \mathbb{P}$, sig(f) total und $(\mathbb{P},\varphi)$ akzeptabel ist, ist $\lambda x E(S(k,x),j)$ eine totale Funktion. Somit folgt aus der Entscheidbarkeit des Äquivalenzproblems für $\mathcal{E}_0$ die Lösbarkeit des Halteproblems für Turingmaschinen. Als Folgerung aus diesem Satz können wir mit Lemma 2.9 schließen, daß $f \leq g$ und damit auch die Frage: "definieren drei Funktionen durch beschränkte Rekursion eine neue Funktion ?" in allen $\mathcal{E}_i$ nicht entscheid-

bar ist. Die Definition von Grzegorzcyk führt deshalb zu nicht entscheidbaren akzeptablen Programmklassen. Durch Modifikation der beschränkten Rekursion lassen sich aber auch akzeptable entscheidbare Programmklassen für $\mathcal{E}_0$ finden. Als weitere Folgerung dieses Satzes ergibt sich, daß für die primitiv rekursiven, die elementaren und die subelementaren Funktionen die Äquivalenz nicht entscheidbar ist.
Als nächstes zeigen wir, daß es zwei Funktionenklassen gibt, deren Funktionen höchstens polynomiales Wachstum besitzen und für die die funktionale Äquivalenz nicht entscheidbar ist.

Def. 3.3 a: Sei Polm die Klasse von Funktionen, die aus den folgenden Funktionen durch Komposition entsteht: $C_k^n, U_i^n, s, +, \cdot, \dot{-}$ ($x_1 \dot{-} x_2 = x_1 - x_2$ falls $x_1 \geq x_2$ und gleich null sonst).

Def. 3.3 b: Sei Polg definiert wie Polm nur statt $\dot{-}$ werde $\chi_=$ ($\chi_=(x_1,x_2) = 0$, falls $x_1 = x_2$ und gleich eins sonst) als als Ausgangsfunktion gewählt.

Satz 3.4: Für Polm und Polg ist die Äquivalenz nicht entscheidbar.

Beweis: Da $\forall x \forall y (\chi_=(x,y) = 1 \dot{-} (1 \dot{-} ((1 \dot{-} x) + (1 \dot{-} y))))$ ist, gilt Polm $\supset$ Polg, so daß der Beweis wegen Folgerung 2.9 nur für Polg geführt werden muß.

Nach Matijasevič (10-Hilbert'sches Problem) gibt es ein Polynom f_1 (dieses habe m+2-Stellen) und zu jedem einstelligen rekursiv aufzählbaren Prädikat T ein t, so daß gilt:
$\forall x (Tx \not\leftrightarrow \exists y_1 \ldots \exists y_m [f_1(t,x,y_1,\ldots,y_m) = 0]$ mit $x,t,y_1,\ldots,y_m \in \mathbb{N}$.
Sei t so gewählt, daß $T = \mathcal{K}$ und sei f das Polynom, welches aus f_1 durch Einsetzung von diesem t entsteht.
Ein Polynom f ist gleich der Differenz zweier Polynome p und q, welche keine Subtraktion und nur positive Koeffizienten enthalten. Damit gilt:
$\forall \underline{z} \in \mathbb{N}^{m+1} (f(\underline{z}) = 0 \not\leftrightarrow \chi_=(p(\underline{z}), q(\underline{z})) = 0)$. Sei nun $(\mathbb{P}, \varphi)$ eine akzeptable Programmklasse für $\mathcal{F} \supset$ Polg, dann gibt es $i \in \mathbb{P}^{(m+1)}$, so daß $\varphi_i \equiv \chi_=(p,q)$, da $\chi_=(p,q) \in$ Polg ist. Weiter sei $j \in \mathbb{P}^{(m)}$, so gewählt, daß $\varphi_j \equiv C_1^m$. Nun können wir schließen:

$$\forall x (x \notin K \not\leftrightarrow \neg \exists y_1 \ldots \exists y_m f_1(k,x,y_1,\ldots,y_m) = 0 \not\leftrightarrow \neg \exists y_1 \ldots y_m \varphi_i(x,y_1,\ldots,y_m) = 0$$

$$\not\leftrightarrow \forall y_1 \ldots \forall y_m \varphi(x,y_1,\ldots,y_m) = 1 \not\leftrightarrow \varphi_{S(i,x)} \equiv \varphi_j \not\leftrightarrow E(S(i,x),j) = 0$$

Die Lösbarkeit des Äquivalenzproblems bedingt also auch hier die Lösbarkeit des Halteproblems für Turingmaschinen.
Für die Klasse Pol der Polynome über $\mathbb{Z}$ ist die funktionale Äquivalenz dagegen durch Koeffizientenvergleich lösbar.

4. Loopprogramme

Die Klasse L der Loopprogramme wurde von Meyer und Ritchie (1967) eingeführt. Wir wollen diese und zwei Erweiterungen im Folgenden (der Kürze wegen nur informal) definieren. Durch Loopprogramme werden natürliche Zahlen, die in beliebig vielen Registern X_i abgespeichert sind manipuliert. Der Sprachschatz der Loopprogramme lautet:

Def. 4.1: Ein Loopprogramm $P \in L$ ist eine endliche Folge von Befehlen, die mit $In(X_1,\ldots,X_n)$ beginnt und mit $Out(X_o)$ endet. Die übrigen Befehle haben die Form $X_i := X_j, X_i := X_i+1, X_i := 0$, $Loop(X_i)$ oder End. Für $P \in L(\dot{-}1)$ sind außerdem noch die Befehle $X_i := X_i \dot{-} 1$ und für $P \in L(\dot{-}1,+)$ die Befehle $X_i := X_i \dot{-} 1$ und $X_i := X_j + X_k$ erlaubt. Die Befehle $Loop(X_i)$ und End müssen immer wie öffnende und schließende Klammern angeordnet sein.

Def. 4.2: Ein Loopprogramm wird interpretiert, indem die Befehle der Reihe nach abgearbeitet und die Registerinhalte entsprechend verändert werden. Zu Beginn seien alle Registerinhalte gleich 0. Dann liest $In(X_1,\ldots,X_n)$ das Argument n-Tupel ein. Die Interpretation der Befehle $X_i := X_i+1, X_i := 0, X_i := X_j$ und $X_i = X_j+X_k$ ist offensichtlich. $X_i := X_i \dot{-} 1$ wird abgearbeitet indem der Inhalt des Registers X_i um 1 erniedrigt wird, falls er größer null ist, und sonst gleich null bleibt. Ist als nächster Befehl $Loop(X_i)$ abzuarbeiten, so suche man das zugehörige End und arbeite das Programmstück, welches zwischen $Loop(X_i)$ und diesem End liegt so oft ab, wie der Inhalt des Registers X_i beim Eintrit in diese Schleife angibt. Anschließend führe man die Interpretation des diesem End folgenden Befehls aus. $Out(X_o)$ gibt schließlich den Inhalt des Registers X_o als Ergebnis aus. Jedem Loopprogramm P wird durch diese Interpretation φ eine Funktion φ_P zugeordnet.
Zur Abkürzung stehe $L(\sim)$ für $L, L(\dot{-}1)$ oder $L(\dot{-}1,+)$.

Def. 4.3: Für alle n sei $L_n(\sim)$ die Klasse von Programmen aus $L(\sim)$, deren $Loop(X_i)$-End-Paare höchstens n-mal geschachtelt sind, das heißt $L_o(\sim)$ enthalte kein $Loop(X_i)$-End-Paar, $L_1(\sim)$ nur einfache Schleifen etc.

Def. 4.4: $\mathcal{L}(\sim)$ bzw. $\mathcal{L}_n(\sim)$ seien die zu $L(\sim)$ bzw. $L_n(\sim)$ und der Interpretation φ gehörenden Funktionenklassen. $\mathcal{L}(\sim)$ sind gerade die primitiv rekursiven Funktionen.

Lemma 4.5: Die Klassen $(L_n(\sim), \varphi)$ sind entscheidbare akzeptable Programmklassen.

Beweisskizze: Der einzige nicht offensichtliche Punkt ist das

S_n^m-Theorem. Wie man die Funktion S erhält, sei hier am Beispiel S_1^1 gezeigt. Ist P :≡ $In(X_1,X_2);P'$, dann lautet $S_1^1(P,x)$ = $In(X_1);X_2 := X_1;$ $X_1 := O;$ $\underbrace{X_1 := X_1+1;\ldots,X_1 := X_1+1}_{x\text{-mal}};P'$.

Für den Beweis des nächsten Satzes benötigen wir die Presburger Arithmetik. Dies ist die Theorie, die zu dem üblichen Axiomensystem für O, ' (Nachfolger) und + in der Logik der 1. Stufe mit Gleichheit gehört. Die n+1-stelligen rechtseindeutigen Ausdrücke definieren n-stellige berechenbare Funktionen.

Satz 4.6: Für $\mathcal{L}_1(\dot{-}1)$ ist die funktionale Äquivalenz entscheidbar.

Beweis: Wie wir im nächsten Satz zeigen, läßt sich $L_1(\dot{-}1)$ rekursiv in die Presburgerarithmetik übersetzen. Da die Presburgerarithmetik entscheidbar ist (vgl. Presburger (1930)), ist das Äquivalenzproblem dort lösbar und somit nach Lemma 2.5 auch in $L_1(\dot{-}1)$.

Def. 4.7: Eine Funktion heiße "partiell linear", wenn sie durch Komposition aus den Ausgangsfunktion $C_o^n, U_i^n, s, +, \dot{-}, w(w(x_1,x_2)=x_1$ für $x_2 \neq O \wedge =O$ sonst), [x/k] und [x,k] (abgeschnittene Division bzw. Rest bei Division durch eine Konstante) gewonnen werden kann. Dies sind gerade um $\dot{-}$ als Ausgangsfunktion erweiterten "simple functions".

Satz 4.8: a) Die Klasse $L_1(\dot{-}1)$ läßt sich in die Klasse der "partiell linearen" Funktionsausdrücke übersetzen. b) Die "partiell linearen" Funktionsausdrücke lassen sich in die Presburgerarithmetik übersetzen.

Beweisskizze: Zu a) Hierzu kann der Beweis des Theorems 4 von Tsichritzis (1970) so erweitert werden, daß auch der Befehl $X:=X\dot{-}1$ miterfaßt wird. Zu b) Zunächst werden für die Grundfunktionen der "partiell linearen" Funktionen äquivalente Presburgerausdrücke angegeben. So kann z.B. $y=x_1\dot{-}x_2$ in $x_1=y+x_2 \vee (y=O \wedge \exists z(x_1+z=x_2))$ und $y=[x/k]$ in $\exists v(\exists w(v+w+1=k) \wedge x=\underbrace{y+\ldots+y}_{k\text{-mal}}+v)$ übersetzt werden.

Sei f aus g,h_1 und h_2 durch Substitution gewonnen, also $f(x) = g(h_1(x),h_2(x))$ und seien H_1,H_2 bzw. G Presburgerausdrücke für h_1,h_2 und g, dann ist
$F \equiv \exists y_1 \exists y_2 (H_1(x,y_1) \wedge H_2(x,y_2) \wedge G(y_1,y_2,y))$ ein Ausdruck für f. Somit kann zu jedem partiell linearen Funktionsausdruck durch Induktion über seinen Aufbau ein äquivalenter Presburger Ausdruck gefunden werden.

5. Cleave-Ritchie-Hierarchien

Die bisher betrachteten Loophierarchien $L_i(\sim)$ werden nun weiter verfeinert, indem der Bereich zwischen zwei benachbarten Loopklassen in

unendliche Hierarchien aufgeteilt wird. Dabei seien die $L_i^k(\sim)$-Programme gerade die Programme aus $L_{i+1}(\sim)$, die höchstens k-verschiedene Loop-End Paare der Schachtelungstiefe i+1 besitzen. $\mathcal{L}_i^k$ seien die zugehörigen Funktionenklassen. Diese sind nun im allgemeinen nicht mehr gegenüber der Komposition von Funktionen abgeschlossen. Daß sie aber gegenüber der Substitution von Konstanten abgeschlossen sind, zeigt:

Lemma 5.1: Die Klassen $L_i^k[\sim]$ sind akzeptable entscheidbare Programmklassen.

Beweis: Der Beweis geht wie für Lemma 4.5.

Satz 5.2: Für die Klassen $\mathcal{L}_1^2$ und $\mathcal{L}_2^1(\dot{-}1)$ ist die Äquivalenz nicht entscheidbar.

Beweisskizze: Sind p und q beliebige Polynome ohne Subtraktionen, so läßt sich $\chi_=(p,q)$ mit L_2^2 bzw. $L_2^1(\dot{-}1)$-Programmen berechnen. Damit ergibt sich dann wie beim Beweis des Satzes 3.4 ein Widerspruch zwischen der Unlösbarkeit des 10. Hilbert'schen Problems und der Entscheidbarkeit der Äquivalenz.

Schluß: Insgesamt zeigt sich also, daß die funktionale Äquivalenz nicht erst in der Klasse $\mathcal{E}$ der elementaren Funktionen, die exponentiell-wachsende Funktionen enthält, sondern schon in kleineren Klassen, deren Funktionen höchstens polynomial oder linear wachsen unentscheidbar werden kann. Andernseits lassen sich die "simple functions" noch ohne Verlust der Entscheidbarkeit etwas erweitern.

Literaturverzeichnis

Grzegorczyk, A.: "Some classes of recursive functions", Rozprawy Mathematyczne 4. (1953)

Matijasevič, J.V.: "Enumerable sets are Diophantine", Soviet Math. Dokl. 11, 354-358. (1970)

Meyer, A.R., D.M. Ritchie: "The complexity of loop programs", Proc. 22nd Nat. Conf. ACM, 1967, ACM Pub P-67, pp. 465-469

Presburger, M.: "Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchen die Addition als einzige Operation hervortritt", Comptes-rendus du I Congrès des Mathématiciens des Pays Slaves, Warsaw (1930) pp. 92-101, 395

Tsichritzis, D.: "The equivalence problem of simple programs", Journal of the Association for Computing Machinery 17: 729-738. (1970)