Complexity of Presburger Arithmetic

Alexander N. Rybalov
Omsk State University
Omsk, Russia

January 11, 2007

Presburger Arithmetic (PA)

is the first-order theory of the model $< \mathbb{N}, + >$.
It is known that PA is:

- consistent
- complete
- decidable

Worst-Case Complexity of PA

Theorem (Fischer, Rabin). *There is no algorithm for deciding of Presburger arithmetic with worst-case complexity less than* $2^{2^{cn}}$ *with some universal constant* $c > 0$.

M.J.Fischer, M.O.Rabin. Super-Exponential Complexity of Presburger Arithmetic. // Proceedings of the SIAM-AMS Symposium in Applied Mathematics Vol. 7: 27–41, 1974.

Diagonalization

The Halting Problem (HP): Does Turing machine M halt on $\delta(M)$?

Suppose there is a machine H deciding HP:

$$H(\delta(M)) = \begin{cases} 1, & M(\delta(M)) \downarrow, \\ 0, & M(\delta(M)) \uparrow . \end{cases}$$

Then there is a machine G:

$$G(\delta(M)) = \begin{cases} \text{loops}, & M(\delta(M)) \downarrow, \\ 0, & M(\delta(M)) \uparrow . \end{cases}$$

Contradiction: $G(\delta(G)) \downarrow \Leftrightarrow G(\delta(G)) \uparrow$

Diagonalization

A Restricted Halting Problem (RHP): Does Turing machine M halt on $\delta(M)$ in time less than $2^{|\delta(M)|}$?

RHP is decidable. Suppose there is a machine H deciding RHP in time less than 2^n:

$$H(\delta(M)) = \begin{cases} 1, & t_M(\delta(M)) < 2^{|\delta(M)|}, \\ 0, & t_M(\delta(M)) \geq 2^{|\delta(M)|}. \end{cases}$$

Then there is a machine G such that:

$t_G(\delta(M)) > 2^{|\delta(M)|}$, if $t_M(\delta(M)) < 2^{|\delta(M)|}$,

$t_G(\delta(M)) < 2^{|\delta(M)|}$, if $t_M(\delta(M)) \geq 2^{|\delta(M)|}$.

Contradiction:

$$t_G(\delta(G)) < 2^{|\delta(G)|} \Leftrightarrow t_G(\delta(G)) \geq 2^{|\delta(G)|}$$

Diagonalization for PA

To prove super-exponential complexity of Presburger Arithmetic we can polynomially reduce the following Restricted Halting Problem:

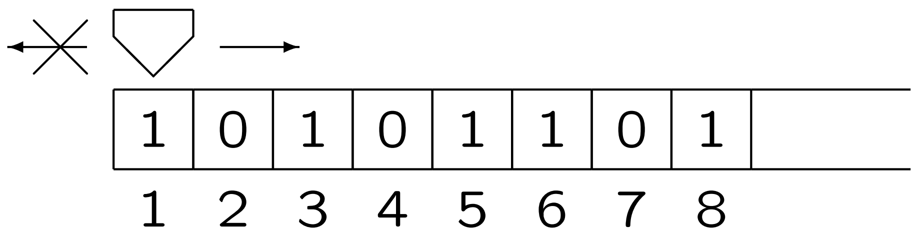
Does machine M halt on w in time $< 2^{2^{|w|}}$?

to the problem:

Is formula $\Phi_{M,w}$ true in Presburger Arithmetic?

Computational Model

Turing machines with one-way tape over alphabet $\{0, 1\}$.



Program of such machine is a list of rules

$$
\begin{array}{lcl}
(1,0) & \to & (i_1, s_1, T_1) \\
(1,1) & \to & (i_2, s_2, T_2) \\
\dots & & \\
(k-1,1) & \to & (i_{2(k-1)}, s_{2(k-1)}, T_{2(k-1)})
\end{array}
$$

where i_j is a state from $1, \dots, k-1$ or k – the final state, $s_j \in \{0, 1\}$, T_j – a shift from $\{R, L\}$.

Modelling of Machine

All $T = 2^{2^n}$ steps of M on w is completely described by:

binary strings $W_1, \ldots, W_T$ of length T, where W_i is content of the first T cells of the tape on the i-th step,

strings $U_1, \ldots, U_T$ of length T over alphabet $\{0, \ldots, k\}$, where $U_i = 0^{p_i} s_i 0^{q_i}$ with s_i – state of M and p_i – position of the head on i-th step ($p_j + q_j + 1 = T$).

We join these strings in two $W = W_1 \ldots W_T$ and $U = U_1 \ldots U_T$ of length T^2.

Modelling of Machine

Strings W and U together code T steps of a halting computation of M on w iff

1. $W_1 = w0^{T-|w|}$ — content of tape on the start.

2. $U_1 = 10^{T-1}$ — starting position of the head and starting state of M.

3. If $U(i) = 0$ and $i+T < T^2$ then $W(i+T) = W(i)$ — that means cells, which are not scanned by the head on current step, are not changed on the next step.

Modelling of Machine

4. If $U(i) = q$, $i+T < T^2$, $0 < q < k$, $W(i) = 0$ and the program of M has a rule $(q, 0) \to (p, R, 1)$. Then $W(i+T) = 1$. Similarly for other rules and tape-symbol combinations.

5. If $U(i) \neq 0$, $T < i < T^2$ then exactly one of $U(i-T)$, $U(i-T-1)$, $U(i-T+1)$ does not equal to 0. Also if $U(i) \neq 0$ then $U(i \pm 1) = U(i \pm 2) = 0$.

6. There exists some i, $1 \leq i \leq T^2$ such that $U(i) = k$ — final state. And if $i + T < T^2$ then $U(i+T) = k$ and $W(i+T) = W(i)$.

Expression by a Formula

W is coded by one integer $x \leq 2^T$. U is a string of alphabet $\{1, \ldots, k\}$, so at first we express every $1, \ldots, k$ by binary words of equal length $p = \lceil log(k) \rceil$ integers, and later represent U as p integers $x_1, \ldots, x_p \leq 2^T$. So

$$\Phi_{M,w} = \exists x \exists x_1 \ldots \exists x_p (E_1 \wedge \ldots \wedge E_6).$$

Here formulas E_i express the conditions 1-6.

Expression by a Formula

To express formulas E_i in PA it is enough to have predicates ($f(n) = 2^{2^n}$):

- $I_n(b) \Leftrightarrow b < f(n)^2$.
- $J_n(b) \Leftrightarrow b = f(n)$.
- $S_n(x, y)$ codes all binary strings of length $f(n)^2$ in the following way: for all $b \in \{0, 1\}^*$, $|b| = f(n)^2$ there exists an integer a such that $S_n(a, i)$ is true iff $b(i) = 1$.
- $H_w(x)$ is true for a iff the first $f(n)$ symbols of the string, coded by a (via predicate S_n), has the form $w0^p$, $p = f(n) - |w|$.

Expression by a Formula

The predicates I_n and J_n are used to control the ranges of indexes of cells, H_w - for starting configuration, S_n - for transition conditions. For example, E_3 is

$$\neg S_n(x_1, y) \wedge \ldots \wedge \neg S_n(x_p, y) \wedge I_n(y+z) \to$$

$$\to (S_n(x, y+z) \leftrightarrow S_n(x, y)).$$

Other E_i are constructed in the similar way.

Formula for S_n

To code a binary word w of length $2^{2^{n+1}}$ we use the integer $x \leq 2^{2^{2^{n+1}}}$ with binary expression w. The predicate $S_n(x, y)$ just says that $x \leq 2^{2^{2^{n+1}}}$, $y \leq 2^{2^{n+1}}$ and i-th bit of x is 1. We can express it by the following formula:

$$\exists z(2^y \leq z < 2^{y+1} \wedge z \leq x \wedge 2^{y+1} | x - z).$$

We need now compact ($O(n)$) formulas for: divisibility ($x|y$), power ($x = 2^y$), and order ($x \leq y$) for very big numbers $x, y \leq 2^{2^{2^{n+1}}}$. This will implies, in particular, expressibility of I_n and J_n.

Order and Divisibility

Formula for order:

$$x \leq y \Leftrightarrow \exists z(y = x + z).$$

If we can express multiplication relation $x = yz$ then we get a formula for divisibility:

$$y|x \Leftrightarrow \exists z(x = yz).$$

Multiplication for "Small" Numbers

Lemma. *For any n there exists formula $M_n(x, y, z)$ such that*

$$M_n(x, y, z) \Leftrightarrow x < 2^{2^n} \wedge xy = z$$

and $|M_n| = O(n)$.

Induction on n. For $n = 0$ we have

$M_n(x, y, z) = (x = 0 \wedge z = 0) \vee (x = 1 \wedge z = y)$.

Suppose we have M_n, to get M_{n+1} note that $x < 2^{2^{n+1}}$ iff

$$\exists x_1, x_2, x_3, x_4 < 2^{2^n} x = x_1 x_2 + x_3 + x_4$$

and so

$$z = xy = x_1(x_2 y) + x_3 y + x_4 y,$$

hence M_{n+1} is expressible by M_n.

Multiplication for "Small" Numbers

The Problem: In this expression M_n can be more than once (that is bad for bound on the formula length), for example

$$M_n(x_1, y_1, z_1) \wedge M_n(x_2, y_2, z_2)$$

and the size of M_{n+1} increases more than on constant. To avoid this we need to rewrite such formula in an equivalent formula with only one occurrence of M_n:

$$\forall x, y, z((x = x_1 \wedge y = y_1 \wedge z = z_1) \vee$$

$$\vee(x = x_2 \wedge y = y_2 \wedge z = z_2)) \rightarrow M_n(x, y, z).$$

Multiplication for "Big" Numbers

Lemma. *For any n there is a formula $Prod_n(x, y, z)$ such that for any $a, b, c \leq 2^{2^{2^{n-1}}}$ it holds*

$$Prod_n(a, b, c) \Leftrightarrow ab = c.$$

And $|Prod_n| = O(n)$.

By the Chinese Remainder Theorem (CRT) if for $x, y, z < 2^{2^{2^{n+1}}}$ and for all primes $p < 2^{2^n}$

$$res(x, p)res(y, p) = res(z, p)$$

then

$$res(x, g)res(y, g) = res(z, g), \quad (*)$$

where

$$g = \prod_{p < 2^{2^n}} p > 2^{\pi(2^{2^n})} > 2^{\frac{2^{2^n}}{2^n}} = 2^{2^{2^{n-1}}}$$

– by the Prime Number Theorem. So $(*)$ implies that $xy = z$.

Multiplication for "Big" Numbers

$$res(x,y) = z \Leftrightarrow \exists u(uy + z = x) \wedge (z < y)$$

can be expressed by compact formulas with M_n for "small" numbers $y, z \leq 2^{2^n}$ and any number x.

$$Prime(x) \Leftrightarrow \forall y \forall z(x = yz) \to (y = 1 \vee z = 1)$$

can be expressed by compact formulas with M_n for "small" number $x \leq 2^{2^n}$.

Now

$$Prod_n(x,y,z) \Leftrightarrow \forall p(p < 2^{2^n} \to$$

$$\to res(x,p)res(y,p) = res(z,p))$$

can be expressed by compact formula for "big" numbers $x, y, z \leq 2^{2^{2^{n-1}}}$.

Formula for Power

Lemma. *For any n there is a formula $Pow_n(x, y, z)$ such that for all $a, b \leq 2^{2^n}$ and $c \leq 2^{2^{2^n}}$*

$$Pow_n(a, b, c) \Leftrightarrow a^b = c$$

and $|Pow_n| = O(n)$.

Induction on n. Again if $x \leq 2^{2^{n+1}}$ then

$$x = x_1 x_2 + x_3 + x_4$$

for some $x_1, x_2, x_3, x_4 \leq 2^{2^n}$ and

$$y^x = (y^{x_1})^{x_2} \cdot y^{x_3} \cdot y^{x_4}.$$

So Pow_{n+1} can be expressed in terms of Pow_n and $Prod_n$. And

$$|Pow_{n+1}| \leq |Pow_n| + C$$

for some constant C.

Formula for H_w

$H_w(x)$ is true for a iff the first $f(n)$ symbols of the string, coded by a (via predicate S_n), has the form $w0^p$, $p = f(n) - |w|$.

Define for binary word u by induction on $|u|$ formulas:

$$\begin{aligned}
&K_0(z) \Leftrightarrow z = 0,\\
&K_1(z) \Leftrightarrow z = 1,\\
&K_{u0}(z) \Leftrightarrow \exists y(K_u(y) \wedge z = y + y),\\
&K_{u1}(z) \Leftrightarrow \exists y(K_u(y) \wedge z = y + y + 1).
\end{aligned}$$

Clearly $K_w(z) \Leftrightarrow w(i) = z(i)$ for $i < |w|$ and $z(i) = 0$ for $i \geq |w|$. So we can express H_w as

$$\forall z \forall i (K_w(z) \wedge i < 2^n \wedge x(i) = z(i))$$

and use $S_n(x, y)$ and $J_n(y)$.