SIAM J. COMPUT.
Vol. 11, No. 1, February 1982

0097-5397/82/1101-0001 $01.00/0

STRAIGHT-LINE PROGRAMS WITH ONE INPUT VARIABLE*

OSCAR H. IBARRA† AND BRIAN S. LEININGER†

Abstract. Let $\mathbb{C}$ be the set of all straight-line programs with *one* input variable, x, using the following instruction set: $y \leftarrow 0$, $y \leftarrow 1$, $y \leftarrow y + w$, $y \leftarrow y - w$, $y \leftarrow y * w$, and $y \leftarrow \lfloor y/w \rfloor$. We show that two programs in $\mathbb{C}$ are equivalent over integer inputs if and only if they are equivalent on all inputs x such that $|x| \leqq 2^{2^{\lambda r^2}}$ (λ is a fixed positive constant and r is the maximum of the lengths of the programs). In contrast, we prove that the zero-equivalence problem (deciding whether a program outputs 0 for all inputs) is undecidable for programs with *two* input variables. An interesting corollary is the following: Let $\mathbb{N}$ be the set of natural numbers and f be any total one-to-one function from $\mathbb{N}$ onto $\mathbb{N} \times \mathbb{N}$ (f is called a pair generator. Such functions are useful in recursive function theory and computability theory.) Then f cannot be computed by any program in $\mathbb{C}$.

Key words. straight-line program, equivalence, zero-equivalence, decidable, undecidable, Hilbert's tenth problem, pair generator

1. Introduction. In this paper, we study the problem of deciding the equivalence of straight-line programs over integer inputs using the operations $+$, $-$, $*$, and $/$, where division is $\lfloor x/y \rfloor$ = the greatest integer $\leqq x/y$ (e.g., $\lfloor 5/4 \rfloor = 1$, $\lfloor -4/3 \rfloor = -2$, etc.).[1] Two programs are equivalent if they are defined at the same points and equal wherever they are defined. Our main result is that equivalence is decidable for straight-line programs with *one* input variable. (There is no restriction on the number of auxiliary and output variables.) More precisely, we show that two programs with one input variable over the instruction set $\{y \leftarrow 0, y \leftarrow 1, y \leftarrow y + w, y \leftarrow y - w, y \leftarrow y * w, y \leftarrow \lfloor y/w \rfloor\}$ are equivalent if and only if they are equivalent on all inputs x such that $|x| \leqq 2^{2^{\lambda r^2}}$, where λ is a fixed positive constant and r is the maximum of the lengths of the programs. (The length of a program is the number of instructions in it.) The double exponential bound cannot be reduced substantially since we can show that for infinitely many r's there are nonequivalent programs with at most r instructions that are equivalent on all inputs x such that $|x| \leqq 2^{2^{\lambda' r}}$ (λ' is a fixed positive constant). In contrast, we can show that the zero-equivalence problem (deciding whether a program outputs 0 for all inputs) is undecidable for programs with *two* input variables. An interesting corollary is that no pair generator can be computed by a program using the instruction set above. A pair generator is any one-to-one function from $\mathbb{N}$ (set of natural numbers) onto $\mathbb{N} \times \mathbb{N}$. Such functions are useful in recursive function theory and computability theory. The undecidability of the zero-equivalence problem for programs with *two* input variables should be contrasted with a recent result in [6]. It was shown in [6] that the zero-equivalence problem for $\{y \leftarrow 0, y \leftarrow 1, y \leftarrow y + w, y \leftarrow \lfloor y/w \rfloor\}$-programs[2] with *ten* input variables is undecidable. This result does not use the operations $-$ and $*$.

There are other types of division: $\lceil x/y \rceil$ and $\langle x/y \rangle$. $\lceil x/y \rceil$ is the least integer $\geqq x/y$ (e.g. $\lceil 5/4 \rceil = 2$, $\lceil -4/3 \rceil = -1$, etc.), $\langle x/y \rangle$ is the integral part of x/y (e.g. $\langle 5/4 \rangle = 1$, $\langle -4/3 \rangle = -1$, etc.). Clearly, $\lfloor x/y \rfloor$ and $\langle x/y \rangle$ are identical when $xy \geqq 0$, but may differ when $xy < 0$. The following propositions whose proofs are given in the Appendix show that $\lfloor x/y \rfloor$, $\lceil x/y \rceil$, and $\langle x/y \rangle$ are not independent operations.

* Received by the editors November 28, 1979, and in final form March 31, 1981. This research was supported by the National Science Foundation under grant MCS78-01736.

† Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455.

[1] We assume that division by 0 is undefined. If an instruction attempts to divide by 0, the program goes into an infinite loop, and its output is undefined.

[2] $\{i_1, \cdots, i_k\}$-programs denotes the class of programs using the instruction set $\{i_1, \cdots, i_k\}$.

PROPOSITION 1. *The instruction* $x \leftarrow \langle x/y \rangle$ *can be computed by a fixed program using only instructions of the form* $y \leftarrow 0$, $y \leftarrow 1$, $y \leftarrow y + w$, $y \leftarrow y - w$, $y \leftarrow y * w$, $y \leftarrow \lfloor y/w \rfloor$.

PROPOSITION 2. *The instructions* $x \leftarrow \lfloor x/y \rfloor$ *and* $x \leftarrow \lceil x/y \rceil$ *can be computed by fixed programs using only instructions of the form* $y \leftarrow 0$, $y \leftarrow 1$, $y \leftarrow y + w$, $y \leftarrow y - w$, $y \leftarrow y * w$, $y \leftarrow \langle y/w \rangle$.

For notational convenience, most of the results in the paper are stated using only the division $\lfloor x/y \rfloor$. However, it is obvious from Propositions 1 and 2 that the results remain valid when all types of division are used.

Remark. We could include the construct $y \leftarrow c$ (c is any positive integer) in our instruction set. However, it is clear that $y \leftarrow c$ can be computed by a $\{y \leftarrow 0, y \leftarrow 1, y \leftarrow y + w\}$-program with at most $O(\log c)$ instructions. Thus, inclusion of the construct $y \leftarrow c$ in the instruction set is not necessary.

2. The main result. Our main theorem is a generalization of the fact that two nth degree polynomials (outputs of programs using only $+$, $-$, $*$) are identical if they agree on $\{0, 1, \cdots, n\}$. Division introduces several complications:

(a) division by 0 is possible (so functions become nontotal);
(b) rational functions can be computed instead of polynomials;
(c) truncation permits selective forward conditional branching, by evaluating different functions for different residue classes.

Nevertheless, it is possible to associate a rational function R of the input variable with each program statement, such that $R(x)(\lfloor R(x) \rfloor)$ will be the value computed by the statement. We show in this paper that two such expressions are equivalent if they are identical for some computable initial segment of the integers. This is so because rational functions behave asymptotically as polynomials; in particular, they are nonzero for large values of the input, unless they are identically zero. Thus, (a) and (b) can be handled. Truncation may introduce branching, but the degree of branching is bounded by the number of residue classes of the greatest value computed by the program (relative to all possible moduli, i.e., relative to all possible moduli smaller than this value). Thus the number of cases is finite for each program, and can be tested. Our task then is to formally prove that the strategy above can be carried out. For convenience, we introduce the following notation.

Notation. Let $c = m/n$ be a rational number (positive, negative, or zero), where m and n are integers with $\gcd(m, n) = 1$. If $c = 0$, take $m = 0$ and $n = 1$. We use the following notation: Num $(c) = |m|$ = absolute value of m and Denom $(c) = |n|$.

Our first lemma concerns polynomial division. It says that, for sufficiently large values of x, a rational form $r(x)$ behaves like a polynomial. Lemma 1 gives a sufficient condition on x for a good approximation.

LEMMA 1. *Let* $r(x) = s(x)/z(x) = p(x) + q(x)/z(x)$, *where* $s(x)$, $z(x)$, $p(x)$ *and* $q(x)$ *are polynomials with rational coefficients such that* $z(x) \neq 0$ ($p(x)$ *and* $q(x)$ *are the polynomials given by the division algorithm*). *Let*

$$b = \max\{2, \text{Num}(c), \text{Denom}(c) | c \text{ is a rational coefficient in } s(x) \text{ or } z(x)\},$$

$$d = \max\{\text{degree}(s(x)), \text{degree}(z(x))\},$$

$$\alpha = b^{2(d+2)^3}.$$

Then for all $x \geqq \alpha$

$$\lfloor r(x) \rfloor = \lfloor s(x)/z(x) \rfloor = \begin{cases} \lfloor p(x) \rfloor & \text{if } p(x) \text{ is not an integer or if} \\ & \text{sign}(q(\alpha)/z(\alpha)) \text{ is nonnegative,} \\ \lfloor p(x) - 1 \rfloor & \text{otherwise.} \end{cases}$$

Moreover, if c is a coefficient of $p(x)$, *then* Num (c), Denom $(c) \leqq b^{2(d+1)^2}$.

Proof. Let h be the least positive integer such that $h \cdot s(x)$ and $h \cdot z(x)$ are polynomials with integer coefficients. Clearly, the absolute values of the coefficients of $h \cdot s(x)$ and $h \cdot z(x)$ do not exceed the bound b^{2d+2}. So without loss of generality we may assume that the coefficients of $s(x)$ and $z(x)$ are integers and their absolute values are bounded by b^{2d+2}.

Let $s(x) = s_1 x^k + \cdots + s_k x + s_{k+1}$ and $z(x) = z_1 x^l + \cdots + z_l x + z_{l+1}$ for some $k, l \geqq 0$. Assume similar notation for the other polynomials in this proof. Also, by multiplying $s(x)$ and $z(x)$ by -1 if necessary, we can assume $z_1 > 0$. For convenience, define $z_j = 0$ for $j \geqq l+2$. Then by the division algorithm

$$r(x) = \frac{s(x)}{z(x)} = p(x) + \frac{q(x)}{z(x)}.$$

Since we would like to derive the worst possible bound on c, we assume the following:

(i) $k \geqq l$ since, otherwise, $p(x) = 0$ and $q(x) = s(x)$.

(ii) $l \geqq 1$ (i.e., degree $(z(x)) \geqq 1$). Note that this implies that degree $(q(x)) < l$.

One can easily check that these assumptions given rise to a worst-case bound on c.

We can find the coefficients of $p(x)$ by the division algorithm. We have

$$p(x) = p_1 x^{k-l} + \cdots + p_{k-l} x + p_{k-l+1}, \quad \text{where}$$

$$p_1 = \frac{s_1}{z_1}, \tag{1}$$

$$p_i = \frac{s_i - \sum_{j=1}^{i-1} p_j z_{i+1-j}}{z_1} = \frac{s_i - \sum_{j=1}^{i-1} p_j z_{i+1-j}}{z_1} \cdot \frac{z_1^{i-1}}{z_1^{i-1}} \quad \text{for } 2 \leqq i \leqq k-l+1. \tag{2}$$

Also, by induction on i,

$$p_i z_1^i \text{ is an integer for } 1 \leqq i \leqq k-l+1. \tag{3}$$

From (1)–(3) and the fact that $s_1, \cdots, s_{k+1}, z_1, \cdots, z_{l+1}$ are integers with absolute values bounded by b^{2d+2}, we have

$$\begin{aligned} &\text{Num } (p_i) \leqq 2^{i-1} b^{(2d+2)i} \quad \text{and} \\ &\text{Denom } (p_i) \leqq b^{(2d+2)i} \quad \text{for } 1 \leqq i \leqq k-l+1. \end{aligned} \tag{4}$$

Since $b \geqq 2$ and $d \geqq k \geqq l \geqq 1$, (4) becomes

$$\begin{aligned} &\text{Num } (p_i) \leqq 2^{d-1} b^{(2d+2)d} \leqq b^{2(d+1)^2} \quad \text{and} \\ &\text{Denom } (p_i) \leqq b^{(2d+2)d} \leqq b^{2(d+1)^2} \quad \text{for } 1 \leqq i \leqq k-l+1. \end{aligned} \tag{5}$$

Now $q(x) = s(x) - p(x)z(x)$. Then

$$\begin{aligned} q_1 x^{l-1} + \cdots + q_{l-1} x + q_l = &(s_1 x^k + \cdots + s_k x + s_{k+1}) \\ &- (p_1 x^{k-l} + \cdots + p_{k-l} x + p_{k-l+1})(z_1 x^l + \cdots + z_l x + z_{l+1}). \end{aligned}$$

So

$$q_{l-r} = s_{k-r+1} - \sum_{j=0}^{r} (p_{k-l-j+1})(z_{l+j-r+1}) \quad \text{for } r = 0, 1, \cdots, l-1.$$

Letting $r = l - i$, we have

$$q_i = s_{k-l+i+1} - \sum_{j=0}^{l-i} (p_{k-l-j+1})(z_{i+j+1}) \quad \text{for } i = 1, 2, \cdots, l. \tag{6}$$

From (5) and (6), we easily obtain upper and lower bounds on the absolute values of the *nonzero* q_i's:

$$|q_i| \leqq b^{2d+2} + lb^{2(d+1)^2} b^{(2d+2)} \leqq b^{2d+2} + db^{2(d+1)^2+(2d+2)} \leqq b^{2(d+2)^2}, \tag{7}$$

$$|q_i| \geqq \frac{1}{(b^{2(d+1)^2})^l} \geqq \frac{1}{b^{2(d+1)^2 d}} \geqq \frac{1}{b^{2(d+1)^3}}. \tag{8}$$

We can write $p(x) = u(x)/m$, where m is the least positive integer such that $u(x)$ is a polynomial with integer coefficients. Now choose the least nonnegative integer β such that

$$\left|\frac{q(x)}{z(x)}\right| < \frac{1}{m} \quad \text{for all } x \geqq \beta.$$

The bounds for m and β are found as follows:

From (5),

$$m \leqq (b^{2(d+1)^2})^{\text{degree}\,(p(x))+1} \leqq b^{2(d+1)^3}. \tag{9}$$

Now, for all x, $|q(x)/z(x)| < 1/m$ if and only if $|z(x)| - m|q(x)| > 0$. Hence from (7) and (9), for sufficiently large x,

$$\begin{aligned} |z(x)| - m|q(x)| &\geqq x^l - l(b^{2d+2} + mb^{2(d+2)^2})x^{l-1} \\ &\geqq (x - d(b^{2d+2} + b^{2(d+1)^3} b^{2(d+2)^2}))x^{l-1} \\ &= (x - d(b^{2d+2} + b^{2(d+1)^3+2(d+2)^2}))x^{l-1} > 0. \end{aligned}$$

Let $\beta = b^{2(d+2)^3}$. Then $\beta > d(b^{2d+2} + b^{2(d+1)^3+2(d+2)^2})$ and, for all $x \geqq \beta$, $|q(x)/z(x)| < 1/m$.

Let α be the least integer such that $\alpha \geqq \beta$ and sign $(q(x)/z(x)) =$ sign $(q(\alpha)/z(\alpha))$ for all $x \geqq \alpha$. We consider two cases:

Case 1. sign $(q(\alpha)/z(\alpha))$ is nonnegative or $p(x)$ is not an integer. Then clearly $\lfloor r(x) \rfloor = \lfloor s(x)/z(x) \rfloor = \lfloor p(x) \rfloor$.

Case 2. sign $(q(\alpha)/z(\alpha))$ is negative and $p(x)$ is an integer. Then $\lfloor r(x) \rfloor = \lfloor s(x)/z(x) \rfloor = \lfloor p(x) - 1 \rfloor = p(x) - 1$.

The bound on α is derived as follows: Let the degree of $q(x)$ be $l - i$ for some $1 \leqq i \leqq l - 1$. Then from (7) and (8), for sufficiently large x,

$$\begin{aligned} |q(x)| &\geqq \frac{x^{l-i}}{b^{2(d+1)^3}} - (l-i)b^{2(d+2)^2} x^{l-i-1} \\ &\geqq \left(\frac{x - (l-i)b^{2(d+1)^3+2(d+2)^2}}{b^{2(d+1)^3}}\right) x^{l-i-1} \\ &\geqq \left(\frac{x - db^{2(d+1)^3+2(d+2)^2}}{b^{2(d+1)^3}}\right) x^{l-i-1} > \left(\frac{x - b^{2(d+2)^3}}{b^{2(d+1)^3}}\right) x^{l-i-1} \end{aligned}$$

and

$$|z(x)| \geqq x^l - lb^{2d+2} x^{l-1} \geqq (x - db^{2d+2})x^{l-1} > (x - b^{3(d+1)})x^{l-1}.$$

Let $\alpha = \max\{\beta, b^{2(d+2)^3}, b^{3(d+1)}\} = b^{2(d+2)^3}$. Then for all $x \geqq \alpha$, $\operatorname{sign}(q(x)/z(x)) = \operatorname{sign}(q(\alpha)/z(\alpha))$. □

It is well known that every nonzero polynomial has a finite number of zeros. The next proposition bounds the absolute value of a zero.

PROPOSITION 3. *Let $p(x)$ be a nonzero polynomial with rational coefficients. Let $d = \text{degree}(p(x))$ and $b = \max\{\text{Num}(c), \text{Denom}(c) | c$ is a rational coefficient in $p(x)\}$. Let $\beta = db^2 + 1$. Then $p(x) \neq 0$ for all $x \geqq \beta$.*

Proof. Clearly, $p(x) \neq 0$ for all x such that $x^d/b - dbx^{d-1} > 0$. □

We will also need the following proposition which is easily verified.

PROPOSITION 4. *Let $(m_1, n_1), \cdots, (m_k, n_k)$ be pairs of integers such that $0 \leqq m_i < n_i$ for $1 \leqq i \leqq k$. Let l be a positive integer. If there exists an integer x_0 such that $x_0 > l + n_1 n_2 \cdots n_k$ and $x_0 \bmod n_i = m_i$ for[3] $1 \leqq i \leqq k$, then there exists another integer x_0' such that $l \leqq x_0' \leqq l + n_1 \cdots n_k$ and $x_0' \bmod n_i = m_i$ for $1 \leqq i \leqq k$.*

Proof. Let $x_0' = x_0 - rn_1 \cdots n_k$, where r is the largest positive integer such that $x_0 - rn_1 \cdots n_k \geqq l$. □

Let P be a $\{y \leftarrow 0, y \leftarrow 1, y \leftarrow y + w, y \leftarrow y - w, y \leftarrow y * w, y \leftarrow \lfloor y/w \rfloor\}$-program with one input variable x. The next lemma shows that the value of any variable y at the end of r instructions can be described by a finite nonempty set $S(r, y)$ of congruence classes of the input. Each congruence class is a pair $(p(x), T)$, where $p(x)$ is a polynomial in x with rational coefficients and T is a finite nonempty set of pairs of integers. $(p(x), T)$ in $S(r, y)$ means that the value of y on large enough input x_0 at the end of r instructions (if defined) is equal to $p(x_0)$ if and only if $x_0 \bmod n = m$ for each (m, n) in T.

LEMMA 2. *Let P be a $\{y \leftarrow 0, y \leftarrow 1, y \leftarrow y + w, y \leftarrow y - w, y \leftarrow y * w, y \leftarrow \lfloor y/w \rfloor\}$-program with one input variable x (but with an arbitrary number of auxiliary and output variables). Assume without loss of generality that x does not appear on the left-hand side (LHS) of any instruction in P. Let y be any variable in P (possibly x), and $r \geqq 1$. Then there is a finite nonempty set $S(r, y)$ with elements of the form $(p(x), T)$, where $p(x)$ is a polynomial in x with rational coefficients and T is a finite nonempty set of pairs of integers. $S(r, y)$ has the following properties:*

(1) Let x_0 be an input such that $x_0 \geqq 2^{2^{3r^2}}$. If the value of y is defined at the end of r instructions then there is a unique element $(p(x), T)$ in $S(r, y)$ such that $x_0 \bmod n = m$ for all (m, n) in T, and the value of y on input x_0 (at then end of r instructions) is given by $p(x_0)$. (We say in this case that $(p(x), T)$ uniquely defines y on input x_0.) Moreover, if x_0' is such that $x_0' \geqq 2^{2^{3r^2}}$ and $x_0' \bmod n = m$ for all (m, n) in T, then $(p(x), T)$ also uniquely defines y on input x_0'.

(2) Let $x_0 \geqq 2^{2^{3r^2}}$. Suppose y on input x_0 is uniquely defined by $(p(x), T)$ in $S(r, y)$ (at the end of r instructions). If $p(x)$ is not the zero polynomial, then the value of $y = p(x_0) \neq 0$.

(3) $|S(r, y)| =$ cardinality of $S(r, y) \leqq 2^{2^{5(r+1)^2}}$.

(4) If $(p(x), T)$ is in $S(r, y)$, then $\text{degree}(p(x)) \leqq 2^r$.

(5) If c is a rational coefficient in $p(x)$, then $\text{Num}(c), \text{Denom}(c) \leqq 2^{2^{3r^2}}$.

(6) If (m, n) is in T, then $0 \leqq m < n \leqq 2^{2^{3r^2}}$.

(7) $|T| = \leqq 2^{r^2}$.

Proof. The proof is an induction on r. At the start, all variables except x have the value 0. Let y be the variable on the LHS of the first instruction. By assumption,

[3] For integers u and v with $v > 0$, let r = remainder of $|u|/v$. Then

$$u \bmod v = \begin{cases} r & \text{if } r = 0 \text{ or } u \geqq 0, \\ v - r & \text{if } r > 0 \text{ and } u < 0. \end{cases}$$

$y \neq x$. Then

$$S(1, x) = \{(x, \{(0, 1)\})\} \quad \text{and}$$

$$S(1, w) = \{(0, \{(0, 1)\})\} \quad \text{for all } w \neq x,\ w \neq y.$$

There are four cases to consider for $S(1, y)$:

(i) If the first instruction is $y \leftarrow c (c = 0 \text{ or } 1)$, then let $S(1, y) = \{(c, \{(0, 1)\})\}$.

(ii) If the first instruction is $y \leftarrow y + w$ or $y \leftarrow y - w$ and $w \neq x$, then let $S(1, y) = \{(0, \{(0, 1)\})\}$.

(iii) If the first instruction is $y \leftarrow y + x$ or $y \leftarrow y - x$, then let $S(1, y) = \{(x, \{(0, 1)\})\}$ or $S(1, y) = \{(-x, \{(0, 1)\})\}$, respectively.

(iv) If the first instruction is $y \leftarrow y * w$ (w can be x) or $y \leftarrow \lfloor y/x \rfloor$ then let $S(1, y) = \{(0, \{(0, 1)\})\}$.

Clearly, properties (1)–(7) hold. Assume now that the lemma holds for sequences of $r \geqq 1$ instructions. We show that it also holds for sequences of $r+1$ instructions. Let y be the variable on the LHS of the $(r+1)$st instruction (note that $y \neq x$). Then, for each variable $w \neq y$, define $S(r+1, w) = S(r, w)$. Obviously, properties (1)–(7) hold for $S(r+1, w)$. If $w = y$, we consider 3 cases.

Case 1. The $(r+1)$st instruction is $y \leftarrow c$, where $c = 0$ or 1. Then let $S(r+1, y) = \{(c, \{(0, 1)\})\}$. Clearly, properties (1)–(7) hold.

Case 2. The $(r+1)$st instruction is $y \leftarrow y$ op w, where op is $+$, $-$ or $*$. Let x_0 be an input such that $x_0 \geqq 2^{2^{3(r+1)^2}}$. Suppose that on input x_0, y and w are defined at the end of r instructions and their values are uniquely defined by $(s(x), T_1)$ in $S(r, y)$ and $(z(x), T_2)$ in $S(r, w)$, respectively. Then $(s(x)$ op $z(x), T_1 \cup T_2)$ uniquely defines y on input x_0 at the end of $r+1$ instructions, and $(s(x)$ op $z(x), T_1 \cup T_2)$ should be in the set $S(r+1, y)$. It is straightforward to verify that properties (1) and (4)–(7) hold for $(s(x)$ op $z(x), T_1 \cup T_2)$. That property (2) is satisfied follows from Proposition 3. The elements of $S(r+1, y)$ are obtained by varying the value of $x_0 \geqq 2^{2^{3(r+1)^2}}$. Now $|T_1 \cup T_2| \leqq 2^{r^2} + 2^{r^2} \leqq 2^{(r+1)^2}$, and if (m, n) is in $T_1 \cup T_2$, then $0 \leqq m < n \leqq 2^{2^{3r^2}} \leqq 2^{2^{3(r+1)^2}}$. It follows from Proposition 4 that there are at most $2^{2^{5(r+1)^2}}$ values of x_0 giving rise to distinct elements of $S(r+1, y)$. Hence, $|S(r+1, y)| \leqq 2^{2^{5(r+1)^2}}$, showing property (3).

Case 3. The $(r+1)$st instruction is $y \leftarrow \lfloor y/w \rfloor$. Let x_0 be an input such that $x_0 \geqq 2^{2^{3(r+1)^2}}$. Suppose that on input x_0, y and w are defined at the end of r instructions and their values are uniquely defined by $(s(x), T_1)$ in $S(r, y)$ and $(z(x), T_2)$ in $S(r, w)$, respectively. If $z(x)$ is not the zero polynomial, then the value of y at the end of $r+1$ instructions is given by $\lfloor s(x_0)/z(x_0) \rfloor$. Note that by induction hypothesis, $z(x_0) \neq 0$.

We show how to construct an element $(u(x), T)$ in $S(r+1, y)$ uniquely defining y on input x_0. Now,

$$\left\lfloor \frac{s(x)}{z(x)} \right\rfloor = \left\lfloor p(x) + \frac{q(x)}{z(x)} \right\rfloor,$$

where $p(x)$ and $q(x)$ are obtained from $s(x)$ and $z(x)$ by the division algorithm. By the induction hypothesis, degree $(s(x))$, degree $(z(x)) \leqq 2^r = d$. Moreover, if c is a coefficient in $s(x)$ or $z(x)$ then Num (c), Denom $(c) \leqq 2^{2^{3r^2}} = b$. Let

$$p(x) = c_1 x^k + \cdots + c_k x + c_{k+1}, \quad \text{where } k \leqq d.$$

For $1 \leqq i \leqq k$, let $c_i = v_i/n_i$, where v_i and n_i are integers such that $n_i > 0$ and gcd $(v_i, n_i) = 1$. (If $c_i = 0$, take $v_i = 0$ and $n_i = 1$). Then, for $1 \leqq i \leqq k$,

$$c_i x_0^{k-i+1} = \lfloor c_i x_0^{k-i+1} \rfloor + \frac{l_i}{n_i},$$

where $l_i = (v_i x_0^{k-i+1}) \bmod n_i$. Then

$$\begin{aligned}\left\lfloor \frac{s(x_0)}{z(x_0)} \right\rfloor &= \left\lfloor p(x_0) + \frac{q(x_0)}{z(x_0)} \right\rfloor \\ &= \left\lfloor c_1 x_0^k + \cdots + c_k x_0 + c_{k+1} + \frac{q(x_0)}{z(x_0)} - \sum_{i=1}^{k} \frac{l_i}{n_i} + \sum_{i=1}^{k} \frac{l_i}{n_i} \right\rfloor \\ &= \left\lfloor \left(c_1 x_0^k - \frac{l_1}{n_1} \right) + \cdots + \left(c_k x_0 - \frac{l_k}{n_k} \right) + c_{k+1} + \frac{q(x_0)}{z(x_0)} + \sum_{i=1}^{k} \frac{l_i}{n_i} \right\rfloor \\ &= c_1 x_0^k + \cdots + c_k x_0 - \sum_{i=1}^{k} \frac{l_i}{n_i} + \left\lfloor c_{k+1} + \frac{q(x_0)}{z(x_0)} + \sum_{i=1}^{k} \frac{l_i}{n_i} \right\rfloor.\end{aligned}$$

Now $c_1 x_0^k + \cdots + c_k x_0 - \sum_{i=1}^{k} l_i/n_i$ is an integer. Then by Lemma 1, since $x_0 \geqq 2^{2^{3(r+1)^2}} \geqq (2^{2^{3r^2}})^{2(2r+2)^3} = b^{2(d+2)^3} = \alpha$,

$$\left\lfloor c_{k+1} + \frac{q(x_0)}{z(x_0)} + \sum_{i=1}^{k} \frac{l_i}{n_i} \right\rfloor = \begin{cases} \left\lfloor c_{k+1} + \sum_{i=1}^{k} \frac{l_i}{n_i} \right\rfloor & \text{if } c_{k+1} + \sum_{i=1}^{k} l_i/n_i \text{ is not an integer or} \\ & \text{if sign } (q(\alpha)/z(\alpha)) \text{ is nonnegative,} \\ \left\lfloor c_{k+1} + \left(\sum_{i=1}^{k} \frac{l_i}{n_i} \right) - 1 \right\rfloor & \text{otherwise.} \end{cases}$$

Let $u(x) = c_1 x^k + \cdots + c_k x + c'_{k+1}$, where

$$c'_{k+1} = \begin{cases} -\sum_{i=1}^{k} \frac{l_i}{n_i} + \left\lfloor c_{k+1} + \sum_{i=1}^{k} \frac{l_i}{n_i} \right\rfloor & \text{if } c_{k+1} + \sum_{i=1}^{k} l_i/n_i \text{ is not an integer or if} \\ & \text{sign } (q(\alpha)/z(\alpha)) \text{ is nonnegative,} \\ -\sum_{i=1}^{k} \frac{l_i}{n_i} + \left\lfloor c_{k+1} + \left(\sum_{i=1}^{k} \frac{l_i}{n_i} \right) - 1 \right\rfloor & \text{otherwise.} \end{cases}$$

The bounds on the coefficients can be obtained using Lemma 1: For $1 \leqq i \leqq k+1$,

$$\text{Num } (c_i), \text{ Denom } (c_i) \leqq b^{2(d+1)^2} \leqq (2^{2^{3r^2}})^{2(2r+1)^2} \leqq 2^{2^{3(r+1)^2}}.$$

It follows that for $1 \leqq i \leqq k$, $0 \leqq l_i < n_i \leqq b^{2(d+1)^2}$. Hence,

$$\text{Denom } (c'_{k+1}) \leqq (b^{2(d+1)^2})^k \leqq b^{2(d+1)^3} \leqq 2^{2^{3(r+1)^2}}$$

and

$$\begin{aligned}\text{Num } (c'_{k+1}) &\leqq k b^{2(d+1)^2 k} + b^{2(d+1)^2 k} (b^{2(d+1)^2} + k + 1) \\ &\leqq d b^{2(d+1)^2 d} + b^{2(d+1)^2 d} (b^{2(d+1)^2} + d + 1) \leqq b^{2(d+2)^3} \leqq 2^{2^{3(r+1)^2}}.\end{aligned}$$

Now let $m_i = x_0 \bmod n_i$ for $1 \leqq i \leqq k$. Then l_i depends only on m_i. Define $T = T_1 \cup T_2 \cup \{(m_1, n_1), \cdots, (m_k, n_k)\}$. Clearly, properties (1), (4), (5) and (6) hold. Also $|T| = |T_1| + |T_2| + k \leqq 2^{r^2} + 2^{r^2} + 2^r \leqq 2^{(r+1)^2}$. Hence property (7) is satisfied. The proof that property (3) holds is similar to that of case 2. Now $u(x)$ has degree at most $d = 2^r$, and if c is a coefficient in $u(x)$, Num (c), Denom $(c) \leqq b^{2(d+2)^3}$, where $b = 2^{2^{3r^2}}$. Hence, by Proposition 3, unless $u(x) = 0$, $u(x_0) \neq 0$ for all $x_0 \geqq 2^{2^{3(r+1)^2}} > 2^r((2^{2^{3r^2}})^{2(2^r+2)^3})^2 = d(b^{2(d+2)^3})^2$. Thus, (2) holds. □

To handle inputs which cause division by 0, we need the next lemma.

LEMMA 3. *Let P and r be as in Lemma 2. There is a (possibly empty) set $Z(r)$ with elements that are finite nonempty sets of pairs of integers. $Z(r)$ has the following*

properties:

(1) $|Z(r)| \leqq 2^{2^{5(r+1)^2}}$.

(2) *If* T *is in* $Z(r)$, *then* $|T| \leqq 2^{r^2}$.

(3) *If* (m, n) *is in* T, *then* $0 \leqq m < n \leqq 2^{2^{3r^2}}$.

(4) *A division by* 0 *on input* $x_0 \geqq 2^{2^{3r^2}}$ *occurs during the first* r *instructions if and only if there is a* T *in* $Z(r)$ *such that* $x_0 \bmod n = m$ *for all* (m, n) *in* T.

Proof. We describe the construction of $Z(r)$. For convenience, define $Z(0) = \varnothing$. Now assume that $Z(r)$ has been constructed for $r \geqq 0$ and it satisfies (1)–(4) of the lemma. If the $(r+1)$st instruction is not a division instruction, let $Z(r+1) = Z(r)$. Now suppose that $(r+1)$st instruction is $y \leftarrow \lfloor y/w \rfloor$. Let x_0 be an input such that $x_0 \geqq 2^{2^{3r^2}}$. The instruction $y \leftarrow \lfloor y/w \rfloor$ will contribute to a division by zero on input x_0 if and only if w is defined and is equal to 0 at the end of r instructions. By Lemma 2 (property (2)), the element $(p(x), T)$ in $S(r, w)$ uniquely defining w (at the end of r instructions) on input x_0 must have $p(x)$ identically equal to 0. Add T to $Z(r+1)$. Clearly, the number of T's added to $Z(r+1)$ is at most $|S(r, w)| \leqq 2^{2^{5(r+1)^2}}$, and hence, $|Z(r+1)| \leqq 2^{2^{5(r+2)^2}}$. □

We are now ready to prove our main theorem.

THEOREM 1. *Let* P_1 *and* P_2 *be two* $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y + w,\ y \leftarrow y - w,\ y \leftarrow y * w,\ y \leftarrow \lfloor y/w \rfloor\}$*-programs with one input variable* x (*but with an aribtrary number of auxiliary and output variables*). *Assume that* P_1 *and* P_2 *have the same number of output variables. Let* $r = \max\{r_1, r_2\}$, *where* r_i = *number of instructions in* P_i. *Then* P_1 *and* P_2 *are equivalent over nonnegative integer inputs if and only if they are equivalent on all inputs* $0 \leqq x_0 \leqq 2^{2^{\lambda r^2}}$, *where* λ *is some fixed constant.*

Proof. By Lemma 2, the value of any variable y of P_i on input $x_0 \geqq 2^{2^{3r_i^2}}$ at the end of r_i instructions (if defined) is uniquely determined by an element $(p(x), T)$ in $S(r_i, y)$ where $p(x)$ is a polynomial of degree at most 2^{r_i} with rational coefficients, and T is a set of integers of the form (m, n), $0 \leqq m < n \leqq 2^{2^{3r_i^2}}$, and $|T| \leqq 2^{r_i^2}$. Similarly, by Lemma 3, the values of $x_0 \geqq 2^{2^{3r_i^2}}$ which cause program P_i to divide by 0 are determined by the set $Z(r_i)$, where an element of $Z(r_i)$ is a nonempty set T of integers (m, n). (Again, $0 < m < n \leqq 2^{2^{3r_i^2}}$ and $|T| \leqq 2^{r_i^2}$.) Now two polynomials of degree at most 2^r are identical if and only if they agree on $2^r + 1$ points. It follows from Proposition 4 that P_1 and P_2 are equivalent if and only if they are equivalent on all inputs

$$x_0 \leqq (2^r + 1)[2^{2^{3r^2}} + (2^{2^{3r^2}})^{2r^2}] \leqq 2^{2^{5r^2}}. \qquad \square$$

The double exponential bound of Theorem 1 cannot substantially be reduced since we can prove the following proposition.

PROPOSITION 5. *There are nonequivalent programs* P_1 *and* P_2 *with at most* $r \geqq 5$ *instructions that are equivalent on all inputs* $\leqq 2^{2^{r-4}}$.

Proof. Let P_1 and P_2 be the following programs (x is the input/output variable):

P_1:		P_2:	
$y \leftarrow 1$		$y \leftarrow 1$	
$y \leftarrow y + y$		$y \leftarrow y + y$	
$y \leftarrow y * y$ $\vdots$ $y \leftarrow y * y$	$r-3$	$y \leftarrow y * y$ $\vdots$ $y \leftarrow y * y$	$r-4$
$x \leftarrow \lfloor x/y \rfloor$		$x \leftarrow \lfloor x/y \rfloor$	

Clearly, P_1 and P_2 agree on all inputs $x \leqq 2^{2^{r-4}}$. But P_1 and P_2 are not equivalent. □

Theorem 1 also holds when the input variable can assume negative values:

COROLLARY 1. *Let P_1, P_2, and r be as in Theorem* 1. *Then P_1 and P_2 are equivalent over integer inputs if and only if they are equivalent on all inputs x such that $|x| \leqq 2^{2^{\lambda r^2}}$ (λ is some fixed constant). Moreover, the upper bound cannot be reduced substantially, since for infinitely many r's there are nonequivalent programs with at most r instructions that are equivalent on all inputs x such that $|x| \leqq 2^{2^{\lambda' r}}$ (λ' is a fixed constant).*

Proof. Let P_1 and P_2 be two programs. For $i = 1, 2$, construct program P_i' by inserting the following code at the beginning of P_i (z is a new variable):

$$z \leftarrow 0$$
$$z \leftarrow z - x$$
$$x \leftarrow 0$$
$$x \leftarrow x + z.$$

Then P_1 and P_2 are equivalent over (positive, negative, or zero) integer inputs if and only if P_1 and P_2 are equivalent over nonnegative integer inputs and P_1' and P_2' are equivalent over nonnegative integer inputs. The result now follows from Theorem 1 and Proposition 5. □

Our next result shows that Theorem 1 does not hold for programs with two input variables.

THEOREM 2. *The zero-equivalence problem for $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y + w,\ y \leftarrow y - w,\ y \leftarrow y * w,\ y \leftarrow \lfloor y/w \rfloor\}$-programs with two input variables (over nonnegative integer inputs) is undecidable. The result holds for programs with no more than 6 program variables. Moreover, the programs are total in that no division by 0 occurs.*

Proof. The proof uses the undecidability of Hilbert's tenth problem [3]. Let F be a Diophantine polynomial with r variables. Let

$$F = F(x_1, \cdots, x_r) = \sum_{j=1}^{m} c_j x_1^{j_1} \cdots x_r^{j_r},$$

where $|c_j| > 0$ and $j_k \geqq 0$ for $1 \leqq j \leqq m$ and $1 \leqq k \leqq r$. We shall construct a program P_F with input variables x and y and output variable z such that P_F outputs 0 for all nonnegative integer values of x and y if and only if F has no nonnegative integer solution in $x_1, \cdots, x_r$. The result would then follow from the fact that it is undecidable to determine if an arbitrary Diophantine polynomial has a nonnegative integer solution [3].

Given a nonnegative integer x and a positive integer y, we can think of x as a number in base y,

$$x = x_0 + x_1 y^1 + x_2 y^2 + \cdots + x_r y^r + v y^{r+1},$$

where $x_0, x_1, \cdots, x_r$, and v are nonnegative integers with $0 \leqq x_i < y$. Clearly, $(x_1, \cdots, x_r)$ can be made to assume all possible r-tuples by varying x and y. The program P_F decodes $x_1, \cdots, x_r$ and computes $F(x_1, \cdots, x_r)$. P_F then outputs 0 if and only if $F(x_1, \cdots, x_r) \neq 0$. The program P_F is given by the following code, which is

easily translated to a program using the instruction set $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y+w,\ y \leftarrow y-w,\ y \leftarrow y*w,\ y \leftarrow \lfloor y/w \rfloor\}$:

$z \leftarrow 0$

$y \leftarrow 2y+1$ (makes y nonzero)

α_{10}

α_{11}
$\vdots$
$\alpha_{1(r+1)}$

α_{20}

α_{21}
$\vdots$
$\alpha_{2(r+1)}$
$\vdots$
α_{m0}

α_{m1}
$\vdots$
$\alpha_{m(r+1)}$

β

where

(1) For $1 \leqq j \leqq m$, α_{j0} is the code

$w \leftarrow 1$

(2) For $1 \leqq j \leqq m$ and $1 \leqq k \leqq r$, α_{jk} is the code

$v \leftarrow \lfloor x/y^k \rfloor$

$s \leftarrow \lfloor x/y^{k+1} \rfloor y$

$v \leftarrow v - s$

$w \leftarrow w * v^{j_k}$

At the end of α_{jk}, w will contain $x_1^{j_1} \cdots x_k^{j_k}$.

(3) For $1 \leqq j \leqq m$, $\alpha_{j(r+1)}$ is the code

$z \leftarrow z + c_j w$

Clearly, at the end of $\alpha_{m(r+1)}$, z will contain $F(x_1, \cdots, x_r) = \sum_{j=1}^{m} c_j x_1^{j_1} \cdots x_r^{j_r}$.

(4) The code for β is

$z \leftarrow \lfloor (z^2+1)/(2z^2+1) \rfloor$.

Then $z = 0$ if and only if $F(x_1, \cdots, x_r) \neq 0$. It follows that P_F outputs 0 for all nonnegative integer values of x and y if and only if F has no solution. □

Theorem 2 remains valid when the input variables can assume negative values:

COROLLARY 2. *Same as Theorem* 2, *but now the input variables can assume all integer values.*

Proof. Modify P_F by using the following code for β:

$$x \leftarrow \lfloor x/(x^2+1) \rfloor$$
$$y \leftarrow \lfloor y/(y^2+1) \rfloor$$
$$z \leftarrow z^2+x^2+y^2$$
$$z \leftarrow \lfloor (z+1)/(2z+1) \rfloor.$$

Call the new program P'_F. Then P'_F outputs 0 for all integer values of x and y if and only if F has no nonnegative integer solution. □

Remark. By a slightly more complicated coding the number of program variables in Theorem 2 and Corollary 2 can be reduced to 5.

3. An application. Let $\mathbb{N}$ be the set of natural numbers. It is well known that there exist effectively computable total one-to-one functions from $\mathbb{N}$ onto $\mathbb{N}\times\mathbb{N}$. Such functions are called *pair generators* [5]. Pair generators are useful in recursive function theory and computability theory (see, e.g., [2], [4], [5], [7]). Our next theorem shows that pair generators are "not easy" to compute.

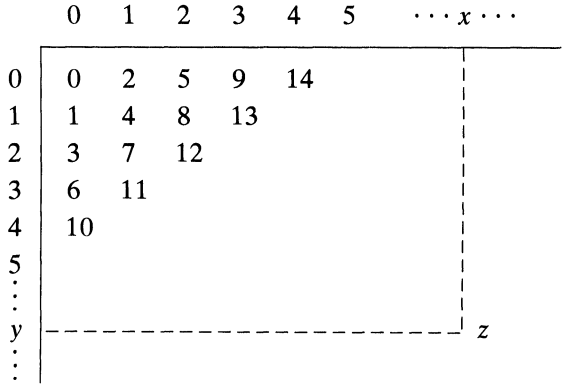
THEOREM 3. *No* $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y+w,\ y \leftarrow y-w,\ y \leftarrow y*w,\ y \leftarrow \lfloor y/w \rfloor\}$-*program* (*with one input variable and two output variables*) *can compute a pair generator.*

Proof. The proof is by contradiction. Suppose that $f: \mathbb{N} \to \mathbb{N}\times\mathbb{N}$ is a pair generator which is computed by a program P. Let u be the input variable of P and x, y be its output variables. For each program P_F constructed in the proof of Theorem 2, we define a new program P'_F:

P
P_F

We assume that P_F has input variables x and y and these are the only variables that P_F has in common with P. Now, P'_F has one input variable u, and P'_F outputs 0 for all nonnegative integer values of u if and only if P_F outputs 0 for all nonnegative integer values of x and y. The result now follows from Theorems 1 and 2. □

Theorem 3 does not hold for inverses of pair generators. (The inverses are called *pairing functions* [4], [7].) There are pair generators with easily computable inverses. For example, consider the pair generator f shown below:



$$f: \mathbb{N} \to \mathbb{N}\times\mathbb{N}$$
$$z \to (x, y).$$

$f^{-1}: \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is given by

$$f^{-1}(x, y) = z = \left\lfloor \frac{(x+y)^2 + 3x + y}{2} \right\rfloor.$$

$f^{-1}(x, y)$ is computable by a $\{x \leftarrow 1,\ x \leftarrow x+y,\ x \leftarrow x * y,\ x \leftarrow \lfloor x/2 \rfloor\}$-program (with two input variables). The function f is defined by two functions g_1 and g_2 (see [2]):

$$x = g_1(z) = \left\lfloor \frac{Q_2(z) - Q_1(z)}{2} \right\rfloor,$$

$$y = g_2(z) = Q_1(z) - \left\lfloor \frac{Q_2(z) - Q_1(z)}{2} \right\rfloor,$$

where

$$Q_1(z) = \left\lfloor \frac{\lfloor \sqrt{8z+1} \rfloor + 1}{2} \right\rfloor - 1,$$

$$Q_2(z) = 2z - (Q_1(z))^2.$$

Hence, there are pair generators that are computable by $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y+w,\ y \leftarrow y-w,\ y \leftarrow y * w,\ y \leftarrow \lfloor y/w \rfloor,\ y \leftarrow \lfloor \sqrt{y} \rfloor\}$-programs, and from Theorem 2 we have

COROLLARY 3. *The zero-equivalence problem for* $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y+w,\ y \leftarrow y-w,\ y \leftarrow y * w,\ y \leftarrow \lfloor y/w \rfloor,\ y \leftarrow \lfloor \sqrt{y} \rfloor\}$*-programs with one input variable is undecidable.*

4. Extension. We can use "forward" **if** statements in our straight-line programs and the results of §§ 2 and 3 still apply. Specifically, we can add the following constructs: **skip** l, **if** $p(y)$ **then skip** l (where l is a nonnegative integer), and **halt**. $p(y)$ is a predicate of the form $y > 0$, $y \geqq 0$, or $y = 0$, and **skip** l causes the $(l+1)$st instruction following the current instruction to be executed next. A program can terminate a computation in three ways: by executing a **halt** instruction, by executing a transfer to a nonexistent instruction, or by executing the last statement of the program.[4]

The following proposition shows that the **if** constructs are not independent. (The proof is given in the Appendix.)

PROPOSITION 6. *The instructions* **if** $y > 0$ **then skip** l *and* **if** $y \geqq 0$ **then skip** l *can be expressed in terms of the instruction* **if** $y = 0$ **then skip** l.

Notation. Let L be the instruction set $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y+w,\ y \leftarrow y-w,\ y \leftarrow y * w,\ y \leftarrow \lfloor y/w \rfloor$, **skip** l, **if** $p(y)$ **then skip** l, **halt**$\}$.

Referring now to the proof of Lemma 2, we see that in order to extend the proof to L-programs, we need only consider (by Proposition 6) two other cases.

Case 4. The $(r+1)$st instruction is **skip** l. Let $S(r+1+l, w) = S(r, w)$ for each variable w. Then continue the construction with instruction $r+l+2$.

Case 5. The $(r+1)$st instruction is **if** $y=0$ **then skip** l. Let x_0 be an input such that $x_0 \geqq 2^{2^{3(r+1)^2}}$. Suppose that on input x_0, y is defined at the end of r instructions and its value is uniquely defined by $(s(x), T)$ in $S(r, y)$. If $s(x)$ is not the zero polynomial, then let $S(r+1, w) = S(r, w)$ for each variable w and continue the construction with instruction $r+2$. If $s(x)$ is the zero polynomial, then let $S(r+1+l, w) = S(r, w)$ for each variable w and continue the construction with instruction $r+l+2$. The construction is completed when a **halt** instruction is encountered, or when a transfer to a nonexistent instruction is executed, or when the last instruction of the program has been considered.

[4] By convention, the program goes into an infinite loop when a division by 0 occurs.

Thus, Lemma 2 holds for the extended language, and all the results of §§ 2 and 3 apply. In particular, we have

THEOREM 4. *Let P_1 and P_2 be two L-programs with one input variable. Then P_1 and P_2 are equivalent if and only if they are equivalent on all inputs x such that $|x| \leqq 2^{2^{\lambda r^2}}$ (r is the maximum of the lengths of P_1 and P_2, and λ is a fixed positive constant).*

THEOREM 5. *No L-program can compute a pair generator.*

Appendix. Proofs of Propositions 1, 2, and 6.

Proof of Proposition 1. The following program which can easily be translated to a program over the instruction set $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y+w,\ y \leftarrow y-w,\ y \leftarrow y*w,\ y \leftarrow \lfloor y/w \rfloor\}$ computes $\langle x/y \rangle$:

$$w \leftarrow \lfloor 1/(4xy+2) \rfloor \qquad w = \begin{cases} 0 & \text{if } xy \geqq 0, \\ -1 & \text{if } xy < 0 \end{cases}$$

$$v \leftarrow \lfloor (x^2+w)/y^2 \rfloor - \lfloor x^2/y^2 \rfloor - w \qquad v = \begin{cases} 0 & \text{if } y \neq 0 \text{ and } w = 0, \\ 0 & \text{if } y \neq 0,\ w = -1, \text{ and } x \text{ is a multiple of } y \\ 1 & \text{if } y \neq 0,\ w = -1, \text{ and } x \text{ is not a multiple of } y \\ \text{undefined} & \text{if } y = 0 \end{cases}$$

$$x \leftarrow \lfloor x/y \rfloor$$

$$x \leftarrow x+v$$

□

Proof of Proposition 2. Programs P_1 and P_2 below (which can easily be transformed to programs over the instruction set $\{y \leftarrow 0,\ y \leftarrow 1,\ y \leftarrow y+w,\ y \leftarrow y-w,\ y \leftarrow y*w,\ y \leftarrow \langle y/w \rangle\}$) compute $\lfloor x/y \rfloor$ and $\lceil x/y \rceil$, respectively.

Program P_1

$$w \leftarrow \langle 3xy/(3xy+1) \rangle \qquad w = \begin{cases} 0 & \text{if } xy \geqq 0, \\ 1 & \text{if } xy < 0 \end{cases}$$

$$v \leftarrow \langle (x^2-w)/y^2 \rangle - \langle x^2/y^2 \rangle + w \qquad v = \begin{cases} 0 & \text{if } y \neq 0 \text{ and } w = 0, \\ 0 & \text{if } y \neq 0,\ w = 1, \text{ and } x \text{ is a multiple of } y, \\ 1 & \text{if } y \neq 0,\ w = 1, \text{ and } x \text{ is not a multiple of } y, \\ \text{undefined} & \text{if } y = 0. \end{cases}$$

$$x \leftarrow \langle x/y \rangle$$

$$x \leftarrow x-v$$

Program P_2

$$w \leftarrow \langle 5x/(4x+1) \rangle \qquad w = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{if } x \neq 0. \end{cases}$$

$$v \leftarrow \langle (x^2-w)/y^2 \rangle - \langle x^2/y^2 \rangle + w$$

$$x \leftarrow \lfloor x/y \rfloor$$

$$x \leftarrow x+v$$

□

Proof of Proposition 6. The constructions are straightforward. For example, **if** $y>0$ **then skip** l can be coded as

$u \leftarrow 1$

$v \leftarrow 1$

$v \leftarrow v+u$ $\quad v=2$

$z \leftarrow 0$

$z \leftarrow z+y$ $\quad z=y$

$z \leftarrow z*v$ $\quad z=2y$

$z \leftarrow z-u$ $\quad z=2y-1$

$w \leftarrow 0$

$w \leftarrow w+z$ $\quad w=2y-1$

$w \leftarrow \langle w/v \rangle$

$w \leftarrow w*v$

$z \leftarrow z-w$ $\quad z=\begin{cases} 1 & \text{if } y>0, \\ -1 & \text{if } y \leqq 0 \end{cases}$

$z \leftarrow z-u$ $\quad z=\begin{cases} 0 & \text{if } y>0, \\ -2 & \text{if } y \leqq 0 \end{cases}$

if $z=0$ **then skip** l

where u, v, w, and z are new variables. □

Acknowledgment. We would like to thank the referees for their suggestions and detailed comments which improved the presentation of our results.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] M. DAVIS, *Computability and Unsolvability*, McGraw-Hill, New York, 1958.

[3] M. DAVIS, Y. MATIJASEVIČ, AND J. ROBINSON, *Hilbert's tenth problem. Diophantine equations: Positive aspects of a negative solution*, Proc. Symp. Pure Mathematics, 28 (1976), pp. 323–378.

[4] F. HENNIE, *Introduction to Computability*, Addison-Wesley, Reading, MA, 1977.

[5] J. E. HOPCROFT AND J. D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[6] O. H. IBARRA AND B. S. LEININGER, *The complexity of the equivalence problem for straight-line programs*, Proc. 12th Annual ACM Symposium on Theory of Computing, 1980, pp. 273–280.

[7] H. ROGERS, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.