MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

Computation Structures Group Memo 79

Rabin's Proof of the Undecidability of the Reachability Set Inclusion Problem of Vector Addition Systems

by

Henry G. Baker, Jr.

This research was supported in part by the National Science Foundation under research grant GJ-34671, and in part by the Advanced Research Projects Agency of the Department of Defense under ARPA Order No. 433 which was monitored by ONR Contract No. N00014-70-A-0362-0001.

July 1973

Vector addition systems were invented by Michael Rabin in 1966 in order to investigate several questions about Petri nets which were posed to him by Richard Karp and Raymond Miller [Karp, Miller]. Although posed in a different formalism, some of the questions they wanted to answer were (1) Given an arbitrary Petri net, [Petri, Holt Commoner], is it live? (2) Given a particular marking of a Petri net, is it in the marking class? (3) Are the marking classes of two given Petri nets equal? (4) Given two Petri nets, is the marking class of the first a subset of the marking class of the second? Karp and Miller conjectured that marking classes of Petri nets were always semilinear sets, which were well known from the study of context free languages by Parikh and Ginsburg [Parikh, Ginsburg, Spanier]. Since semilinear sets are also isomorphic to Presburger sets -- those sets definable by formulae of the first order functional calculus over the non-negative integers with only "+" and "=" -- and Presburger arithmetic was shown to be decidable by Presburger in 1929, any elementary question about any semilinear set is decidable using Presburger's decision procedure [Presburger]. Rabin destroyed this conjecture by showing that semilinear sets are a proper subset of vector addition systems, as well as show that a particular aspect of Petri nets was undecidable, namely question (4) above.

Let N be the set of non-negative integers, Z be the set of integers, N^p be the set of p-tuples of non-negative integers, and Z^p be the set of p-tuples of integers, where p is a positive integer. We denote by N^+ the set of positive integers and by N_k^+ the set of positive integers less than or equal to k, i.e. $N_k^+ = \{i \in N \mid 1 \leq i \leq k\}$. We denote by the generic symbol $\bar{0}$ the p-tuple of zeros for any $p \in N^+$; the value of p will be clear from the context in which $\bar{0}$ is used. If $x, y \in Z^p$, then x is greater than or equal to y, (denoted $x \geq y$), if and only if $x_i \geq y_i$ for all components i, $1 \leq i \leq p$. Addition and subtraction of vectors will be componentwise.

Definition Let $I \in N^p$ be the initial vector (or origin) and $T \subseteq Z^p$ be a finite set of periods of the vector addition system (T,I). We define the reachability set of (T,I)--denoted by $RS(T,I)$:

$$RS(T,I) = \{ x \in N^p \mid \exists k \in N, \exists t_1 \in T, \exists t_2 \in T, \ldots, \exists t_k \in T \text{ such that } x = I + t_1 + t_2 + \ldots + t_k \text{ and } \forall i \leq k (I + t_1 + t_2 + \ldots + t_i \geq \bar{0}) \}$$

We construct a behavior graph for a VAS (T,I) from the reachability set RS(T,I) by adding arcs (x,y) from point x to point y labelled t, whenever x+t=y and $t \in T$. Finally, a period $t \in T$ is said to be enabled at a point x in RS(T,I) if $x+t \geq \bar{0}$.

Thus $x \in RS(T,I)$ is the sum of the origin plus a linear combination of periods in which all the partial sums are greater than or equal to $\bar{0}$. RS(T,I) can be constructed recursively by 1) the initial vector $I \in RS(T,I)$ and 2) $x \in RS(T,I)$ if and only if there exists a point $y \in RS(T,I)$ and a period $t \in T$ such that x=y+t. Intuitively, $x \in RS(T,I)$ means that x can be reached from I by a path of vectors selected from T which never leaves the first orthant (N^p). These paths of vectors are isomorphic to directed paths starting from I of the behavior graph of (T,I). Thus the terminal point of such a path is equal to I plus the sum of all the labels on the arcs of the path. It will be seen that vector addition systems are a natural generalization of linear sets to be defined later.

Definition A linear set is the reachability set of a vector addition system (T,I) in which the periods are all greater than or equal to zero; i.e. $T \subseteq N^p$. If $S = \bigcup_{i=1}^{r} RS(T_i, I_i)$ where (T_i, I_i) are linear sets of the same dimension p, $1 \leq i \leq r$, then S is called a semilinear set, because S is a finite union of linear sets.

These definitions are motivated by the fact that the expression $I+n_1t_1+n_2t_2+\ldots+n_kt_k$ ($k=|T|$), as well as all its partial sums, will always be in N^p, for any choice of the n_i's from N. Linear sets are the generalization of arithmetic sequences to many dimensions; these sets are periodic, hence the name periods for $t \in T$. The class of semilinear sets is closed under union, intersection, complementation with respect to N^p, and projection; it is also isomorphic to the class of Presburger sets -- those sets definable by well-formed formulae of the first order functional calculus over N, with the functions and predicates 0, +, =.

Projection

Given a relation R of n dimensions, i.e. a set of n-tuples of elements taken from some domain, it is often useful to consider projections of R onto a smaller dimensional subspace. However, if we consider any dimension reducing operation as a projection, there are at least two we would like to consider. The first meaning is that of an intersection with a smaller

dimensional subspace--for example, if our relation were the unit circle in the Euclidean plane, a projection of this kind would be the intersection of the x-axis with that circle, yielding exactly two points $\{-1,1\}$. This operation, which we will call sectioning, can be interpreted as a restriction of the original relation (i.e. a subset), followed by a throwing away of a coordinate from every tuple in the set to produce a section of the original relation.

The second type of projection we will consider is that of finding a domain of a relation. For example, a relation $R\subseteq N^5$ can also be thought of as a relation $R\subseteq(N^3)\times(N^2)$ over two domains N^3 and N^2 rather than over five domains. Thus, an algorithm which produces the set of triples which is the first domain of the relation $R\subseteq(N^3)\times(N^2)$ we will also consider to be a projection, but one which produces a domain. This projection is a very natural operation which we do all the time in solving problems. Sometimes we are given a relation $R\subseteq N^5$, for example, and we notice that really $R=R'\times N^2$ for some $R'\subseteq N^3$. This is just what happens when we realize that the last two variables in the problem are extraneous--i.e. independent of the other variables in the relation. We then project onto the first 3 coordinates to produce R', then solve the problem.

Definitions Given a relation $R\subseteq D^n$ over some domain set D and some positive integer n, a projection in the sense of domain, or simply a domain, of R will be a relation R' of dimension $m\leq n$ such that there exists a 1-1 function $s:N_m^+\to N_n^+$ called the selection vector, and

$$R' = \{\, x\in D^m \mid \exists y\in R \text{ such that } x_i=y_{s(i)},\ 1\leq i\leq m \,\}.$$

Given a relation $R\subseteq D^n$ for some domain set D and some positive integer n, a projection in the sense of section, or simply a section, of R will be a relation R' of dimension $m\leq n$ such that there exists a selection vector s and a restriction vector $r:(N_n^+\text{-range } s)\to D$ such that

$$R' = \{\, x\in D^m \mid \exists y\in R \text{ s.t. } x_i=y_{s(i)},\ 1\leq i\leq m \text{ and } y_j=r(j),\ j\in(N_n^+\text{-range } s) \,\}.$$

Semilinear sets are closed under both section projection and domain projection. Vector addition systems do not have such nice properties, however. But in some instances, section projection of reachability sets do still result in reachability sets of new vector addition systems. The next theorem gives one of these special cases and is not the most general theorem which could be proved in this context.

Theorem Given a p-dimensional vector addition system (T,I) for which there exists an integer k, $1 \leq k \leq p$, such that $I_k=0$ and there exist two periods $t,t' \in T$ such that 1) t,t' are the only periods in T which have non-zero values in coordinate k and 2) $t_k=-1$ and is the only negative coordinate of t, $t'_k=+1$ and is the only positive coordinate of t', and 3) the set of non-zero coordinates of t intersects the set of non-zero coordinates of t' only at k, then there exists a (p-1)-dimensional vector addition system (T',I') whose reachability set RS(T',I') is the section projection of RS(T,I) in which the kth coordinate was restricted to zero.

Proof:

Consider the VAS (T^-,I) where $T^-=T-\{t,t'\}\cup\{t+t'\}$. The kth coordinate of I and the kth coordinates of all the periods in T^- are zero. By induction, the kth coordinates of all the points in the reachability set $RS(T^-,I)$ must be zero. Every point in the set $RS(T^-,I)$ is included in the set RS(T,I). Assume that there exists a point $x \in RS(T^-,I)$ but not in RS(T,I). Consider the closest such point to I in terms of the number of vectors from T^- needed to reach x. Clearly x cannot be I, because I is also in RS(T,I). But every point along the path from I to x is also in RS(T,I), including x itself. This is because every period used in the path is either a period of T or the period t+t'. All we must show is that at the point y from which the period t+t' was used, the vector t could be used in RS(T,I) followed by the vector t'. Assume that t is not enabled at y. Then t+t' would also not be enabled at y, since enabling depends only upon the negative coordinates of t which all appear in t+t' due to the hypothesis. Therefore t can be used in RS(T,I) to reach a point y'=y+t which has a k coordinate of +1. Since the only negative coordinate of t' is the kth, t' is enabled and can be used to reach the next point on the path in $RS(T^-,I)$ to x. Thus, by induction on the length of the path, every point on the path in $RS(T^-,I)$ is in RS(T,I). Therefore $RS(T^-,I) \subseteq RS(T,I)$.

Finally, no point in the set $RS(T,I)-RS(T^-,I)$ has a zero in the kth coordinate. Assume that there exists a point x in $(RS(T,I)-RS(T^-,I))$ such that $x_k=0$. Then there exists a shortest path of vectors from I to x. Clearly, if t appears in that path ℓ times, then t' must also appear in

that path exactly k times, since t,t' are the only periods in T which have non-zero kth coordinates. But since x is not in $RS(T^-,I)$ it could not have been reached by a path from I using the other vectors and t+t'. No t' vector could appear in the path before the first t vector since t is the only vector which can make the kth coordinate non-zero. But at the point just following the application of the t vector, the first t' vector becomes enabled and persists until t' is finally applied. Therefore, we could have moved the first t' vector forward in the path in order to create a new path in which t' directly followed t. But then, we could have applied t+t' instead of t,t' and shortened the path. In this way, every t' vector can be moved next to a t vector, which shows that each pair could be replaced by t+t'. This argument shows that x is indeed reachable by a path of vectors from T^-, contradicting the assumption. Therefore $RS(T^-,I)$ is exactly the set RS(T,I) restricted to zero in the kth coordinate. Finally, since a section projection is simply a restriction followed by a domain projection, T' and I' are constructed by deleting the kth coordinates of the periods in T^- and the origin I. QED

Given an n-dimensional vector addition system (V,I), we can construct the corresponding Generalized Petri net by the method described in either [Keller] or [Hack]. Hack also shows that Generalized Petri nets have reachability sets (called marking classes in Petri net terminology) which are simple linear transformations of marking classes of normal Petri nets.

Most of the time we can also make the reverse construction: Given a Generalized Petri net, we can produce a vector addition system having the same reachability set as the marking class of the Petri net. The rest of the time, we must increase the dimension of the Petri net slightly, but again a simple linear transformation of the reachability set will give us back the marking class of the given Generalized Petri net [Hack].

The major question to be answered is this: Is RS(T,I) for a vector addition system (T,I) always a semilinear set? In general the answer is no. Before we can prove this proposition, however, we will need more machinery in order to program these reachability sets. We will look at a type of Turing machine studied by Minsky and others called register or counter machines (CM's) [Minsky]. These machines are composed of a finite state control coupled with n registers or counters, capable of holding any non-negative integer. The finite state control can increment or decrement a counter by 1, or test it for zero and branch. A CM is fed input by starting it with an input number in one of the counters, zero in the others; it leaves its output in one of the counters and may or may not clear the rest. Minsky has shown that there exists a 2-counter universal machine; that is, one capable of simulating any normal Turing machine made up of a finite state control and an infinite read-write tape [Minsky].

Counter machine finite state control units are programmed in an assembly type language consisting of only five types of instructions:

1) $c_i \leftarrow c_i+1$;
2) $c_i \leftarrow c_i-1$;
3) if $c_i=0$ then goto ℓ_1;
4) goto ℓ_1;
5) stop;

where c_i denotes the contents of counter i and ℓ_i denotes a label. These instructions are assembled into a sequence in which each label is replaced by a positive integer which specifies the position in the sequence at which the instruction with that label can be found. The special label "*+1" is always interpreted as "this instruction plus one". By convention, the CM is started on the first instruction in the sequence, called the initial instruction.

We will be looking at restricted non-deterministic counter machines (RNCM's) that are 1) restricted in that they cannot test for zero and are 2) non-deterministic in that they have a multiway branch instruction. Thus, the only instructions are:

1) $c_i \leftarrow c_i+1$;
2) $c_i \leftarrow c_i-1$;
3) goto $\ell_1, \ell_2, \ldots, \ell_k$;
4) stop;

Instructions of type 3) are interpreted as: split yourself into k copies and in copy i goto ℓ_i. If instructions of type 2) drive any counter negative, then this copy of the non-deterministic machine will fail--that is die or disappear. A simulation of a RNCM will produce a (possibly infinite) tree of instructions which were executed, the root of this tree being the initial instruction. A sequence of instructions listing all the nodes of this tree which lie on any finite path from the root is called a computation. If such a path ends by failing, then the sequence is a failing computation; if the path ends by stopping, then the sequence is an accepting computation. Any computation which is not accepting and not failing is called an incomplete computation.

If these machines were only deterministic RCM's, they would be quite stupid. In order to compute almost anything useful, they would have to test for zero, which they cannot do. However, Rabin noticed that if they were made non-deterministic, it would not be necessary for any particular copy to test for zero; if one copy failed, other copies could still go on to succeed. That this is so can be seen from the following example. We want to construct a 2-counter RNCM to copy a number from its first counter into its second counter. Our program for it will be:

```
test:goto loop,end;
loop:c1←c1-1;
     c2←c2+1;
     goto test;
end: stop;
```

At test, we do not really test c_1 for zero, but just non-deterministically guess whether we are done or not. If we guess that we are not done when $c_1=0$, then the instruction "loop" will fail; if we guess that we are done when $c_1>0$, then we will not have copied all of c_1 into c_2 by the time that we stop; if we guess that we are done just when $c_1=0$, then the program will have worked correctly (assuming it was intended to copy). This example is the motivation for our next definition.

Definition A counter configuration of an m-counter machine is an m-tuple $x=(c_1,c_2,\ldots,c_m)$. A function $f:N^n \to N$ is said to be weakly computable (w.c.) on a restricted non-deterministic m-counter machine F ($m \geq n$) if for every non-negative integer $k \leq f(x_1,x_2,\ldots,x_n)$ there exists a computation by F which leads from the counter configuration $X=(x_1,x_2,\ldots,x_n,0,\ldots,0)$ to $U=(u_1,u_2,\ldots,u_n,k,u_{n+2},\ldots,u_m)$ and conversely a computation from X to U implies that $k \leq f(x_1,x_2,\ldots,x_n)$. Generalizing, a function $g:N^n \to N^k$ is weakly computable on a restricted non-deterministic m-counter machine G ($m \geq n+k$) if for every tuple of non-negative integers $(k_1,k_2,\ldots,k_k) \leq g(x_1,x_2,\ldots,x_n)$, there exists a computation by G from $X=(x_1,x_2,\ldots,x_n,0,\ldots,0)$ to $U=(u_1,u_2,\ldots,u_n,k_1,k_2,\ldots,k_k,u_{n+k+1},\ldots,u_m)$ and conversely: a computation from X to U implies that $(k_1,k_2,\ldots k_k) \leq g(x_1,x_2,\ldots,x_n)$.

Thus, in our example above, the machine weakly computed the identity function. It should be easy to see that only monotonic increasing functions are weakly computable; we will be considering only monotonic increasing polynomial functions.

Theorem If $f(x_1,x_2,\ldots,x_n)$ is a polynomial with positive integer coefficients, then f is weakly computable by some RNCM F.

Proof: We show how to write a program F which weakly computes f. The proof will be by induction on the structure of the polynomial expression for f; sums and products are weakly computable and the composition of monotonic increasing weakly computable functions is weakly computable.

Let $g(x,y)=x+y$ and $h(x,y)=x \cdot y$.

Addition is weakly computable

$g(c_1,c_2) \to c_3$:

```
loopx:  goto loopy,*+1;          guess whether c1=0
        c1←c1-1;                 pump c1 into c3
        c3←c3+1;
        goto loopx;
loopy:  goto end,*+1;            guess whether c2=0
        c2←c2-1;                 pump c2 into c3
        c3←c3+1;
        goto loopy;
end:    goto next_function;
```

i.e. c_3 is the result of pumping 1's out of c_1 for a while, then pumping 1's out of c_2 for a while; the total number of 1's in c_3 is $\leq c_1+c_2$.

Multiplication is weakly computable

```
h(c1,c2)→c4:
loopback:goto end,Decx,*+1;
         c3←c3-1;                pump c3 back into c2
         c2←c2+1;
         goto loopback;
Decx:    c1←c1-1;                c1 controls the number of iterations of
loopfor: goto loopback,*+1;      loopforward.
         c2←c2-1;                pump c2 into c3 and c4
         c3←c3+1;
         c4←c4+1;
         goto loopfor;
end:     goto next_function;
```

i.e. pump c_2 into c_3,c_4; pump c_3 into c_2; repeat whole sequence c_1 times; total number of 1's pumped into c_4 is $\leq c_1 \cdot c_2$.

Composition of weakly computable functions is weakly computable.

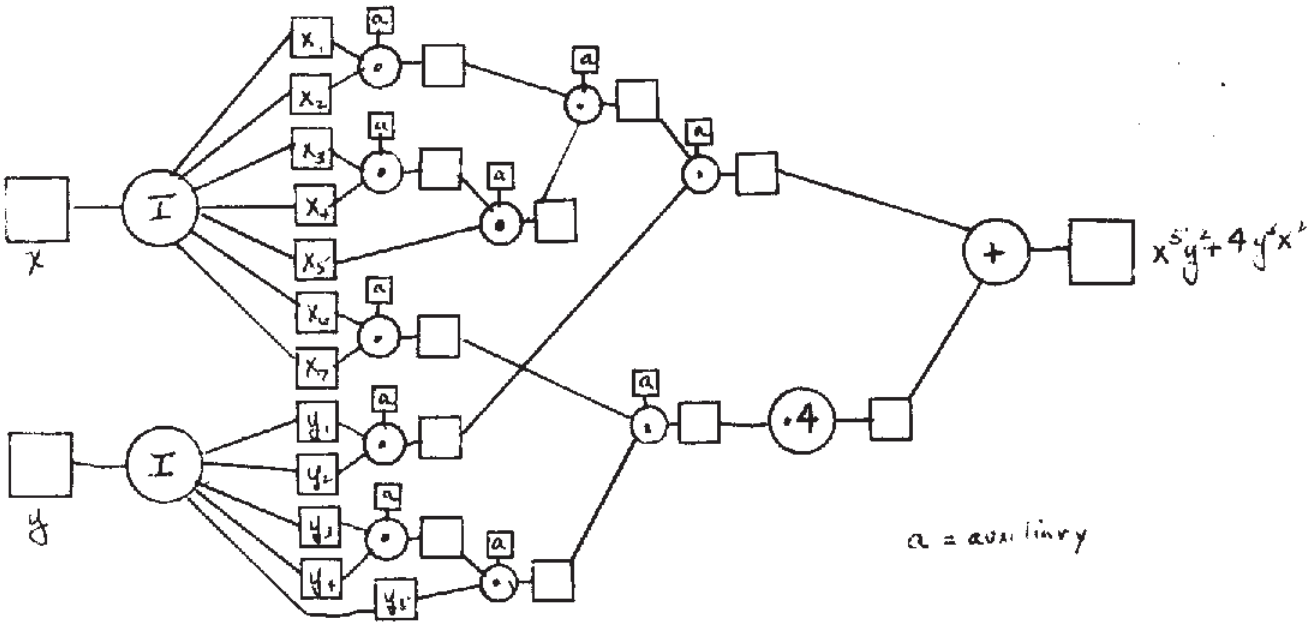
Consider the polynomial $P(x,y)=x^5y^2+4y^3x^2$. Since we will require 7 copies of x, perform weak identity:

```
copyloop:goto end,*+1;
         x←x-1;                  deflate x
         x1←x1+1;                while inflating x1,x2,...,x7
         x2←x2+1;
         ...
         x7←x7+1;
         goto copyloop;
end:     goto next_function;
```

Similarly, make 5 weak copies of y. Suppose y^3x^2 has been weakly computed into counter z. We can weakly compute $z'=4y^3x^2=4z$ by the following program:

```
mult4:   goto end,*+1;
         z←z-1;                  deflate z but
         z'←z'+1;                inflate z' 4 times as much
         z'←z'+1;
         z'←z'+1;
         z'←z'+1;
         goto mult4;
end:     goto next_function;
```

We can use our definitions of addition and multiplication as schemas or macros in order to weakly compute P(x,y) using 36 registers (2 input, 1 output, 23 intermediate result, and 10 auxiliary registers for multiplications).



Given this example, it should be easy to see how to program any polynomial with positive integer coefficients on these machines. QED

If we define the weak graph of a function $f:N^n \to N$, denoted G_f, to be the set $G_f = \{x \in N^{n+1} \mid x_{n+1} \leq f(x_1, x_2, \ldots, x_n)\}$ then the following corollary proves that the weak graph of a weakly computable function is computable

by a restricted non-deterministic counter machine started with all counters clear, where computable here means that the domain projection of the set of counter configurations onto the first n+1 coordinates is exactly the set G_f.

Corollary The weak graph of a weakly computable function is computable on a restricted non-deterministic counter machine started with all counters zero.

Proof: Given a weakly computable function $f:N^n \to N$, using the theorem we can produce an m-counter RNCM F which weakly computes that function. We show how to modify F in order to compute the weak graph of f. Assume that F is given its arguments $\bar{x}$ in the first n counters and weakly computes $f(\bar{x})$ in its (n+1)st counter. We add n new counters and relable all the counters as follows: the new counters become the first n counters, the values of $f(\bar{x})$ will be computed in the n+1st counter, the input $\bar{x}$ will be placed in the next n counters, and the rest of the counter will follow. Since the machine will be started with all counters zero, we must insert instructions at the beginning of F's program in order to guess a value $\bar{x}$ with which the machine will weakly compute $f(\bar{x})$. While the machine is guessing the vector $\bar{x}$, it makes two copies: the first copy in the first n counters and the second in the n+2nd to 2n+1st counters. Once the machine has guessed the value $\bar{x}$, it transfers to the program which actually does the weak computation of $f(\bar{x})$. This program will destroy the second copy of $\bar{x}$ in order to compute $f(\bar{x})$, but the first copy is left in the first n counters untouched. Thus, at all times, the first n coordinates will contain a value $\bar{x}$ and the n+1st coordinate will contain a value $\leq f(\bar{x})$. Furthermore, for all $\bar{x}$, there exists a computation which actually produces the value $f(\bar{x})$ in the n+1st coordinate.

The inserted instructions at the beginning of the program look like this:

F':

```
x1:     goto x2,*+1;          stop incrementing x1?
        c1←c1+1;              pump up 1st copy
        cn+2←cn+2+1;          and 2nd copy same amount
        goto x1;
```

```
x_2:    goto x_3,*+1;          stop incrementing x_2?
        c_2 ← c_2+1;
        c_{n+3} ← c_{n+3}+1;
        goto x_2;
x_3:
        ...
x_n:    goto end,*+1;
        c_n ← c_n+1;
        c_{2n+1} ← c_{2n+1}+1;
        goto x_n;
end:    goto F;                compute f(c_{n+2},c_{n+3},...,c_{2n+1})
```

It should be clear that for any $\bar{x}\in N^n$, there exist computations of F' which will weakly compute $f(\bar{x})$ in counter n+1. Thus F' computes the weak graph of f. QED

We now show how vector addition systems can completely mimic the calculations of an RNCM.

Theorem For every restricted non-deterministic counter machine C with m counters and ℓ instructions, there exists a finite set of periods $T_C \subseteq Z^{m+\ell}$ such that for every input $\bar{k}=(k_1,k_2,\ldots,k_m)\in N^m$ to C, there exists an $I_{C,\bar{k}}\in N^{m+\ell}$, where $I_{C,\bar{k}}=(k_1,k_2,\ldots,k_m,1,0,\ldots,0)$, such that the reachability set $RS(T_C,I_{C,\bar{k}})=\{(b_1,b_2,\ldots,b_m,-,-,\ldots,-)\in N^{m+\ell} \mid (b_1,b_2,\ldots,b_m)$ is a counter configuration of C$\}$; i.e. the domain projection of $RS(T_C,I_{C,\bar{k}})$ onto the first m coordinates.

Proof: We must show how to code C in the vector addition system $(T_C,I_{C,\bar{k}})$. Since we are given the coding for the initial vector $I_{C,\bar{k}}$, we now show how to obtain the period vectors $t_i\in T_C$. The basic idea is this: each point x in the reachability set will be an instantaneous description (i.d.) of the state of a particular copy of C in the midst of a particular computation, but between two instructions. We code the contents of the first m counters of C as the values of the first m coordinates of x. We code the state of the finite state control of C--i.e. the program counter--into the last ℓ coordinates of x. These last ℓ coordinates will be used to mimic the program counter for our simulated computation; each coordinate corresponds

to a particular instruction in the program in a one-for-one manner. Therefore, we call the last ℓ coordinates program counter coordinates. Since only one instruction will be enabled at any point during the simulation of any one copy of C, only one of the ℓ program counter coordinates will ever be non-zero, for any x in the reachability set. A 1 in the ith program counter coordinate of a point x in $RS(T_{C,}, I_{C,\bar{k}})$ indicates that the copy of the RNCM whose i.d. is coded by x is initiating the ith instruction of C.

Each instruction of types 1), 2), and 4) of C will be coded in a one-for-one manner as a single vector in T_C. An instruction of type 3) oc C is coded as k different vectors in T_C, where k is the number of labels in that branch instruction. When the jth instruction is to be initiated, we will be at an i.d. $x=(x_1,x_2,\ldots,x_m,0,0,\ldots,1,0,\ldots,0)$ where the 1 appears in the jth program counter coordinate. Therefore, if $j:c_k \leftarrow c_k-1$; then we code the corresponding vector t_j: $(0,\ldots,-1,0,\ldots,0;0,\ldots,-1,1,0,\ldots,0)$ where the first -1 appears in position k in order to decrement c_k and the -1,1 appear in the jth and j+1st program counter coordinates so that the -1 can turn off j's initiating flag and the +1 can turn on (j+1)'s initiating flag--thus transfering control to the next sequential instruction. If the value x_k in the i.d. x is already zero, neither t_j nor any other period will be enabled at this i.d. (since all other periods have -1's in positions corresponding to 0's in x's program counter coordinates), therefore the simulation would hang up--i.e. the computation would fail. If $j:c_k \leftarrow c_k+1$; then we would code the vector t_j:$(0,\ldots,+1,0,\ldots,0;0,\ldots,-1,1,0,\ldots,0)$ where the first +1 appears in position k in order to increment c_k and the -1,1 appear in the jth and (j+1)st p.c.c.'s. If j:stop; then we would code the vector t_j:$(),\ldots,0;0,\ldots,-1,0,\ldots,0)$ where the -1 appears in the jth p.c.c. in order to turn off j's initiation flag. Since t_j does not turn on any other initiation flag, the simulation must stop; this situation is different from the failing situation because when a computation stops on purpose, all of the p.c.c.'s in that i.d. are zero. Finally, if j:goto $\ell_1,\ell_2,\ldots,\ell_k$; then we code k different vectors t_{ji}:$(0,\ldots,0;0,\ldots,-1,0,\ldots,0,+1.0,\ldots,0)$ where the -1 turns off j's initiation flag and the +1 turns on ℓ_i's initiation flag thus transfering control to ℓ_i. The k different vectors

are all enabled when j's flag comes on, but they are all mutually exclusive in the sense that once one of them is chosen for the simulation, the others are thereby disabled.

If we consider the behavior graph of $(T_C, I_{C,\bar{k}})$, it should be clear that since every node is an instantaneous description of C and every arc is the execution of an instruction of C, then any directed path from I through this graph is isomorphic to the simulation of a particular copy of C and thus is isomorphic to a computation of C. QED

Corollary There exists a reachability set RS(T,I) which is not semilinear.
Proof: Construct a restricted courter machine which weakly calculates $y=x^2$. Mimic this machine with a VAS (T,I). If its reachability set were semilinear, then its domain projection onto the first two coordinates would also be semilinear. But $\{(x,y)\in N^2 \mid y\leq x^2\}$ is not a semilinear set, which is a contradiction. Therefore, the reachability set RS(T,I) is not semilinear. QED

Theorem (The Reachability Set Inclusion Problem) The problem to decide whether $RS(T_1,I_1) \stackrel{?}{=} RS(T_2,I_2)$ where (T_i,I_i) are p-dimensional vector addition systems is not recursively solveable.
Proof: We show that Hilbert's tenth problem is reducible to this inclusion problem [Hilbert]. Since Hilbert's tenth problem is known to be recursively undecidable, this set inclusion problem is also recursively undecidable [Davis, Putnam, Robinson; Matijasevic].

Hilbert's tenth problem is whether there exists a decision procedure for finding the integral roots of polynomials with integral coefficients. That is, given a polynomial $P(x_1,x_2,\ldots,x_n)$ with integral coefficients, do there exist integers $x_1,x_2,\ldots,x_n$ such that $P(\bar{x})=0$? Given a decision procedure for the existence of non-negative integers x_i such that $P(\bar{x})=0$, we could solve the general problem simply by calling this procedure 2^n times and asking $P(x_1,x_2,\ldots,x_n) \stackrel{?}{=} 0$, $P(x_1,x_2,\ldots,x_{n-1},-x_n) \stackrel{?}{=} 0$, $P(x_1,\ldots,-x_{n-1},x_n) \stackrel{?}{=} 0$, and so on until $P(-x_1,-x_2,\ldots,-x_n) \stackrel{?}{=} 0$. Furthermore, since $P^2(\bar{x})$ and $P(\bar{x})$ both have the same (numerical) roots. We never need to produce negative numbers in the evaluation of a polynomial in the process of solving Hilbert's problem. Finally,

if we separate the positive and negative coefficients of $P^2(\bar{x})$ such that $A(\bar{x})$ is the polynomial with only the positive coefficients and $-B(\bar{x})$ is the polynomial with only the negative coefficients, then $P^2(\bar{x})=A(\bar{x})-B(\bar{x})$ and A,B both have only positive coefficients. But $P^2(\bar{x})\geq 0$; therefore $A(\bar{x})-B(\bar{x})\geq 0$ and $A(\bar{x})\geq B(\bar{x})$. But a vector $\bar{x}\in N^n$ such that $P^2(\bar{x})=0$ is also a solution of $A(\bar{x})=B(\bar{x})$ and vice versa. So if for all $\bar{x}\geq\bar{0}$, $A(\bar{x})\geq B(\bar{x})+1$, then there is <u>no</u> solution to $A(\bar{x})=B(\bar{x})$ or $P^2(\bar{x})=0$ and if for some $\bar{x}$, $A(\bar{x})<B(\bar{x})+1$ then $A(\bar{x})=B(\bar{x})$, $P^2(\bar{x})=0$ and $P(\bar{x})=0$. Thus, we can reduce Hilbert's tenth problem to the question $\exists\bar{x}\in N^n[A(\bar{x})<B(\bar{x})+1]$? for non-negative coefficients in $A(\bar{x})$ and $B(\bar{x})$. Furthermore, Hilbert's problem reduces to the question $G_A \overset{?}{=} G_{B+1}$ since any point $\bar{z}=\bar{x}_0 x y$ in the graph of $B(\bar{x})+1$ but <u>not</u> in the graph of $A(\bar{x})$ would satisfy the criterion $A(\bar{x})<y$ where $y=B(\bar{x})+1$.

But given $A(\bar{x})$ and $B(\bar{x})$, we can compute the weak graphs G_A and G_{B+1} on our restricted non-deterministic counter machines. Call the RNCM for G_A "A" and the RNCM for G_{B+1} "B". Now Hilbert's problem reduces to the problem of determining whether the set of counter configurations of A, denoted by CC_A, domain projected onto the first n+1 coordinates, includes the set CC_B, projected onto the first n+1 coordinates. However, we do not have to perform the projection outside of the RNCM's; they can do it for us! Suppose that A has $n+1+m_A$ total counters and B has $n+1+m_B$ total counters. We add to the machine with the smaller number of counters, enough <u>dummy</u> counters so that they both have $n+1+m$ counters, where $m=\max(m_A,m_B)$. A <u>dummy counter</u> is initialized to zero and is never incremented by the program. How can we "project out" the m scratch counters from the set of counter configurations of an RNCM, say A. To project out a coordinate, we make it <u>independent</u> of the other coordinates. This can be done by adding a <u>coda</u> to A's program which "liberates" the m scratch counters:

```
Coda:
DecS1:   goto incS1,*+1;
         c_{n+2} ← c_{n+2} -1;          decrement scratch counter 1 for a while
         goto DecS1;
IncS1:   goto DecS2,*+1;
         c_{n+2} ← c_{n+2} +1;          increment scratch counter 1 for a while
         goto IncS1;
DecS2:
    ...
IncSm:   goto stop,*+1;
         c_{n+1+m} ← c_{n+1+m} +1;      increment scratch counter m for a while
         goto IncSm;
stop:    stop;
```

It can be seen that for any combination of values $\bar{s} \in N^m$, there exists a computation of the coda, that stops with $\bar{s}_i$ in c_{n+1+i}. Thus, the set of counter configurations of A', where A' is the RNCM A with the coda added, is equal to $G_A \times N^m$; i.e. $CC_{A'} = G_A \times N^m$. Similarly, we can construct an RNCM B' which appends a coda to the program for B, which liberates B's scratch counters. Thus $CC_{B'} = G_{B+1} \times N^m$. Clearly, $G_A \times N^m \supseteq G_{B+1} \times N^m$ if and only if $G_A \supseteq G_{B+1}$. Thus, we have reduced Hilbert's problem to the inclusion problem of counter configuration sets of restricted non-deterministic counter machines.

But we are trying to prove that the inclusion problem for reachability sets is undecidable. Using the techniques of the previous theorem, we can construct a vector addition system $(T_{B'}, I_{B'})$ of dimension $n+1+m+\ell_{B'}$ corresponding to the RNCM B'. We then add $\ell_{A'}$ zero coordinates to $I_{B'}$ and all the periods $t \in T_{B'}$ to construct $(T'_{B'}, I'_{B'})$ of dimension $n+1+m+\ell_{B'}+\ell_{A'}$. Since the last ℓ_A coordinates of $I'_{B'}$ are zero and the last $\ell_{A'}$ coordinates of $t \in T'_{B'}$ are zero, then for any $x \in RS(T'_{B'}, I'_{B'})$ the last $\ell_{A'}$ coordinates of x are zero.

Before converting A' into a VAS, we must do more surgery to produce the RNCM A" which has $m+\ell_{B'}$ scratch counters. This can be done by simply adding $\ell_{B'}$ more dummy counters which are never referenced in the body of the weak computation for $A(\bar{x})$, but which will be liberated along with the rest of the scratch counters in the coda for A". Hilbert's problem

can now be reduced to the set inclusion problem: $CC_{A''} \stackrel{?}{\supseteq} RS(T_{B'}, I_{B'})$ since any $x \in RS(T_{B'}, I_{B'})$ which is not in $CC_{A''}$ must be such that $A(x_1, x_2, \ldots, x_n) < x_{n+1}$ since $CC_{A''} = G_A \times N^{m+\ell_{B'}}$ due to the liberation of the last $m+\ell_{B'}$ counters. Finally, A" is converted into a VAS $(T_{A''}, I_{A''})$ of exactly $n+1+m+\ell_{B'}+\ell_{A''}$ dimensions by the standard technique of the theorem. Now the set of $x \in RS(T_{A''}, I_{A''})$ such that the last $\ell_{A''}$ coordinates of x are zero, is just the set of instantaneous descriptions in which A" has halted. If we define this set $H = \{x \in RS(T_{A''}, I_{A''}) \mid x_i = 0,\ n+1+m+\ell_{B'}+1 \leq i \leq n+1+m+\ell_{B'}+\ell_{A''}\}$, then $H \subseteq RS(T_{A''}, I_{A''})$. But $CC_{A''}$ is just that <u>section projection</u> of $RS(T_{A''}, I_{A''})$ for which the last $\ell_{A''}$ coordinates are zero; i.e. $CC_{A''}$ is the domain projection of H. But $RS(T'_{B'}, I'_{B'}) = RS(T_{B'}, I_{B'}) \times 0^{\ell_{A''}}$ and $H = CC_{A''} \times 0^{\ell_{A''}}$, therefore $CC_{A''} \supseteq RS(T_{B'}, I_{B'})$ if and only if $H \supseteq RS(T'_{B'}, I'_{B'})$. Finally, since $(RS(T_{A''}, I_{A''}) - H) \cap RS(T'_{B'}, I'_{B'}) = \emptyset$, $H \supseteq RS(T'_{B'}, I'_{B'})$ if and only if $RS(T_{A''}, I_{A''}) \supseteq RS(T'_{B'}, I'_{B'})$. We have finally reduced Hilbert's tenth problem to the inclusion problem for reachability sets, therefore the undecidability of the first implies the undecidability of the second. QED

REFERENCES

Davis, M., H. Putnam, J. Robinson. The Decision Problem for Exponential Diophantine Equations. Annals of Math. Vol. 74 No. 3, Nov. 1961, pp 425-436.

Ginsberg S. and E. H. Spanier Semigroups, Presburger Formulas, and Languages. Pacific J. Math. Vol. 16 1966 pp 285-296.

Hack, M. The Equivalence of Generalized Petri Nets and Ordinary Petri Nets. Computation Structures Note 9, Project MAC, M.I.T., April 1973

Hilbert, D. Mathematische Probleme. Vortag, gehalten auf dem internationalen Mathematiker-Kongress zu Paris 1900. Nachr. K. Ges. Wiss. Göttingen, Math.-Phys. Kl. 1900, pp 253-297. Translation: Bull. Amer. Math. Soc., 8(1901-1902), pp 437-479.

Holt, A. W., and F. Commoner Events and Conditions. Applied Data Research, New York 1970.

Karp, R. M., and Miller, R. E. Parallel program schemata: A mathematical model for parallel computation. IEEE Conference Record, Eighth Annual Symposium on Switching and Automata Theory, Oct. 1967, pp 55-61.

Keller, R. M. Vector Replacement Systems: A Formalism for Modeling Asynchronous Systems. Tech. Rep.117, Princeton University Computer Science Laboratory, Dec.1972.

Matijasevic, Ju. V. Enumerable Sets are Diophantine. Soviet Math. Dokl. Vol. 11 (1970), No. 2, pp 354-357.

Minsky, M. Computation: Finite and Infinite Machines. Prentice-Hall, Inc. Englewood Cliffs, N. J. 1967, pp 255-258.

Parikh, R. J. Language generating devices, M.I.T. Research Lab. Electronics Quart. Progress Report 60, 1961, pp 199-212.

Petri, C. A. Communication with Automata. Supplement 1 to Tech. Rep. RADC-TR-65-377, Vol. 1, Griffiss Air Force Base, New York 1966. Originally published in German: Kommunikation mit Automaten, University of Bonn, 1962.

Presburger, M. Über die Vollstandigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchen die Addition als einzige Operation hervortritt. Comptes Rendus du Ier Congries des Mathematiciens des pays Slaves. Warsaw 1929, pp 92-101,395.