SOME COMPLEXITY BOUNDS FOR PROBLEMS CONCERNING FINITE AND 2-DIMENSIONAL VECTOR ADDITION SYSTEMS WITH STATES

Rodney R. Howell, Dung T. Huynh,
Louis E. Rosier, and Hsu-Chun Yen

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-86-08 March 1986

Some Complexity Bounds for Problems Concerning Finite and 2-Dimensional Vector Addition Systems with States

Rodney R. Howell[1], Dung T. Huynh[2], Louis E. Rosier[1], and Hsu-Chun Yen[1]

Abstract

In this paper, we analyze the complexity of the reachability, containment, and equivalence problems for two classes of vector addition systems with states (VASSs): finite VASSs and 2-dimensional VASSs. Both of these classes are known to have effectively computable semilinear reachability sets (SLSs). By giving upper bounds on the sizes of the SLS representations, we achieve upper bounds on each of the aforementioned problems. In the case of finite VASSs, the SLS representation is simply a listing of the reachability set; therefore, we derive a bound on the norm of any reachable vector based on the dimension, number of states, and amount of increment caused by any move in the VASS. The bound we derive shows an improvement of two levels in the primitive recursive hierarchy over results previously obtained by McAloon, thus answering a question posed by Clote. We then show this bound to be optimal. We feel that the techniques we use in deriving our upper bounds represent an original approach to the problem, and since they yield improvements over previous results, we feel these techniques may have applications to other problems. In the case of 2-dimensional VASSs, we analyze an algorithm given by Hopcroft and Pansiot that generates a SLS representation of the reachability set. Specifically, we show that the algorithm operates in $2^{2^{c*l*n}}$ nondeterministic time, where l is the length of the binary representation of the largest integer in the VASS, n is the number of transitions, and c is some fixed constant. We also give examples for which this algorithm will take $2^{2^{d*l*n}}$ nondeterministic time for some positive constant d. Finally, we give a method of determinizing the algorithm in such a way that it requires no more than $2^{2^{c*l*n}}$ deterministic time. From this upper bound and special properties of the generated SLSs, we derive upper bounds of DTIME($2^{2^{c*l*n}}$) for the three problems mentioned above.

1. Introduction

The containment and equivalence problems for vector addition systems (VASs) (or equivalently vector addition systems with states (VASSs) or petri nets) are, in general, undecidable [2, 11]. They are decidable, however, for classes of VASs (VASSs, petri nets) whose reachability sets are effectively computable semilinear sets (SLSs). Such classes include finite VASs [19], 3-dimensional VASs [34], 5-dimensional VASs (or, equivalently 2-dimensional VASSs) [12], conflict free VASs [7], persistent VASs [9, 22, 23, 28], weakly persistent VASs [36], and regular VASs [10, 35]. For each of these classes, the algorithm which generates the SLS representation of the reachability set is a search procedure that is guaranteed to terminate. However, no analysis of when termination will occur is provided, and thus no complexity results are obtained. The perhaps best studied class is that of symmetric VASs. For this class

[1]Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712.

[2]Department of Computer Science, Iowa State University, Ames, IA 50011.

the equivalence and reachability problems have been shown to be exponential space complete [6, 14, 27]. The best known lower bound for the general reachability problem is exponential space [21]. Few other complexity results appear to be known.

In this paper, we concern ourselves with examining the complexity of the containment and equivalence problems for two classes of VASSs -- finite VASSs and 2-dimensional VASSs. Recently, Mayr and Meyer [26] showed that the containment and equivalence problems for finite VASs are not primitive recursive. Subsequently, McAloon [25] showed that the problems are primitive recursive in the Ackermann function, and Clote [5] showed the finite containment problem to be DTIME(Ackermann) complete. Let $f_1(x)=2x$ and $f_n(x)=f_{n-1}^{(x)}(1)$ for $n>1$, where $f_i^{(m)}$ is the m-th fold composition of f_i. Using a combinatorial argument, McAloon showed an upper bound for the time complexity of the finite-containment problem that can be shown to be at least $f_{k+1}(m)$, where k is the dimension and m is the maximum sum of the elements of any vector in the VAS (see also [5]). Clote [5] subsequently used Ramsey theory to give an upper bound of approximately $f_{k+6}(m)$ and posed a question as to whether McAloon's bound could be improved. It follows that these bounds also hold for the size of finite VASs. McAloon's bound on the size of finite VASs is close to optimal. See [23, 26, 29, 35].

Let BV(k,b,n) be the class of k-dimensional n-state finite VASSs where the maximum increase in the norm of a vector (i.e., the sum of the absolute values of its elements) caused by any move is b. (Assume the start vector is **0**.) In Section 2, we use a tree construction technique to derive an upper bound on the largest norm of any vector reachable in BV(k,b,n). We do this by first examining the problem for finite VASs (i.e., systems without states) where the start vector is not required to be **0**. We then extend the results to BV(k,b,n). The bound we derive for k-dimensional VASs is $f_{k-1}(d*m^2)$, $k \geq 2$, ($f_{k-1}(d*m)$ for $k \geq 4$), where m is the maximum sum of the elements of any vector in the VAS, and d is a constant. By then considering the addition of states and the restriction of the start vector to **0**, we derive a bound of $f_k(c*\max(n,\log b))$ on the norm of the largest vector reachable in BV(k,b,n), where $k \geq 3$ and c is a constant. Furthermore, we show that this bound is tight for b=1. (I.e., we illustrate for each k and m a VASS in BV(k,1,m*(2*k-1)+2) that can generate a vector with norm $f_k(m)$.) These results immediately yield, for the k-dimensional VAS finite containment problem, a bound of $f_{k-1}(d'*m)$ time, for $k \geq 4$ and some constant d'. This bound represents an improvement of two levels in the primitive recursive hierarchy over McAloon's result, thus answering the question posed by Clote. Since we do not know of any attempts to use tree construction techniques similar to ours in analyzing combinatorial problems, and because our techniques yield better results than the standard combinatorial techniques applied in the past to this problem, we surmise that our techniques may have other applications. Finally, we show that the containment and equivalence problem (for BV(k,b,n)) require at least time $f_{k-c}(d*n)$ infinitely often for some constants c and d. The proof is such that each position in the constructed VASS can be bounded by $f_{k-c}(d*n)$. Hence, if we considered the entire class of VASSs whose positions were bounded by $f_k(n)$ (rather than just BV(k,b,n)) our lower bound would be tight. We surmise, therefore, that the constant c can be eliminated. The section concludes by examining special cases where k is fixed. For example, the problems for BV(1,1,n) are shown to be complete for $A\Pi^L_2$ -- the second level of the alternating logspace hierarchy [4].

In Section 3, we utilize the ideas inherent in the previous section to provide an analysis of the algorithm given in [12], which generates from an arbitrary 2-dimensional VASS the SLS representation of its reachability set. As a result of the analysis, we obtain upper bounds for the containment, equivalence and reachability problems in the case of 2 dimensions. Let VASS(2,l,n) denote the class of 2-dimensional VASSs whose integers can each be represented in l bits and such that n is the maximum of the number of states and the number of transitions. Specifically, we show that the algorithm of Hopcroft and Pansiot [12] operates on any VASS in VASS(2,l,n) in NTIME($2^{2^{c*l*n}}$) for some constant c. Furthermore, there are instances that require $2^{2^{d*l*n}}$ steps for some positive constant d; hence our analysis (of the algorithm) is tight. We then give a minor modification to the algorithm that reduces its complexity to DTIME($2^{2^{c'*l*n}}$) for some constant c′. The SLS constructed by the resulting algorithm contains O($2^{2^{c'*l*n}}$) linear sets. Each of these linear sets has a base with norm O($2^{2^{c'*l*n}}$) and O(2^n) periods with norm O($2^{d'*l*n}$) for some constant d′. From these properties we derive an upper bound of DTIME($2^{2^{c*l*n}}$) for the reachability, equivalence, and containment problems for VASS(2,l,n). Now the best known lower bounds for these problems are significantly smaller (e.g., NLOGSPACE (NP) for the reachability problem of VASS(2,1,n) (VASS(2,l,n)) [30]). Hence, there is still much room for improvement. However, the two algorithms for the general reachability problem in [20, 24] do not appear to yield better upper bounds for 2-dimensional VASSs. Hence, whether or not these bounds can be tightened we leave as an open question.

2. Finite VASSs

Let Z (N, N^+, R) denote the set of integers (nonnegative integers, positive integers, rational numbers, respectively), and let Z^k (N^k, R^k) be the set of vectors of k integers (nonnegative integers, rational numbers). For a vector $v \in Z^k$, let v(i), $1 \leq i \leq k$, denote the i-th component of v. For a given value of k, let **0** in Z^k denote the vector of k zeros (i.e., **0**(i)=0 for i=1, . . . ,k). Now given vectors u, v, and w in Z^k we say:

- v=w iff v(i)=w(i) for i=1, . . . ,k;
- $v \geq w$ iff $v(i) \geq w(i)$ for i=1, . . . ,k;
- v>w iff $v \geq w$ and $v \neq w$;
- u=v+w iff u(i)=v(i)+w(i) for i=1, . . . ,k.

A k-dimensional *vector addition system* (VAS) is a pair (v_0,A) where v_0 in N^k is called the *start vector*, and A, a finite subset of Z^k, is called the set of *addition rules*. The *reachability set* of the VAS (v_0,A), denoted by $R(v_0,A)$, is the set of all vectors z, such that $z=v_0+v_1+\cdots+v_j$ for some $j \geq 0$, where each v_i $(1 \leq i \leq j)$ is in A, and for each $1 \leq i \leq j$, $v_0+v_1+\cdots+v_i \geq \mathbf{0}$. A k-dimensional *vector addition system with states* (VASS) is a 5-tuple (v_0,A,p_0,S,δ) where v_0 and A are the same as defined above, S is a finite set of states, δ $(\subseteq S \times S \times A)$ is the transition relation, and p_0 is the initial state. Elements (p,q,x) of δ are called *transitions* and are usually written $p \rightarrow (q,x)$. A *configuration* of the VASS is a pair (p,x), where p is in S and x is a vector in N^k. (p_0,v_0) is the initial configuration. The transition $p \rightarrow (q,x)$ can be applied to the configuration (p,v) and yields the configuration $(q,v+x)$, provided that $v+x \geq \mathbf{0}$. In this case, $(q,v+x)$ is said to *follow* (p,v). Let σ_0 and σ_t be two configurations. Then σ_t is said to be *reachable* from σ_0 iff $\sigma_0=\sigma_t$ or there exist configurations $\sigma_1,\ldots,\sigma_{t-1}$ such that σ_{r+1} follows σ_r for $r=0,\ldots,t\text{-}1$. We then say $\sigma=<\sigma_0,\ldots,\sigma_t>$ is a *path* in (v_0,A,p_0,S,δ). The *reachability set* of the VASS (v_0,A,p_0,S,δ), denoted by $R(v_0,A,p_0,S,\delta)$, is the subset of $S \times N^k$ containing all configurations reachable from (p_0,v_0).

We find it convenient to define $VASS(k,l,n)$ as the set of VASSs (v_0,A,p_0,S,δ) such that $\{v_0\} \cup A \subseteq Z^k$, l is the maximum length of the binary representation of any integer in the system, and $n=\max(|S|,|\delta|)$. Note that this definition differs from the one in [30], where n represents the number of states. We alter the definition in this manner so that we may use in our analysis either the number of states or the number of transitions, whichever is more applicable to the particular problem. In this section, we will assume the start vector is $\mathbf{0}$. (Note that $R(v_0,A,p_0,S,\delta)=R(\mathbf{0},A \cup v_0,q,S \cup \{q\},\delta')$ for some $q \notin S$ and some δ'.) Let $BV(k,b,n)$ be the set of all VASSs $(\mathbf{0},A,p_0,S,\delta)$ such that $R(\mathbf{0},A,p_0,S,\delta)$ is finite, $A \subseteq Z^k$, $|S|=n$, and $\max\{\sum_{i=1}^{k} v(i) : v \in A\}=b$. For any $v \in Z^k$, we define the *norm* of v, $||v||$, as $\sum_{i=1}^{k} |v(i)|$. (Note that this is often called the *1-norm*.) We define $\mu(k,b,n)$ as the maximum norm of any vector reachable by a VASS in $BV(k,b,n)$. Let σ be a path in a VASS. We define the *monotone increasing component* of σ, $\iota(\sigma)$, to be the sequence of configurations σ_i in σ for which all previous configurations in σ having the same state as σ_i have a vector with strictly smaller norm than that of σ_i. If σ is a path in a VASS in $BV(k,b,n)$, then $\iota(\sigma)$ clearly has finite length.

In this section, we will examine two related bounds, an upper bound on the time complexity of the finite containment problem and an upper bound for $\mu(k,b,n)$. In order to compare our results with those of McAloon [25], we define the following hierarchy of primitive recursive functions (see also [18]):

$$f_1(x)=2x$$
$$f_i(x)=f_{i-1}^{(x)}(1), \text{ for } i>1.$$

[25] gives an upper bound for the time complexity of the finite containment problem for k-place petri nets; the result clearly holds for k-dimensional VASs as well. It is easy to show that this upper bound is at least $f_{k+1}(m)$, where m is the maximum of the norm of the start vector and the increase in norm caused by any vector in the VAS. Our tightest results, however involve the VASS model rather than

either the petri net model or the VAS model. We can show that $\mu(k,b,n) \leq f_k(k,\max(n,\log b))$, for $k \geq 3$ and some constant d independent of k, b, and n. Furthermore, we exhibit a VASS in BV(k,1,m*(2*k-1)+2) that can produce a vector with norm $f_k(m)$. In order to compare our results with those of McAloon, however, we must phrase our upper-bound in terms of VASs. We are able to show that for any finite k-dimensional VAS, $k \geq 4$, with start vector v_0, such that no move causes an increase in norm of more than b, the containment problem can be solved in time $f_{k-1}(c^*\max(\log b,||v_0||))$, where c is a constant independent of k, b, and v_0. Our upper bound, therefore, represents an improvement of two levels of the primitive recursive hierarchy over that of McAloon. The bounds we get for k=2 and k=3 are similar.

2.1. Bounds on the Sizes of Finite VASSs

The general idea in what follows is to arrange the monotone increasing component of a path in a VASS into a tree in which any proper subtree contains only configurations whose states are the same and whose vectors have identical values in certain positions. In particular, in a subtree rooted at depth i (where the root of the tree is defined to be at depth 0), $i \geq 1$, all vectors will agree in at least i-1 positions. The resulting tree has certain properties which allow us to give a tight upper bound on its size, and hence, on the length of the monotone increasing component. The following lemma relates the length of a monotone increasing component to the norms of its constituent vectors.

Lemma 2.1: Let σ be a path in a VASS in BV(k,b,n), and let $\iota(\sigma)=<\sigma_0, \ldots, \sigma_t>$. Then the vector in σ_r has norm no more than r*b, $0 \leq r \leq t$.

Proof. By induction on r. The vector in σ_0 is $\mathbf{0}$, so the induction is well-based. Assume that for some $r>0$, σ_r has a vector with size u>r*b, but for all s, $0 \leq s<r$, the vector in σ_s has size no more than s*b. Clearly, no vector in any σ_s, $0 \leq s<r$, has size more than (r-1)*b. But since the size can be increased by no more than b in one move, σ must pass through a configuration with a vector having size u', (r-1)*b<u'<u, before entering σ_r--a contradition. Therefore, the vector in σ_r has size no more than r*b.

□

We now define $\mathcal{T}$(k,b,n) as the set of trees T having the following properties:

1. T has height $\leq$ k (i.e., the longest path from the root to a leaf is no more than k);
2. The root node of T is labelled 0 and has no more than n-1 children;
3. The nodes in T have integer labels such that for any node labelled r>b, there is a node labelled s, r-b $\leq$ s<r;
4. The label of any node in T is less than the label of any of its children;
5. The number of children of any node of depth i, $1 \leq i \leq k-1$, is no more than the node's label.

The following lemma shows that any monotone increasing component in BV(k,b,n) can be arranged into a tree in $\mathcal{T}$(k,b,n). We will subsequently derive an upper bound on the number of nodes in any tree in $\mathcal{T}$(k,b,n), thus yielding an upper bound on the length of any monotone increasing component in BV(k,b,n), and finally an upper bound on μ(k,b,n).

Lemma 2.2: Let σ be a path in a VASS in BV(k,b,n), $\iota(\sigma)=<\sigma_0, \ldots, \sigma_t>$. There is a tree $T \in \mathcal{T}$(k,b,n) with t+1 nodes whose labels are the norms of the vectors in $\iota(\sigma)$.

Proof. We first construct a tree T′ with nodes $[A_r, \sigma_r]$, $0 \leq r \leq t$, that satisfies the following:

1. The root node is $[\emptyset, \sigma_0]$.
2. The children of the root node are $\{[\emptyset, \sigma_r]$: σ_r contains the first occurrence in $\iota(\sigma)$ of some state q}.
3. If $[A_r, \sigma_r]=[A_r, <q_r, v_r>]$ is the parent of $[A_s, \sigma_s]=[A_s, <q_s, v_s>]$, and $[A_r, \sigma_r]$ is not the root node, then
   a. $r<s$;
   b. $q_r=q_s$;
   c. $\forall i \in A_r$, $v_r(i)=v_s(i)$;
   d. $A_s=A_r \cup \{i\}$, $i \notin A_r$, such that $v_s(i)<v_r(i)$.
4. If $[A \cup \{i\}, <q,v>]$ and $[A \cup \{i\}, <q,v'>]$ are children of $[A, \sigma_r]$, then $v(i) \neq v'(i)$.

We show by induction on t that T′ can be so constructed.

Clearly, T′ can be constructed if t=0. Suppose t>0, and assume we can construct a tree T″ from $\iota(\sigma')=<\sigma_0, \ldots, \sigma_{t-1}>$. Let $\sigma_t=<q, v_t>$. If the state q has not appeared in $\iota(\sigma')$, $[\emptyset, \sigma_t]$ can be added as a child of the root node, and all the conditions are clearly satisfied. Now suppose state q has appeared in $\iota(\sigma')$ for the first time in σ_r. If we stipulate that $[A_t, \sigma_t]$ is added as a leaf at depth 2 or deeper, conditions 1, 2, and 3a continue to hold. By adding $[A_t, \sigma_t]$ to the subtree rooted at $[\emptyset, \sigma_r]$, condition 3b is satisfied. Let $[A_s, \sigma_s]=[A_s, <q, v_s>]$ be any node in the subtree rooted at $[\emptyset, \sigma_r]$ such that $\forall i \in A_s$, $v_t(i)=v_s(i)$. ($[A_s, \sigma_s]=[\emptyset, \sigma_r]$ satisfies this.) There must exist an i ($\notin A_s$) such that $v_t(i)<v_s(i)$; otherwise, the VASS would be unbounded. If $[A_s, \sigma_s]$ has no child $[A_s \cup \{i\}, <q,v>]$ such that $v_t(i)=v(i)$, then $[A_s \cup \{i\}, \sigma_t]$ can be added as a child of $[A_s, \sigma_s]$, satisfying the remaining conditions. Otherwise, by induction on the height of T″, we can add $[A_t, \sigma_t]$, where $A_s \cup \{i\} \subseteq A_t$, to the subtree rooted at $[A_s \cup \{i\}, <q,v>]$.

We now construct T. To do so, we change every node label $[A_r, <q_r, v_r>]$ in T′ to $\|v_r\|$. We claim that $T \in \mathcal{T}$(k,b,n). Assume T′ has a node $[A_r, \sigma_r]=[A_r, <q, v_r>]$ at depth k, and $[A_r, \sigma_r]$ has a child $[A_s, \sigma_s]=[A_r \cup \{i\}, <q, v_s>]$. Clearly, A_s must contain all the integers 1, . . . ,k. Therefore, for all $j \neq i$

$1 \leq j \leq k$, $v_s(j)=v_r(j)$, and $v_s(i)<v_r(i)$, so $v_s<v_r$. But this contradicts the fact that σ_r occurs before σ_s in $\iota(\sigma)$. Therefore, T′ has height no more than k. Clearly, the root node is labelled 0 and has no more than n-1 children. As was shown in Lemma 2.1, if there exists a node label $||v_r||$, there must exist a node label $||v_s||$, $||v_r||\text{-}b \leq ||v_s||<||v_r||$. Clearly, the label of any node in T is less than the label of any of its children. From conditions 3d and 4 of the construction of T′, for each i, $1 \leq i \leq k$, $[A_r,\sigma_r]$ can have no more than $v_r(i)$ children, for a total of not more than $\sum_{i=1}^{k} v_r(i)$. Since $v_r \geq \mathbf{0}$, this is $||v_r||$. Therefore, $T \in \mathcal{T}(k,b,n)$. □

We will now show that a tree in $\mathcal{T}(k,b,n)$ having maximal size (i.e., a tree in $\mathcal{T}(k,b,n)$ having as many nodes as any other tree in $\mathcal{T}(k,b,n)$) is one whose depth-first (preorder) traversal visits its nodes in increasing order of their labels. We show this in the next three lemmas by using a rearrangement strategy.

Lemma 2.3: For any tree $T \in \mathcal{T}(k,b,n)$, there is a tree $T' \in \mathcal{T}(k,b,n)$ with the same number of nodes as T such that the labels on all nodes of any given depth are nondecreasing from left to right.

Proof. Suppose j is the smallest integer such that depth j of T is unordered; i.e., there exist nodes c and d of depth j such that c<d and d is to the left of c. We will show that the subtrees rooted at c and d may be swapped without disobeying the properties of $\mathcal{T}(k,b,n)$. Clearly, properties 1, 2, 3, and 5 are preserved, and if c and d have the same parent, property 4 is also preserved. Suppose, then, that d's parent is a, and c's parent is b, $a \leq b$ (see Figure 2-1). From property 4, a<d and b<c. Therefore, b<c<d and $a \leq b<c$, so if the subtrees rooted at c and d are swapped, property 4 is preserved. This swapping may be repeated until all depths are ordered, yielding $T' \in \mathcal{T}(k,b,n)$. □

Lemma 2.4: For any k, b, and n, $\mathcal{T}(k,b,n)$ contains a tree of maximal size.

Proof. By induction on k. If k=1, $\mathcal{T}(k,b,n)$ clearly contains a tree of maximal size. Suppose k>1, and assume that for any b and n, $\mathcal{T}(k\text{-}1,b,n)$ contains a tree of maximal size. Now assume that $\mathcal{T}(k,b,n)$ does not contain a maximal-sized tree for some b and n. We will first show that under this assumption, there exist n_0 and u_0 such that $\mathcal{T}(k,b,n_0)$ has no maximal-sized tree, but for any node label x occurring at depth 1 in a tree in $\mathcal{T}(k,b,n_0)$, $x<u_0$; i.e., the nodes at depth 1 in $\mathcal{T}(k,b,n_0)$ have bounded labels. First note that the nodes having depth 1 in the trees in $\mathcal{T}(k,b,n+1)$ have unbounded node labels, because we can add to any tree in $\mathcal{T}(k,b,n)$ a node at depth 1 with a label as large as any other label in the tree, thus yielding a tree in $\mathcal{T}(k,b,n+1)$. Clearly, the nodes with depth 1 in the trees in $\mathcal{T}(k,b,2)$ have bounded node labels. Let n_0 be the largest integer such that the nodes with depth 1 in the trees in $\mathcal{T}(k,b,n_0)$ have bounded labels. Consider an arbitrary tree in $\mathcal{T}(k,b,n_0+1)$. If we remove all nodes having a label at least as large as the largest label in depth 1 (call this label x), we get a tree in $\mathcal{T}(k,b,n_0)$ with some node labelled $x' \geq x\text{-}b$. Since x can be arbitrarily large, $\mathcal{T}(k,b,n_0)$ has arbitrarily large trees.

Since the nodes with depth 1 in the trees in $\mathcal{T}(k,b,n_0)$ have bounded labels, there exists a u such that no tree in $\mathcal{T}(k,b,n_0)$ has more than u nodes with depth 2. Let T be any tree in $\mathcal{T}(k,b,n_0)$. We now rearrange T by moving all subtrees with roots having depth 2 to depth 1; i.e., the roots of these subtrees become new children of the root of T. Since there are now no longer any nodes with depth k, $T' \in \mathcal{T}(k\text{-}1,b,n_0+u)$. Therefore, for any tree T in $\mathcal{T}(k,b,n_0)$ there is a tree T′ in $\mathcal{T}(k\text{-}1,b,n_0+u)$ with the same number of nodes as T. But since $\mathcal{T}(k\text{-}1,b,n_0+u)$ has a maximal-sized tree, $\mathcal{T}(k,b,n_0)$ must also have a maximal-sized tree--a contradiction. Therefore, for any k, b, and n, $\mathcal{T}(k,b,n)$ contains a tree of maximal size. □

Lemma 2.5: Any tree in $\mathcal{T}(k,b,n)$ having maximal size has its node labels arranged in order of a depth-first (preorder) traversal.

Proof. Assume T is a maximal-sized tree in $\mathcal{T}(k,b,n)$ whose node labels are not arranged in order of a depth-first traversal. We will construct a tree $T' \in \mathcal{T}(k,b,n)$ having more nodes than T has. From Lemma 2.3, we can assume without loss of generality that the node labels in each level of T are nondecreasing from left to right. If T has two nodes with the same label, we can clearly add 1 to the labels of one of these two nodes and all nodes having larger labels. Hence, we can assume that no node labels are repeated. Furthermore, we can clearly assume that the number of children of any node having depth $< k$ is the same as the node's label. Consider a traversal of T in order of increasing node labels, and let s_0 be the first node label reached that does not appear in depth-first order. Let $t > s_0$ be the label of a node appearing in a valid position for s_0 if the traversal were required to be depth-first (see Figure 2-2). Thus, t is at a lower level than s_0. Let A denote the position of s_0 in T, and let B denote the position of t in T. Let r be the parent of t, and let s_j the leftmost descendant of s_0 having the same depth as t. Also, let $s_1, \ldots, s_{j-1}$ be the nodes between s_0 and s_j, with s_i being the largest number in $s_0, \ldots, s_j$ less than t. Since position B is a valid position for s_0, $r < s_0$. Furthermore, all ancestors of r are less than s_0, so the subtree rooted at s_0 must be to the right of t; hence, $t < s_j$. So we have $r < s_0 < \cdots < s_i < t < s_{i+1} < \cdots < s_j$.

We make the following modifications to T:

1. Remove node t and the subtrees rooted at $t\text{-}s_0$ of t's children (if t has children).

2. Move node s_0 to position B (since $r < s_0$).

3. Move nodes $s_1, \ldots, s_i$ up one level in the tree (since $s_1, \ldots, s_i$ are smaller than each of their siblings).

4. Insert node t between nodes s_i and s_{i+1}, or into position A if i=0 (since $s_i < t < s_{i+1}$). We now have room for $(s_1\text{-}s_0) + \cdots + (s_i\text{-}s_{i-1}) + (t\text{-}s_i) = t\text{-}s_0$ subtrees below $s_1, \ldots, s_i, t$.

5. Insert the subtrees removed in Step 1 into the "holes" left in Step 4.

Notice that since each of the subtrees removed in Step 1 has been moved upward in the tree, there is now room for more nodes at the bottoms of these subtrees. By adding one node, we get a tree $T' \in \mathcal{T}(k,b,n)$ with more nodes than T. This contradicts the assumption that T is a maximal-sized tree in $\mathcal{T}(k,b,n)$. Since $\mathcal{T}(k,b,n)$ has a maximal-sized tree, it must be a tree whose node labels are arranged in order of a depth-first traversal. □

Corollary 2.1: Let S(k,b,n,i,x) be the set of subtrees in $\mathcal{T}(k,b,n)$ whose root is at depth i and has label x. The largest element of S(k,b,n,i,x) has its node labels arranged in order of a depth-first (preorder) traversal.

We can now give our upper bounds, first for VASs, then for $\mu(k,b,n)$. The idea is to first derive an upper bound on the largest node label in some tree in $\mathcal{T}(k,b,n)$. From this bound and Lemma 2.2, we can derive bounds on the norms of vectors generated by finite VASs and VASSs. In deriving our bounds, we use the following functions:

$g_b(x)=x+b$
$h_{1,b}(x)=x$
$h_{i,b}(0)=0$
$h_{i,b}(x)=(h_{i-1,b} \circ g_b)^{(x)}(x)$, for $i>1$, $x \neq 0$
$F_{i,b}(1)=0$
$F_{i,b}(x)=(h_{i,b} \circ g_b)^{(x-1)}(0)$, for $x>1$
$\lambda_1(b,n)=n*b$
$\lambda_2(b,n)=n*\max(\log b,1)$
$\lambda_i(b,n)=\max(n,\log b)$, for $i \geq 3$

Lemma 2.6: A subtree S with height $i \leq k-1$ and root label x in a tree $T \in \mathcal{T}(k,b,n)$ whose node labels are arranged in depth-first order has for its largest node label $u \leq h_{i+1,b}(x)$.

Proof. By induction on i. If i=0, then the largest node label in S is $x=h_{1,b}(x)$. Suppose $i>0$, and assume that any subtree of depth i-1 and root label y in a tree $T \in \mathcal{T}(n,s,b)$ whose node labels are arranged in depth-first order has for its largest node label $u_y \leq h_{i,b}(y)$. Now x has no more than x children, and the label of the 1st child is no more than $x+b=g_b(x)$. Since the labels of S are in depth-first order, the label u_j of the *j*th child, $1<j \leq x$, is no more than b plus the largest label in the subtree rooted at u_{j-1}. By the induction hypothesis, $u_j \leq h_{i,b}(u_{j-1})+b=g_b(h_{i,b}(u_{j-1}))$, so $u_x \leq g_b(h_{i,b} \circ g_b)^{(x-1)}(x)$. Now the largest label in S is in the subtree rooted at u_x, so from the induction hypothesis, the value of the largest label is $u \leq h_{i,b}(u_x) \leq (h_{i,b} \circ g_b)^{(x)}(x)=h_{i+1,b}(x)$. □

Theorem 2.1: There exist constants c and d (independent of k and b) such that for any k-dimensional finite VAS (v_0,A) with $\max\{\sum_{i=1}^{k} v(i) \ : \ v \in \{v_0\} \cup A\} = b$, $k \geq 2$, we have $\forall v \in R(v_0,A)$, $||v|| \leq f_{k-1}(c*\lambda_{k-1}(b,||v_0||))$.

Proof. Let $x=||v_0||$, and assume without loss of generality that $x \neq 0$. Clearly, for some n there exists a tree $T \in \mathcal{T}(k,b,n)$ with a subtree of height k-1 having root node labelled x. Therefore, from Lemmas 2.2, 2.5, and Corollary 2.1, the maximum $||v||$ in $R(v_0,A)$ must be bounded by $h_{k,b}(x)$. Now $h_{2,b}(x) = (h_{1,b} \circ g_b)^{(x)}(x) = g_b^{(x)}(x) = x*(b+1) \leq 2*b*x = f_1(\lambda_1(b,||v_0||))$. Also, $h_{3,b}(x) = (h_{2,b} \circ g_b)^{(x)}(x) \leq 2^{2*x*b^x*x} \leq 2^{c*x*\max(\log b,1)} = f_2(c*\lambda_2(b,||v_0||))$ for some constant c. In order to show the case for $k \geq 4$, we must first show by induction on y that for $k \geq 4$, $f_{k-1}^{(y)}(c*\lambda_k(b,x)) \leq f_k(c*\lambda_k(b,x)-x+y-1)$. Using this, we can show by induction on k that for $k \geq 4$, $h_{k,b}(x)+b \leq f_{k-1}(c*\lambda_{k-1}(b,x))$, from which the result follows. These two induction proofs are straightforward and are therefore omitted. □

Lemma 2.7: The largest node label in a tree $T \in \mathcal{T}(k,b,n)$ whose node labels are in depth-first order is no more than $F_{k,b}(n)$.

Proof. The root of T has no more than n-1 children, one of which is $u_1 \leq b$. Since the labels of T are in depth-first order, the label u_i of the *i*th child, $1<i<n$, is no more than b plus the largest label in the subtree rooted at u_{i-1}. Since the subtree rooted at u_i has height k-1, from Lemma 2.6, its largest node label is no more than $h_{k,b}(u_{i-1}+b)=h_{k,b} \circ g_b(u_{i-1})=(h_{k,b} \circ g_b)^{(i)}(0)$. Therefore, the largest node label in T is no more than $(h_{k,b} \circ g_b)^{(n-1)}(0)=F_{k,b}(n)$. □

Theorem 2.2: There exists a constant c (independent of k, b, and n) such that $\mu(k,b,n) \leq f_k(c*\lambda_k(b,n))$.

Proof. From Lemma 2.2, $\mu(k,b,n)$ is bounded by the largest node label in $\mathcal{T}(k,b,n)$, which, from Lemmas 2.5 and 2.7, is no more than $F_{k,b}(n)$. Now $F_{1,b}(n) = (h_{1,b} \circ g_b)^{(n-1)}(0) = (n-1)*b \leq f_1(\lambda_k(b,n))$. We now show the result for k=2; for $k \geq 3$, a similar induction is used. We proceed by induction on n. $F_{2,b}(1) = 0 \leq 2^{c*n*\max(\log b,1)} = f_2(c*\lambda_2(b,n))$, where c is the maximum of 3 and the constant from Theorem 2.1. Assume for some $n \geq 2$ that $F_{2,b}(n-1) \leq 2^{c*(n-1)*\max(\log b,1)}$. Then $F_{2,b}(n) = h_{2,b}(g_b((h_{2,b} \circ g_b)^{(n-2)}(0))) = h_{2,b}(g_b(F_{2,b}(n-1))) \leq h_{2,b}(2^{c*(n-1)*\max(\log b,1)}+b) = (b+1)(2^{c*(n-1)*\max(\log b,1)}+b) \leq 2^{c*n*\max(\log b,1)} = f_2(c*\lambda_2(b,n))$. □

We now turn to the lower bound.

Theorem 2.3: For any $k \geq 2$, $m \geq 1$, there is a VASS in BV(k,1,m*(2*k-1)+2) that can produce a vector with norm $f_k(m)$.

Proof. Consider the VASS V shown in Figure 2-3. V is bounded, because every loop in the state graph causes one position to decrease each iteration. V contains m copies of V_k. Since V_2 has 3 states, and V_i has 2 more states than V_{i-1}, V_k has 2*k-1 states, and V contains m*(2*k-1)+2 states. We now show by induction on i that V_i can produce $(0, \ldots ,0,f_{i-1}(n))$ when starting from $(0, \ldots ,0,n)$. Suppose i=2. Then the loop on q_1 and q_2 can be executed n times, yielding (2*n,0). The loop on q_3 can then produce $(0,2*n)=(0,f_1(n))$. Now suppose i>2. Assume V_{i-1} can produce $(0, \ldots ,0,f_{i-2}(n))$ when starting on $(0, \ldots ,0,n)$. The first time the loop on q_1 in v_1 is executed, it produces $(0, \ldots ,0,f_{i-2}(1)-1,n-1)$. As this loop is repeated, the input to $V_{i-1}^{(i)}$ on the j-th iteration is $(0, \ldots ,0,f_{i-2}^{(j)}(1),n-j)$. Therefore, when q_2 is reached after n iterations, the vector is $(0, \ldots ,0,f_{i-2}^{(n)}(1),0)$. The loop on q_2 can produce

$(0, \ldots, 0, f_{i-2}^{(n)}(1)) = (0, \ldots, 0, f_{i-1}(n))$.

The input to the first copy of V_k is $(0, \ldots, 0, 1)$, so this copy can produce $(0, \ldots, 0, f_{k-1}(1))$. The input to copy j can therefore be $(0, \ldots, 0, f_{k-1}^{(j-1)}(1))$, and the output can be $(0, \ldots, 0, f_{k-1}^{(j)}(1))$. Therefore V can produce $(0, \ldots, 0, f_{k-1}^{(m)}(1)) = (0, \ldots, 0, f_k(m))$. □

Corollary 2.2: For any $k \geq 2$, $m \geq (1)$, there is a finite VASS in VASS(k,1,m*(4*k-3)) that can produce a vector with size $f_k(m)$.

We have not been able to find a VAS in VASS(k,l,1) (for any constant l) to match the upper bound given in Theorem 2.1, although technically a k-dimensional VAS with $\max\{\sum_{i=1}^{k} v(i) : v \in \{v_0\} \cup A\} = 1$ can be shown whose maximum reachable vector has norm $O(f_{k-1}(n))$, where n is the norm of the start vector. The problem with all such VASs that we have seen is that their constituent vectors contain very large (i.e., $O(f_{k-1}(n))$) positive numbers in some positions and very large negative numbers in other positions, so that the net gain caused by each vector is only 1.

2.2. The Finite Containment and Equivalence Problems

In this subsection we concern ourselves with the complexity of the equivalence and containment problems for finite VASSs. If u is an upper bound on the norm of any vector reachable by a k-dimensional VASS (or VAS), clearly u^k is an upper bound on the number of vectors in the reachability set. From [19], we can therefore generate the reachability set in time $O(u^k)$. It then follows that the finite containment and equivalence problems can be solved in time $O(u^{2k})$. Thus, from Theorems 2.1 and 2.2 we have the following result, which represents an improvement over the bound provided by [25].

Theorem 2.4 There exists a positive constant c (independent of k, b, and n) such that the containment and equivalence problems can be solved in time

1. $f_k(c^*\lambda_k(b,n))$ for BV(k,b,n), $k \geq 2$;
2. $f_1^{4}(c^*n^*b)$ for 2-dimensional finite VASs whose vectors cause increments of no more than b and whose start vectors have norm n;
3. $f_{k-1}(c^*\lambda_{k-1}(b,n))$ for k-dimensional finite VASs, $k \geq 3$, whose vectors cause increments of no more than b and whose start vectors have norm n.

Now [25] gives an upper bound for the finite containment problem (and, hence, the finite equivalence problem) for petri nets; this bound can be shown to be at least $f_{k+1}(\lambda_3(b,n))$. Note that since our analysis for VASs applies also to petri nets, our result improves the upper bound of McAloon [25] by two levels in the primitive recursive hierarchy. A natural question to consider is whether one can establish a similar lower bound. Certainly the lower bound must in some fashion grow within the hierarchy since the problems are complete within the Ackermann function [5]. We would like to show that there exists a positive constant d such that the problems require $f_k(d^*n)$ time infinitely often. At this time, however, we are only able to show the following:

Theorem 2.5: There exist positive constants a, b and c (independent of k and n) such that the containment and equivalence problems for BV(k,1,n) require $f_{k-a}(b^*n)$ time infinitely often whenever $k>c$.

Proof. The proof is a refinement of the one in [26], and hence only a sketch will be given. In [26], the complexity of the finite containment problem for VASs was shown to be non-primitive recursive. The proof was done by reducing the bounded version of Hilbert's tenth problem to the containment problem; this is similar to Rabin's proof of the undecidability of the containment problem for arbitrary VASs. More precisely, let A(k) be a function that majorizes the primitive recursive functions (for instance, $f_k(k)$, which was defined in the previous section). Mayr and Meyer showed how to reduce the *Bounded Polynomial Inequality Problem* (BPI) (given two r-variable polynomials p, q and a positive integer k, decide whether $\forall\, \bar{y} \in \{0,1,...,A(k)\}^r$, $p(\bar{y}) \leq q(\bar{y})$) to the containment problem for two VASs V_p and V_q, such that the BPI has a solution iff $R(V_p) \subseteq R(V_q)$. For an instance of the BPI, V_p (V_q) basically consists of two VASs, say V and V′, connected in series. V computes the function A(k), while V′ simulates the computation of the polynomial p (q). Then, according to the result by [1], the complexity of BPI is greater than $\log(\log(\log(A(m^{1/4})^{1/5})))$ infinitely often. Therefore, the non-primitive recursive lower bound for the containment problem for VASs is obtained. Notice that in [26] the complexity is measured in terms of the overall size of the VASs, which is, in some sense, rough. A careful analysis will further indicate that the number of coordinates (i.e., the dimension of the vector) needed in V′ depends only on the number of variables and the order of the polynomial. This, combined with the fact that the polynomials in the BPI can be further restricted to have a fixed number of variables and fixed order (see [1]), gives us that the two instances of the containment problem are in $BV(k+c_1,1,c_2^*(k+Q\{|p|,|q|\}))$, for fixed constants c_1 and c_2 and some polynomial Q. Since the construction is very much the same as that in [26], the details are omitted. Let $a=c_1$, $b=1/c_2$ and $c=\max\{3,c_1\}$. The equivalence and containment problems for BV(k,1,n) require $f_{k-a}(b^*n)$ time infinitely often whenever $k>c$. □

In the proof of the previous theorem note that the construction is such that each position is bounded by $f_{k-a}(b^*n)$. Let $\mathbf{V}_i(n)$ denote the set of finite VASSs whose reachability sets are bounded by $f_i(n)$. Now given an arbitrary instance (p,q,k) of BPI, we can construct VASSs V_p and V_q in $\mathbf{V}_k(c_2^*(k+Q\{|p|,|q|\}))$ such that (p,q,k) has a solution iff $R(V_p) \subseteq R(V_q)$.

Corollary 2.3 There exist positive constants c and d (independent of i and n) such that the time complexity of the containment and equivalence problems for $\mathbf{V}_i(n)$ are bounded above (below) by $f_i(d^*n)$ ($f_i(c^*n)$).

With respect to the difficulty of these problems for small fixed values of k, not much is known. For example, the problems are clearly in PSPACE for $\cup_{b,n}BV(1,b,n)$. One can easily conclude that the problems are NP-hard (PSPACE-hard) when $k \geq 2$ ($k \geq 4$) from results in [30] concerning the boundedness problem. (Similar gaps in knowledge currently exist for the case of symmetric VASs where the equivalence problem is known to be PSPACE-complete (NP-hard, in PTIME, respectively) for 6- (4-, 1-, respectively) dimensional VASs [15, 16].) We are, however, able to establish a completeness result for $\cup_n$BV(1,1,n).

Theorem 2.6: The containment and equivalence problems for $\cup_n$BV(1,1,n) are $A\Pi_2^L$-complete.

Proof. To derive the upper bound, we show how to construct a log n space-bounded, 1-alternating ATM M whose initial state is universal, such that M accepts an input string of two VASSs V and V′ in BV(1,1,n) iff $V \subseteq V'$. (For definitions of ATM's see [4].) First notice that any reachable configuration of V (V′) can contain a vector with norm at most n; otherwise, a pumpable positive loop exists; this contradicts the fact that V (V′) is bounded. Therefore, to reach a given configuration, one only needs to consider a path of length c*n, for some constant c. We now sketch the computation of M as follows.

A computation of M has two phases--first the <u>universal</u> phase and then the <u>existential</u> phase. In the first phase, M traverses all paths of V (of length at most c*n) and records the information of the current configuration (which is a pair [p,x]) on the first track of the work tape. Note here that all states in the this phase are universal. Now, from each universal state, M can enter the second phase to simulate the computation of V′. In the second phase, M (nondeterministically) traverses a path in V′ (of length at most c*n) and keeps the information of the current [p,x] on the second track of the work tape. At any time if the contents of the first and the second tracks are the same, M enters an accepting state. It is reasonably easy to see that M accepts the input (i.e., the string representing V and V′) iff $R(V) \subseteq R(V')$. Furthermore, M needs only logarithmic space.

Now, we show the lower bound. Let M be a log n space-bounded 1-alternating ATM whose initial state is universal. Given an input string we show how to construct two VASSs V and V′ in BV(1,1,n) in such a way that M accepts x iff V=V′. Let |x| denote the length of x. A configuration of M is a 3-tuple [p,i,s], where p is the current state, i is the input head position and s is the content of the work tape (including the head position). Since M uses only O(log |x|) space, the number of distinct configurations is polynomial in |x|. A configuration is called a universal (existential, accepting, rejecting) configuration iff p is a universal (existential, accepting, rejecting) state. Let T be the set of all configurations of M on x. Now, $V=(<0>,\{<0>\},p_0,S,\delta_0)$ is constructed as follows (see Figure 2-4):

1. $S=T\cup((T\cup\{\theta\})\times T)$

2. $p_0=c_0$, where c_0 is the initial configuration of M,

3. δ:

   (a) $(q,<0>)\in\delta(p)$, where p is a universal configuration and M can reach q from p in one step,

(b) ([r,r],<0>) $\in \delta$(r) for every existential configuration r,

(c) $\forall$r $\in$ T ([q,r],<0>) $\in \delta$([p,r]), where p is an existential configuration and M can reach q from p in one step,

(d) $\forall$r $\in$ T ([Θ,r],<0>) $\in \delta$([v,r]), where v is an accepting configuration,

(e) $\forall$r $\in$ T ([Θ,r],<0>) $\in \delta$([r,r]).

V′ is exactly the same as V except that rule 3(e) is removed. Clearly, M accepts x iff every node labelled [Θ,r], for some r, can be reached. This, in turn, can happen iff R(V)=R(V′). Furthermore, clearly V and V′ are in BV(1,1,n), where n is polynomial in |M| and |x|. Since the equivalence problem can be reduced to the containment problem, this completes the proof. □

The last result may be of independent interest since few natural problems complete for $A\Pi_2^L$ appear to be known. See [31, 32] for other examples.

3. 2-dimensional VASSs

The containment and equivalence problems for VASSs are, in general, undecidable [2, 11]. In fact, using essentially the same proof it can be shown that there exists a constant k such that the containment and equivalence problem are undecidable for $\cup_{l,n}$VASS(k,l,n). In Rabin's proof, he reduced Hilbert's tenth problem, which is well known to be undecidable, to the equivalence problem for VASs. More precisely, given an arbitrary polynomial he showed how to construct two VASs to compute, in some sense, the polynomial in such a way that the Diophantine equation has a solution iff the two VASs are equal. Therefore, the undecidability results of the equivalence and containment problems for VASs are obtained. In fact, one can further restrict the Diophantine equation to have a fixed order and a fixed number of variables [8]. In other words, there exists a universal polynomial P which contains a special variable i such that for an arbitrary integer j it is undecidable whether the Diophantine equation P_j=0, where P_j is the new polynomial obtained by substituting j into the variable i, has a nonnegative integer solution. Furthermore, a detailed analysis of Rabin's proof will reveal that the dimension of the VASs depends only on the order and the number of variables of the polynomial. Consequently, applying the same proof to the universal polynomial, the containment and equivalence problems are undecidable for $\cup_{l,n}$VASS(k,l,n), for some fixed constant k. However, at this moment the best upper bound for k is still unknown. It is known, however that the containment and equivalence problems are decidable for k=2 [12]. In this section, our goal is to establish a complexity bound for the reachability, containment and equivalence problems for 2-dimensional VASSs. In order to do this we establish a bound on the algorithm of Hopcroft and Pansiot [12], which, when given a 2-dimensional VASS, generates the corresponding SLS representation of the reachability set. There are at least two reasons one might want to consider these problems for 2-dimensional VASSs. First, note that the reachability set is not, in general, semilinear when the dimension is greater than two [12]. Also, the problems for 2-dimensional VASSs appear to be easier to work with than they do for other classes of VASs whose reachability sets are also effectively computable SLSs. Perhaps more importantly, we hope to be better equipped to attack the complexity of the general reachability problem.

In the subsequent discussion, we closely examine the algorithm provided in [12]. We first show that this algorithm operates in NTIME($2^{2^{b*l*n}}$) for some constant b independent of l and n on any VASS in VASS(2,l,n). We then prove that, for any VASS in VASS(2,l,n), there is a DTIME($2^{2^{c*l*n}}$) algorithm to generate the corresponding SLS representation whose size is bounded by O($2^{2^{d*l*n}}$), where c and d are constants independent of l and n. This SLS has the additional properties that each of its constituent linear sets has O(2^n) periods with norm O($2^{c'*l*n}$) for some constant c′ independent of l and n. These properties allow us to derive upper bounds of DTIME($2^{2^{c*l*n}}$) for the reachability, equivalence, and containment problems for VASS(2,l,n). Although we are unable to establish the corresponding lower bound, we are able to show that the search procedure of Hopcroft and Pansiot requires $2^{2^{c*l*n}}$ steps. Thus, our analysis of their algorithm is tight. However, at this time we do not know whether exploring the entire tree is necessary. It is possible that only a portion of the tree is needed to generate the SLS. If so, some other strategy like breadth-first search might result in a more efficient algorithm. Neither do we know whether there exists a more efficient algorithm not based on the Hopcroft and Pansiot tree construction. So far, the best lower bound we know for $\cup_{l,n}$VASS(2,l,n) is NP [30]. Hence, there is still much room for improvement. Now, before continuing to the detailed discussion, the following definitions are required.

For any vector $v_0 \in N^k$ and any finite set $P(=\{v_1,\ldots,v_m\}) \subseteq N^k$, the set $\mathcal{L}(v_0,P)=\{x:\exists k_1,\ldots,k_m \in N^k \text{ and } x=v_0+\sum_{i=1}^{m} k_i v_i\}$ is called the *linear set* over the set of *periods* P. The *size* of the linear set $\mathcal{L}(v_0,P)$, denoted by $|\mathcal{L}(v_0,P)|$, is defined to be $\sum_{i=0}^{m} k \times \log_2 ||v_i||$. (I.e., the number of bits needed to represent the linear set.) A finite union of linear sets is called a *semilinear set* (SLS, for short). The size of a SLS is the sum of the sizes of its constituent linear sets. The *cone* generated by v_0 and P, denoted by $\mathcal{C}(v_0,P)$, is the set $\{x:\exists k_1,\ldots,k_m \in R^k, k_1,\ldots,k_m \geq 0, \text{and } x=v_0+\sum_{i=1}^{m} k_i v_i\}$.

Given a VASS=(v_0,A,p_0,S,δ) and a path l in the state graph, $l= s_1 \xrightarrow{v_1} s_2 \xrightarrow{v_2} \cdots s_{t-1} \xrightarrow{v_{t-1}} s_t$ where $s_i \rightarrow (s_{i+1},v_i)$ $(1 \leq i \leq t-1)$ is in δ, l is a *short loop* iff $s_1 = s_t$ and $s_i \neq s_j$ $(1 \leq i < j \leq t)$. The *displacement* of l, denoted by $|l|$, is $\sum_{i=1}^{t-1} v_i$. l is a *short positive loop* (*p-loop*, for short) iff l is a short loop and $|l| > 0$.

In what follows, our analysis heavily depends on the algorithm given in [12]. Hence, for the sake of completeness, the algorithm is listed below. However, only a brief description will be given. The reader is encouraged to refer to [12] for more details. Given a 2-dimensional VASS V, the main idea behind the algorithm is to construct a tree in which each node is labelled by a 3-tuple $[x,p,A_x]$, where $x \in N^2$, $p \in S$ and $A_x \subseteq N^2$, to represent the reachability set generated by V. In what follows, each A_x is called a *loop set*. Each v in A_x is called a *loop vector*. The label $[x,p,A_x]$ indicates that $\{(p,v)\colon v \in \mathcal{L}(x,A_x)\} \subseteq R(v_0,A,p_0,S,\delta)$. The intuitive idea of why the procedure works is the following. The tree is built in such a way that each path, in a sense, corresponds to a computation of the VASS. Each time an executable (valid) p-loop is encountered, that particular p-loop will be added (if necessary) to the loop set since clearly that loop can be repeated as many times as we want. If, along any path of the tree, there is an ancestor $[z,p,A_z]$ of $[x,p,A_x]$ such that $A_x=A_z$ and $x \in \mathcal{L}(z,A_z)$, then that particular path terminates at $[x,p,A_x]$. (This condition will be referred to as the *terminating condition*.) In [12], it was shown that a point (p,v) in $S \times N^2$ is reachable in V iff there exists a node with the label $[x,p,A_x]$ such that $v \in \mathcal{L}(x,A_x)$. (In other words, the reachability set coincides exactly with the SLS associated with the tree construction.) Furthermore, the tree construction will eventually terminate. Now, in order to put complexity bounds on this procedure, some measure of the tree is needed. In particular, we will see later that in order to derive the upper bound of the Hopcroft-Pansiot algorithm, it suffices to consider the following two quantities:

(1) $\max\{||v||\colon \exists\ [x,p,A_x] \in T \text{ such that } v \in A_x\}$,

(2) $\max\{||x||\colon [x,p,A_x] \in T\}$.

Intuitively, the first quantity tells us how "large" each linear set can be; while the second quantity indicates the number of linear sets required to build the SLS.

<u>Algorithm:</u> (from [12])

Create root labelled $[x_0,p_0,\emptyset]$;
while there are unmarked leaves **do**
begin
Pick an unmarked leaf $[x,p,A_x]$;
Add to A_x all displacements of short positive loops from p valid at x;
if A_x is empty and there exists an ancestor $[z,p,A_z]$ with $z<x$,
then add x-z to A_z;
if there exists $c \in N^2$, $c=(0,\gamma)$ or $(\gamma,0)$ such that
(a) c is not colinear to any vector of A_x, and
(b) either
(i) there exists an ancestor $[z,p,A_z]$ of $[x,p,A_x]$ such that x-z=c, or
(ii) for some short nonpositive loop from p valid at x with displacement a and some $b \in A_x$, there exists $\alpha, \beta \in N$ such that $\alpha a+\beta b=c$
then add c to A_x;
if there exists an ancestor $[z,p,A_z]$ of $[x,p,A_x]$ such that
$\mathcal{L}(z,A_z)$ contains x and $A_z=A_x$

```
        then mark [x,p,A_x]
        else
              for each transition p→(q,v) do
              begin
                    Let A_x={v_1,...,v_k}
                    for each a, a=α_1v_1+...+α_kv_k where
                          (α_1,...,α_k) is a minimal k-tuple such that x+a+v ≥ 0,
                    do construct a son [y,q,A_y] where y=x+a+v and A_x=A_y;
              end
        if [x,p,A_x] has no son then mark [x,p,A_x];
end
```

3.1. The Upper Bound

Now, we are ready to derive an upper bound on the algorithm's complexity. Given a VASS V in VASS(2,l,n) and some path s in the corresponding tree T, one can easily see the following facts:

- V has at most n states;
- there are at most 2^n distinct p-loops in any loop set;
- in addition to those p-loops, at most one non-axis vector and 2 axis vectors can occur in any of the loop sets in s (in what follows, if they exist, they will be referred to as u_1, γ_1 and γ_2, respectively);
- of all the vectors appearing in the loop sets in s, only u_1, γ_1 and γ_2 can have a norm greater than $n*2^l$.

Consider an arbitrary path in the tree generated by the algorithm. Let $h_{u_1}(l,n)$, $h_{\gamma_1}(l,n)$ and $h_{\gamma_2}(l,n)$ denote the maximum norm of all the vectors added before u_1, γ_1 and γ_2 are added, respectively. Also, let $h_k(l,n)$ denote the maximum norm of all vectors ever occurring in the system before the k-th loop vector is added. For two arbitrary nodes $d_1=[x_1,p_1,A_1]$ and $d_2=[x_2,p_2,A_2]$, $d_1 \rightarrow d_2$ iff d_1 is an ancestor of d_2 in T. d_2 is said to be *redundant* with respect to d_1, denoted by $d_1 \bullet\!< d_2$, iff $p_1=p_2$, $A_1=A_2$, $x_2\text{-}x_1 \in \mathcal{L}(\mathbf{0},A_1)$ and $d_1 \rightarrow d_2$. We also say that a node d is redundant iff there exists a d′ such that $d' \bullet\!< d$. (Note that, according to the terminating condition, if $d_1 \bullet\!< d_2$, then d_2 is a leaf.) A sequence of nodes $d_1=[x_1,p_1,A] \rightarrow d_2=[x_2,p_2,A] \rightarrow \cdots \rightarrow d_t=[x_t,p_t,A]$ is said to be *monotonic* (*strongly monotonic*) if $||x_1|| \leq ||x_2|| \leq \cdots \leq ||x_t||$ ($x_1 \leq x_2 \leq \cdots \leq x_t$). In what follows, we first derive the quantity max{$||v||$: $\exists$ $[x,p,A_x] \in T$ such that $v \in A_x$}, which is one of the two values we are most interested in. Hence, we must derive bounds for $||u_1||$, $||\gamma_1||$ and $||\gamma_2||$. The next lemma and its corollary provide a bound for $||u_1||$.

Lemma 3.1: $h_{u_1}(l,n) \leq h_1(l,n)=O(2^{c^{*l*n}})$, for some constant c independent of l and n.

Proof. In any path in the tree, no node $[x,p,\emptyset]$ can occur such that $[x',p,\emptyset] \rightarrow [x,p,\emptyset]$ where $x'<x$, unless u_1 is added. Therefore, from Theorem 2.2, $h_1(l,n)=O(2^{c^{*l*n}})$. Since u_1 can only be added to an empty loop set, the result follows. □

Corollary 3.1: $||u_1|| \leq a*2^{b*l*n}$, for some constants a and b independent of l, and n.

Proof. Let π: $s_0 \rightarrow s$ be a path in T such that s_0 is the root and u_1 is added in s. Clearly any node $[x,p,A_x]$ in π must have that $||x|| \leq h_1(l,n)$. From the algorithm, $u_1 = x_2 - x_1$, for some $d_1 = [x_1,p,\emptyset]$ and $d_2 = [x_2,p,\emptyset]$ in π. Thus, according to Lemma 3.1, we have that $||u_1|| = ||x_2 - x_1|| \leq \max\{||x_1||, ||x_2||\} \leq h_1(l,n) = O(2^{b*l*n})$. The result follows. □

As long as the loop set is empty, a path in T corresponds exactly with a path in the associated VASS. After the first loop vector is added, however, the correspondence no longer remains exact. Therefore, we must find an upper bound on the gain in norm caused by one step in T.

Lemma 3.2: Let $u = [x,p,A_x] \rightarrow u' = [x',p',A_{x'}]$ be two consecutive nodes in T. Let $r = \max\{||v||: v \in A_x\}$. Then $||x'-x|| \leq c*(r*2^l)^d$, for some constants c and d independent of r, and l. (I.e., the maximum gain in one step in T is bounded by $c*(r*2^l)^d$.)

Proof. To show this, first note that given a node $u = [x,p,A_x]$ in T, the successor $u' = [x',p',A_{x'}]$ can be obtained if $p \rightarrow (p',v)$ is in δ and $x' = x + v + \sum_{i=1}^{k} \alpha_i v_i \geq \mathbf{0}$, where $A_x = \{v_1,...,v_k\}$ and $(\alpha_1,...,\alpha_k)$ is a minimal k-tuple such that $x' \geq \mathbf{0}$. (See the algorithm.) According to the result in [3], we know that if the above linear equation has a nonnegative solution, there must exist a solution $(\beta_1,...,\beta_k)$ such that $|\beta_i| \leq c'*(\max\{||v||,||v_j||: 1 \leq j \leq k\})^2$, which is no greater than $c'*(r*2^l)^2$. Clearly by a direct substitution, the net gain $||x'-x||$ is no more than $c*(r*2^l)^d$, for some constants c and d. □

In deriving bounds for $||\gamma_1||$ and $||\gamma_2||$, the idea is to show that if a monotonic sequence of some specified length exists, then a strongly monotonic sequence of a certain length must also exist. The following lemma gives a bound on the length of a strongly monotonic sequence over the same loop set.

Lemma 3.3: Consider a nonempty loop set $A = \{v_1,...,v_m\}$, where $v_1,...,v_m$ are arbitrary loop vectors. Let $\beta = \max\{||v||: v \in A\}$. If $d_1 = [x_1,p,A] \rightarrow d_2 = [x_2,p,A] \rightarrow \cdots \rightarrow d_{\beta^2+1} = [x_{\beta^2+1},p,A]$ is a strongly monotonic sequence, then there exist i and j, $1 \leq i,j \leq \beta^2+1$, such that $d_i \bullet\!\!< d_j$.

Proof. Let w_v and w_h be the vectors in A with the maximum and minimum slopes, respectively. It follows from [12] that the sequence is contained in the cone $\mathcal{C}(x_1,A)$. Since $x_1 \leq x_2 \leq \cdots \leq x_{\beta^2+1}$, it can be shown easily that, for all i, $2 \leq i \leq \beta^2+1$, there exist $\bar{x}_i$, $a_{i,v}$ and $a_{i,h}$ such that $x_i = x_1 + \bar{x}_i + a_{i,v} w_v + a_{i,h} w_h$. Furthermore, they satisfy the following conditions:

(1) $\mathbf{0} \leq \bar{x}_i \leq \langle \beta-1, \beta-1 \rangle$, and

(2) $\forall$ i and j, $1 \leq i \leq j \leq \beta^2+1$, $\langle a_{i,v}, a_{i,h} \rangle \leq \langle a_{j,v}, a_{j,h} \rangle$.

By the pigeon-hole principle, there exist i and j, i<j, such that $\bar{x}_i = \bar{x}_j$. Hence, $x_j - x_i = (a_{j,v} - a_{i,v})*w_v + (a_{j,h} - a_{i,h})*w_h$, which is in $\mathcal{L}(\mathbf{0},A)$ (actually, in $\mathcal{L}(\mathbf{0},\{w_h,w_v\})$). Therefore, $d_i \bullet\!\!< d_j$. □

We note here that although an upper bound for $\max\{||x|| : [x,p,A_x]\}$ can be derived directly from

Lemma 3.3, this bound is not tight (i.e., it would cost us additional levels of exponentiation). Hence, we wish to derive, via the next two lemmas, a better bound for $||\gamma_1||$.

Lemma 3.4: Let σ: $d \rightarrow d'$ be a path in T. Let A_d be the loop set in d. Assume that $A_d \neq \emptyset$ and no axis vector exists in A_d. If $d_1=[x_1,p,A] \rightarrow d_2=[x_2,p,A] \rightarrow \cdots \rightarrow d_t=[x_t,p,A]$ is a monotonic sequence in σ, then it is also strongly monotonic.

Proof. For two arbitrary nodes d_i and d_j, $1 \leq i \leq j \leq t$, let $x_j - x_i = \sum_{m=1}^{k} w_m$, where w_m's are displacements of short loops. One of the following two cases must be true:

(Case 1:) all w_m's are of the form (x,y), such that $x>0$ and $y>0$.

If so, clearly $x_j \geq x_i$ <u>or</u>

(Case 2:) all w_m's and all vectors in A_d are colinear.

In this case, $||x_j|| \geq ||x_i||$ implies $x_j \geq x_i$.

(Note that for other cases, an axis vector would have been added. See also [12].) This completes the proof.

□

Lemma 3.5: $h_{\gamma_1}(l,n) \leq c'*2^{d'*l*n}$, for some constants c' and d' independent of l and n.

Proof. Let s' be the node at which γ_1 is added. Let $p_1, p_2, \ldots, p_t$ be those nodes (in sequence), along the path from s_0 to s', where new loop vectors are added ($p_t = s'$). We define a function f(k), $1 \leq k \leq t$, such that f(k) is the maximum norm of any vector ever occurring in the system before p_k is reached. Consider the following two cases:

(Case 1:) t=1; i.e., γ_1 is the first loop vector.

Clearly, the result follows from Lemma 3.1.

(Case 2:) $t>1$.

Clearly, $f(1)=h_1(l,n)$. In what follows, we calculate f(k) recursively. Consider the path from p_k to p_{k+1} (excluding p_{k+1}). During this period the loop set, say A_k, remains the same. Let v_h and v_v be the vectors with the minimum and maximum slopes in A_k. Let $\beta'_k = \max\{||v||: v \in A_k\}$. According to Corollary 3.1, $\beta'_k \leq a*2^{b*l*n}$, for some constants a and b. Now, applying the result of Lemma 3.2, the maximum gain in one step is bounded by $a_1*2^{b_1*l*n}$, for some constants a_1 and b_1. (Let β denote this amount.) Suppose a node p' contains a vector with norm $\geq$ $f(k)+n*\beta^3$. By the pigeon-hole principle there must exist a monotonic sequence, ϵ: $d_1 \rightarrow \cdots \rightarrow d_{\beta^2+1}$, over the same state (see Figure 3-1). Furthermore, according to Lemma 3.4, ϵ is also strongly monotonic. Thus, by Lemma 3.3, the procedure should terminate, which is clearly a contradiction. Therefore, $f(k+1) \leq f(k)+n*\beta^3$. Inductively, one can easily get $f(t) \leq h_1(l,n)+t*(n*\beta^3)$. Since t (the number of p-loops) $\leq 2^n$, we have that $f(t) \leq c'*2^{d'*l*n}$, for some constants c' and d'. The result follows.

□

Corollary 3.2: $||\gamma_1|| \leq c*2^{d*l*n}$, for some constants c and d independent of l, and n.

Proof. γ_1 can be added either because

(Case 1:) $\exists$ an ancestor $[z,p,A_z]$ of $[x,p,A_x]$ such that $x-z=\gamma_1$.

Clearly, $||\gamma_1||=||x-z||\leq h_{\gamma_1}(l,n)$ <u>or</u>

(Case 2:) $\exists$ α, α' in N, $a\in A_x$ and a non-positive loop b, such that $\alpha a+\alpha' b=\gamma_1$.

In this case, $||\gamma_1||\leq m*(h_{\gamma_1}(l,n))^r$, for some constants m and r. (Due to the result of [3].)

□

We are now ready to derive a bound on $||\gamma_2||$, using the following two lemmas:

Lemma 3.6: Consider a loop set $A=\{v_1,\ldots,v_m\}$ that contains a vertical (horizontal) axis vector. (We include the case in which both axis vectors exist.) Let $\beta = \max\{||v||: v\in A\}$. Let b and f be arbitrary positive integers. If $d_1=[x_1,p,A]\rightarrow d_2=[x_2,p,A]\rightarrow \cdots \rightarrow d_{\beta*b+1}=[x_{\beta*b+1},p,A]$ is a <u>monotonic</u> sequence contained in the area $\{<x,y>: f\leq x\leq f+b-1 \text{ and } 0\leq y\}$ ($\{<x,y>: f\leq y\leq f+b-1 \text{ and } 0\leq x\}$), then there exist i and j, $1\leq i<j\leq \beta*b+1$, such that $d_i \bullet\!< d_j$.

Proof. Without loss of generality, we only consider the case with a vertical axis vector. (The other case is symmetric.) Furthermore, with no loss of generality, we also assume that f=0. Let w_v be the axis vector. Clearly, $||w_v||\leq\beta$. Now, consider b vertical lines L_k's, $0\leq k\leq b-1$, where $L_k=\{<k,y>: y\in N\}$. By the pigeon-hole principle there must exist some line L_j that contains more than β points. Let $d_{i_1},d_{i_2},\ldots,d_{i_{\beta+1}}$ be such a sequence. Clearly, there exist r and s, $1\leq r<s\leq\beta+1$, such that $x_{i_s}-x_{i_r}=d*\beta$, for some $d\in N$. Thus, $d_{i_r}\bullet\!< d_{i_s}$.

□

Lemma 3.7: $h_{\gamma_2}(l,n)\leq a'*2^{b'*l*n}$, for some constants a' and b' independent of l and n.

Proof. Without loss of generality, assume that γ_1 exists and is a vertical axis vector. Suppose s' is the node where γ_1 is added. Let $\beta=\max\{c*2^{d*l*n},n*2^l\}$ and $\mathbf{b}=c'*2^{d'*l*n}$, where c, d and c', d' are the constants mentioned in Corollary 3.2 and Lemma 3.5, respectively. (Thus, β bounds the largest norm of any loop vector added before γ_2 and **b** bounds the largest norm of any vector appearring before γ_1 is added.) Let β' ($=O(2^{c''*l*n})$, for some constant c'') be the maximum gain in one step. Let v be the vector in $A_{s'}$ with the minimum slope. Let D and D_i, $1\leq i\leq 2^n*\beta^2$, be:

$D=\{<x,y>: 0\leq x<\mathbf{b}, 0\leq y\}$,

$D_i=\{<x,y>: \mathbf{b}+(i-1)*\beta'\leq x<\mathbf{b}+i*\beta', 0\leq y\}$.

See Figure 3-2. Suppose a node contains a vector with norm greater than $\beta'*\beta*\mathbf{b}*n+(\beta')^2*2^n*\beta^4*n$. Clearly, there must exist a monotonic sequence consisting of $\beta*\mathbf{b}+\beta'*2^n*\beta^3$ nodes over the same state. One of the following three cases must be true:

(Case 1:) D contains a monotonic sequence over the same state with $\beta*\mathbf{b}+1$ nodes -- a contradiction (according to Lemma 3.6).

(Case 2:) $\exists$ i, $1 \le i \le 2^n * \beta^2$, such that D_i contains a monotonic sequence over the same state with $\beta*\beta'+1$ nodes -- a contradiction (according to Lemma 3.6).

(Case 3:) $\exists$ a monotonic sequence $d_1 \to \cdots \to d_{2^n * \beta^2}$, such that d_i is in D_i for $1 \le i \le 2^n * \beta^2$.

Note that in the above sequence, the horizontal component is always incremented. Since no horizontal axis vector exists, this sequence must be also strongly monotonic. (Otherwise, a horizontal axis vector would be added.) Furthermore, since at most 2^n loops will be added, there must exist i and j, $1 \le i,j \le 2^n * \beta^2$, $j-i > \beta^2$, such that no p-loop is added during the period from d_i to d_j. This clearly contradicts the conclusion of Lemma 3.3.

□

Corollary 3.3: $||\gamma_2|| \le a*2^{b*l*n}$, for some constants a and b independent of l, and n.

Proof. Similar to the proof of Corollary 3.2.

□

According to Corollaries 3.1-3, we have:

Theorem 3.1: Given an arbitrary V in VASS(2,l,n) and its corresponding tree T, $\max\{||v||: \exists [x,p,A_x] \in T \text{ such that } v \in A_x\} = O(2^{c*l*n})$, for some constant c independent of V, l, and n.

Now, according to Lemma 3.2 and Theorem 3.1, we have:

Corollary 3.4: In T, the maximum gain in one step is bounded by $c_1 * 2^{2^{c_2 * l * n}}$, for some constants c_1 and c_2 independent of l and n.

Since the above quantity will be used frequently in the subsequent discussion, for ease of expression, let $\beta = c_1 * 2^{2^{c_2 * l * n}}$ hereafter. We now wish to derive an upper bound on the second quantity, $\max\{||x||:[x,p,A_x] \in T\}$. We first give the following lemma, which allows us to derive a recurrence relation for $h_k(l,n)$.

Lemma 3.8: Let $w=[x,p,A_x] \to w'=[x',p',A_{x'}]$ be a path in T. If $||x'|| > ((t+1)*\beta)^t * (||x||+1)$, then from w to w′ there must exist a <u>strongly monotonic</u> sequence with t nodes.

Proof. To prove this, we first show that given a path $w_1=[x_1,p_1,A_1] \to w_2=[x_2,p_2,A_2]$ such that $||x_2|| > (n+1)*\beta*||x_1|| > 0$, then from w_1 to w_2, either

(1) $\exists$ $w''=[x'',p'',A_{x''}]$ such that $||x''|| \le ||x_2||$ and $x'' > x_1$, <u>or</u>

(2) $\exists$ a strongly monotonic sequence consisting of n nodes.

Since the maximum gain in each step is at most β, there must exist a monotonic sequence $d_1,\ldots,d_{n*||x_1||}$ from w_1 to w_2. Let $D=\{<x,y>: 0 \le x \le ||x_1||, 0 \le y\}$ and $D'=\{<x,y>: 0 \le y \le ||x_1||, 0 \le x\}$. If (1) is false, then all d_i's, $1 \le i \le n*\beta$, must be in the area D (or D′). Then by the pigeon-hole principle, one line in D (or D′) must contain at least n nodes. In this case, (2) is true.

Now, we prove the theorem by induction on t.

Induction base: t=1. Trivial.

Induction hypothesis: Assume the assertion is true for t=k, $k \geq 1$.

Induction step: t=k+1.

Let $v=[y,q,A_y]$ be the first node from w to w′ such that $((k+2)*\beta)^{k}*(||x||+1)>||y||>((k+1)*\beta)^{k}*(||x||+1)$. According to the induction hypothesis, there exists a strongly monotonic sequence $s_1,\ldots,s_k=[x_k,p_k,A_k]$ on the path from w to v. Clearly, $||x_k|| \leq ((k+2)*\beta)^{k}*(||x||+1)$. Now, consider the path from s_k to w′. Since $||x'||>(k+2)*\beta||x_k||$, it must be true that either

(a) $\exists$ $v'=[y',q',A_{y'}]$ such that $||y'|| \leq ||x'||$ and $y'>x_k$, <u>or</u>

(b) $\exists$ a strongly monotonic sequence consisting of k+1 nodes.

In either case, the assertion is true. □

Lemma 3.9: $h_{k+1}(l,n) \leq c*2^{2^{d*l*n}}*(h_k(l,n))$, for some constants c and d independent of k, l, and n.

Proof. Consider a path σ: $s_0 \rightarrow s'$ in T such that s_0 is the root and in s′ the (k+1)-st loop vector is added. Let s″ be the node where the k-th loop vector is added. Let σ': $s'' \rightarrow s'$ be the subpath of σ starting at s″. It is easy to see that for every node $[x,p,A_x]$ in σ' (except the end node s′), $A_x=\overline{A}$, for some set $\overline{A}$, i.e., every node in σ' has the same loop set. Clearly, $||s''|| \leq h_k(l,n)$. Let $t=\beta^2*n$. Now, if $||s'||>((t+1)*\beta)^{t}*h_k(l,n)$, according to Lemma 3.8, there must exist a strongly monotonic sequence π: d_1, d_2, ..., d_{β^2+1} over the same state, say p. Now according to Lemma 3.3, there exist i and j, $1 \leq i \leq j \leq \beta^2+1$, $d_i \bullet\!< d_j$ -- a contradiction. Therefore, $h_{k+1}(l,n) \leq ((t+1)*\beta)^{t}*h_k(l,n)$, which is bounded by $c*2^{2^{d*l*n}}*h_k(l,n)$, for some constants c and d. This completes the proof. □

Corollary 3.5: $h_k(l,n)=O(2^{k*2^{c*l*n}})$, for some constant c independent of k, l, and n.

Since there will be at most 2^n+3 vectors in any A_x, the following theorem is obtained:

Theorem 3.2: For an arbitrary V in VASS(2,l,n) and its corresponding tree T, $\max\{||x||: [x,p,A_x] \in T\}=O(2^{2^{d*l*n}})$, for some constant d independent of V, l, and n.

We are now ready to construct an algorithm to generate a SLS representation of the reachability set of a given VASS. The reader, at this point, should recall that in the original Hopcroft-Pansiot algorithm, no upper bound is given for the size of the SLS representation, neither does it tell how quick the SLS can be generated. In what follows, we utilize the results obtained earlier in this section to construct a modified version of the Hopcroft-Pansiot algorithm. More precisely, we have:

Theorem 3.3: Given a VASS $V=(v_0,A,p_0,S,\delta)$ in VASS(2,l,n) and a state p in S, we can construct a SLS $\mathcal{SL}=\cup_{i=1}^{k}\mathcal{L}_i(x_i,P_i)$ in DTIME($2^{2^{c*l*n}}$), for some constant c independent of V, l, and n, such that,

(1) $\mathcal{SL}=\{x: (p,x)\in R(v_0,A,p_0,S,\delta)\}$,

(2) $k=O(2^{2^{d_1*l*n}})$, for some constant d_1 independent of k, l, and n,

(3) $\forall i$, $1\leq i\leq k$, $||x_i||=O(2^{2^{d_2*l*n}})$, for some constant d_2 independent of k, l, and n,

(4) $\forall i$, $1\leq i\leq k$, $|P_i|=O(2^n)$, where $|P_i|$ is the number of vectors in P_i,

(5) $\forall v\in P_i$, $1\leq i\leq k$, $||v||=O(2^{d_3*l*n})$, for some constant d_3 independent of k, l, and n.

Proof. First recall that each node in a Hopcroft-Pansiot tree T is a 3-tuple $[x,p,A_x]$. According to the algorithm and Theorems 3.1 and 3.2, $\forall$ $[x,p,A_x]$ in T,

- $|A_x|\leq c_1*2^n$ for some constant c_1 (because according to the algorithm at most 2^n+3 loop vectors can exist),
- $||x||\leq c_2*2^{2^{d_2*l*n}}$ for some constants c_2 and d_2 (Theorem 3.2),
- $\forall$ $v\in A_x$, $||v||\leq c_3*2^{d_3*l*n}$ for some constants c_3 and d_3 (Theorem 3.1). Furthermore, at most 3 loop vectors in A_x can be of that norm (the others are bounded by $O(n*2^l)$).

As a result, the number of possible <u>distinct</u> nodes in T is bounded by $O(2^{2^{d_1*l*n}})$, for some constant d_1. However, in the original tree construction ([12]) nodes are not necessarily distinct. This is due to the fact that, even if two different paths reach the same node, the rest of both paths still have to be explored separately (because one path may terminate earlier than the other). Note, however, that since the maximum norm of vectors that a path can reach is bounded by $c_2*2^{2^{d_2*l*n}}$, instead of checking the termination condition we can explore the entire tree (up to the above bound) so that only distinct nodes will appear in the tree. The new tree is generated as in the original algorithm with the following exceptions:

1. An axis vector γ is added to the loop set only if $||\gamma||\leq c_3*2^{d_3*l*n}$;
2. the terminating condition is not checked;
3. a new leaf $[y,q,A_y]$ is added only if
   a. $[y,q,A_y]$ does not occur elsewhere in the tree, <u>and</u>
   b. $||y||\leq c_2*2^{2^{d_2*l*n}}$.

Clearly this procedure can be done in DTIME($2^{2^{c*l*n}}$), for some constant c (since there are at most $O(2^{2^{d_1*l*n}})$ nodes). One can easily see that

- every node in the original tree must also appear in the new tree, and
- for every node d in the new tree, either d is in the original tree or there exists a node d′ in the original tree such that $d'\bullet\!< d$.

Consequently, the two trees represent the same SLS, i.e., (1) is true. The difference is that perhaps a more succinct SLS will be generated. Furthermore, it is easy to see that the description of the SLS satisfies conditions (2)-(5). This completes the proof. □

From Theorem 3.3 we want to show that the reachability, containment and equivalence problems for VASS(2,l,n) can be solved in DTIME($2^{2^{c*l*n}}$) for some fixed constant c. While the proof for the reachability problem for VASS(2,l,n) is quite straightforward, the complexity results for the equivalence problem for SLSs [13, 17] do not directly yield the desired upper bound for the containment and equivalence problems for VASS(2,l,n). However, we will show in the following that a careful application of the proof techniques in [17] yields the desired upper bound for the containment and equivalence problems also.

In view of Theorem 3.3, we consider in the following SLSs that are subsets of N^r where $r>0$ is fixed. Furthermore, each SLS $\mathcal{SL}=\cup_{i=1}^{k}\mathcal{L}(x_i,P_i)$ satisfies the conditions:

(C1) k is $O(2^{2^{d_1*N}})$,

(C2) $\forall i$, $1\leq i\leq k$, $||x_i||$ is $O(2^{2^{d_2*N}})$,

(C3) $\forall\, v\in P_i$, $1\leq i\leq k$, $||v||$ is $O(2^{d_3*N})$,

where d_1, d_2, d_3 are some fixed constants.

The following two lemmas will enable us to obtain the DTIME($2^{2^{c*l*n}}$) upper bound for the reachability, containment and equivalence problems for VASS(2,l,n).

Lemma 3.10: Let $\mathcal{SL}_1$ and $\mathcal{SL}_2$ be two SLSs that satisfy the conditions (C1-C3). Then $\mathcal{SL}_1 \neq \mathcal{SL}_2$ iff there exists a vector w in the symmetric difference of $\mathcal{SL}_1$ and $\mathcal{SL}_2$ so that $||w||$ is $O(2^{2^{c*N}})$, where c is some fixed constant (depending on r, d_1, d_2 and d_3 only).

Lemma 3.11: The membership, containment and equivalence problems for SLSs that satisfy conditions (C1-C3) can be solved in DTIME($2^{2^{c*l*n}}$), where c is a fixed constant (depending on r, d_1, d_2 and d_3 only).

In the following we proceed to show Lemmas 3.10 and 3.11. We will show Lemma 3.10 by applying the proof techniques in [17]. To this end we reproduce here some important technical notions from the theory of polyhedra (cf. [33] for a complete treatment). The reader is referred to [17] for the proofs of several facts that are used in establishing Lemma 3.10.

Let A be an m$\times$r matrix with integer coefficients. Let $b=(b(1),\ldots,b(m))\in Z^m$ and $x=(x(1),\ldots,x(r))$ be a vector of unknowns. For i=1,...,m, let A_i denote the ith row of the matrix A. If $A_i\neq \mathbf{0}$, then the rational solutions set of the linear equality $A_ix^T=b(i)$ and the linear inequality $A_ix^T\leq b(i)$ are called a *hyperplane* and a *halfspace*, respectively. The rational solutions set $\mathcal{S}$ of the finite system of linear inequalities $Ax^T \leq b$ is called a *polyhedron*. If $b=\mathbf{0}$, then S is called a *polyhedral cone*. If $A_i \neq \mathbf{0}$, then the hyperplane defined by $A_i x^T = b(i)$ is called a *boundary plane* of S.

The following facts state that every (finitely generated) cone, as defined at the beginning of this section, is also a polyhedral cone.

Fact 3.12. Let $C=C(\mathbf{0},P) \subseteq R^r$ be a cone such that $P \subseteq N^r$ is finite, and each $v \in P$ satisfies the condition that $||v||=O(2^{d_3 * N})$, where d_3 is some fixed constant. Then C may be represented as a polyhedral cone $Ax^T \leq \mathbf{0}$ so that $||A||=O(2^{c*N})$, where c is some fixed constant (depending on r and d only).

Proof. The proof of Fact 3.12 is similar to that of Lemma A.4 in [17], and is therefore omitted. $\square$

As a corollary of Fact 3.12, we obtain

Fact 3.13. Let $C=C(x,P) \subseteq R^r$ be a cone such that $x \in N^r$, $P \subseteq N^r$ is finite, $||x||=O(2^{2^{d_2 * N}})$, and each $v \in P$ satisfies $||v||=O(2^{d_3 * N})$, where d_2, d_3 are some fixed constants. Then C may be represented as a polyhedral cone $Ax^T \leq b$ so that

(1) $||A||=O(2^{c_1 * N})$,

(2) $||b||=O(2^{2^{c_2 * N}})$,

where c_1, c_2 are some fixed constants (depending on r, d_2, d_3 only).

Furthermore, if $v \in Z^r$ is some vector that does not belong to C, then there is some row A_i of A such that for the halfplane H defined by $A_i x^T \geq b(i)+1$ it holds that $v \in H$ and $H \cap C = \emptyset$.

Proof. Similar to the proof of Corollary A.5 in [17]. $\square$

We are know in position to prove Lemma 3.10.

Proof of Lemma 3.10. Let SL_1 and SL_2 be two SLSs in N^r that satisfy conditions (C1-C3). Suppose that the symmetric difference Δ of SL_1 and SL_2 is not empty. We want to show that Δ contains a "small" vector that witnesses the fact that $\Delta \neq \emptyset$.

Let w be some vector in Δ. Without loss of generality, we may assume that $w \in L \setminus SL_2$, where $L=L(x,P)$ is a linear set in SL_1 and $SL_2=\cup_{i=1}^{k} L(x_i,P_i)$. Let $C=C(x,P)$ and $C_i=C(x_i,P_i)$ for $i=1,\ldots,k$. Without loss of generality, let $w \in C_1 \cap \ldots \cap C_m$ and $w \notin C_{m+1} \cup \ldots \cup C_k$, where $1 \leq m \leq k$. For each $j=m+1,\ldots,k$, let H_j be the halfspace as obtained in Fact 3.13 for w and C_j. Then let C_w denote the intersection

$$C_w = C \cap C_1 \cap \ldots \cap C_m \cap H_{m+1} \cap \ldots \cap H_k$$

With these notations, we can show

Fact 1. C_w may be represented as a polyhedron of the form $C_w = \mathrm{conv}(E) + C(\mathbf{0},F)$, where $E \subseteq R^r$, $F \subseteq N^r$ are finite sets of nonnegative vectors, conv(E) denotes the convex hull

$$\{\textstyle\sum_{v\in E} \mathcal{G}_v v \colon \mathcal{G}_v \in R,\ \mathcal{G}_v \geq 0 \text{ and } \sum_{v\in E} \mathcal{G}_v = 1\},$$

and for subsets U, V $\subseteq R^r$, $U+V=\{u+v\colon u\in U, v\in V\}$.

Furthermore, E and F can be chosen so that

(1) $||F|| = O(2^{c_1 * N})$,

(2) $||E|| = O(2^{2^{c_2 * N}})$,

where c_1, c_2 are some fixed constants (depending on r, d_1, d_2 and d_3 only).

Proof of Fact 1. Similar to the proof of Lemma 2.1 in [17]. □

Now, consider w and C_w. We have $w \in C_w$. In what follows we will show that in C_w if $||w||$ is too large, i.e., w is "far away" from conv(E), then we can find a "small" witness w′ for the fact $\Delta \neq \emptyset$. To this end, consider the linear sets $L(\mathbf{0},P)$, $L(\mathbf{0},P_1)$, ..., $L(\mathbf{0},P_m)$ and the cone $C(\mathbf{0},F)$. Obviously, $C(\mathbf{0},F) \subseteq C(\mathbf{0},P) \cap C(\mathbf{0},P_1) \cap \ldots \cap C(\mathbf{0},P_m)$. Therefore, each $v \in F$ may be expressed as a nonnegative linear combination of $\leq$ r linearly independent vectors in P′, where P′ is any of the sets P, P_1, ..., P_m (cf. Caratheodory's Theorem for cones [33]). Hence, there are nonnegative integers λ, λ_1, ..., λ_m such that $\lambda v \in L(\mathbf{0},P)$, $\lambda_1 v \in L(\mathbf{0},P_1)$, ..., $\lambda_m v \in L(\mathbf{0},P_m)$, where λ, λ_1, ..., λ_m may be chosen, by Cramer's rule, as some subdeterminants of the matrices formed by vectors in P, P_1', ..., P_m, respectively. Thus, for some fixed constant c_3, λ, λ_1, ..., λ_m are $O(2^{c_3 * N})$. From this, it follows that the least multiple λ_v of λ, λ_1, ..., λ_m is $O(2^{2^{c_4 * N}})$ for some fixed constant c_4 (even when m may be doubly exponential in N). We therefore obtain

Fact 2. For each $v \in F$, there exists an integer λ_v of $O(2^{2^{c_4 * n}})$ such that $\lambda_v v \in L(\mathbf{0},P) \cap L(\mathbf{0},P_1) \cap \ldots \cap L(\mathbf{0},P_m)$, where c_4 is some fixed constant (depending on r, d_1, d_2 and d_3 only).

Let $G = \{\lambda_v v\colon v \in F\}$. Each $\lambda_v v$ is a "superperiod" from which w can be subtracted so that a "small" witness w′ can be obtained. We formalize this idea in the following. Suppose that $||w|| > ||E||$. Then C_w is an unbounded polyhedron, and F (or equivalently G) is not empty.

Consider the lattice points in C_w, i.e., elements in $C_w \cap N^r$. Let $u \in C_w \cap N^r$. By Caratheodory's Theorem for cones (cf. [33]), u may be expressed as $u = \sum_{y\in E} \mathcal{G}_y y + \sum_{z\in G'} \delta_z z$, where $\mathcal{G}_y$, $\delta_z \in R$, $\mathcal{G}_y$, $\delta_z \geq 0$, $\sum_{y\in E} \mathcal{G}_y = 1$, and $G' \subseteq G$ is a linearly independent subset. Therefore, $u' = u - \sum_{z\in G'} \lfloor \delta_z \rfloor z$ is $\leq$ u and

$u' \in C_w \cap N^r$. Let U denote the set of all such lattice points u' in $C_w \cap N^r$. Obviously, U is finite, and $||U||$ can be bounded in terms of $||E||$ and $||G||$. It can easily be seen that $||U||$ is $O(2^{2^{c_5 * N}})$ for some fixed constant c_5. Furthermore, it holds that $C_w \cap N^r = \cup_{u \in U, G' \subseteq G} \mathcal{L}(u,G')$, where G' runs over all subsets of $\leq$ r linearly independent vectors in G.

Fact 3. For each $u \in U$, the intersection $\mathcal{L}(u,G') \cap \mathcal{L}(,P)$ is a SLS of the form $\cup_{y \in Y} \mathcal{L}(y,G')$ so that $||Y||$ is $O(2^{2^{c_6 * N}})$ for some fixed constant c_6.

Proof of Fact 3. Similar to the proof of Lemma 2.2 in [17]. □

We are now in position to conclude the proof of Lemma 3.10. Observe that, $w \in (C_w \cap N^r) \cap \mathcal{L}(x,P)$. So for some $y \in Y$, $G' \subseteq G$, a subset of $\leq$ r linearly independent vectors, $w \in \mathcal{L}(y,G')$. Defining w' to be y, we have that $w' \in \mathcal{L}(x,P)$. On the other hand, it is clear that $w' \notin \mathcal{L}(x_1,P_1) \cup \ldots \cup \mathcal{L}(x_m,P_m)$, since w would belong to $\mathcal{L}(x_1,P_1) \cup \ldots \cup \mathcal{L}(x_m,P_m)$ otherwise. Thus, $w' \in \mathcal{L} \setminus \mathcal{SL}_2$, and this completes the proof of Lemma 3.10. □

Proof of Lemma 3.11. In view of Lemma 3.10, it suffices to show that the following membership problem can be solved in $DTIME(2^{2^{c*N}})$, where c is some fixed constant.

Input.

> A SLS $\mathcal{SL}$ satisfying conditions (C1-C3) and a vector $v \in N^r$ with $||v|| = O(2^{2^{c_1 * N}})$ for some fixed constant c_1.

Question. Does v belong to $\mathcal{SL}$?

This membership problem is reduced to the problem of checking the existence of a nonnegative integer solution of a system of equations $Ax^T = b$, where $A \in Z^{r \times m}$, $b \in Z^r$, $||A||$ is $O(2^{2^{d_3 * N}})$, m is $O(2^{d_4 * N})$ and $||b||$ is $O(2^{2^{c_1 * N}})$, where d_3 is the constant in condition (C3), and d_4 is some fixed constant (depending on r and d_3). From a result in [3], it follows that if such a system has nonnegative integer solutions, it has one whose entries are $O(2^{2^{c_2 * N}})$ for some fixed constant c_2. By exhaustive search, this, and hence the membership problem mentioned above, can be solved in $DTIME(2^{2^{c*N}})$, where c is some fixed constant. This completes the proof of Lemma 3.11. □

From Theorem 3.3 and Lemmas 3.10 and 3.11, we have:

Theorem 3.4: For VASS(2,l,n), the reachability, containment and equivalence problems can be solved in $DTIME(2^{2^{c*l*n}})$, for some constant c independent of l and n.

3.2. The Lower Bound

In what follows, we show that the upper bound we obtained for the Hopcroft-Pansiot algorithm in Section 3.1 is tight.

Theorem 3.5: There exists a VASS in VASS($2,l,3*n+4$) whose Hopcroft-Pansiot tree can reach a vector with norm $2^{2^{d*l*n}}$, for some positive constant d independent of l and n. Furthermore, the longest path in the tree can have at least $2^{2^{c*l*n}}$ nodes for some positive constant c independent of l and n.

Proof. Consider the VASS in Figure 3-3. (Without loss of generality, assume n is even.) Now consider the path shown in Figure 3-4. The computation proceeds by phases, where each phase contains n+1 stages (the last stage consists of states b_1 and b_2). For example, in stage j of phase i (assume j is even), the system starts at state a_j with the vector $<2^{(i*n+j)*l}+i,0>$. First, the transition $a_j \rightarrow (a_j, <-1,1>)$ is involved i times in order to obtain the vector $<2^{(i*n+j)*l},i>$. After that, $a_j \rightarrow (a_j, <-1,2^l>)$ will be applied repeatedly until the vector $<0,2^{(i*n+j+1)*l}+i>$ is obtained. Finally, $a_j \rightarrow (a_{j+1}, <0,0>)$ is used to enter the next stage. Proceeding in this manner, the vector $<2^{(i+1)*l*n},i>$ will be obtained in state a_n. The function of b_1, b_2 and their associated transitions is to increment the norm of the vector by 1 before the end of a phase. Now, one can easily see the following facts:

- V is in VASS($2,l,3*n+4$).
- No non-axis vector can exist in the loop set. (Since there is no p-loop and the first vector added is an axis vector.)
- The two axis vectors (in sequence) are $<2^{n*l},0>$ and $<0,2^{n*l}>$, respectively. (They are added when entering stage 1 of phase 1.)
- No redundant nodes can exist in the same stage. (This is because in the same stage one component is incremented while the other one is decremented.)
- No redundant nodes exist between different phases during phases 0 through 2^{l*n}-1. (Let v_i and v_j, $i<j$, be two vectors in different phases. It must be the case that $||v_j||-||v_i||=b*2^{l*n}+r$, where $0<r<2^{l*n}$ for some b. As a result, v_j can not be in $\mathcal{L}(v_i,\{<2^{n*l},0>,<0,2^{n*l}>\})$.)

Consequently, the computation can proceed, in a zigzag fashion (see Figure 3-5), 2^{l*n} phases without having redundant nodes. We can then conclude that the system can produce a vector of size $2^{2^{d*l*n}}$, for some positive constant d. Furthermore, the length of the path described above is $2^{2^{c*l*n}}$ for some positive constant c. □

Acknowledgment: We would like to thank Professor Vidal-Naquet for pointing out reference [25].

References

[1] Adleman, L. and Manders K., Computational Complexity of Decision Procedures for Polynomials, *Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science* (1975), pp. 169-177.

[2] Baker, H., Rabin's Proof of the Undecidability of the Reachability Set Inclusion Problem of Vector Addition Systems, MIT Project MAC. CSGM 79, Cambridge, MA, 1973.

[3] Borosh, I. and Treybis, L., Bounds on Positive Integral Solutions of Linear Diophantine Equations, *Proc. AMS 55*, 2 (March 1976), pp. 299-304.

[4] Chandra, A., Kozen, D. and Stockmeyer, L., Alternation, *JACM 28*, 1 (January 1981), pp. 114-133.

[5] Clote, P., The Finite Containment Problem for Petri Nets, to appear in *Theoret. Comp. Sci.*.

[6] Cardoza, E., Lipton. R. and Meyer, A., Exponential Space Complete Problem for Petri Nets and Commutative Semigroups, *Proceedings of the 8th Annual ACM Symposium on Theory of Computing* (1976), pp. 50-54.

[7] Crespi-Reghizzi, S. and Mandrioli, D., A Decidability Theorem for a Class of Vector Addition Systems, *Information Processing Letters 3*, 3 (1975), pp. 78-80.

[8] Davis, M., Matijasevic, Y. and Robinson, J., Hilbert's Tenth Problem. Diophantine Equations: Positive Aspects of a Negative Solution, *Proceedings of Symposium on Pure Mathematics 28* (1976), pp. 323-378.

[9] Grabowski, J., The Decidability of Persistence for Vector Addition Systems, *Information Processing Letters 11*, 1 (1980), pp. 20-23.

[10] Ginzburg, A. and Yoeli, M., Vector Addition Systems and Regular Languages, *J. of Computer and System Sciences 20* (1980), pp. 277-284.

[11] Hack, M., The Equality Problem for Vector Addition Systems is Undecidable, *Theoret. Comp. Sci. 2* (1976), pp. 77-95.

[12] Hopcroft, J. and Pansiot, J., On the Reachability Problem for 5-Dimensional Vector Addition Systems, *Theoret. Comp. Sci. 8* (1979), pp. 135-159.

[13] Huynh, D., The Complexity of Semilinear Sets, *J. Information Processing & Cybernetics 6* (1982), pp. 291-338.

[14] Huynh, D., The Complexity of the Equivalence Problem for Commutative Semigroups and Symmetric Vector Addition Systems, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (1985), pp. 405-412.

[15] Huynh, D., Complexity of the Word Problem for Commutative Semigroups of Fixed Dimension, *Acta Informatica 22* (1985), pp. 421-432.

[16] Huynh, D., The Complexity of the Membership Problem for Two Subclasses of Polynomial Ideals, to appear in *SIAM J. on Computing*, 1986.

[17] Huynh, D., A Simple Proof for the Σ_2^P Upper Bound of the Inequivalence Problem for Semilinear Sets, *J. Information Processing & Cybernetics 22* (1986), pp. 147-156.

[18] Ibarra, O., Leininger, B. and Rosier, L., A Note on the Complexity of Program Evaluation, *Mathematical Systems Theory 17*, 2 (1984), pp. 85-96.

[19] Karp, R. and Miller, R., Parallel Program Schemata, *J. of Computer and System Sciences 3*, 2 (1969), pp. 147-195.

[20] Kosaraju, R., Decidability of Reachability in Vector Addition Systems, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing* (1982), pp. 267-280.

[21] Lipton, R., The Reachability Problem Requires Exponential Space, Yale University, Dept. of CS., Report No. 62, Jan., 1976.

[22] Landweber, L. and Robertson, E., Properties of Conflict-Free and Persistent Petri Nets, *JACM 25*, 3 (1978), pp. 352-364.

[23] Mayr, E., Persistence of Vector Replacement Systems is Decidable, *Acta Informatica 15* (1981), pp. 309-318.

[24] Mayr, E., An Algorithm for the General Petri Net Reachability Problem, *SIAM J. Comput. 13*, 3 (1984), pp. 441-460.

[25] McAloon, K., Petri Nets and Large Finite Sets, *Theoret. Comp. Sci. 32* (1984), pp. 173-183.

[26] Mayr, E. and Meyer, A., The Complexity of the Finite Containment Problem for Petri Nets, *JACM 28*, 3 (1981), pp. 561-576.

[27] Mayr, E. and Meyer, A., The Complexity of the Word Problems for Commutative Semigroups and Polynomial Ideals, *Advances in Mathematics 46* (1982), pp. 305-329.

[28] Muller, H., Decidability of Reachability in Persistent Vector Replacement Systems, *Proceedings of the 9th Symposium on Mathematical Foundations of Computer Science* , LNCS 88 (1980), pp. 426-438.

[29] Muller, H., Weak Petri Net Computers for Ackermann Functions, to appear in *Elektronische Informationserarbeitung und Kybernetik.*

[30] Rosier, L. and Yen, H., A Multiparameter Analysis of the Boundedness Problem for Vector Addition Systems, *Fundamentals of Computation Theory 85*, LNCS 199 (1985), pp. 361-370. (To appear in *J. of Computer and System Sciences*, 1986.).

[31] Rosier, L. and Yen, H., Logspace Hierarchies, Polynomial Time and the Complexity of Fairness Problems Concerning ω-Machines, *Proceedings of the 3rd Annual Symposium on Theoretical Aspects of Computer Science,* LNCS 210 (1986), pp. 306-320.

[32] Rosier, L. and Yen, H., On the Complexity of Deciding Fair Termination of Probabilistic Concurrent Finite-State Programs, to be presented at the *13th International Colloquium on Automata, Languages and Programming,* France, July 1986.

[33] Stoer, J. and Witzgall, C., *Convexity and Optimization in Finite Dimensions, I,* (Springer Verlag, 1970).

[34] Van Leeuwen, J., A Partial Solution to the Reachability Problem for Vector Addition Systems, *Proceedings of the 6th Annual ACM Symposium on Theory of Computing* (1974), pp. 303–309.

[35] Valk, R. and Vidal-Naquet, G., Petri Nets and Regular Languages, *J. of Computer and System*

Sciences 23 (1981), pp. 299-325.

[36] Yamasaki, H., On Weak Persistency of Petri Nets, *Information Processing Letters 13*, 3 (1981), pp. 94-97.

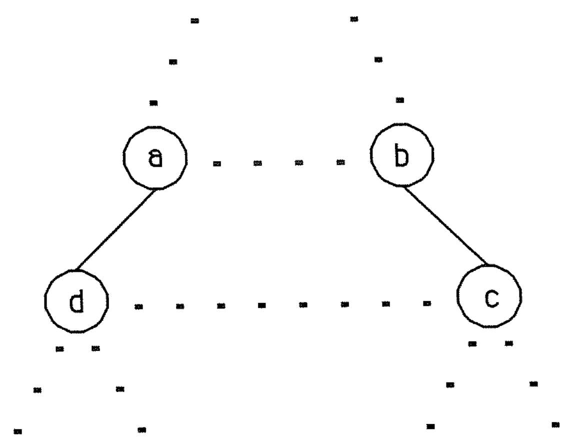

Figure 2-1: Swapping nodes c and d.

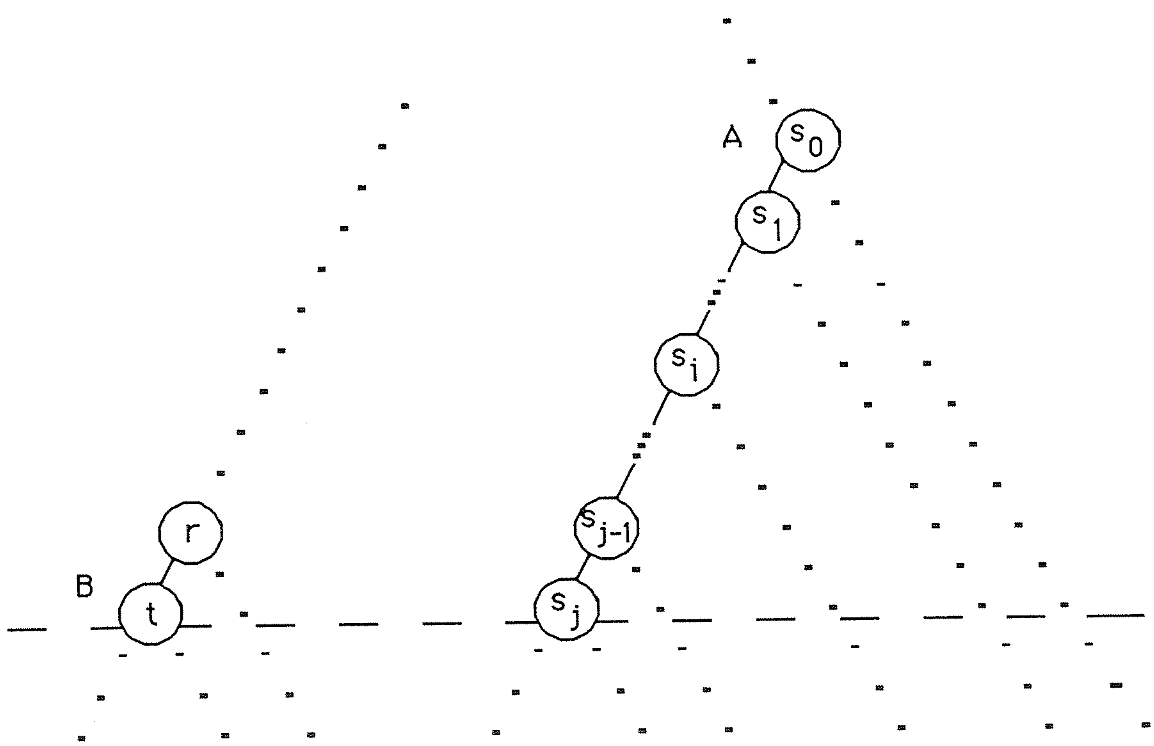

Figure 2-2: Swapping nodes s_0 and t.

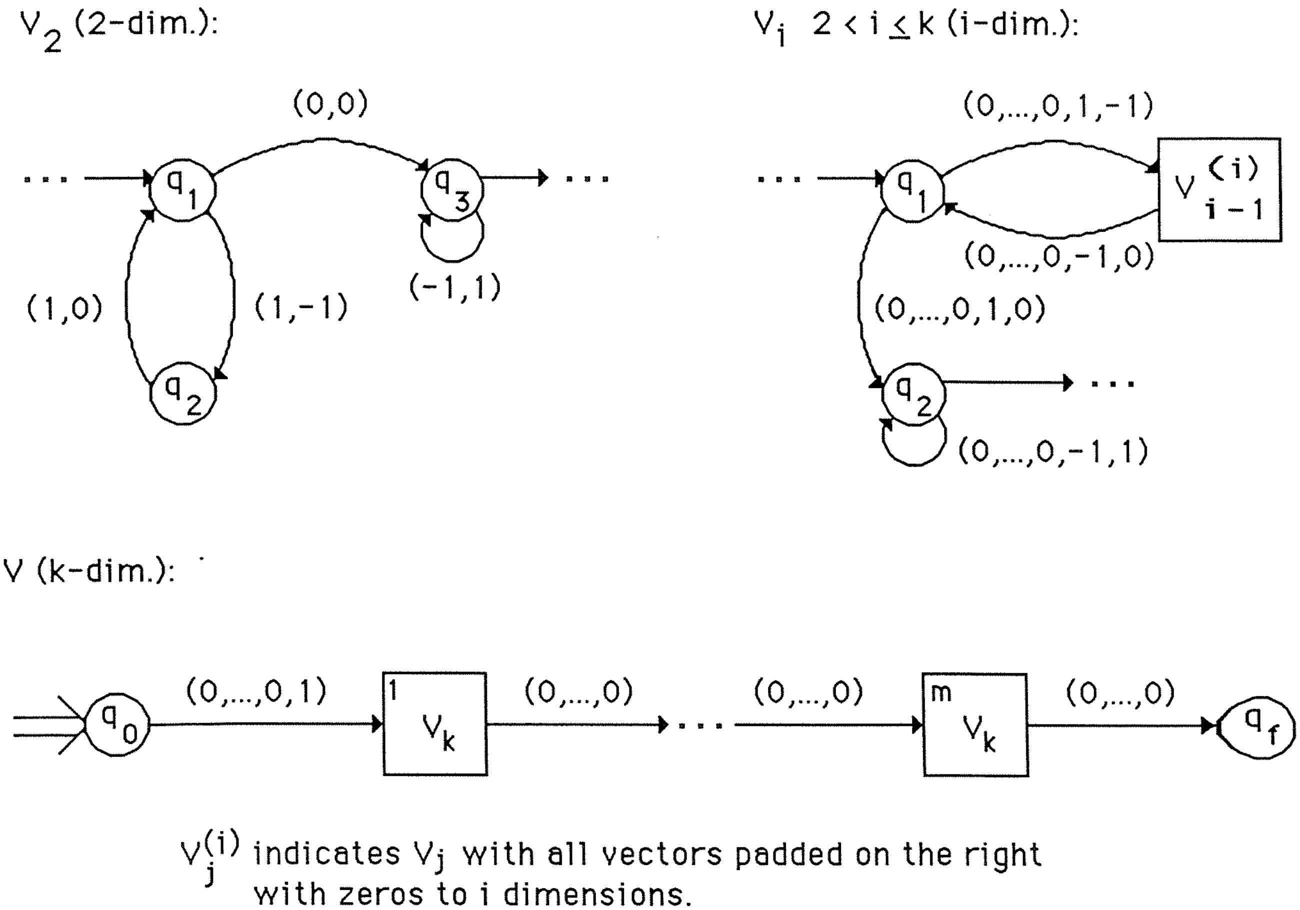

Figure 2-3: A VASS in BV(k,1,m*(2*k-1)+2).

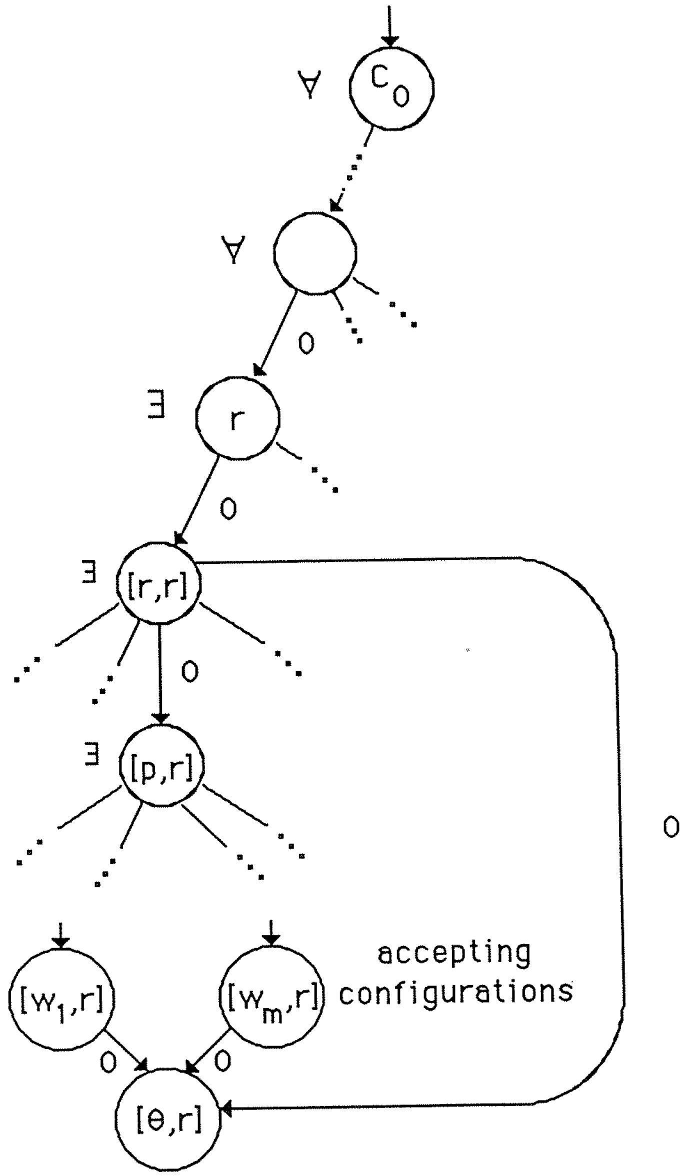

Figure 2-4: The VASS V.

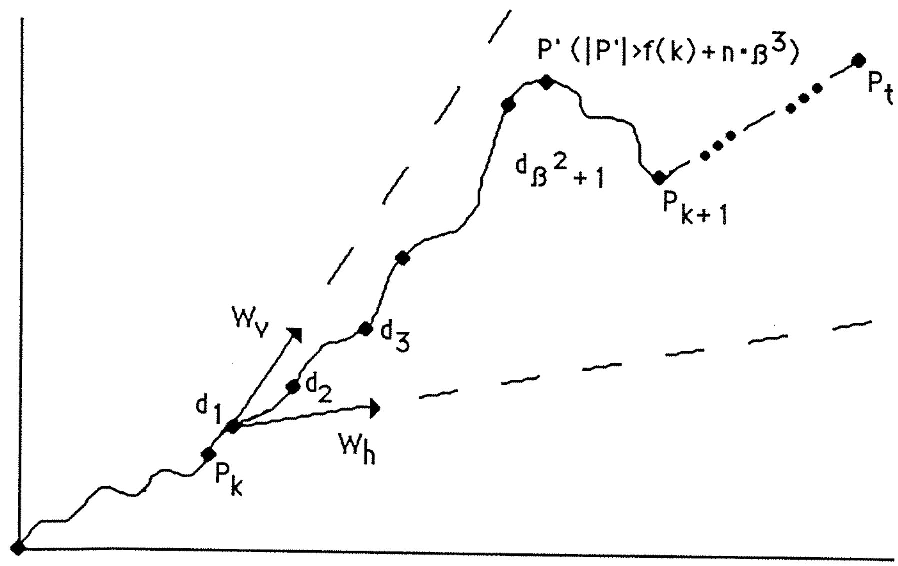

Figure 3-1: A path without an axis vector.

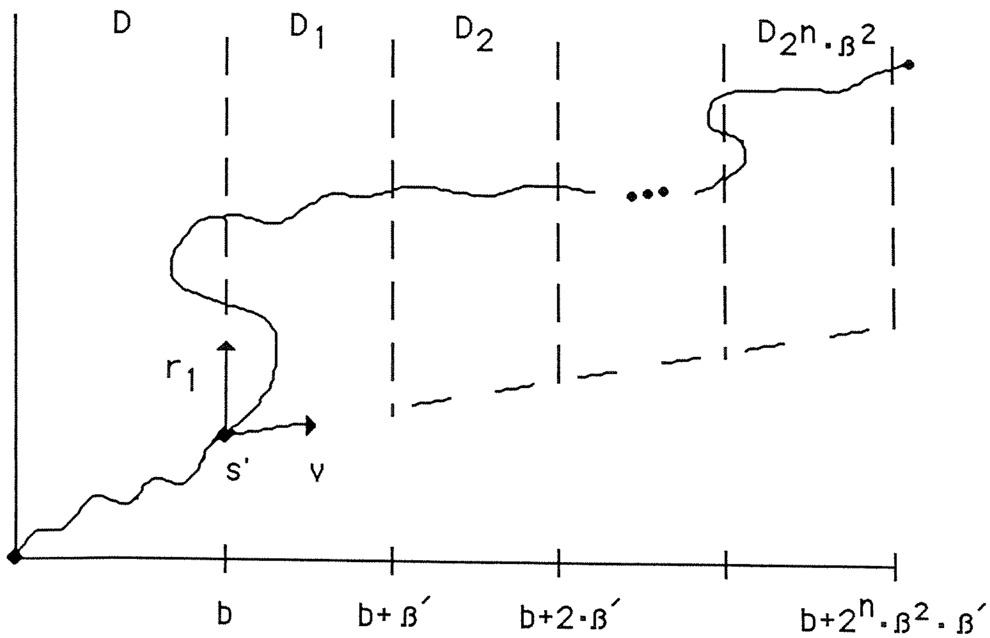

Figure 3-2: A path with an axis vector.

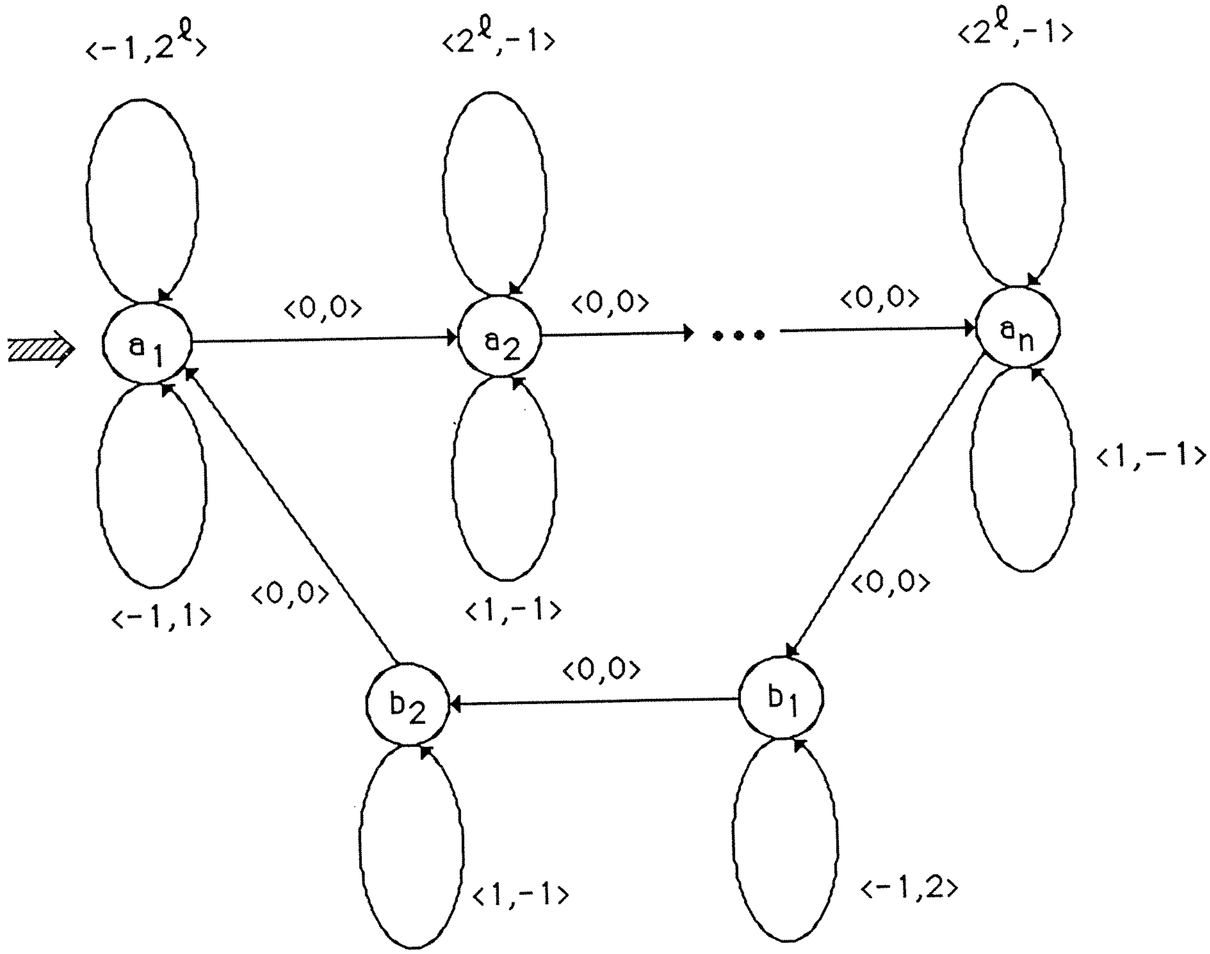

Figure 3-3: An example to illustrate the worst case.

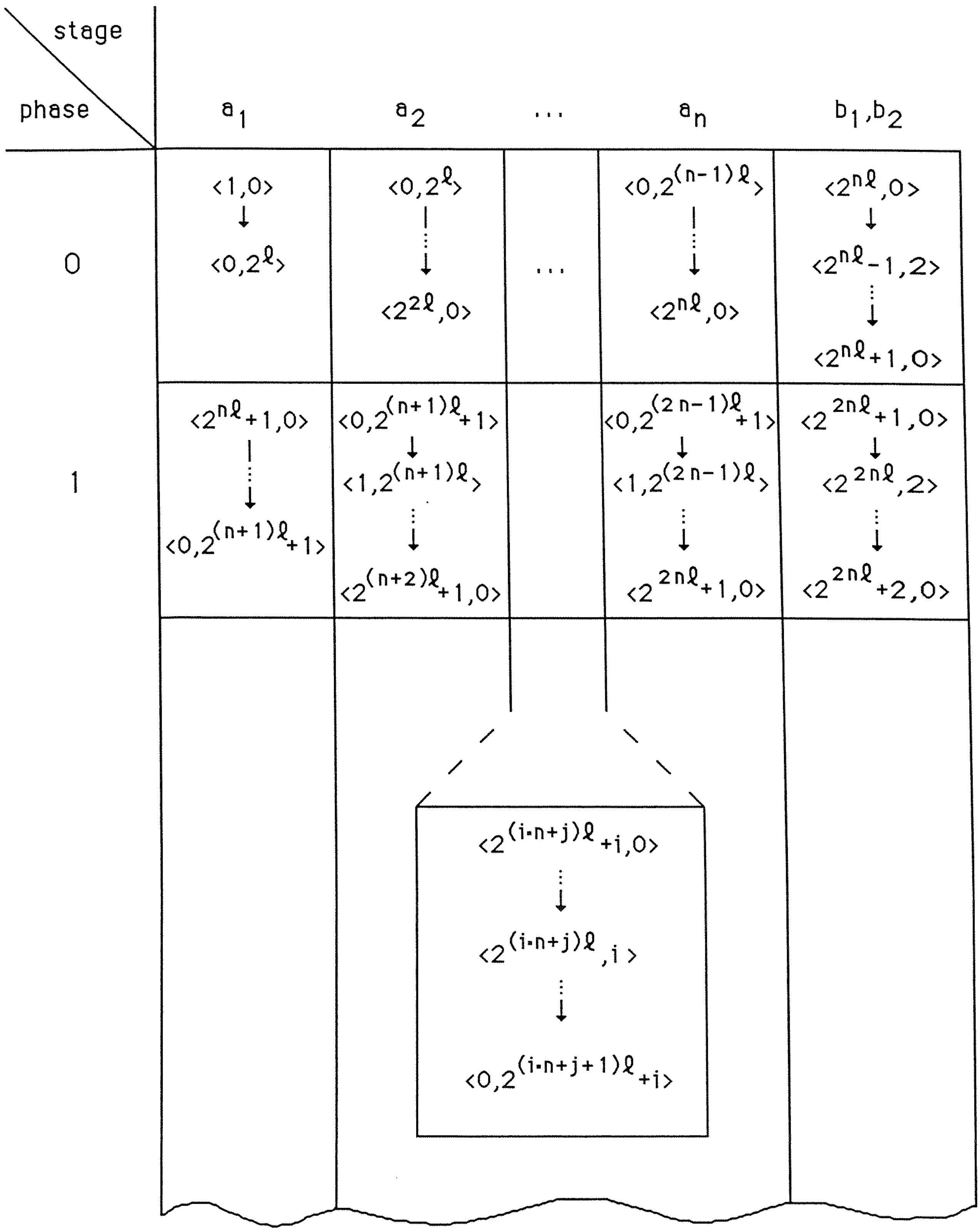

Figure 3-4: A path.

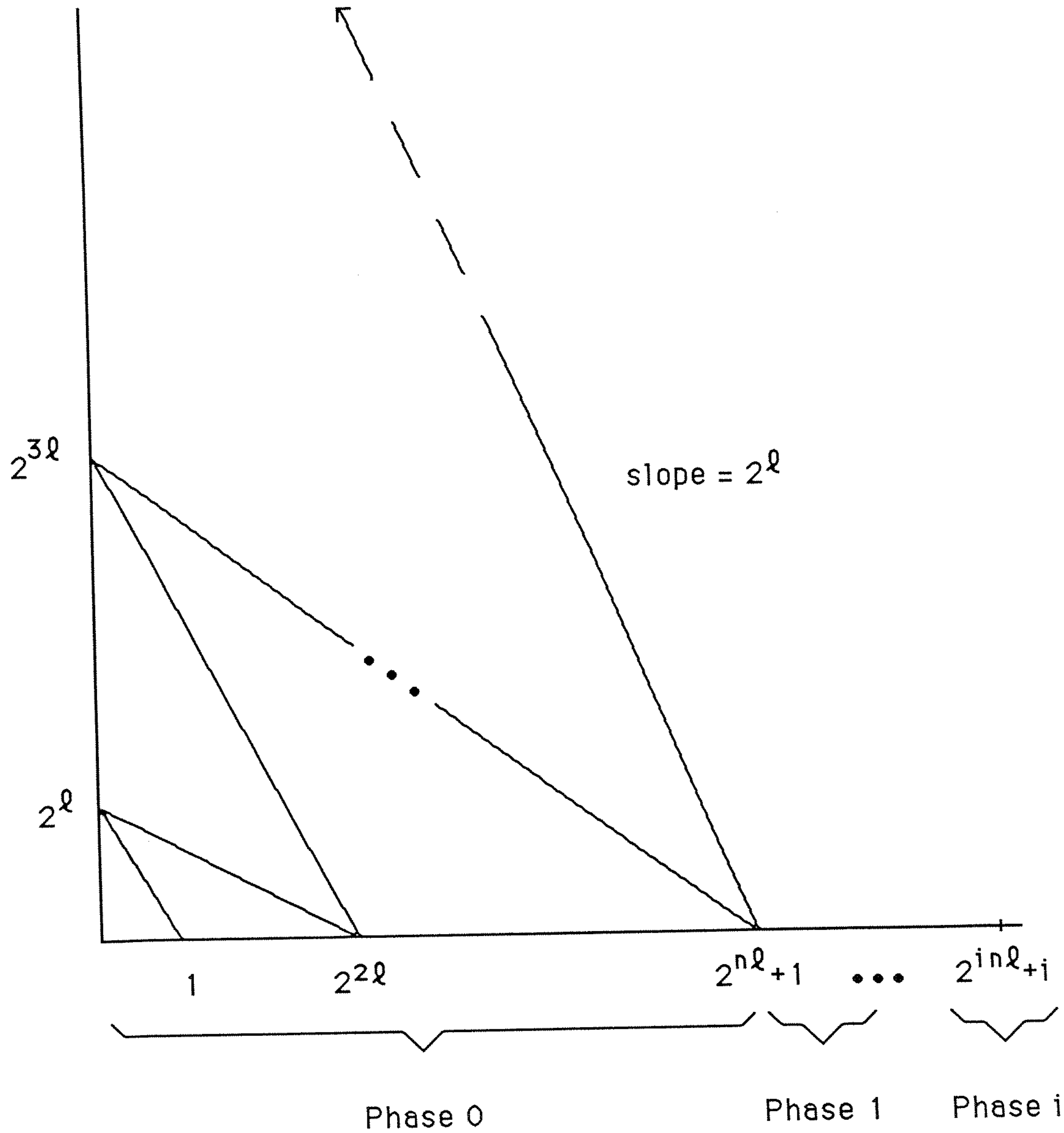

Figure 3-5: A pictorial description of the path in Figure 3-4.