A MULTIPARAMETER ANALYSIS OF THE BOUNDEDNESS PROBLEM FOR VECTOR ADDITION SYSTEMS

Louis E. Rosier and Hsu-Chun Yen

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

TR-85-03 January 1985

Abstract

Let VASS(k,l,n) denote the class of k-dimensional n-state *Vector Addition Systems with States*, where the largest integer mentioned, in an instance, can be represented in l bits. Using a modification of the technique used by Rackoff, we show that the *Boundedness Problem* (BP), for VASS(k,l,n), can be solved in $O((l+\log n)*2^{c*k*\log k})$ nondeterministic space. By modifying Lipton's result, a lower bound is then shown of $O((l+\log n)*2^{c*k})$ nondeterministic space. Thus, the upper bound is optimal with respect to parameters l and n, and is nearly optimal with respect to the parameter k. This yields an improvement over the result of Rackoff, especially when compared with the lower bound of Lipton. This is because the lower bound, of $O(2^{c*k})$ space, was essentially given for VASS(k,1,1). Now Rackoff's corresponding upper bound, just for the instances of VASS(k,1,1) constructed by Lipton, is no better than $O(2^{c*k^2*\log k})$ space. (In general, it can get much worse.) Our result, however, yields an upper bound of $O(2^{c*k*\log k})$, over the entire class. We also investigate the complexity of this problem for small, but fixed, values of k. We show that the BP is PSPACE-complete for 4-dimensional VASS's, and NP-hard for 2-dimensional VASS's. The above results can then be extended for the case without states. In particular, we are able to show that the BP is NP-hard for VASS(3,l,1) and PSPACE-complete for VASS(4,l,1). Extensions to related problems (e.g. covering and reachability) are also discussed.

Table of Contents

1. Introduction 1
2. The Upper Bound 3
3. The Complexity of the BP When k is a Fixed Known Constant 6
4. The Lower Bound 11

1. Introduction

In this paper, we investigate the complexity of the boundedness problem (BP) for *Vector Addition Systems* (VAS's), or equivalently *Vector Addition Systems with States* (VASS's). This problem was first considered in [14], where it was shown to be decidable. The algorithm presented there was, however, basically an unbounded search and consequently no complexity analysis was shown. Subsequently, in [17], a lower bound of $O(2^{c*m})$ space was shown, where m represents the dimension of the problem instance (and c is some constant). Finally, an upper bound of $O(2^{c*n*\log n})$ space was given in [23]. Here, however, n represents the size or number of bits in the problem instance. A close analysis of the result in [17] reveals, in terms of n, a lower bound of $O(2^{c*\sqrt{n}})$ space, since the size of the systems constructed in [17] required $O(m^2)$ bits.

The VASS model was introduced in [10], where it was shown that a k+3-dimensional VAS could simulate a k-dimensional VASS. Although, the addition of states adds no expressive power to the notion of VAS, it does add to the functional ease in which these systems can be used to model processes. For example, these systems can be used to simulate or model networks of CFSM's (c.f. [4,6,7,24,27]), where each machine is constrained to be able to send only a single type of message to every other machine in the network. The BP for such networks of CFSM's was first considered in [4], where an unbounded search algorithm, similar to the one in [14], was presented. A faster algorithm was later shown for networks of 2 such CFSM's in [27]. As it turns out, it is reasonably easy to show that such a network of k CFSM's can be simulated by a 2*k-dimensional VASS where the number of moves defined in the VASS is proportional to the number of moves defined in the network. (In fact, the maximum size of any integer mentioned in the definition of the simulating VASS need be no more than 1.) Likewise, a k-dimensional VASS, where the maximum size of any integer mentioned in the VASS is 1, can be simulated by such a network of CFSM's. The simulating network requires no more than k+1 machines and, as before, the number of moves defined in the network is proportional to the number of moves defined in the VASS. (In each case the number of states in the VASS is polynomially related to the sum of the number of states in the CFSM's.)

In [8], the deadlock detection problem was considered for the class of networks consisting of k CFSM's where each machine is constrained to send only a single type of message (hereafter this class of networks is referred to as CFSM(k)). In a later paper enlarging upon these results, the BP for CFSM(k) was also considered[6]. It was noted there that the result of [23] could be used to show that the BP for CFSM(k) was solvable in PSPACE, providing k was considered to be a fixed known constant. It was left open whether, in such cases, the result could be improved to PTIME.

In this paper, we consider the complexity of the BP for VASS's with respect to the following three natural numeric parameters of a given instance:

1. the dimension

2. the maximum size of any integer mentioned in the description, and

3. the number of states.

For ease of expression, we let VASS(k,l,n) denote the class of k-dimensional n-state VASS's where the largest integer mentioned, in an instance, can be represented in l bits. In section 2, we use a modification of Rackoff's technique to show that the BP, for VASS(k,l,n), can be solved in $O((l+\log n)*2^{c*k*\log k})$ nondeterministic space. (All of our results are expressed with respect to nondeterministic complexity classes unless otherwise stated. This could also be said for [17] and [23], although in their case such algorithms can be determinized without altering the complexity class, as only the constant c gets changed[26]. Our subsequent discussion, however, involves subexponential complexity classes where this is

not necessarily the case.) This offers a slight improvement over the result in [23], especially when compared with the best known lower bound. This is because the lower bound, of $O(2^{c*k})$ space, was essentially given for VASS(k,1,1). Now the corresponding upper bound from [23], just for the instances of VASS(k,1,1) considered in [17], is no better than $O(2^{c*k^2*\log k})$ space. (In general, it can get much worse.) Our result, however, yields an upper bound of $O(2^{c*k*\log k})$, over the entire class.

Whenever k is a fixed known constant, however, the algorithm requires at most nondeterministic linear space in terms of l. In section 3, we show that the BP is PSPACE-complete for 4-dimensional VASS's. Although we are unable to provide the corresponding result for either 2 or 3-dimensional VASS's, we are able to show that the problem is NP-hard for these classes. The above results can then be extended for the case without states. In particular, we are able to show that the BP is NP-hard for VASS(3,l,1) and PSPACE-complete for VASS(4,l,1). Consequently, it is unlikely that the problem can be solved in PTIME with respect to the parameter l. When k=1, however, the BP can be solved in PTIME. When both k and l are considered to be fixed known constants the problem becomes NLOGSPACE-complete. Hence, the complexity of the algorithm, given in section 2, is unlikely to be improved with respect to the parameter n either. Now recall that for each member of CFSM(k), the simulating VASS is such that the parameter l is 1. Hence, the BP for CFSM(k) can be solved in NLOGSPACE (and consequently PTIME[3]), providing k is considered to be a fixed known constant.

Now in the last section, we show how the proof of [17] can be augmented to obtain a lower bound of $O((l+\log n)*2^{c*k})$ nondeterministic space for VASS(k,l,n). In order to do this, we define a class of problems each of which has 3 natural numeric parameters, say k, l and n, whose nondeterministic space complexity is bounded by $O(2^{c*k}*(l+\log n))$, for some positive constant c. We then show that the BP for VASS(k,l,n) is as "hard" as any problem in this class. Our goal here is to provide a simultaneous lower bound over the three parameters, as opposed to examining the lower bound when one (or two) of the parameters is fixed (as was done in section 3). (Thus, an algorithm for this problem with nondeterministic space complexity $O(2^{c*k}+l+\log n)$ cannot exist. Note, however, that the existence of such an algorithm was not precluded by the results given in the previous section.) Thus, the algorithm presented in section 2, is optimal with respect to parameters l and n, and is nearly optimal with respect to the parameter k. We do not know at this time, however, whether the log k factor can be eliminated. This result can also be used to show that although the BP for CFSM(k) is solvable in PTIME (for fixed k), it is likely that the order of the polynomial depends on k.

The results of this paper thus provide improved upper and lower bounds for the VASS boundedness problem. The multiparameter analysis used here also sheds some light on the gap between the previously best known upper and lower bounds of [23] and [17]. As it turns out this gap is at least partially caused by the complexity being measured as a function of the input size. In terms of that measure it is questionable whether the gap can be improved.

Lastly, we briefly mention how our results can be extended to some other related problems concerning VASS's. For example, the lower bounds established in Section 4 hold for the covering and reachability problems. (Variations of those given in Section 3 hold as well.) Using a similar approach, as was used in Section 2, an upper bound of $O((l+\log n) * 2^{c*k*\log k})$ can be shown for the covering problem. (Finding better upper/lower bounds for this problem was mentioned as an open problem in [18].) Previously, the best known bounds for this problem were essentially the same as those for the boundedness problem. See [17, 18, 23].

2. The Upper Bound

Let Z (N) denote the set of integers (nonnegative integers, respectively). 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 $z=v_0+v_1+...+v_j$ for some $j\geq 0$, where each v_i $(1\leq i\leq j)$ is in A and for each $1\leq i\leq j$, $v_0+v_1+...+v_i\geq 0$. 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: $S\rightarrow S\times A$ is the transition function, and p_0 is the initial state. 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 vector t in state p and yields the vector (or, point in k space) t+x, in the state q, provided that $t+x\geq 0$. The *reachability set* of the VASS (v_0,A,p_0,S,δ) is the set $R(v_0,A,p_0,S,\delta)= \{(p,x)|\ (p,x) \in \delta^*(p_0,v_0)\}$, where δ^* represents the reflexive transitive closure of δ. The *boundedness problem* (BP) for a VAS (or a VASS) is to determine whether the reachability set, of a given instance, is finite.

The boundedness problem for vector addition systems was first studied in [14], where a nonprimitive recursive decision procedure was proposed. This procedure is based on a search algorithm which generates the reachability set $R(v_0,A)$, and at the same time, attempts to find a "pumpable loop", that can exploited to reach an arbitrary number of distinct vectors. It was shown that either the reachability set is finite or such a loop exists. Hence, the algorithm must eventually terminate. Unfortunately, the size of $R(v_0,A)$ is not bounded by a primitive recursive function of the size of (v_0,A). Rackoff, in [23], basically showed that if such a pumpable loop could be executed by some computation of the system, then it could be executed by a "short" computation. As a result, Rackoff presented an algorithm which required at most $2^{c*m*\log m}$ space, for some constant c. Here, the parameter m represents the size (i.e. number of bits) of the instance. Actually, this parameter, upon careful examination, can be decomposed into three natural parameters k, l and n, where k is the dimension of the VAS, l is the maximum length of the binary representation of each component in A (the set of addition rules) and n is the number of rules (i.e., the number of vectors in A). (The reader should note that these parameters are equally natural for Petri Nets - a notational variant of VAS's. There k would represent the number of places, l the maximum number of inputs or outputs of a transition and n the number of transitions. See e.g. [22].) However, it is always the case that $n \leq (2*2^l+1)^k$. Hence, for simplicity, we use VAS(k,l) to denote the class of VAS's with parameters k and l. Later, when we discuss VASS's, we introduce another parameter m to represent the number of states. It should be clear that the three parameters k, l, and m are mutually independent. Therefore, the class of VASS's with parameters k, l and m will be denoted as VASS(k,l,m).

Our motivation for separating the parameter m into (k,l) (or (k,l,m)) comes from an observation that, in some cases when one or two parameters are fixed, the exponential space result can be improved. (It also serves to analyse what effect large numbers have with respect to the complexity of this problem.) For example, consider a network of two CFSM's in which each machine sends only one type of message to the other machine. It is not difficult to see, that this kind of network can be simulated by an instance of VASS(2,1,m), for some m. Furthermore, the BP for this kind of network is known to be NLOGSPACE-complete[24]. This seems to suggest that, the parameters k, l and m may contribute differently to the complexity of the problem. Therefore, by considering the complexity of the BP, concurrently, with respect to these three parameters, we hope to develop new insight into the problem. As a result, we are able to show for VAS(k,l), an upper bound of $O((l+\log n)*2^{c*k*\log k})$ nondeterministic space, where c is a constant and n is the number of addition rules contained in the instance. Recall that in Rackoff's result, the BP can be solved in $O(2^{d*m*\log m})$ space where m is the size of the VAS. In general, m can be greater than k^2, and hence, the use of distinct parameters does improve the complexity result to some degree.

In the remainder of this section, we show how to obtain an upper bound of $O((l+\log n)*2^{c*k*\log k})$ nondeterministic space, where c is a constant and n is the number of rules, for the boundedness problem of a VAS(k,l). What we actually show, as did Rackoff, is a bound on the length of the shortest system computation, if one exists, which will execute a pumpable loop. In fact, our proof is a modification of the one in [23]. As a result, many of the details in the following proof are omitted especially when they are similar to those in [23].

Before going into detail, we require the following definitions. Most of them are the same as in [23]. The following notation is also taken, more or less, from [23]. For a VAS (v,A), a finite sequence of vectors $w_1,w_2,...,w_m \in Z^k$ is said to be a *path* in (v,A), of length m, if $w_1=v$ and if $w_{i+1}-w_i \in A$ for all i, $1\leq i<m$. Let $w \in Z^k$ and $0\leq i\leq k$. The vector w is *i-bounded* if $w(j)\geq 0$ for $1\leq j\leq i$. If $r \in N^+$ is such that $0\leq w(j)<r$ for $1\leq j\leq i$, then w is called *i-r bounded*. Let $p = w_1,...,w_m$ be a sequence of vectors, we say p is i-bounded (i-r bounded) if every member in p is i-bounded (i-r bounded). Moreover, if $w_j<w_{m'}$ for some j, $1\leq j<m$, then p is said to be *self-covering*. p is called an *i-loop* if in the first i places $v_1=v_m$ and $v_{j_1}\neq v_{j_2}$ for all $1\leq j_1<j_2\leq m$, i.e., p is a path such that the start and end vectors have their first i components identical and no other intermediate points have this property. The loop value of p is defined to be v_m-v_1. Let $0\leq i\leq k$. For each $v \in Z^k$, define m'(i,v) to be the length of the shortest i-bounded, self-covering path in (v,A), if one exists; otherwise, define m'(i,v)=0. Define $g(i)=\max\{m'(i,v) \mid v \in Z^k\}$. This function g (of A) then, will give us a bound on the length of the shortest computation which can execute a pumpable loop. Note that this upper bound does not depend on the start vector v.

To prove the upper bound result, we need the following lemma concerning the bounds of solutions of linear equations. The lemma is from [23]. (The proof is essentially from [2].)

Lemma 2.1: Let d_1, $d_2 \in N^+$, let B be a $d_1\times d_2$ integer matrix and let b be a $d_1\times 1$ integer matrix. Let $d\geq d_2$ be an upper bound on the absolute values of the integers in B and b. If there exists a vector v $\in N^d2$ which is a solution to $Bv\geq b$, then for some constant c independent of d, d_1, d_2, there exists a vector v $\in N^d2$ such that $Bv\geq b$ and $v_i<d^{cd}1$ for all i, $1\leq i\leq d_2$.

Lemma 2.2: If there exists an i-r bounded, self-covering path in (v,A), then there exists an i-r bounded, self-covering path with length $< (r*2^l)^{k^c}$, for some constant c.

Proof : Let $v_1,...,v_{m_0},w_1,...,w_{m_1}$ be an i-r bounded self-covering path and $w_1<w_{m_1}$. First, consider the path $v_1,...,v_{m_0}$. Clearly, we can assume $m_0\leq r^i\leq r^k$; since otherwise, there exists some i-loop and then we can make it short. Next, consider the path $w_1,...,w_{m_1}$. This path, clearly, can be decomposed into a path s and some i loops, such that the length of $s\leq(r^k+1)^2$. Let the loop values of those i-loops be $l_1,...,l_p$. Furthermore, it should be clear that a loop value is just the sum of at most r^k members of A, and therefore each place of l_i is of absolute value $\leq 2^l r^k$. Hence, there are at most $(2(2^l r^k)+1)^k$ distinct loop values. To find a shorter path, it suffices to find $n_1,...,n_p$ such that $n_1l_1+...+n_pl_p+(s)>0$ where (s) is the difference of the end and start points of the path s. By letting $d_1=k$ and $d=(2^l r)^{c'k^2}$ and using lemma 2.1, we are able to find $n_1,...,n_p$ such that $n_i\leq(2^l r)^{k^{c''}}$, and therefore, $m_1\leq(2^l r)^{k^c}$, for some c. By combining the above results, the lemma is proved. □

Lemma 2.3: $g(0)<(2^l n)^{ck}$, for some constant c.

Proof : Recall that n is the number of rules in A. To evaluate g(0), it suffices to solve $m_1v_1+...+m_nv_n>0$ where $A=\{v_1,...,v_n\}$. Choose $d_1=k$ and $d=2^l n$. Using lemma 2.1 we can find a solution with $m_i<(2^l n)^{c'k}$. Clearly then the lemma is proved. □

Lemma 2.4: $g(i+1)<(2^{2l}*g(i))^{k^c}$, for some constant c.

Proof :

(Case 1:) If there is an (i+1)-2^l*g(i) bounded self-covering path in (v,A), then from lemma 2.2, there exists a short one with length $<(2^l*2^l*g(i))^{k^c}$.

(Case 2:) Otherwise, let $v_1,...,v_{m_0},v_{m_0+1},...,v_n$ be the path such that v_{m_0} is the first one not $2^l g(i)$ bounded. Without loss of generality, assume that $v_{m_0}(i+1)>(2^l g(i))$. Now no two of $v_1,...,v_{m_0}$ can agree on the first i+1 places, for then the sequence could be made even shorter. Hence, $m_0<(2^l g(i))^{i+1}$. Let p be an i-bounded self-covering path in (v_{m_0},A) of length $<$ g(i). Since $v_{m_0}(i+1)>2^l g(i)$ and since each place in each vector in A is at most 2^l in absolute value, p must be also (i+1)-bounded. So the path $v_1,...,v_{m_0-1}$,p is an (i+1)-bounded, self-covering path of length $(2^{2l}g(i))^{i+1}+g(i) < (2^{2l}g(i))^{k^c}$. □

Theorem 2.1: The BP can be decided in $O(2^{c*k*\log k}*(l+\log n))$ nondeterministic space, for some constant c.

Proof : By recursively applying lemmas 2.3 and 2.4, it is easy to derive the result that $g(k)\leq(2^l*n)^{2^{c*k*\log k}}$. This means if a VAS is unbounded, there must exist a path, of length no more than $(2^l*n)^{2^{c*k*\log k}}$, that leads to the "pumpable loop". Therefore, the BP can be solved in $O((l+\log n)*(2^{c*k*\log k}))$ nondeterministic space. □

Now consider a VASS in the class VASS(k,*l*,m). Clearly the number of possible rules is no more than $m*(2*2^l+1)^k$. Then according to Theorem 2.1, we have the following corollary:

Corollary 2.1 : The BP for the class VASS(k,*l*,m) can be solved in $O((l+\log m)*2^{c*k*\log k})$ nondeterministic space, for some constant c.

Notice that the number of edges, between two nodes in such a graph, can be as great as $2^{l+1}+1$. Thus, the actual size or number of bits in the description, of a VAS (or VASS), can be exponential in *l*. Note, however, that the size of the systems constructed in the previous theorem are only polynomial in *l*. What's, perhaps, somewhat surprising is that such an increase in the density of edges does not alter the complexity, of the problem, in terms of the parameters k, *l*, and m.

3. The Complexity of the BP When k is a Fixed Known Constant

In this section, we focus our attention on the BP for VASS(k,l,m) when k is fixed. A summary of these results can be found in Figure 3.1. We show that the boundedness problem is PSPACE-complete for $k \geq 4$. Consequently, it is unlikely that one can improve the complexity of the aforementional algorithm with respect to the parameter l. At this time, we do not know whether the PSPACE-complete result can be improved for the case when k=2 or 3. However, we are able to show that it is NP-hard when k=2. Later, we show that when k=1, there exists a PTIME algorithm. If l is also fixed, then the problem becomes NLOGSPACE-complete. However, we do not know whether the previous PTIME result can be improved to NLOGSPACE. Finally, we consider the case where k=1 such that zero detection is allowed. (When k=2, this augmentation results in the BP becoming undecidable[7,9,22].) In this case, the BP becomes NP-complete. These results should be compared with earlier PSPACE-hard results concerning VAS's (or VASS's) in the literature [12], where the parameter k instead of l is allowed to vary.

Theorem 3.1 : The boundedness problem for k-dimensional VASS's, when $k \geq 4$, is PSPACE-complete.

Proof : The result of being in PSPACE was shown in section 2. It suffices, therefore, to show that it is PSPACE-hard.

Consider a nondeterministic linear bounded automata (LBA, for short) $W=<Q,\Sigma,\Gamma,\delta,q_0,\#,\$,F>$, where

Q is the (finite) set of states,

Σ is the (finite) set of input symbols,

Γ is the finite set of allowable tape symbols,

δ is the transition or next move function,

q_0 is the initial state,

and $ are the left and right endmarkers, respectively,

$F \subseteq Q$ is the set of accepting states.

Let $l=\#a_1,\ldots,a_n\$$ be an input to W. Without loss of generality, we assume that each a_i has value 0 or 1. Then one can construct a 4-dimensional VASS V to simulate the execution of W on l in such a way that W accepts l iff the VASS V is unbounded. Since the membership problem for LBA's is well known to be PSPACE-complete, the boundedness problem for k-dimensional VASS's ($k \geq 4$) is, therefore, PSPACE-hard.

Basically, the state of V contains the information of the current state and the current head position of W. The 4-dimensional vector <A,A',B,B'> is used to encode the contents of the tape in such a way that, at some instant, the numbers B and A represent the tape contents to the left and to the right of the current head position, respectively, and B' and A' represent the complement of the tape in a similar way. More specifically, assume that the current head position is at i, then the encodings are as follows: (assume $a_0=a'_0=a_{n+1}=a'_{n+1}=0$)

$B= \Sigma_{j=0}^{i-1} a_j * 2^{n+1-j}$,

$A= \Sigma_{j=i}^{n+1} a_j * 2^{n+1-j}$,

$B' = \Sigma_{j=0}^{i-1} a'_j * 2^{n+1-j}$,

$A' = \Sigma_{j=i}^{n+1} a'_j * 2^{n+1-j}$.

where a'_j is the complement of a_j, for $1 \leq j \leq n$.

The crux of the simulation is the ability to read the i-th symbol on the tape by doing some arithmetic computation on the numbers A, A', B, and B'. This can be done by introducing the following transitions: Assuming that the head position is at i and the current state is p.

t_0: $p \rightarrow (p_0, <0,-2^{n+1-i},0,0>)$

t_1: $p \rightarrow (p_1, <-2^{n+1-i},0,0,0>)$

Clearly, the transition t_0 (t_1) is executable iff a_i equals 0 (1). Now, consider a transition t of W:

t: $(p,a_t) \rightarrow (q,a_l,\Delta)$, where Δ (=0, -1, or +1) indicates the direction of the head movement.

In what follows, we show how the transition t will be simulated on the VASS. Without loss of generality, we assume that a_t=0 and a_l=1. The other three cases can be handled in a very similar way. Recall that Δ stands for the direction of the head movement. Therefore, we have the following three cases:

1. **no move :**

   We add the following transitions to the VASS. t_0 can be executed only when a_i=0. t_m is used to update the state as well as the input tape.

   t_0: $p \rightarrow (p_0, <0,-2^{n+1-i},0,0>)$

   t_m: $p_0 \rightarrow (q, <+2^{n+1-i},0,0,0>)$

2. **move right :**

   We introduce the following transitions to the VASS. Notice that in t_m, the number 2^{n+1-i} is added to the position B. This corresponds to the fact that, a "1" has been written on the input tape and the input head has been moved to the right one position. Here we also assume that the "q" contains the information of the current state and the input head position.

   t_0: $p \rightarrow (p_0, <0,-2^{n+1-i},0,0>)$

   t_m: $p_0 \rightarrow (q, <0,0,+2^{n+1-i},0>)$

3. **move left :**

   The left_move transition is a bit more involved, and consists of the execution of 2n stages in V, as shown in Figure 3.2. The first two steps copy the updated ith bit from vector A (A') to vector B (B'). The remaining bits of A (A') are then copied unchanged to B (B'). Then they are all copied back from B (B') to A (A'). Finally, all bits left of the i-1st bit are copied from A (A') to B (B'). Now the simulation is ready to continue since the vector is now positioned correctly with respect to the input head.

Finally, for each accepting state q_f in F, we introduce the transition

$q'_f \rightarrow (q'_f, <+1,+1,+1,+1>)$

to the corresponding state q'_f in V. Clearly then, W accepts the input $\#a_1,...,a_n\$$ iff V is unbounded. In addition, the number of vectors we use is a polynomial with respect to n, and therefore, the above construction can be done in NLOGSPACE. □

In [10], it is shown that an n-dimensional VASS can be simulated by an (n+3)-dimensional VAS. Using this fact, it follows immediately that for $k \geq 7$, the BP for k-dimensional VAS's is PSPACE-complete. However, using a more careful encoding, we are able to utilize the positions of the VAS more efficiently during the simulation of the LBA. As a result, we obtain the following improved Corollary:

Corollary 3.1: The BP for k-dimensional VAS's, when $k \geq 4$, is PSPACE-complete.

Proof : As shown in [10], each state can be simulated by a sequence of transitions as follows. For each state q_i, let $a_i=i$ and $b_i=(k+1)(k+1-i)$, where k is the number of states. Assume that the current configuration is (q_i,v). Then a transition $q_i \rightarrow (q_j,w)$ in the VASS can be simulated as (see [10] for details):

$(a_i,b_i,0,v) \rightarrow (0,a_{k-i+1},b_{k-i+1},v) \rightarrow (b_i,0,a_i,v) \rightarrow (a_j,b_j,0,v+w)$.

Notice that three extra positions have been used to simulate a state. Upon a careful examination, of the previous proof, one can see that at most one position will be subtracted at a time. This suggests a more efficient way to encode the tape contents using the VASS. For example, let a_i and b_i be the same as before. Let n denote the input length of the LBA. During the simulation of the LBA, assume that the current configuration is $(q_i,<A,A',B,B'>)$ and the transition $q_i \rightarrow (q_j,<-a,b,c,d>)$, where a,b,c and d are nonnegative integers, is applicable. Then our method is to use the vector $(A,A',a_i*2^{n+1}+B,b_i*2^{n+1}+B')$ to represent the configuration $(q_i,<A,A',B,B'>)$. The basic idea here is that, we use positions B and B' to store both state and tape information in such a way that, the high order bits represent the state and the low order bits represent the tape. Then since no subtraction of the low order bits of B or B' will be made for this transition, the arguments in [10] and in the proof of Theorem 3.1 still work. Similarly, for a transition $q_i \rightarrow (q_i,<a,b,-c,d>)$, the vector $(a_i*2^{n+1}+A,b_i*2^{n+1}+A',B,B')$ will be used. In addition, it should be clear that some rule like $(a_i*2^{n+1},b_i*2^{n+1},-a_i*2^{n+1},-b_i*2^{n+1})$ will be added if necessary. The rest of the simulation is then straightforward. □

At the present time, we do not know whether the PSPACE result can be extended for the case when k=2 or 3. We surmise PSPACE is needed, however, since there exists instances in VASS(2,*l*,m), where the shortest path that leads to a "pumpable loop" requires an exponential number of steps in *l* and m. To show this, consider the following instance in VASS(2,*l*,m) (see also figure 3.3):

1. $(q_1,<1,0>)$ is the initial configuration.

2. The transitons are:

   a. for $1 \leq i \leq m-1$ and i is odd,

      $q_i \rightarrow (q'_i,<-1,1>)$

$q'_i \rightarrow (q_i, <0,1>)$

$q_i \rightarrow (q_{i+1}, <0,0>)$

b. for $1 \leq i \leq m-1$ and i is even,

$q_i \rightarrow (q'_i, <1,-1>)$

$q'_i \rightarrow (q_i, <1,0>)$

$q_i \rightarrow (q_{i+1}, <0,0>)$

c. $q_m \rightarrow (q_{m+1}, <-2^m,0>)$

d. $q_{m+1} \rightarrow (q_{m+1}, <1,1>)$

It is then clear that the VASS is unbounded iff the state q_{m+1} is reachable. Hence 2^m steps are required.

On the other hand, we are able to show that the problem is NP-hard. In order to show that the boundedness problem for k-dimensional VASS's, when $k \geq 2$, is NP-hard, we reduce the Knapsack problem to the boundedness problem. Since the Knapsack problem is known to be NP-complete, it follows immediately that the boundedness problem for VASS's is NP-hard. Recall that the Knapsack problem is the following:

Instance: Finite set U, for each $u \in U$, a size $s(u) \in Z^+$ and a value $v(u) \in Z^+$, and positive integers B and K.

Question: Is there a subset $V \subseteq U$, such that $\Sigma_{u \in V} s(u) \leq B$ and $\Sigma_{u \in V} v(u) \geq K$?

Theorem 3.2: The boundedness problem for k-dimensional VASS's, when $k \geq 2$, is NP-hard.

Proof : Consider an instance of the Knapsack problem as stated above. Let $U = \{u_1,...,u_n\}$. In what follows, we are going to construct a 2-dimensional VASS in such a way that the Knapsack problem has a solution iff the VASS is unbounded. Consequently, the boundedness problem for 2-dimensional VASS's is, at least, as "hard" as the Knapsack problem.

The VASS we construct has n+3 states, namely, $p_0, p_1, ..., p_{n+1}, p'_{n+1}$. Let p_0 and $<0,0>$ be the initial state and vector, respectively. The transitions of the VASS are the following:

1. $p_0 \rightarrow (p_1, <0,B>)$

2. For i=1,...,n-1

$p_i \rightarrow (p_{i+1}, <+v(u_i), -s(u_i)>)$

$p_i \rightarrow (p_{i+1}, <0,0>)$

3. $p_n \rightarrow (p_{n+1}, <-K,0>)$

   $p_n \rightarrow (p'_{n+1}, <0,0>)$

4. $p_{n+1} \rightarrow (p_{n+1}, <+1,+1>)$

   $p'_{n+1} \rightarrow (p'_{n+1}, <0,0>)$

It should be clear that the VASS is unbounded iff the state p_{n+1} is reachable via a path from p_0. On the other hand, such a path must correspond to a solution of the Knapsack problem. Notice that, since we only introduce n+3 states, the above construction can easily be done in NLOGSPACE. □

Using similar encoding techniques as described in Corollary 3.1, we can obtain the following corollary:

Corollary 3.2: The BP is NP-hard for k-dimensional VAS's when $k \geq 3$.

Finally, we have the following theorems concerning 1-dimensional VASS's:

Theorem 3.3: The BP is NLOGSPACE-complete for 1-dimensional VASS's, when the parameter l is also 1.

Proof : The problem was shown to be decidable in NLOGSPACE, in the previous section, whenever k and l were considered to be fixed constants. The hardness follows from an easy reduction from the graph reachability problem[26]. See [12,24] for similar reductions. □

Theorem 3.4: For 1-dimensional VASS's, the BP can be decided in PTIME.

Proof : Consider an instance of VASS(1,l,n) with the initial configuration (q_0,v_0). It is well known that the VASS is unbounded iff there exists a reachable state q_i, such that from q_i a loop with a positive gain can be executed [14]. Based on this idea, the BP can be solved by detecting the existence of this kind of loop: $(q_0,v_0) \rightarrow (q_l,v) \rightarrow (q_l,v+\Delta)$, where each segment is no longer than n steps. Now the largest value obtainable in any state q within at most n steps can be calculated iteratively as follows:

$$d_0^0 = v_0,\ d_i^0 = -\infty,\ i \neq 0.$$

$d_i^{k+1} = \max\{d_i^k, \max_j\{d_j^k + t \mid \text{where } \delta(q_j) = (q_i,t) \text{ is a transition in the VASS and } d_j^k + t \geq 0\}\}$.

Now, such a pumpable loop exists $\Leftrightarrow \exists\ 0 \leq j \leq n,\ 0 \leq s < t \leq n+1$, such that $d_j^t > d_j^s$.

□

The last result should be compared with the related results concerning the shortest and longest path problems studied in [16]. (See also [5].) The difference here is, of course, that we do not require the path in question to visit each node at most once.

Examples illustrating the limitations of the modeling power of VAS's [1,15,20,21,22] indicate that the

limiting factor is the inability of the VAS to test a position for zero and take a particular action on the outcome of the test. (See also [25].) As a result, the literature contains many extensions to the basic model which allow such a zero test. (See e.g. [22].) In most cases, when zero testing is allowed, two potentially unbounded positions are sufficient to render the BP undecidable. (See [9,22].) However, when only one position is allowed, we have the following:

Theorem 3.5: The BP is NP-hard for 1-dimensional VASS's when zero-detection is allowed.

Proof : This can be proved by reducing the PARTITION problem to the BP. Since the PARTITION problem is known to be NP-complete, it follows that the BP is NP-hard.

Recall that an instance of the PARTITION problem consists of a finite set A and a "size" $s(a) \in Z^+$ for each $a \in A$. Let $T=\sum_{i=1}^{n} s(i)$. Let n denote the number of elements in A. We construct an instance of 0-detecting VASS(1,l,n+3) as follows:

1. (q_1,T) is the initial configuration.

2. For i=1,...,n:

   $q_i \rightarrow (q_{i+1},-s(i))$

   $q_i \rightarrow (q_{i+1},0)$

3. $q_{n+1} \rightarrow (q,-T/2)$

4. $q \rightarrow (q_f,0)$ if the vector in q equals zero

5. $q_f \rightarrow (q_f,+1)$

It should be clear that the PARTITION problem has a solution iff the corresponding VASS is unbounded (i.e., the state q_f is reachable). Therefore, the BP is NP-hard. □

We surmise, but are unable to show, that the aforementional problem is solvable in NP. First, note that a "pumpable loop" cannot contain a 0-move. Now, consider the shortest path, which executes a "pumpable loop". Let B denote the last conditional move executed on that path. Then the number of steps executed after B can be no more than n+1. However, as noted earlier, the number of steps proceeding B can be exponential. But, it has to be the case, that every loop proceeding B must either cause a net loss or contain a 0-move. As a result, we have that the computation before B is bounded by $O(n*2^l)$. The best we can do, at this time, then is to deduce that the problem is doable in PSPACE.

4. The Lower Bound

In this section, we establish a lower bound for the VASS boundedness problem in terms of the three natural numeric parameters, k, l and n. Our goal is to provide a simultaneous lower bound over the three parameters, as opposed to examining the lower bound when one (or two) of the parameters is fixed (as was done in Section 3). In order to do this, we define a general class of problems whose complexity can also be stated in terms of three natural numeric parameters. In particular, we focus on the following class which we denote as C:

C: A problem P belongs to C if P is a problem with three natural numeric parameters, say k, l and n, can be solved in $O((l+\log n)*2^{c^*k})$ nondeterministic space (for some constant

c), and for any instance x, of P, the following two conditions are satisfied:

1. $|x| \geq \max\{k,l,n\} \geq \log |x|$, ($|x|$ represents the length of x), *and*

2. k, l and n can be computed from x and written down in deterministic O(log $|x|$) space.

We will show that the BP for the class VASS(k,l,n) is hard for C, with respect to logspace reductions. However, since the current best upper bound for the BP is $O(2^{d*k*\log k}(l+\log n))$ nondeterministic space (which was shown in section 2), we do not know, at this time, whether the BP for VASS's is complete for C.

To prove our result, we require the following lemma, which is similar to the one in [17], where Lipton constructed a VAS to simulate a multiple counter machine. Therefore only a proof sketch will be given here.

Lemma 4.1 : There exists a positive integer constant h, such that for any 3-tuple of integers k, l and n one can construct a VASS in VASS(h*k,l,n), that can manipulate a counter, whose value can range from 0 to $(n*2^l)^{2^k}$. Furthermore, this counter can be incremented, decremented and tested for zero. (I.e. the resulting VASS can simulate a counter machine with a finite state control.)

Proof sketch : Basically, the construction is a straightforward generalization of the one shown in [17]. There, Lipton proved this theorem for the case when l and n were equal to 1. Lipton showed how to maintain and store a number, whose value ranged between 0 and 2^{2^k}, by such a system. (Lipton actually showed this for a class of parallel programs which correspond quite naturally to VASS's.) This number could then be incremented by 1 (as long as the current value was below the upper limit), decremented by 1 (as long as the current value exceeded 0), and tested for zero. The zero-test was the hard part, the crux of which is the following. Let r=p*q. Let (z,z'), (x,x') and (y,y') be three pairs of positions such that z+z'=r, x+x'=p and y+y'=q. Lipton showed that if one could test the the positions x, x', y, and y' for zero, then the position z (and z') could be tested for zero. Using this method recursively, he was then able to build the required VASS. The base case, of this recursion, required one to test a position, whose value was either 0, 1 for zero, (which certainly can be done using a pair of positions). The only difference in our construction, is the base case, where we allow a position to have a value between 0 and $n*2^l$. Certainly, such a pair of positions x, x', whose sum is $n*2^l$, can be so tested. In fact, it can be done using a fixed number of positions and n states, providing we are allowed to use numbers with up to l bits. The result then is a pair of positions z, z' which contain values between 0 and $(n*2^l)^{2^k}$, such that z+z' always equals the largest value, and the indicated operations can be carried out on z and z'.

Whereas, it seems that the total number of positions required by such a construction would be exponential, it turns out that, only a constant number of positions, say h, is required for each additional allowable invocation of the aforemention recursion. (In order to do this, Lipton showed how to simulate the recursion by what he called a primitive type of parameter passing and subroutine calling mechanism, which can be implemented by a VASS of the required type.) Therefore, the above VASS is in the class VASS(h*k,l,n). Further details are left to the reader, which should consult [17]. □

Using the above result, one can construct a VASS in VASS(h*k,l,n) such that, a pair of positions (z,z') can be used to simulate a counter. By using no more than three times the number of positions one can then construct such a VASS that can simulate a 3 counter machine, ala Minsky[19]. Such a 3 counter machine (3CM) has a read only input tape, a finite state control and 3 counters. During a single

computation step, such a machine can, depending on the current state and input symbol, check each counter for zero, and then move according. It can move its input head either one cell to the left or to the right and increment or decrement each of its counters by 1 (as long as they remain nonzero). For more details, see [11,19]. We now are ready to show the following:

Theorem 4.1 : The BP for VASS(k,l,n) is hard for C, with respect to logspace reductions.

Proof : This is shown by reducing an arbitrary problem in C to the BP for VASS(k,l,n). Let P be an arbitrary problem in C. Then there exists a nondeterministic TM M that solves P in $d'*2^{d*k}(l+\log n)$ space, for some positive constants d and d'. It is known that this kind of TM can be simulated by a 3CM M' in such a way that, each counter counts up to no more than $c'*(n*2^l)^{2^{c*k}}$, where c' (c) is a constant depending solely on d' (d). Let p be the number of states in M'. In what follows, given an input x for M, we show how to construct a VASS $V_{M',x}$ to simulate the computation of M' on x such that, M' accepts x iff $V_{M',x}$ is unbounded.

Informally, the state of $V_{M',x}$ is a pair (q,i), where q is a state in M' and i indicates the head position. The contents of the input tape will be implicitly embedded in the transitions of the VASS. Three pairs of positions will be used to simulate the three counters in M'. Moreover, from the previous lemma the operations of adding, subtracting and testing for zero can be applied to each of these counters. The simulation is then straightforward (given Lemma 4.1). At this point, it is important to realize that at most $O(p*|x|+n)$ states will be required in $V_{M',x}$. The first part, $p*|x|$, indicates the maximum number of (q,i)'s; while the second part, n, represents the number of states needed for zero testing (see Lemma 4.1).

Now consider the complexity of constructing $V_{M',x}$, given x. Notice that since x is an instance of P, x must satisfy the three conditions mentioned above. From the 1st condition, $|x| \geq \max\{k,l,n\}$. Recall that at most h*k positions and $p*|x|+n$ states will be required in the VASS. Therefore, one needs only d"*(log |x|) bits, for some constant d", for the working space during the construction, i.e., the space required for storing a pointer (p is considered to be a constant here). This result, associated with the 2nd condition, indicates that the above construction can be done in c"*(log |x|) space, for some constant c".

Hence, the BP for VASS(k,l,n) requires $r*((l+\log n)*2^{r'*k})$ nondeterministic space, for some positive constants r and r', which depend solely on d' and d. As a result, the BP for VASS(k,l,n) is "harder" than any problem in C. □

Note that given an LBA, we could construct an equivalent 3CM where, two counters are used to simulate the tape and the third is used as a working counter. Moreover, each counter counts up to no more than $2^{c*|x|}$ for some constant c, where |x| is the length of the LBA's input string. Based on the discussion in this section, one can then construct a VASS in VASS(6,c*|x|,n) to simulate the 3CM in a way that, each counter is represented by a pair of positions. Consequently, for $k \geq 6$, the BP becomes PSPACE-hard. The reader should recall, however, that in section 3, we were able to show that the BP was PSPACE-hard for $k \geq 4$.

In the remainder of this section, we examine the consequence of Theorem 4.1 with respect to the BP for VASS(k,1,n) and CFSM(k), both of which can be solved in PTIME providing k is a known fixed constant. The reader should recall the fact that, there is a strong connection between VASS(k,1,n) and CFSM(k). More precisely, a k-dimensional VASS, where the maximum size of any integer mentioned in the VASS is 1, can be simulated by a network of CFSM's. Moreover, the simulating network requires no more than k+1 machines, and the number of states in the network is polynomially related to the number of states in the VASS. The BP for CFSM(k) was considered in [6], where it was noted that the problem was solvable in PSPACE, providing k was a fixed known constant. However, it was left open there

whether the result could be improved to PTIME. The fact that the problem is solvable in PTIME follows from the result of section 2. In what follows, we show that under a conjecture used in [13], the BP for CFSM(k) requires $O(n^{2^{k/c}})$ deterministic time, for some constant c. To show this, we require the following corollary which can be derived directly from the proof of Theorem 4.1:

Corollary 4.1: The BP for VASS(k,1,n) requires $O(2^{k/h}*\log n)$ nondeterministic space, for some constant h.

In [13], it was conjectured that for any $\epsilon>0$, NLOGSPACE($2^k*\log n$) $\subsetneq$ DTIME($n^{2^k-\epsilon}$). This indicates that for any k, NLOGSPACE($2^k*\log n$) contains problems which require at least $O(n^{2^k})$ deterministic time. Based on this conjecture, we have the following corollary concerning the time complexity of the BP for VASS(k,1,n) and CFSM(k):

Corollary 4.2: For some constant c, the BP for VASS(k,1,n) (or CFSM(k)) requires $O(n^{2^{k/c}})$ deterministic time, under the above conjecture.

REFERENCES

[1] Agerwala, T. and Flynn, M., Comments on Capabilities, Limitations, and 'Correctness' of Petri Nets, *Proceedings of the First Annual Symposium on Computer Architectures*, New York: ACM, 1973, pp. 81-86.

[2] Borosh, I. and Treybis, L., Bounds on Positive Integral Solutions of Linear Diophantine Equations, *Proc. AMS*, Vol. 55, No. 2, pp. 299-304, March 1976.

[3] Cook, S., Characterizations of Pushdown Machines in Terms of Time Bounded Computers, *J. ACM*, Vol. 18, 1971, pp. 4-18.

[4] Cunha, P. and Maibaum, T., A Synchronous Calculus for Message Oriented Programming, *Proc. 2nd International Conf. on Distributed Computing Systems*, April 1981, pp. 443-445.

[5] Garey, M. and Johnson, D., "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman and Company, San Francisco, 1979.

[6] Gouda, M., Gurari, E., Lai, T. and Rosier, L., Deadlock Detection in Systems of Communicating Finite State Machines, Tech. Rep. No. 84-11, University of Texas at Austin, Department of Computer Sciences, April 1984.

[7] Gouda, M. and Rosier, L., Priority Networks of Communicating Finite State Machines, accepted for publication in the *SIAM Journal on Computing.*

[8] Gurari, E. and Lai, T., Deadlock Detection in Communicating Finite State Machines, *ACM SIGACT NEWS*, Vol. 15, No. 4, and Vol. 16, No. 1, Winter - Spring 1984, pp. 63-64.

[9] Hack, M., Decidability Questions for Petri Nets, M.I.T., LCS, TR. 161, June 1976.

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

[11] Hopcroft, J. and Ullman, J., "Introduction to Automata Theory, Languages, and

Computation", Addison-Wesley, Reading, Mass., 1979.

[12] Jones, N., Landweber, L. and Lien, Y., Complexity of Some Problems in Petri Nets, *Theoret. Comp. Sci.*, Vol. 8, 1979, pp. 277-299.

[13] Kasai, T. and Iwata, S., Gradually Intractable Problems and Nondeterministic Log-Space Lower Bounds, to appear in *Math. Systems Theory.*

[14] Karp, R. and Miller, R., Parallel Program Schemata, *J. Comput. System Sci.*, 3, 1969, pp. 147-195.

[15] Kosaraju, S., Limitations of Dijkstra's Semaphore Primitives and Petri Nets, *Operating Systems Review*, Vol. 7, No. 4, Oct. 1973, pp. 122-126.

[16] Lawler, E., "Combinatorial Optimization: Networks and Matroids", Holt, Rinehart and Winston, New York, 1976.

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

[18] 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.

[19] Minsky, M., "Computation: Finite and Infinite Machines", Prentice Hall, Englewood Cliffs, NJ, 1967.

[20] Parnas, D., On a Solution to the Cigarette Smokers' Problem (Without Conditional Statements), *Communications of the ACM*, Vol. 18, No. 3, March 1975, pp. 181-183.

[21] Patil, S., Limitations and Capabilities of Dijkstra's Semaphore Primitives for Coordination Among Processes, Group Memo 57, Project MAC, MIT, February 1971, 18 pages.

[22] Peterson, J., "Petri Net Theory and the Modeling of Systems", Prentice Hall, Englewood Cliffs, NJ, 1981.

[23] Rackoff, C., The Covering and Boundedness Problems for Vector Addition Systems, *Theoret. Comp. Sci.*, 6, 1978, pp. 223-231.

[24] Rosier, L. and Gouda, M., On Deciding Progress for a Class of Communication Protocols, in the *Proceedings of the Eighteen Annual Conference on Information Sciences and Systems*, Princeton Univ., 1984, pp. 663-667.

[25] Rosier, L. and Yen, H., Boundedness, Empty Channel Detection and Synchronization for Communicating Finite State Machines, in the *Proceedings of the Second Annual Symposium on Theoretical Aspects of Computer Science*, Saarbrucken, West-Germany, 1985, pp. 287-298.

[26] Savitch, W., Relationships between Nondeterministic and Deterministic Tape Complexities, *J. of Computer and System Sciences*, Vol. 4, No. 2, 1970, pp. 177-192.

[27] Yu, Y. and Gouda, M., Unboundedness Detection for a Class of Communicating Finite State Machines, *Information Processing Letters*, 17 (1983), pp. 235-240.

	VASS (k,ℓ,m)	VAS (k,ℓ)
PSPACE-Complete	$k \geq 4$	$k \geq 4$
NP-hard	$k \geq 2$	$k \geq 3$
PTIME	$k = 1$	------
trivial	-------	$k = 1$

Figure 3.1 The complexity of the BP for VASS's (VAS's) where k is fixed.

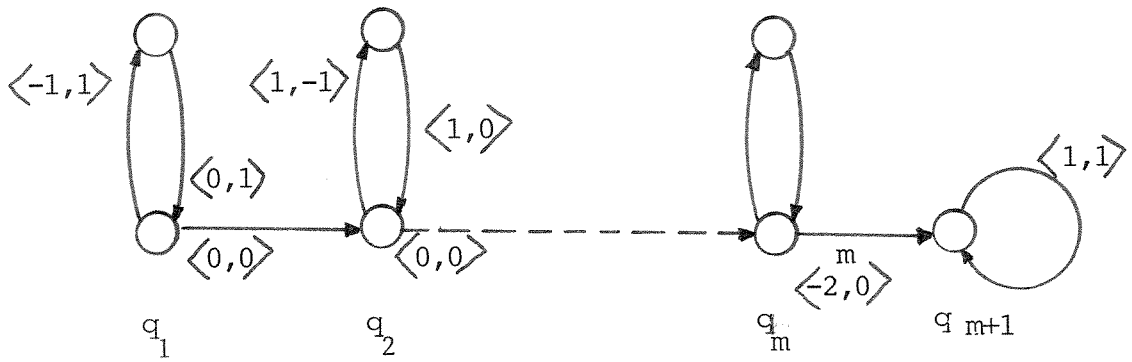

Figure 3.3 An unbounded VASS that requires an exponential number number of steps.

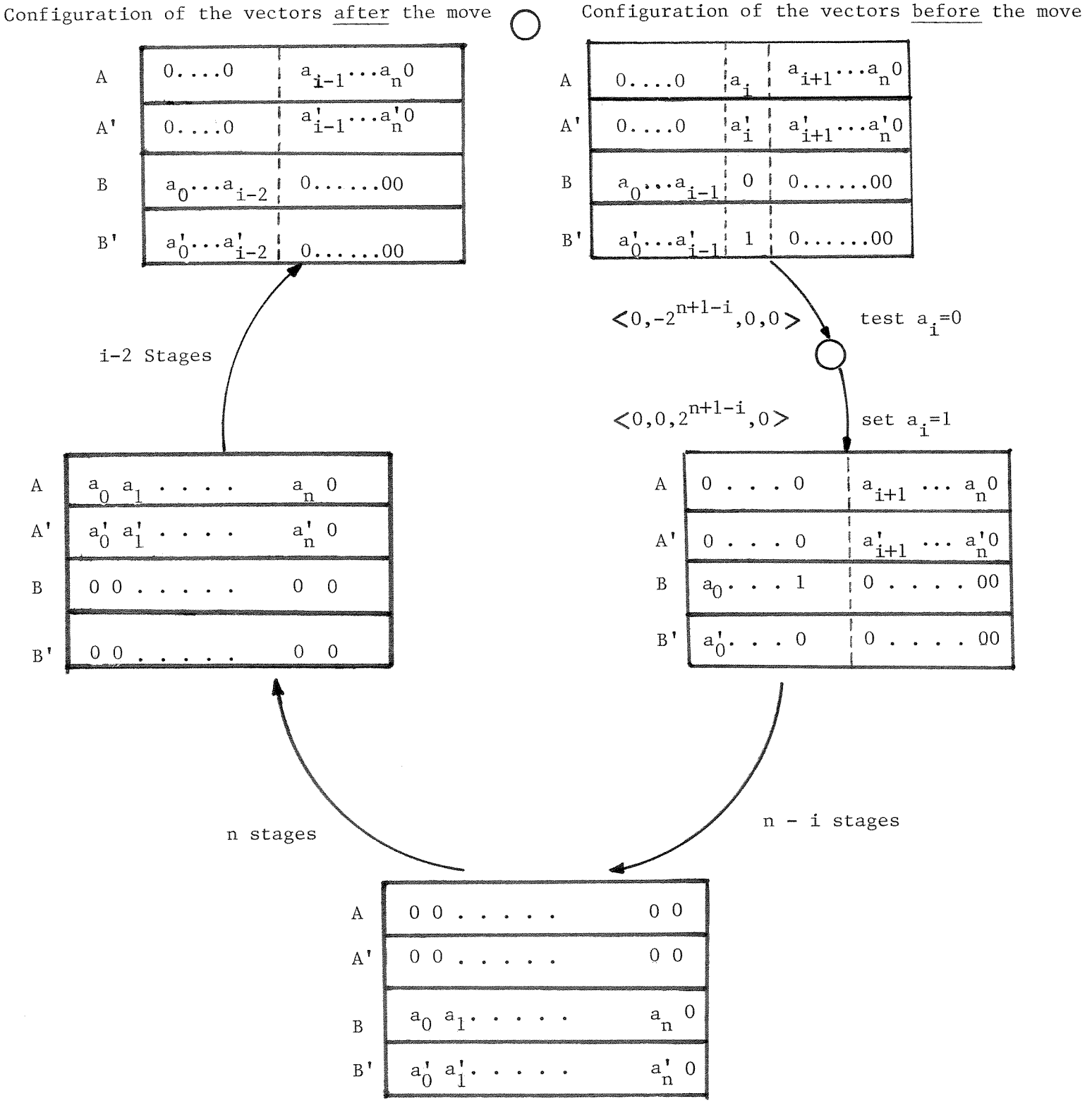

Figure 3.2 The simulation of the left-move transition.