THE BOUNDEDNESS PROBLEM FOR CONFLICT-FREE VECTOR REPLACEMENT SYSTEMS IS PTIME-COMPLETE

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

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

TR-86-13 May 1986

Keywords: Boundedness problem, complexity, conflict-free, Petri nets, vector addition systems, vector replacement systems.

1. Introduction

Vector addition systems (VASs) were introduced in [8], and were later shown by Hack [2] to be equivalent to Petri nets. The boundedness problem for VASs has been studied in [9, 11], where exponential-space upper and lower bounds were shown. (See also [12].) Crespi-Reghizzi and Mandrioli [1] introduced a subclass of VASs called conflict-free VASs. Landweber and Robertson [10] subsequently showed an upper bound of exponential time for the boundedness problem for conflict-free Petri nets.

Since the definitions given in [1, 10] are somewhat different from one another, and since translations between the two do not seem to preserve sharp complexity bounds, we introduce the notion of conflict-free vector replacement systems (VRSs; see [7]). Our definition is general enough to include both previous definitions as special cases (although conflict-free Petri nets have a somewhat more succinct representation), yet it is also restrictive enough to allow us to prove the boundedness problem PTIME-complete for all three definitions. In particular, we give a time $O(n^{1.5})$ algorithm for deciding boundedness for conflict-free VRSs (or VASs). In terms of the Petri net model, the time complexity is $O(n^2)$, an improvement over the result of Landweber and Robertson. (For our complexity measures, n is the total number of bits used in encoding the problem.) Now most problems concerning VRSs or equivalent formalisms have been shown to be intractable (see, e.g., [3, 4, 6, 9, 12]); therefore, since our algorithm has a time complexity of such a small-order polynomial, it may have real-world applications.

2. Definitions

Let Z (N) denote the set of integers (nonnegative integers, respectively), and let Z^k (N^k) be the set of vectors of k integers (nonnegative integers), and $Z^{k\times m}$ ($N^{k\times m}$) be the set of $k\times m$ matrices of integers (nonnegative integers). For a vector $v\in Z^k$, let v(i), $1\leq i\leq k$, denote the i^{th} component of v, and for a matrix $V\in Z^{k\times m}$, let V(i,j), $1\leq i\leq k$, $1\leq j\leq m$, denote the element in the i^{th} row and j^{th} column of V. For a given value of k, let $\mathbf{0}$ in Z^k denote the vector of k zeros (i.e., $\mathbf{0}(i)=0$ 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 $k\times m$ *vector replacement system* (VRS), is a triple (v_0,U,V), where $v_0\in N^k$, $U\in N^{k\times m}$, and $V\in Z^{k\times m}$, such that for any i,j, $1\leq i\leq k$, $1\leq j\leq m$, $U(i,j)+V(i,j)\geq 0$. v_0 is known as the *start vector*, U is known as the *check matrix*, and V is known as the *addition matrix*. A column u_j of U is called a *check vector*, and a column v_j of V is called an *addition rule*. For any $x\in N^k$, we say addition rule v_j is *enabled* at x iff $x\geq u_j$. The *reachability set* of the VRS (v_0,U,V), denoted by $R(v_0,U,V)$, is the set of all vectors z, such that $z=v_0+v_1+\cdots+v_n$ for some $n\geq 0$, where each v_j ($1\leq j\leq n$) is a column of V, and for each $1\leq j\leq n$, v_j is enabled at $v_0+v_1+\cdots+v_{j-1}$. Let $\sigma=<w_0,...,w_t>$ be a sequence of vectors in N^k. If $w_0=v_0$, and for every r, $1\leq r\leq t$, there is a j such that $w_r=w_{r-1}+v_j$ and $w_{r-1}\geq u_j$, then we say $<w_0,...,w_t>$ is a *path* in (v_0,U,V). If there exist r and s, $1\leq r<s\leq t$, such that $w_r\leq w_s$ ($w_r<w_s$), then we say that $\pi=<w_r,...,w_s>$ is a *loop* (*positive loop*), and that π is *enabled* at w_{r-1}.

A VRS (v_0,U,V) is said to be *conflict-free* iff (1) no number in U is greater than 1; (2) no number in V is less than -1; (3) no row in V has more than one -1; and (4) if $V(i,j)=-1$, then $U(i,j)=1$, and all other elements in row i of U are 0. (v_0,U,V) is said to be *bounded* iff for each row i, there is a constant c such that if $v\in R(v_0,U,V)$, then $v(i)<c$. The *boundedness problem* for VRSs is the problem of determining whether a given VRS is bounded.

For a given $k\times m$ addition matrix V, the *minimal check matrix* is a $k\times m$ matrix U in which $U(i,j)=1$ if $V(i,j)=-1$, and $U(i,j)=0$ otherwise. It is easy to see that the set of $k\times m$ conflict-free VRSs with minimal check matrices is equivalent to the set of $k\times m$ conflict-free VASs (see [1]). Furthermore, there is an obvious translation from a conflict-free Petri net (see [10]) with k places and m transitions to a $k\times m$ conflict-free VRS whose addition rules and check vectors have no elements larger than 1. Thus, our definition is general enough to include both previous definitions. In addition, we are able to show that our PTIME lower bound holds even when the VRSs simultaneously satisfy all three definitions. However,

when we translate a conflict-free Petri net to its equivalent VRS, an increase in the size of the problem description may be incurred. In particular, for a $k \times m$ VRS with $m \leq k$, m is clearly $O(n^{0.5})$, where n is the size of the problem description. On the other hand, if n' is the size of the description of the equivalent Petri net, we can only conclude that m is $O(n')$. Thus, the complexity measures for the algorithm operating on the two respective formalisms differ slightly.

3. The Upper Bound

In this section, we present an $O(n^{1.5})$ algorithm to determine boundedness for conflict-free VRSs. Karp and Miller [8] showed that a VRS is unbounded iff it can execute a positive loop. Our algorithm, therefore, looks for a positive loop that can be executed by the VRS. Before a positive loop can be executed, it must be enabled. Lemma 3.1 below gives an algorithm for finding a path that we will later show enables some postive loop if one exists.

Lemma 3.1: For any $k \times m$ conflict-free VRS $\mathcal{V}=(v_0,U,V)$ that is described by n bits, we can construct in time $O(n^{1.5})$ a path σ in which no rule in V is used more than once in σ, such that if some rule w is not used in σ, then there is no path in which w is used.

We construct σ as follows. First, we execute all rules enabled at v_0. Then we repeatedly cycle through U, executing all those rules which are enabled but have not yet been executed. We continue until a complete pass is made through U, during which no position increases in value. (Note that this is a sufficient condition to conclude that no new rules are enabled.) Clearly, no more than k+1 passes are made through U. On each pass except the last, there is at least one rule (say v_j) enabled that was not enabled the previous pass; i.e., some position (say p) which was zero the previous pass is now positive. Furthermore, since $\mathcal{V}$ is conflict-free, if some rule subtracts from position p, that rule must be v_j. Therefore, position p must have never previously been positive. Thus, on each pass except the last some position becomes positive for the first time, so the number of passes is no more than $\min(k,m)+1=O(n^{0.5})$. Therefore, the entire procedure operates in time $O(n^{1.5})$.

Now suppose there is a path σ' using rules not in σ. Let v_r be the first such rule executed in σ'. Then all rules used before v_r in σ' are used in σ. Since v_r is not executed in σ, no position from which v_r subtracts ever decreases in value in σ; hence, these positions are at least as large as they are at the point at which v_r is executed in σ'. Then v_r is enabled by σ, a contradiction. Therefore, if v_r is not used in σ, then there is no path in which v_r is used. □

The following lemma gives necessary and sufficient conditions for determining which addition rules in a conflict-free VRS can appear in a loop.

Lemma 3.2: Given a conflict-free VRS (v_0,U,V), let V' be any matrix of rules in V which can be executed in some path. There exists a path σ containing a loop whose rules used are exactly those in V' iff every row in V' that contains a -1 also contains a positive number. Furthermore, the loop in σ can be required to have each rule in V' used exactly once.

Proof: Assume there is a path σ containing a loop whose rules used are exactly those in V'. Recall that loops have a nonnegative displacement. Suppose some rule v_j in V' subtracts 1 from some position p (i.e., $V'(p,j)=-1$). Since v_j is used in a loop in σ, some rule used in the loop must add to position p. Since every rule in the loop is also in V', row p in V' must also contain a positive number. Hence, every row in V' that contains a -1 also contains a positive number.

Now assume that every row in V' that contains a -1 also contains a positive number. From Lemma 3.1 there is a path σ' that uses all of the rules in V' exactly once. Let rule v_j be the first rule from V' executed in σ'. Since the remainder of the rules in V' are subsequently executed in σ', every position from which v_j subtracted 1 has subsequently increased in value. We can therefore append v_j to σ'. By repeated application of this principle to each of the remaining rules in V' in the order they appear in σ', we can construct a path σ containing a loop whose rules used are exactly those in V'. Furthermore, each rule in V' is used exactly once in the loop. □

We now give our algorithm for determining boundedness in a conflict-free VRS.

Theorem 3.1: The boundedness problem for conflict-free VRSs is solvable in time $O(n^{1.5})$, where n is the number of bits in the description of the VRS.

Proof: Let $\mathcal{V}=(v_0,U,V)$ be an arbitrary $k\times m$ conflict-free VRS that is described by n bits. We first remove all columns j of U and V such that column j of V is all zeros. From Lemma 3.1 we can remove from U and V in time $O(n^{1.5})$ all columns representing rules that can never be used. Now if more than k columns remain in each matrix, some column of V must contain no -1's; hence, a positive loop can be executed. Therefore, without loss of generality, assume no more than k columns remain; i.e., the number of remaining columns is $O(n^{0.5})$. We now wish to remove from U and V all columns that cannot be used in any loop. By Lemma 3.2, if $V(i,j)=-1$ and row i of V contains no positive numbers, column j can be removed from U and V. This can be repeated until every row of V that contains a -1 also contains a positive number. This can clearly be done in time $O(n^{1.5})$. Call the resulting addition matrix V' and the resulting check matrix U'. By Lemma 3.2, there is a path that contains a loop with every rule in V' used exactly once; hence, if some row has a positive sum, $\mathcal{V}$ is unbounded. Therefore, assume without loss of generality that each row of V' that contains a -1 also contains exactly one 1, with the rest of its elements zero. We now give an algorithm that we claim will find a positive loop in V' iff one exists (let $V||v$

indicate a column vector v appended to a matrix V):

```
unbounded ← false
for each column j in V' do
    initialize V'' to the null matrix
    V'' ← V''||v_j
    R ← {i | V'(i,j)=-1}
    C ← {j}
    while R ≠ ∅
        pick an element i' ∈ R
        let j' be the column for which V'(i',j')=1
        if j' ∉ C then
            V'' ← V''||v_j'
            R ← R∪{i | V'(i,j')=-1}
            C ← C∪{j'}
        end if
        R ← R-{i'}
    end while
    if some row of V'' has positive sum then unbounded ← true endif
end for
```

The **for** loop in the above algorithm executes at most $O(n^{0.5})$ times. The sets R and C contain the indices of the rows that are to be examined and the columns that have been examined, respectively, on the current iteration of the **for** loop. Note that no column is examined more than once, and that a row is examined only if some column previously examined contained a -1 in that row. Since no row contains more than one -1, the **while** loop terminates after at most k iterations. Since R can be implemented as a queue, and C has at most m elements, every statement which must be executed every time through the **while** loop takes at most O(m) time. Now the three statements inside the **if** block execute no more than m times (on a given interation of the **for** loop), and each takes at most O(k) time. Thus, the body of the **for** loop takes at most O(n) time, and the entire algorithm operates in time $O(n^{1.5})$.

Now if $\mathcal{V}$ is bounded, from Lemma 3.2 no matrix V'' containing a row with a positive sum can be generated by an iteration of the **for** loop. Suppose $\mathcal{V}$ is unbounded. Let π be the shortest positive loop that can be executed in $\mathcal{V}$. If we let x be a vector designating the number of times each rule in V' is executed in π and w be the displacement of π, we have V'x=w. Let element j of x be a maximal element of x. We will now show that the iteration of the above algorithm which begins with C={j} produces a matrix V'' containing a row with a positive sum. The proof is by contradiction. Since any row in V'' that contains a -1 also contains a 1, the sum of each row in V'' must be nonnegative. In order to obtain the contraction, assume that every row of V'' has a zero sum. If V'(i,j)=-1, there must be exactly one element in row i, say V'(i,j'), such that V'(i,j')=1. Since all other elements of row i are 0, we have $x(j')-x(j)=w(i) \geq 0$; hence, since x(j) is a maximal element of x, x(j')=x(j). Thus it is easily seen that each rule in V'' is executed exactly x(j) times in π, and the sum of all the occurrences of these rules is therefore

0. Now let W be the minimal check matrix for V. Clearly, if σ is any path in $\mathcal{V}$ enabling π, σ also is a path in $\mathcal{V}'=(v_0,W,V)$. From a result in [1], if all x(j) occurrences of the rules from V'' in π are removed, the resulting positive loop π' is also enabled by σ in $\mathcal{V}'$. We now claim that π' is enabled in $\mathcal{V}$ by the path $\sigma\pi$, which is obtained by appending π to σ. This claim is also proved by contradiction. Suppose π' is not enabled by $\sigma\pi$ in $\mathcal{V}$, and let v_r be the first addition rule used in π' that is not enabled. Clearly, π' is enabled by $\sigma\pi$ in $\mathcal{V}'$, so if v_r is not enabled in $\mathcal{V}$, it must be because some position $p_i<U(i,r)=1$, where $V(i,r)\neq -1$. Now since every addition rule used in π' is used in π, at some point in $\sigma\pi$ we must have $p_i \geq 1$. But since $\mathcal{V}$ is conflict-free, if U(i,r)=1 and $V(i,r)\neq -1$, then row i of V cannot have a -1. Hence, p_i cannot decrease in value--a contradiction. Therefore, π' is a positive loop shorter than π that appears in a path in $\mathcal{V}$. This contradicts our original choice of π; hence, some row in V'' must have a positive sum. This completes the proof. □

The preceding algorithm will also work for conflict-free Petri nets; however, due to their more succinct representation, we cannot conclude from the fact that $k \geq m$ that m is $O(n^{0.5})$. The best we can state in this regard is that m is O(n). We therefore have the following corollary, which gives an improvement over the exponential-time result in [10].

Corollary 3.1: The boundedness problem for conflict-free Petri nets is solvable in $O(n^2)$ time, where n is the number of bits necessary to describe the Petri net.

4. The Lower Bound

In this section, we show the boundedness problem for conflict-free VRSs to be PTIME-complete. In order to show the problem to be PTIME-hard, we use a reduction from the path system problem, which is known to be PTIME-complete [5]. A *path system* is a 4-tuple P=(X,R,S,T) where X is a finite set of nodes, $S(\subseteq X)$ is the set of starting nodes, $T(\subseteq X)$ is the set of terminal nodes, and $R(\subseteq X\times X\times X)$ is the set of rules. A node x in X is said to be *admissible* iff either $x\in T$ or there exist $y,z\in X$ such that $(x,y,z)\in R$ and both y and z are admissible. The path system P is said to have a solution iff there is an admissible node in S.

Lemma 4.1: The boundedness problem for conflict-free VRSs is PTIME-hard, even if the largest integer mentioned in the system is 1 and the VRS has a minimal check matrix.

Proof: Given an arbitrary path system P=(X,R,S,T) we want to construct a conflict-free VRS $\mathcal{V}=(v_0,U,V)$ such that $\mathcal{V}$ is unbounded iff P has a solution. Let $X=\{x_1,\ldots,x_k\}$, |S|=m, and |R|=n. Then $\mathcal{V}$ will be a $(k*n+k+n+3)\times(n+m+2)$ conflict-free VRS. For ease of illustration let us denote a configuration of $\mathcal{V}$ (which is simply a (k*n+k+n+3)-dimensional vector) by an assignment of values to the following set of variables:

$\{a_{ij}, b_i, c_j, d_1, d_2, d_3 \mid 1 \leq i \leq k, 1 \leq j \leq n\}$.

Each addition rule in V can then be specified by a set of changes to the variables. Now v_0 is defined by the following variable assignment:

$$\forall i,j,\ 1 \leq i \leq k,\ 1 \leq j \leq n,\ a_{ij} = \begin{cases} 1 \text{ if } x_i \in T \\ 0 \text{ otherwise} \end{cases}$$

$$\forall i,\ 1 \leq i \leq k,\ b_i = \begin{cases} 1 \text{ if } x_i \in T \\ 0 \text{ otherwise} \end{cases}$$

$\forall j,\ 1 \leq j \leq n,\ c_j = 1$

$d_1 = d_2 = d_3 = 0$

The addition rules in V are given below; U is the minimal check matrix for V. For convenience, each addition rule has a type associated with it. Each type 1 addition rule corresponds to a rule in R. Each type 2 addition rule corresponds to an element in S. The two type 3 addition rules will allow the system to become unbounded provided they can be enabled by some path.

Type 1: If (x_r, x_s, x_t) is the j^{th} rule in R, then the following defines an addition rule v_j in V:

$\forall k,\ 1 \leq h \leq n,\ a_{rh} \leftarrow a_{rh}+1$
$a_{sj} \leftarrow a_{sj}-1$
$a_{tj} \leftarrow a_{tj}-1$
$b_r \leftarrow b_r+1$
$c_j \leftarrow c_j-1$

Type 2: If $x_i \in S$ then the following defines an addition rule v_{n+i} in V:

$b_i \leftarrow b_i-1$
$d_1 \leftarrow d_1+1$

Type 3: The following two addition rules are also in V:

1. $d_1 \leftarrow d_1-1$
   $d_2 \leftarrow d_2+1$
   $d_3 \leftarrow d_3+1$

2. $d_1 \leftarrow d_1+1$
   $d_2 \leftarrow d_2-1$
   $d_3 \leftarrow d_3+1$

$\mathcal{V}$ is clearly conflict-free and can be constructed in deterministic logspace. Before we show that $\mathcal{V}$ is unbounded iff P has a solution, we will show that $\mathcal{V}$ is unbounded iff there is a path in $\mathcal{V}$ which enables a type 2 addition rule. Clearly, only the type 3 rules can appear in a matrix of rules $V' \subseteq V$ in which each

row containing a -1 also contains a positive number; therefore, from Lemma 3.2, only the type 3 rules can occur in a loop. Since the type 3 rules can become enabled iff some type 2 rule can become enabled, $\mathcal{V}$ is unbounded iff a type 2 rule can become enabled. As a result, we need only to show that P has a solution iff there exists a path in $\mathcal{V}$ that enables a type 2 addition rule.

Let us say that a node $x_r \in X$ is Q-admissible for some $Q \subseteq R$ iff $(X,Q,\{x_r\},T)$ has a solution. We will now show by induction on $|Q|$ that for all r, x_r is Q-admissible iff there exists a path σ in $\mathcal{V}$ using only type 1 addition rules corresponding to rules in Q, such that σ leaves $b_r > 0$. If $Q=\emptyset$, the claim clearly holds. Let $|Q|>0$ and assume the claim for all Q' such that $|Q'|<|Q|$. Suppose some node x_r is Q-admissible. If $x_r \in T$, the claim clearly holds. If $x_r \notin T$, then there is a rule $(x_r,x_s,x_t) \in R$ such that x_s and x_t are Q'-admissible, $Q'=Q-\{(x_r,x_s,x_t)\}$. From the induction hypothesis, there exists a path σ_s (σ_t) in $\mathcal{V}$ using only type 1 addition rules corresponding to rules in Q', such that σ_s (σ_t) leaves b_s (b_t) > 0. Clearly, at the first point in σ_s (σ_t) at which b_s (b_t) > 0, a_{sh} (a_{th}) > 0 for all h. It follows from the proof of Lemma 3.1 that there exists a path σ' whose addition rules used are those used in σ_s union those used in σ_t. Let (x_r,x_s,x_t) be the j^{th} rule in R. Since $(x_r,x_s,x_t) \notin Q'$, v_j is not used in σ'; therefore, σ' must leave a_{sj} (a_{tj}) > 0. v_j can then be executed, leaving $b_r > 0$. Thus, the claim is satisfied.

Now suppose there is a path in $\mathcal{V}$ using only type 1 addition rules corresponding to rules in Q, after which $b_r > 0$. Let σ be the shortest such path. If σ is the null path, then $x_r \in T$ and the claim is satisfied. Otherwise, some addition rule v_j corresponding to a rule $(x_r,x_s,x_t) \in Q$ was the last addition rule executed in σ. Since a_{sj} (a_{tj}) must be positive for v_j to be executed, b_s (b_t) must clearly become positive in σ. Thus, there is a path σ_s (σ_t) in $\mathcal{V}$ using only type 1 addition rules corresponding to rules in $Q'=Q-\{(x_r,x_s,x_t)\}$ after which b_s (b_t) > 0. From the induction hypothesis, x_s and x_t are Q'-admissible; therefore, x_r is Q-admissible. We have therefore shown our claim.

Since only type 1 rules may be executed before the first type 2 rule is executed, there is an R-admissible (i.e., admissible) node $x_r \in S$ iff there exists a path in V after which $b_r > 0$; i.e., P has a solution iff there exists a path in $\mathcal{V}$ that enables a type 2 addition rule. We can therefore conclude that $\mathcal{V}$ is unbounded iff P has a solution. This completes the proof. □

The following theorem now follows immediately from Theorem 3.1 and Lemma 4.1.

Theorem 4.1: The boundedness problem for conflict-free VRSs (VASs, Petri nets, respectively) is PTIME-complete.

References

[1] 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.

[2] Hack, M., The Recursive Equivalence of the Reachability Problem and the Liveness Problem for Petri Nets and Vector Addition Systems, *15th Annual Symp. on Switching and Automata Theory* (1974), pp. 156-164.

[3] 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.

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

[5] Jones, N., and Laaser, W., Complete Problems for Deterministic Polynomial Time, *Theoret. Comp. Sci. 3* (1977), pp. 105-117.

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

[7] Keller, R.M., *Vector Replacement Systems: A Formalism for Modelling Asynchronous Systems*, TR 117, (Princeton University, CSL, 1972).

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

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

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

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

[12] 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.