Theoretical Computer Science 7 (1978) 311–324

REMARKS ON BLIND AND PARTIALLY BLIND ONE-WAY MULTICOUNTER MACHINES*

S. A. GREIBACH
Department of System Science, University of California, Los Angeles, CA 90024, U.S.A.

Communicated by Ronald Book
Received September 1977
Revised February 1978

Abstract. We consider one-way nondeterministic machines which have counters allowed to hold positive or negative integers and which accept by final state with all counters zero. Such machines are called blind if their action depends on state and input alone and not on the counter configuration. They are partially blind if they block when any counter is negative (i.e., only nonnegative counter contents are permissible) but do not know whether or not any of the counters contain zero. Blind multicounter machines are equivalent in power to the reversal bounded multicounter machines of Baker and Book [1], and for both blind and reversal bounded multicounter machines, the quasirealtime family is as powerful as the full family. The family of languages accepted by blind multicounter machines is the least intersection closed semiAFL containing $\{a^n b^n | n \geq 0\}$ and also the least intersection closed semiAFL containing the two-sided Dyck set on one letter. Blind multicounter machines are strictly less powerful than quasirealtime partially blind multicounter machines. Quasirealtime partially blind multicounter machines accept the family of computation state sequences or Petri net languages which is equal to the least intersection closed semiAFL containing the one-sided Dyck set on one letter but is not a principal semiAFL. For partially blind multicounter machines, as opposed to blind machines, linear time is more powerful than quasirealtime. Assuming that the reachability problem for vector addition systems is decidable [16], partially blind multicounter machines accept only recursive sets and do not accept even $\{a^n b^n | n \geq 0\}^*$, and quasirealtime partially blind multicounter machines are less powerful than general quasirealtime multicounter machines.

In this paper, we shall consider only nondeterministic on-line acceptors, that is, nondeterministic acceptors with a one-way input tape read strictly from left to right.

We regard a counter as an arithmetic register containing an integer which may be positive, zero or negative. In one step, a multicounter machine may increment or decrement a counter by a fixed amount. For most models, adding or subtracting 1 or 0 suffices, so we shall use this convention. The action or choice of actions of the machine is determined by the input symbol currently scanned, the state of the machine, and the sign of each counter: positive, negative or zero. The machine

* This paper was supported in part by the National Science Foundation under Grant DCR74-15091.

starts with all counters empty and accepts if it reaches a designated final state with all counters empty (containing 0).

Unlimited two counter machines accept all recursively enumerable languages [14]. However, various restrictions on the amount of time or space allowed or on the permissible actions of the machine produce proper subclasses of the family of recursive languages (cf. [1, 6, 10]).

We can also restrict the amount of information regarding its counters available to a multicounter machine. We call such a machine *blind* if it is allowed no information regarding the condition of its counters; that is, the same action or choice of action is available regardless of the sign or size of the counters. Such machines (called storage independent in [9]) are not powerless. For example, a blind one-state one-counter machine can accept the set L_1 of strings in $\{a, b\}^*$[1] containing equal numbers of a's and b's (the two-sided Dyck set on one letter) by simply adding 1 for each a read and subtracting 1 for each b: if the counter is initially zero, it returns to zero precisely when the number of a's and b's are equal. We shall show (Theorem 3) that no such machine can accept the subset L_2 of L_1 consisting of strings w in L_1 such that every prefix of w has no more b's than a's (i.e., the one-sided Dyck set on one letter).

We shall show (Theorem 2) that one-way nondeterministic blind multicounter machines are equivalent in accepting power to the one-way nondeterministic reversal-bounded multicounter machines of Baker and Book [1] and for both classes of machines restriction to quasirealtime does not limit the accepting power. This somewhat strengthens the result of Baker and Book [1], who showed this for restriction to linear time. The class of languages accepted by these machines is thus the least intersection closed semiAFL containing $\{a^n b^n | n \geq 0\}$.

A *partially blind* machine is also blind and in addition must have nonnegative counter contents; should any counter go negative, no further transitions are allowed and the machine is blocked. While blind machines know nothing about their counters but can drive them below zero, partially blind machines cannot use nonnegative integers but do know something about their counters, namely, that they are nonnegative, and can learn that they are nonzero (subtract 1 and add 1) but cannot determine whether they are zero. Partially blind multicounter machines are strictly more powerful than blind machines since they can accept the one-sided Dyck set on one letter (Theorem 3). Unlike the situation with blind machines, for partially blind one-way nondeterministic multicounter machines, linear time is more powerful than quasirealtime (Theorem 4). Quasirealtime partially blind machines define the family of computation state sequences or Petri net languages which is thus the least intersection closed semiAFL containing L_2 but is not a principal semiAFL (Theorem 5). Using the claimed decidability of the reachability problem for vector addition systems [16], we observe that one-way

[1] For a set A of strings, $A^* = \{e\} \cup \{x_1 \ldots x_n \mid n \geq 1, x_i \in A\}$, where e is the empty string.

nondeterministic partially blind multicounter machines with no time restriction define only recursive languages and hence cannot accept $(\{a^n b^n | n \geq 0\})^*$, so the quasirealtime subfamily has a decidable emptiness problem and is strictly less powerful than the family of quasirealtime one-way nondeterministic multicounter machines (Theorem 6).

We start by defining multicounter machines and the languages they accept. Let $\mathbf{Z}$ be the set of integers (positive, negative and zero) and let $\mathbf{N}$ be the subset of nonnegative integers. Let $\text{sgn}(x) = 1, 0$ or -1 as $x > 0$, $x = 0$, or $x < 0$, respectively. Let e denote the empty string; input e corresponds to a move that does not advance the input tape.

Definition. A *k-counter machine* $M = (K, \Sigma, \delta, q_0, F)$ consists of a finite set K of *states*, a designated *initial* state q_0, a designated subset F of *final* or *accepting* states, a finite *input* alphabet Σ and a *transition function* δ from $K \times (\Sigma \cup \{e\}) \times \{0, 1, -1\}^k$ into subsets of $K \times \{0, 1, -1\}^k$. An *instantaneous description* (ID) of M is a member of $K \times \Sigma^* \times Z^k$. If $(q', v_1, \ldots, v_k)$ is in $\delta(q, a, u_1, \ldots, u_k)$ and $(q, aw, y_1, \ldots, y_k)$ is an ID with $\text{sgn}(u_i) = \text{sgn}(y_i)$, $1 \leq i \leq k$, then we write $(q, aw, y_1, \ldots, y_k) \vdash (q', w, y_1 + v_1, \ldots, y_k + v_k)$. If $a = e$, this is an *e-move*. For any ID I, we write $I \vdash^0 I$. If $\text{ID}_1 \vdash \text{ID}_2$ and $\text{ID}_2 \vdash^n \text{ID}_3$, we write $\text{ID}_1 \vdash^{n+1} \text{ID}_3$. If $\text{ID}_1 \vdash^n \text{ID}_2$ for any $n \geq 0$, we call it an *n-step computation* and write $\text{ID}_1 \vdash^* \text{ID}_2$; if $\text{ID}_1 = (q_0, w, 0, \ldots, 0)$ and $\text{ID}_2 = (q, e, 0, \ldots, 0)$ for any q in F, it is an *accepting computation for w*. The *language accepted by M* is

$L(M) = \{w \text{ in } \Sigma^* \mid M \text{ has an accepting computation for } w\}$.

We shall deal primarily with machines restricted to operate in realtime or linear time.

Definition. A k-counter machine $M = (K, \Sigma, \delta, q_0, F)$ *accepts L in linear time with linear factor d* if $L = L(M)$ and for any w in L there is an accepting n-step computation with $n \leq d \text{ Max}(|w|, 1)$.[2] Machine M is *quasirealtime of delay d* if whenever $(q, e, y_1, \ldots, y_k) \vdash^n (q', e, y_1', \ldots, y_k')$, $n \leq d$; it is *realtime* if it is of delay 0.

We designate the various families of languages as follows.

Definition. Let

k-COUNTER(n) = $\{L \mid$ There is a quasirealtime k-counter machine accepting $L\}$

k-COUNTER(lin) = $\{L \mid$ There is a k-counter machine accepting L in linear time$\}$,

COUNTER(n) = $\bigcup_k k$-COUNTER(n).

COUNTER(lin) = $\bigcup_k k$-COUNTER(lin).

[2] For a string w, $|w|$ is the length of w.

We restrict our machines to be "blind" by forcing identical action for all counter configurations, and restrict them to be "partially blind" by not allowing transitions for negative counters and by forcing other transitions to ignore counter contents.

Definition. A k-counter machine $M=(K, \Sigma, \delta, q_0, F)$ is *blind* if for each q in K, a in $\Sigma \cup \{e\}$, $\delta(q, a, u_1, \ldots, u_k)=\delta(q, a, v_1, \ldots, v_k)$ for all u_i, v_i in $\{0, 1, -1\}$. A k-counter machine $M=(K, \Sigma, \delta, q_0 F)$ is *partially blind* if (1) for all q in K, a in $\Sigma \cup \{e\}$, $\delta(q, a, u_1, \ldots, u_k)=\phi$ whenever any u_i is -1 and (2) for each q in K, a in $\Sigma \cup \{e\}$, $\delta(q, a, u_1, \ldots, u_k)=\delta(q, a, v_1, \ldots, v_k)$ for all u_i, v_i in $\{0, 1\}$.

Definition. Let

BLIND $=\{L(M) \mid M$ is a blind multicounter machine$\}$,

BLIND$(n)=\{L(M) \mid M$ is a quasirealtime blind multicounter machine$\}$,

PBLIND $=\{L(M) \mid M$ is a partially blind multicounter machine$\}$,

k-PBLIND$(n)=\{L(M) \mid M$ is a quasirealtime partially blind k-counter machine$\}$,

PBLIND$(n)=\bigcup_k k$-PBLIND(n),

BLIND(lin) $=\{L \mid L$ is accepted in linear time by a blind multicounter machine$\}$, and

PBLIND(lin) $=\{L \mid L$ is accepted in linear time by a partially blind multicounter machine$\}$.

Thus, if M is blind we can write its transition function δ as simply a function on $K \times (\Sigma \cup \{e\})$. If it is partially blind, then should any counter go negative, no further transitions are defined and the machine is blocked. So in this case too, one can consider the transition function as a function of state and input alone, with a negative counter equivalent to a block. Notice that condition (1) in the definition of a partially blind machine would by itself cause no loss in power.

We now define the reversal bounded machines. In general, a reversal bounded tape of any type has a bound on the number of times its read-write head can change direction; a reversal bounded counter has a bound on the number of times it can switch from incrementing to decrementing the counter. For convenience, we shall use a bound of 1; justification for this further restriction appears in [1]. Thus, a reversal bounded counter will be one which can never be incremented once it is decremented.

Definition. A k-counter machine $M=(K, \Sigma, \delta, q_0, F)$ is *reversal bounded* if for any counter i, $1 \leq i \leq k$, and any subcomputation

$$(q_0, w, 0, \ldots, 0) \vdash^* (q_1, w_1, y_1, \ldots, y_k) \vdash^* (q_2, w_2, x_1, \ldots, x_k) \vdash^* (q_3, w_3, z_1, \ldots, z_k)$$

if $y_i > x_i$, then $x_i \geq z_i$.

Definition. Let

RBC = $\{L(M) \mid M$ is a reversal bounded multicounter machine$\}$,

RBC(n) = $\{L(M) \mid M$ is a quasirealtime reversal bounded multicounter machine$\}$,

RBC(lin) = $\{L(M) \mid$ There is a reversal bounded multicounter machine accepting L in linear time$\}$.

First we establish that RBC = RBC(lin) = RBC(n) = BLIND(n) = BLIND(lin) = BLIND (i.e., for reversal bounded machines quasirealtime suffices and such machines are equivalent to quasirealtime blind multicounter machines). Along the way, we shall use algebraic or operator characterizations of our families of languages, so we next define semiAFLs.

Definition. A *semi*AFL is a family of languages containing at least one language not ϕ or $\{e\}$ and closed under union, nonerasing homomorphism, inverse homomorphism and intersection with regular sets. A *full semi*AFL is a semiAFL closed under arbitrary homorphism. An AFL (*full* AFL) is a semiAFL (full semiAFL) closed under concatenation and Kleene +.[3] For a language L, the least semiAFL containing L is denoted by $\mathscr{M}(L)$ and called a principal semiAFL; the least intersection closed semiAFL containing L is denoted by $\mathscr{M}_\cap(L)$.

Definition. For a family of languages $\mathscr{L}$,

$\mathscr{H}(L) = \{h(L) \mid L \in \mathscr{L}, h$ a homomorphism$\}$,

$\mathscr{H}^{\text{lin}}(\mathscr{L}) = \{h(L) \mid L \in \mathscr{L}, h$ a homomorphism linearly bounded on $L\}$.

Each of our families can be expressed as the least intersection closed semiAFL containing a certain language and thus they are "principal" as intersection closed semiAFLs although they are not principal semiAFLs. Let $\#_a(w)$ be the number of a's in w and similarly for $\#_b(w)$.

Definition. Let $L_0 = \{a^n b^n \mid n \geq 0\}$,

$$L_1 = \{w \text{ in } \{a, b\}^* \mid \#_a(w) = \#_b(w)\},$$

$$L_2 = \{w \text{ in } \{a, b\}^* \mid w \text{ is in } L_1; \text{ if } w = xy, \text{ then } \#_a(x) \geq \#_b(x)\},$$

and

$$L_3 = (L_2 c)^*.$$

Techniques in [8, 9, and 10] show that RBC(n) is the least intersection closed semiAFL containing L_0, BLIND(n), the least intersection closed semiAFL containing L_1, PBLIND(n) the least intersection closed semiAFL containing L_2, and COUNTER(n), the least intersection closed semiAFL containing L_3

[3] Kleene + is the operation taking a language L into $L^+ = LL^*$.

(equivalently, COUNTER(n) is the least intersection closed AFL containing L_2). The "c" in the generator for COUNTER(n) indicates that counter machines can test whether or not a counter is 0 while partially blind counter machines can know only that a counter is nonnegative ($\#_a(x) \geq \#_b(x)$ for xy in L_2) and that it is zero at the end ($\#_a(w) = \#_b(w)$). The class of languages accepted by any "reasonable" family of machines consists of the homomorphic images of the languages accepted by the quasirealtime subfamily [7], so PBLIND $= \mathscr{H}$(PBLIND(n)). Linear time bounds on the machines correspond to linearly bounded homomorphisms on the languages, (cf. [3, 4, 5]), so RBC(lin) $= \mathscr{H}^{\text{lin}}(\mathscr{M}_\cap(L_0))$. We summarize.

Theorem 1.

(a) RBC(n) $= \mathscr{M}_\cap(L_0)$ and RBC(lin) $= \mathscr{H}^{\text{lin}}(M_\cap(L_0))$,

(b) BLIND(n) $= \mathscr{M}_\cap(L_1)$ and BLIND(lin) $= \mathscr{H}^{\text{lin}}(M_\cap(L_1))$,

(c) PBLIND(n) $= \mathscr{M}_\cap(L_2)$ and PBLIND(lin) $= \mathscr{H}^{\text{lin}}(M_\cap(L_2))$,

(d) COUNTER(n) $= \mathscr{M}_\cap(L_3)$.

We use two lemmas to establish RBC = RBC(n) = BLIND.

Lemma 1. BLIND(n) $\subseteq$ RBC(n)

Proof. Theorem 1 indicates that it suffices to show that L_1 is in RBC(n). Obviously, L_1 is in RBC(lin) since one need only count the a's on one counter and the b's on the other and then compare the two counters by zeroing them at the end. The trick is to show that this can be done in quasirealtime, and in fact with delay 0, by adding another counter. The reversal bounded delay 0 3-counter machine M accepting L_2 behaves as follows. First M guesses whether the number of a's will be odd or even and, if even, whether it will read half the a's before it receives half the b's or whether half the b's will appear before half the a's. Let us first consider the even case and describe M's behavior when it guesses that half the a's will arrive first; the transitions for the other case are obtained by using new states and exchanging the roles of a and b. Initially, M adds 1 to counters 1 and 2 for each a read, and 1 to counter 3 for each b. At some point, M has read n a's and guesses that this is half the total number of a's to be read. Then it still adds 1 to counter 3 for each b read but now for each a read, subtracts 1 from counter 1. Next, after reading m b's, M guesses that it has read half the b's in the input. From now on, M will be in accepting states only and so M will accept precisely when all counters are 0. Now M subtracts 1 from counter 1 for each a and subtracts 1 from counters 2 and 3 for each b. All counters will be 0 when M has read $2n$ a's (so counter 1 is empty) and $2m$ b's (so counter 3 is empty) and $n = m$ (so counter 2 is empty). Thus M accepts all words with equal even numbers of a's and b's. To handle the case where the

number of a's is odd, M performs the same actions except that for the first a and the first b, it leaves the counters alone. □

Now we show that BLIND(lin)$\subseteq$BLIND(n). This proof is more complicated. The basic idea is to move around "small loops" to convert a linear time machine to a quasirealtime one.

Lemma 2. BLIND(lin)$\subseteq$BLIND(n).

Proof. Let $M=(K, \Sigma, \delta, q_0, F)$ be a blind multicounter machine and $r \geq 2$ an integer such that if M accepts a word w it has an accepting computation for w which takes at most r Max $(1, |w|)$ steps. For each subset S of states, we construct a blind quasirealtime counter machine M_S such that $L(M)=\bigcup_{S\subseteq K} L(M_S)$. Since BLIND($n$) is closed under union, this will suffice.

The machine M_S simulates only computations of M passing through every state of S. The finite state control of M_S records which states have been entered at least once and so M_S will not accept unless all states of S have been entered at least once. If a given computation of M passes through a state q, then if M_S simulates that computation plus any "loop" of consecutive moves of M passing from q to q on e-moves (i.e., no input read), M_S is still simulating a legitimate (though perhaps not accepting) computation for the same input. Further, since M is blind, its moves do not depend on the counter configuration and switching an action from one part of the computation to another will not cause a "block". In particular, if q is in S and we move an e-loop around q to a different position in a computation of M_S, then M_S is still simulating a legitimate computation of M and can accept if it ends in an accepting state with all counters zero. To make M_S quasirealtime, we move the e-loops to follow the reading of input.

To each state q in K we can associate a finite (possibly empty) set of "small" loops, LOOP(q), that is, subcomputations of M consisting of consecutive e-moves in M (i.e., no input) from q back to q in at most $\# K$ steps.[4]

So M_S simulates a computation of M passing through all states in S. After each real input (in Σ), it may simulate up to $r-1$ (but no more) loops in LOOP(q) for any q in S. Since the computation must pass through q at some point, this is legitimate. In recording the states passed through, if a loop from LOOP(q) is simulated "out of order", i.e., when q is not actually the current state, the states in the loop, including q itself, are not recorded at that time. In addition to simulating the moves of M on real input and the loops from LOOP(q), M_S is also allowed to simulate, before real input, up to $\# K^2+r$ (but no more) moves of M on e input in the order of the actual computation. Hence, M_S is of delay at most $r(\# K)^2+r$ and so quasirealtime. Since M_S is simulating only legitimate computations of M, although perhaps in a different order, $L(M_S)$ is a subset of $L(M)$.

[4] For a finite set A, $\# A$ is the number of members of A.

If M accepts e, then by hypothesis it does so in some computation of r steps or less and so e will be in some $L(M_S)$. If M accepts a nonempty word w, it has some accepting computation C for w which takes at most $r|w|$ steps. Such a computation has at most $r-1$ e-moves per real input move. Any sequence of $\#K$ e-moves in a row contains a loop of $\#K$ or fewer steps. Divide the e-moves in C into such loops. Let S_1 be the set of entry states for such loops (a loop from q to q has entry state q) and let S_2 be the states C passes through outside of these loops. Let $S = S_1 \cup S_2$ and let C' be C with these loops excised. Thus M_S will have a computation which simulates C' but after each real input simulates up to $r-1$ more loops from $C - C'$ until all loops have been simulated. So M_S accepts w. Thus $L(M) = \bigcup_{S \subseteq K} L(M_S)$. □

Theorem 2. RBC = RBC(lin) = RBC(n) = BLIND(n) = BLIND(lin) = BLIND.

Proof. Lemmas 1 and 2 and our definitions yield:

$$\text{BLIND(lin)} \subseteq \text{BLIND}(n) \subseteq \text{RBC}(n) \subseteq \text{RBC(lin)} \subseteq \text{RBC}.$$

Baker and Book [1] showed that RBC $\subseteq$ RBC(lin). Clearly L_0 is in $\mathcal{M}(L_1)$ ($L_0 = a^*b^* \cup L_1$) so by Theorem 1, RBC(lin) $\subseteq$ BLIND(lin). Thus:

$$\text{BLIND(lin)} \subseteq \text{BLIND}(n) \subseteq \text{RBC}(n) \subseteq \text{RBC(lin)} \subseteq \text{RBC} \subseteq \text{RBC(lin)} \subseteq \text{BLIND(lin)},$$

so equality holds everywhere, which completes the proof. □

Now we show that blind multicounter machines are strictly less powerful than quasirealtime partially blind multicounter machines.

Theorem 3. BLIND $\subsetneq$ PBLIND(n).

Proof. It is known that $\mathcal{M}(L_1)$ and $\mathcal{M}(L_2)$ are incomparable [2], so our algebraic characterization of BLIND(n) will not suffice without further work. However, L_0 is obviously in $\mathcal{M}(L_2)$ ($L_0 = L_2 \cup a^*b^*$), so by Theorem 1, RBC(n) $\subseteq$ PBLIND(n) and by Theorem 2, BLIND $\subseteq$ PBLIND(n).

We complete the proof by showing that L_2 is not in BLIND. As in the proof of Lemma 2, we notice that blind machines are indifferent to their counter contents until the end, when the counters must be zero for acceptance. So if the moves of an accepting computation can be written $C: C_1C_2C_3C_4C_5$ where C_2 and C_4 start in state p and end in state q, then $C': C_1C_4C_3C_2C_5$ is also an accepting computation.

Suppose $L_2 = L(M)$ for a blind k-counter machine $M = (K, \{a, b\}, \delta, q_0, F)$. Let $n = \#K$. Consider the following word in L_2.

$$w = aba^2b^2 \cdots a^{n^2+1}b^{n^2+1}a^{n^2+2}b^{n^2+2}a^{n^2+3}b^{n^2+3}.$$

Let C be an accepting computation of M on w. Let A_i be the subcomputation of C on a^i and B_i the subcomputation on b^i. Let B_i start with state p_i and end with state q_i. Since there are only n^2 distinct pairs of states, we must have $p_i = p_{i+s}$ and $q_i = q_{i+s}$ for some i's with $2 \leq i < i+s \leq n^2+2$. So, exchanging B_i and B_{i+2}, we obtain another accepting computation of M

$$C: A_1 B_1 \cdots A_{i-1} B_{i-1} A_i B_{i+s} \cdots A_{i+s} B_i \cdots A_{n^2+3} B_{n^2+3}$$

which accepts

$$w' = ab \cdots a^{i-1} b^{i-1} a^i b^{i+s} \cdots a^{i+s} b^i \cdots a^{n^2+3} b^{n^2+3}$$

which is not in L_2. This is a contradiction, so L_2 is not in BLIND. □

We observe that, unlike the case for blind machines, partially blind machines can do more in linear time than in quasirealtime. In fact, there is a language accepted in linear time by a partially blind multicounter machine but not accepted in quasireal-time by any multicounter machine.

Theorem 4.

$$\text{PBLIND}(n) \subsetneq \text{PBLIND(lin)} \subseteq \text{PBLIND},$$

and

$$\text{PBLIND(lin)} - \text{COUNTER}(n) \neq \phi.$$

Proof. For a word w in $1\{0, 1\}^*$, let $\nu(w)$ be the integer of which w is the binary expression. Let

$$L = \{wc0^m | m \leq \nu(w),\ w \text{ in } 1\{0, 1\}^*\}.$$

We construct a partially blind 2-counter machine $M = (K, \{0, 1, c\}, \delta, q_0, \{f_1, f_2\})$ to accept L in linear time with linear factor 8. Machine M tries to implement a standard algorithm for translating w into $\nu(w)$ and after reading c, compares m against its guess, A, for $\nu(w)$ and then tries to zero the counters. The crux of the conversion algorithm is to go from a configuration with s in one counter to one with $2s$ in the other by a subroutine which subtracts 1 from the counter and adds 2 to the second for s steps until the first counter is zero. Since M is partially blind, it can only guess when the first counter is empty and this subroutine complete. If all guesses for each application of the subroutine are correct, $A = \nu(w)$. Otherwise, $A \leq \nu(w)$. Let $K = \{q_0\} \cup \{q_i, q_i', \bar{q}_i, b_i, p_i, f_i | i = 1, 2\}$. We describe δ as a function of state and input (or e) alone. Let

$$\delta(q_0, 1) = \{(q_1, 1, 0), (b_1, 1, 0)\}$$

$$\delta(q_1, 1) = \{(\bar{q}_1, 0, 1)\}, \quad \delta(q_1, 0) = \{(q_1', -1, 1)\},$$

$$\delta(q_1', e) = \{(\bar{q}_1, 0, 1), (q_2, 0, 1), (b_2, 0, 0)\}, \quad \delta(\bar{q}_1, e) = \{(q_1', -1, 1)\},$$

$$\delta(q_1, c) = \delta(b_1, c) = \{(p_1, 0, 0)\}, \delta(b_1, 0) = \delta(b_1, 1) = \{(b_1, 0, 0)\},$$

$$\delta(p_1, 0) = \{(p_1, -1, 0)\}, \delta(p_1, e) = \{(f_1, -1, 0)\},$$

$$\delta(f_1, e) = \{(f_1, -1, 0)\}.$$

The transitions for the states with subscript 2 are obtained from those with subscript 1 by exchanging subscripts 1 and 2 and the instructions for counters 1 and 2. Elsewhere, δ is undefined. After reading c, M enters a state p_i with an integer A in counter i, with $1 \leqq A \leqq \nu(w)$; for M to accept, the other counter must be 0. Then M subtracts m from A, blocking if $m > A$. Finally, M can zero counter i and end in state f_i. So $L(M) \subseteq L$. On the other hand, let $y = wc0^m$ be in L and let binary m have $r \leqq |w|$ digits. If $r = |w|$, then M has an accepting computation in which it correctly converts w into $\nu(w)$ in $\nu(w)$ steps, then reads c, reads 0^m subtracting m from $\nu(w)$ in m steps, and zeroes the counters in another $\nu(w) - m$ steps. Since $\nu(w) \leqq 2\,\mathrm{Max}(m, 1)$, M accepts y in at most $4|y|$ steps. Now suppose $r < |w|$. Then M has an accepting computation in which it correctly converts the leftmost $r+1$ digits of w into an integer t, $m \leqq t \leqq 4\,\mathrm{Max}(m, 1)$, in t steps and then uses the b_i states to read up through c without altering the counters, finally comparing m to t and zeroing the counters in t steps. Hence, M accepts y in at most $8|y|$ steps. So L is in PBLIND(lin).

If L were in COUNTER(n), it would be accepted with delay d by some k-counter machine M. For $|w| = n$, M can have at most $(dn + n + d)^k$ distinct counter configurations as it reads c and hence for some r, at most $r\,\mathrm{Max}(n^k, 1)$ total configurations. But M must accept 2^{n-1} words $wc0^{\nu(w)}$ with $|w| = n$. Hence, there must be words $w_1, w_2, |w_1| = |w_2|$, and $\nu(w_1) < \nu(w_2)$, such that M has accepting computations for $w_i c0^{\nu(w_i)}$, $i = 1, 2$ which are in the same total configuration upon reading c. So M accepts $w_1 c0^{\nu(w_2)}$, a contradiction. Thus, L is in PBLIND(lin) − COUNTER (n). □

We close by considering whether PBLIND contains nonrecursive sets and whether PBLIND(n) equals COUNTER(n). For this purpose, we appeal to the connection with Petri net languages and the claimed decidability of the reachability problem for vector addition systems [16].

We define our Petri net machines to resemble counter machines as closely as possible.

Definition. A *k-place Petri net machine* $M = (k, \Sigma, T, F)$ consists of a finite number k of *places*, a finite set Σ of *input symbols*, a subset $F \subseteq \{1, \ldots, k\}$ of *accepting places*, and a finite set $T \subseteq N^k \times \Sigma \times N^k$ of *labelled transitions*. An *instantaneous description* (ID) of M is a member of $\Sigma^* \times N^k$; in ID $(w, n_1, \ldots, n_k)$, we call w the *input to be processed* and n_i the *number of tokens in place i*. If $(u_1, \ldots, u_k, a, v_1, \ldots, v_k)$ is in T, a is in Σ and $(aw, y_1, \ldots, y_k)$ is an ID such that $u_i \leqq y_i$ for all i, $1 \leqq i \leqq k$, then we write $(aw, y_1, \ldots, y_k) \vdash (w, (y_1 - u_1) + v_1, \ldots, (y_k - u_k) + v_k)$ and speak of this move as "firing $(u_1, \ldots, u_k, a, v_1, \ldots, v_k)$". We let $\vdash^*$ be the transitive reflexive closure of $\vdash$. An ID $(e, n_1, \ldots, n_k)$ is *accepting* if for some t in F, $n_t = 1$ and $n_i = 0$ for all $i \neq t$.

The *language accepted by* M is

$$L(M)=\{w \mid \text{there is an accepting ID } I, (w, 1, 0, \ldots, 0)\vdash^* I\}$$

and is called a *computation sequence set* or *Petri net language*. We let CSS_k be the family of languages accepted by k-place Petri net machines and $\mathrm{CSS}=\bigcup_k \mathrm{CSS}_k$.

Observe that a k-place Petri net machine M is already defined as a quasirealtime partially blind k-counter machine if we regard "places" as "counters", with the following restrictions and modifications: (1) M *has no* e-moves and so is nondeterministic realtime, (2) M has no states (or is one-state), (3) M can add or subtract integers other than 1 and in one step first subtracts and then adds, (4) M starts with 1 in the first place (counter) and the others 0, (5) M accepts with all but one place (counter) empty and that place contains 1 and is a designated accepting place. It is tedious but straightforward to construct a quasirealtime partially blind k-counter machine to simulate M. On the other hand, the given restrictions and modifications do not limit accepting power. Standard constructions can always convert a quasirealtime partially blind k-counter machine to one with delay 0. Instead of t states, we can use another t counters with transitions arranged so that one of these counters contains 1 and the others 0 and, further, counter i is 1 if and only if the machine is supposed to be in state q_i. Thus, the following lemma is immediate (and is not new).

Lemma 3.

(a) *For each* $k \geqq 1$, $\mathrm{CSS}_k \subseteq k$-PBLIND$(n)$.

(b) *If* M *is a nondeterministic realtime partially blind* k*-counter machine with* t *states, which does not accept* e*, then* $L(M)$ *is in* $\mathrm{CSS}_{(k+t)}$*; if* e *is in* $L(M)$, $L(M)$ *is in* $\mathrm{CSS}_{(k+t+1)}$.

Hence, CSS is principal as an intersection closed semiAFL but is not a principal semiAFL.

Theorem 5. $\mathrm{CSS}=\mathrm{PBLIND}(n)=\mathscr{M}_\cap(L_2)$ *and for each* k, $\mathrm{CSS}_k \subsetneq \mathrm{CSS}_{k+1}$, k-PBLIND$(n)\subsetneq(k+1)$-PBLIND(n), *and so* CSS *is not a principal semi*AFL.

Proof. For each i, let a_i and b_i be new symbols, let $\Sigma_k=\{a_1, \ldots, a_k, b_1, \ldots, b_k\}$, and let h_i be the homomorphism which turns a_i to a, and b_i to b and erases all other symbols. For each k, let

$$A_k=\{w \text{ in } \Sigma_k^* \mid \text{for } 1 \leqq i \leqq k,\ h_i(w) \text{ is in } L_2\}.$$

A k-place Petri net machine M_k for $a_1 A_k$ has accepted place 1 and transitions for input a_i which add 1 to place i, and for input b_i which decrease place i by 1, leavin

the other places unchanged. The language

$$A_{k+1} \cap a_1^* \cdots a_{k+1}^* b_{k+1}^* \cdots b_1^*$$
$$= \{a_1^{n_1} \cdots a_{k+1}^{n_{k+1}} b_{k+1}^{n_{k+1}} \cdots b_1^{n_1} \mid n_i \geq 0, 1 \leq i \leq k+1\}$$

is not in k-COUNTER(n) [6, 10], and hence not in k-PBLIND(n). Thus, A_{k+1} and $a_1 A_{k+1}$ are not in k-PBLIND(n), which is a semiAFL. Hence, $a_1 A_{k+1}$ is in $\mathrm{CSS}_{k+1} - \mathrm{CSS}_k$ and in $(k+1)$-PBLIND(n) $-$ k-PBLIND(n). Since k-PBLIND(n) is a semiAFL [7, 8], PBLIND(n) = CSS cannot be a principal semiAFL. □

The membership problem for $\mathscr{H}$(CSS) = PBLIND can be related to the reachability problem for vector addition systems and the emptiness problem for Petri net languages in the usual way.

Lemma 4. *The following are equivalent*:

(1) *The reachability problem for vector addition systems is decidable.*

(2) *The emptiness problem for Petri net languages is decidable.*

(3) *The membership problem for* PBLIND *is decidable.*

Proof. Statements (1) and (2) are known to be equivalent [15]. If h is the homomorphism which erases everything, then for any language L, $L \neq \phi$ if and only if $h(L)$ contains e. The transformations that take a Petri net machine M and a homomorphism h into a partially blind multicounter machine M' with $L(M') = h(L(M))$ are algorithmic. Hence (3) implies (2). On the other hand, suppose that the emptiness problem for Petri net languages were decidable, i.e., that there were an algorithm to decide for each Petri net machine M whether $L(M) = \phi$. Let M be a partially blind multicounter machine with input alphabet Σ and c a symbol not in Σ. There is an algorithm to obtain from M and w a partially blind multicounter machine M_w such that $L(M_w) = L(M) \cap \{w\}$. There is an algorithm to obtain from M_w the partially blind multicounter machine $M_{w,c}$ which is identical to M_w for transitions on input from Σ but substitutes for e-moves, transitions on input c. Thus $M_{w,c}$ is a quasirealtime (and delay 0) partially blind multicounter machine such that $L(M_{w,c}) \neq \phi$ if and only if w is in $L(M_w)$ if and only if w is in $L(M)$. There is an algorithm to obtain from $M_{w,c}$ a Petri net machine $M'_{w,c}$ with $L(M_{w,c}) = L(M'_{w,c})$. Hence, there is an algorithm to decide whether $L(M_{w,c}) = \phi$ and thus to decide whether w is in $L(M)$. So (2) implies (3). □

Theorem 6. *If the reachability problem for vector addition systems is decidable, the following hold*

(1) PBLIND *is a proper subset of the family of recursive languages,*

(2) PBLIND *does not contain* L_0^* *and is not closed under Kleene* $+$,

(3) PBLIND(n) *is not closed under Kleene* $+$,

(4) PBLIND(n) *has a decidable emptiness problem,*
(5) PBLIND(n)$\subsetneq$COUNTER(n).

Proof. According to [16], the reachability problem for vector addition systems is decidable and hence each member of PBLIND has a decidable membership problem and is recursive. Since PBLIND can be recursively enumerated (by Godelization of partially blind multicounter machines), it cannot contain all recursive languages. Thus (1) would be a direct consequence of the decidability of the reachability problem for vector addition systems.

Now PBLIND is an intersection closed full semiAFL and contains L_0, so it would contain all recursively enumerable languages if either it contained L_0^* [11, 13], or it were closed under Kleene +, so (1) implies (2), and similarly for PBLIND(n). Lemma 4 and Theorem 5 show that (1) implies (4). Since COUNTER(n) contains L_0^* (and $\hat{\mathscr{R}}$(COUNTER(n)) is the family of recursively enumerable languages), (5) is a consequence of (2). □

Remarks. Paterson [15] states that Hack [12] has shown that CSS is not closed under Kleene +. Wrathall [17] has shown that any context-free language can be defined from L_0 and regular sets by use of the Boolean operations, inverse homomorphism and length-preserving homomorphism. There are context-free languages not in COUNTER(n) [6, 9]. Hence neither BLIND nor PBLIND(n) can be closed under complementation.

A curious point appears when we consider, in the proof of Theorem 4, the binary conversion language L which is in PBLIND(lin). The task involved in L is conversion of an integer n in binary to an integer m in unary with $m \leq n$. However, assuming the decidability of the reachability problem for vector addition systems, partially blind machines cannot convert a binary to a larger unary integer. That is, PBLIND does not contain

$$L' = \{wc0^m \mid w \in 1\{0, 1\}^*, m \geq \nu(w)\}.$$

For if PBLIND contained L', then, since it is an intersection and reversal closed semiAFL, it would contain

$$\{wcx \mid w, x^R \in 1\{0, 1\}^*, \nu(w) \geq \nu(x^R)\},$$

hence $\{wcw^R \mid w \in \{0, 1\}^*\}$ and so all r.e. languages [1].

Related results on commutative languages and vector addition systems appear in [18] and [19], which contain some of the results of this paper in a different notation.

Open questions

Among the questions not considered here are (1) the relationship among off-line (two-way input tape) blind, reversal bounded, partially blind and quasirealtime

multicounter machines, and (2) the relationship between PBLIND and PBLIND(lin).

References

[1] B.S. Baker and R.V. Book, Reversal-bounded multipushdown machines, *J. Comput. System Sci.* **3** (1974) 315–332.

[2] L. Boasson, Two iteration theorems for some families of languages, *J. Comput. System Sci.* **7** (1973) 583–596.

[3] R. Book and S. Greibach, Quasirealtime languages, *Math. Syst. Theory* **4** (1970) 97–111.

[4] R.V. Book, S. Greibach, O. Ibarra and B. Wegbreit, Tape-bounded Turing acceptors and principal AFLs, *J. Comput. System Sci.* **4** (1970) 622–625.

[5] R.V. Book and B. Wegbreit, A note on AFLs and bounded erasing, *Information and Control* **19** (1971) 18–29.

[6] P.C. Fischer, A.R. Meyer and A.L. Rosenberg, Counter machines and counter languages, *Math. Syst. Theory* **2** (1968) 265–283.

[7] S. Ginsburg and S. Greibach, Abstract families of languages, *Memoirs Amer. Math. Soc.* **87** (1969) 1–32.

[8] S. Ginsburg and S. Greibach, Principal AFL, *J. Comput. System. Sci.* **4** (1970) 308–338.

[9] S. Ginsburg and S. Greibach, On AFL Generators for finitely encoded AFA, *J. Comput. System Sci.* **7** (1973) 1–27.

[10] S.A. Greibach, Remarks on the complexity of nondeterministic counter languages, *Theoret. Comput. Sci.* **1** (1976) 269–288.

[11] S. Greibach and S. Ginsburg, Multi-tape AFA, *J. ACM* **19** (1972) 192–221.

[12] M. Hack, Petri net languages, Computation Structures Group Memo 124, Project MAC, Massachusetts Institute of Technology (1975).

[13] J. Hartmanis and J. Hopcroft, What makes some language theory problems undecidable *J. Comput. System Sci.* **3** (1969) 196–217.

[14] M. Minsky, Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines, *Annals Math.* **74** (1961) 437–455.

[15] J.L. Paterson, Computation sequence sets, *J. Comput. System Sci.* **13** (1976) 1–24.

[16] G.S. Sacerdote and R.L. Tenney, The decidability of the reachability problem for vector addition systems, *Proc. Ninth Ann. ACM Symp. on Theory of Computing*, Boulder, Colorado (May 1977) 61–76.

[17] C. Wrathall, Characterizations of the Dyck sets, *RAIRO* **11** (1977).

[18] M. Latteux, Cônes rationnels commutativement Clos, *RAIRO Informat. Théorique* **11** (1977) 29–51.

[19] A. Arnold and M. Latteux, Vector addition systems and semi-Dyck languages, unpublished manuscript (1977).