INFORMATION AND CONTROL 63, 88–112 (1984)

# Can Message Buffers Be Axiomatized in Linear Temporal Logic?*

A. P. SISTLA

*University of Massachusetts, Amherst, Massachusetts*

E. M. CLARKE

*Carnegie-Mellon University, Pittsburgh, Pennsylvania*

N. FRANCEZ

*The Technion, Haifa, Israel*

AND

A. R. MEYER

*Massachusetts Institute of Technology, Cambridge, Massachusetts*

Message passing is one of the primary modes of interprocess communication in a distributed system. In this paper we investigate the possibility of *characterizing* and *axiomatizing* different message passing systems in temporal logic. Specifically, we consider FIFO buffers (queues), LIFO buffers (stacks) and unordered buffers (bags). We show that all bounded buffers are characterizable in propositional temporal logic (PTL) and so are axiomatizable. We prove that the theory of unbounded FIFO buffers is $\pi_1^1$-complete and so is not axiomatizable. We also prove that the theories of unbounded LIFO and unordered buffers are decidable and hence are axiomatizable.

## 1. INTRODUCTION

Exchange of information between executing processes is one of the primary reasons for process interaction. Many distributed systems implement explicit message passing primitives to facilitate intercommunication. Typically, a process executes a *write* command to pass a message to another process, and the target process accepts the message by

* This research was partially supported by NSF Grant MCS79-08365, and NSF Grant MCS80-10707.

0019-9958/84 $3.00

executing a *read* command. The semantics of *write* and *read* may differ considerably depending on the methods used for storing or buffering messages that have been sent but not yet accepted by the receiving process.

Because message passing systems are so widely used, it is important to develop formal techniques for reasoning about them. In this paper we investigate the possibility (impossibility) of using *linear temporal logic* to characterize and axiomatize different message buffering mechanisms. This logic was originally introduced as a formal system for reasoning about sequences of events that are totally ordered in time. Recently, linear temporal logic has been proposed by Manna and Pnueli [2] and Owicki and Lamport [3] as an appropriate formal system for reasoning about parallel programs. The logic permits the description of a program's execution history without the explicit introduction of program states or of time. Moreover, important correctness properties such as mutual exclusion, freedom from deadlock and absence of starvation can be elegantly expressed in this system.

Specifically, we consider FIFO buffers (*queues*), LIFO buffers (*stacks*) and unordered buffers (*bags*). The set of distinct messages that can be written into the buffer is called the *message alphabet*. We specify a message buffer by the set of all valid infinite input/output message sequences.

Characterizing a message buffer in temporal logic consists of obtaining a formula that is true exactly on the set of sequences specifying the buffer. We show that it is possible to characterize bounded buffers over a finite alphabet in propositional linear temporal logic (PTL) and show how to obtain such a characterization. Although such bounded buffers can be characterized using $\omega$-regular expressions, it is not obvious that they can be characterized in PTL since this logic is provably less expressive than $\omega$-regular expressions [4]. Using first order temporal logic, we can give a uniform characterization of bounded buffers which is independent of the message alphabet. We also show that unbounded buffers are not characterizable in PTL.

Since the formulae we obtain for bounded buffers may be quite complicated, we introduce an extension of PTL in which certain atomic propositions are designated as auxiliary. The auxiliary propositions are not interpreted and are treated like existentially quantified monadic predicates. We give simple and succinct formulae in the extended logic which characterize bounded message buffers over a finite alphabet.

We also consider the problem of *axiomatizing* the various types of message buffers. A model of a message buffer is an infinite sequence of states denoting a series of legal read/write operations on the buffer. The theory of a message buffer is the set of all PTL formulae which are true in all models of the buffer. We say that a message buffer is axiomatizable iff the theory of the buffer is recursively enumerable. A simple and complete axiom system for all the bounded buffers can be given using the characterization of these buffers in PTL. We show that, in general, unbounded FIFO buffers are not axiomatizable in PTL. We also show that unbounded LIFO buffers and unbounded unordered buffers are axiomatizable in PTL and in fact, we prove that the theories of these buffers are decidable.

One of our principal motivations for this work has been to study possible axiomatizations of computational behavior by temporal logic, and so we have formulated most of our results and indeed the title of this paper in terms of axiomatizability. Nevertheless, the detailed structure of axiom system actually plays no role in our results. All our non-axiomatizability results are in fact proofs that various theories are not recursively enumerable, and some of our axiomatizability results are proofs that the corresponding theories are recursively enumerable.

The paper is organized as follows: Section 2 defines the syntax and semantics of the linear temporal logic that we use in the remainder of the paper. In Section 3 we specify precisely those properties of message buffers that we would like to capture in temporal logic. Section 4 shows that bounded buffers can be characterized in the logic and describes how uninterpreted auxiliary proposition symbols can be added to simplify this construction. We also prove that it is impossible to give a characterization of unbounded message buffers in PTL. In Section 5 we consider axiomatization of message buffers in PTL. We show that unbounded FIFO message buffers are not axiomatizable in PTL while unbounded LIFO and unordered buffers are axiomatizable. The paper concludes in Section 6 with a summary and discussion of our results.

## 2. Linear Temporal Logic

The language of PTL uses certain symbols called atomic propositions drawn from a finite set $\mathscr{P}$, the propositional connectives $\wedge$, $\sim$, and the temporal modalities $\mathbf{X}$ ("next-time"), $\mathbf{U}$ ("until"), $\mathbf{Y}$ ("last-time"), $\mathbf{S}$ ("since"), together with the parenthesis.

A *well-formed formula* in PTL is either an atomic proposition or is of the form $(\sim f_1)$, $(f_1 \wedge f_2)$, $(\mathbf{X} f_1)$, $(f_1 \mathbf{U} f_2)$, $(\mathbf{Y} f_1)$, $(f_1 \mathbf{S} f_2)$, where $f_1, f_2$ are well-formed formulae. We avoid parenthesis whenever the implied parsing of the formula is understood from the context. In addition, we use the following abbreviations:

$$(f_2 \vee f_2) \equiv \sim(\sim f_1 \wedge \sim f_2), (f_1 \supset f_2) \equiv \sim f_1 \vee f_2,$$
$$\mathbf{F} f \equiv (\text{True}\, \mathbf{U} f), \mathbf{G} f \equiv \sim(\mathbf{F} \sim f).$$

$\mathbf{F}$, $\mathbf{G}$ are the "sometimes," "always" operators, respectively. A state is a mapping from $\mathscr{P}$ into the set {True, False}. Let $\mathscr{M}$ denote the set of all states. Note that $\mathscr{M}$ is finite. A model is a $\omega$-sequence of states. An interpretation is an ordered pair $(t, i)$, where $t$ is a model and $i \geqq 0$ is an integer specifying the present state. We define the truth of a formula $f$ in an interpretation $(t, i)$ $(t, i \models f)$ inductively as follows:

$t, i \models P$ where $P$ is atomic iff $t_i(P) = \text{True}$;

$t, i \models f_1 \wedge f_2$ iff $t, i \models f_1$ and $t, i \models f_2$;

$t, i \models \sim f_1$ iff not $(t, i \models f_1)$;

$t, i \models \mathbf{X} f_1$ iff $t, i+1 \models f_1$;

$t, i \models f_1 \mathbf{U} f_2$ iff there exists a $k \geqq i$ such that $t, k \models f_2$ and for all $j$ such that $i \leqq j < k$, $t, j \models f_1$;

$t, i \models \mathbf{Y} f_1$ iff $i > 0$ and $t, i-1 \models f_1$;

$t, i \models f_1 \mathbf{S} f_2$ iff there exists a $k \leqq i$ $t, k \models f_2$ and for all $j$ such that $k < j \leqq i$ $t, j \models f_1$;

It is to be noted that $(t, i) \models \mathbf{F}(f)$ iff there exists $j \geqq i$ such that $(t, j) \models f$, and $(t, i) \models \mathbf{G}(f)$ iff for all $j \geqq i$ $(t, j) \models f$. A formula is *satisfiable* iff it is true in some interpretation and it is *valid* iff it is true in all interpetations. A consistent and complete axiomatization for the set of validities of PTL is presented in [14]. In [10] such an axiomatic system is presented for a restricted PTL that uses **X**, **U** as the only temporal modalities.

## 3. Modelling Message Buffers

We characterize a message buffer by the set of legal *read/write* sequences allowed on the bugger. A *write* operation writes a message into the buffer; a *read* operation reads a message from the buffer and deletes it. At most one read or write operation is permitted at any instant of time. In the case of bounded buffers a write request will be rejected when the buffer is full; similarly, a read request on an empty buffer will be rejected. Rejected read/write requests are not included in the sequences of legal operations characterizing the buffer. We consider below three types of message buffers: FIFO buffers (*queues*), LIFO buffers (*stacks*), and unordered buffers (*bags*). In FIFO buffers the *earliest* written message currently in the buffer is the ouput for the next read request; with LIFO buffers the *latest* written message currently in the buffer is used; and with unordered buffers any message present in the buffer may be output.

Let $\Sigma$ be the message alphabet and $\mathscr{P}_\Sigma$ be the set of atomic propositions $\{R_\sigma \mid \sigma \in \Sigma\} \cup \{W_\sigma \mid \sigma \in \Sigma\}$ such that $\mathscr{P}_\Sigma \subseteq \mathscr{P}$. Let $ST = \{\phi \mid \phi : \mathscr{P} \rightarrow \{\text{True}, \text{False}\}$ such that $\phi(P) = \text{True}$ for at most one $P$ in $\mathscr{P}_\Sigma\}$. If $R_\sigma(W_\sigma)$ is true in a state, then it indicates that the message $\sigma$ is read (written) from (into) the buffer in that state. Note that at most one operation (a read or a write) occurs in any state in $ST$.

Let $t \in ST^* \cup ST^\omega$ and $i_0 < i_1 < \ldots$, be all the instances at which some messages $\sigma_0, \sigma_1, \ldots$, are read from the buffer, i.e., $t_{i_k}(R_{\sigma_k}) = \text{True}$ for $k \geqq 0$. Then $\pi_r(t)$ denotes the sequence $(\sigma_0, \sigma_1, \ldots,)$. Similarly, we define $\pi_w(t)$. Intuitively, $\pi_r(t)$ $(\pi_w(t))$ denotes the sequence of messages read from the buffer (written into the buffer) in $t$. Let $t[i]$ denote the sequence $(t_0, t_1, \ldots, t_i)$, then $nb(i) = \text{length}(\pi_w(t[i])) - \text{length}(\pi_t(t[i]))$ is the number of messages in the buffer just after the instance $i$.

Let $\text{FIFO}_{\Sigma,k}$ be the set of all infinite sequences of states which denote legal series of read/write operations on a FIFO buffer of size $k$. Similarly, let $\text{LIFO}_{\Sigma,k}$, $\text{UNOR}_{\Sigma,k}$ be the corresponding sets of sequences for LIFO and unordered buffers, respectively. Unbounded buffers will be denoted in this scheme of notation by $k = \infty$. For $k \geqq 0$,

$\text{FIFO}_{\Sigma,k} = \{t \in ST^\omega \mid$ for all $i \geqq 0$, $\pi_r(t[i])$ is a prefix of $\pi_w(t[i])$ and $nb(i) \leqq k\}$;

$\text{LIFO}_{\Sigma,k} = \{t \in ST^\omega \mid$ for all $i \geqq 0$, $0 \leqq nb(i) \leqq k$ and if for some $\sigma \in \Sigma$, $(t, i) \models R_\sigma$ then there exists a $j < i$ such that $(t, j) \models W_\sigma$ and $nb(j) = nb(i-1)$ and for all $l$ such that $j \leqq l < i$ $nb(l) \geqq nb(j)\}$;

$\text{UNOR}_{\Sigma,k} = \{t \in ST^\omega \mid$ for all $i \geqq 0$, $0 \leqq nb(i) \leqq k$ and for all $\sigma \in \Sigma$, the number of writes of the message $\sigma$ up to $i \geqq$ the number of reads of the message $\sigma$ up to $i\}$.

For a finite alphabet $\Sigma$, a formula $f$ in PTL *characterizes* a FIFO message buffer of size $k$ iff $\{t \mid (t, 0) \models f\} = \text{FIFO}_{\Sigma,k}$. Similarly, we define what it means to characterize LIFO and unordered buffers.

We say that a sequence $t$ is a behavior of a FIFO (LIFO or unordered) buffer with *liveness* property iff $t$ is a legal sequence as defined above and the buffer becomes empty infinitely often in $t$. That is, $nb(i) = 0$ in $t$ for infinitely many values of $i$. This property guarantees that every message written into the buffer is eventually read which is a liveness property. As above we define what it means to characterize a message buffer with liveness property.

A *model* or *history* of a message buffer is an infinite sequence of states denoting a legal series of read/write operations on the buffer, as defined above. The *theory* of a message buffer is the set of all PTL formulae which are true in all interpretations of the form $(t, 0)$, where $t$ is a model of the buffer. We say that a message buffer is *axiomatizable* in PTL iff the theory of the buffer is recursively enumerable.

## 4. Characterizing Message Buffers

### 4.1. *Direct Characterization in PTL*

In this section we show how we can characterize bounded buffers over a finite alphabet using PTL. We let $fb_k$, $lb_k$, $ub_k$, respectively, denote formulae in PTL characterizing FIFO, LIFO, and unordered message buffers of size $k$ over the finite message alphabet $\Sigma$. First we describe how to obtain the formulae for the buffers with $k=1$ and $k=2$.

Let $\Sigma$ be a finite message alphabet, and $\mathscr{P}_\Sigma = \{R_\sigma \mid \sigma \in \Sigma\} \cup \{W_\sigma \mid \sigma \in \Sigma\}$ be the set of atomic propositions. Throughout this section we use the following abbreviations:

$$W = \bigvee_{\sigma \in \Sigma} W_\sigma$$

$$R = \bigvee_{\sigma \in \Sigma} R_\sigma$$

$$Ex = \left( \bigwedge_{\sigma_1 \neq \sigma_2} \sim(R_{\sigma_1} \wedge R_{\sigma_2}) \right) \wedge \left( \bigwedge_{\sigma_1 \neq \sigma_2} \sim(W_{\sigma_1} \wedge W_{\sigma_2}) \right) \wedge \sim(W \wedge R)$$

$$I = \mathbf{G}(Ex).$$

$I$ asserts that at any instant at most one operation occurs on the buffer.

In the case of buffer size $=1$, FIFO, LIFO, and unordered buffers are identical and the buffer behavior is as follows:

1. The *writes* and *reads* occur alternately;

2. The message read in each *read* operation is the message written by the previous *write* operation. Thus, $fb_1 = I \wedge f_a \wedge f_b$, where

$$f_a = \mathbf{G}((W \wedge \mathbf{XF}(R \vee W)) \supset \mathbf{X}(\sim W \,\mathbf{U}\, R))$$
$$\wedge\ \mathbf{G}((R \wedge \mathbf{XF}(R \vee W)) \supset \mathbf{X}(\sim R \,\mathbf{U}\, W));$$

$$f_b = \mathbf{G}\left( \bigwedge_{\sigma \in \Sigma} (R_\sigma \supset (\sim W \,\mathbf{S}\, W_\sigma)) \right).$$

The first (second) conjunct in $f_a$ asserts that every write (read) operation which is not the last operation on the buffer is followed by a read (write) operation before any other write (read) operation. It is easily seen that $f_a$ and $f_b$ assert properties (1) and (2), respectively.

Intuitively, the operation of a buffer of size two can be described as follows. Initially, *writes* and *reads* occur alternately. This continues until two writes occur successively without a *read* operation in between, and the buffer becomes full (formula $l_2$ expresses this). Subsequently, *reads* and *writes* will again begin to alternate. After each *read* the buffer will have one message and after each *write* operation the buffer becomes full. This may continue forever, or until two *reads* occur successively without a *write* in between, making the buffer empty ($r_2$ expresses this); now the previous sequence repeats. This behavior is common for FIFO, LIFO, and unordered buffers of size two. The formulae $l_2, r_2$ are given below:

$$l_2 = W \wedge \mathbf{X}(\sim R \,\mathbf{U}\, W)$$

$$r_2 = R \wedge \mathbf{Y}(\sim W \,\mathbf{S}\, R).$$

In the remainder of this section we will frequently use the formula alt($p$,-$q$, $c$) given below:

$$\text{alt}(p, q, c) = [(g \,\mathbf{U}\, c) \vee \mathbf{G}(g \wedge \sim c)] \wedge [(\sim c \,\mathbf{U}\, p) \supset (\sim q \,\mathbf{U}\, p)],$$

where

$$g = (p \supset \mathbf{X}(\sim p \,\mathbf{U}\, q)) \wedge (q \supset [\mathbf{X}(\sim q \,\mathbf{U}\, p) \vee \mathbf{X}(\sim (p \vee q) \,\mathbf{U}\, c)]).$$

The first conjunct in alt($p$, $q$, $c$) asserts that either there is a future instance at which $c$ occurs and until this instance $p, q$ occur alternately, or throughout the future $p, q$ occur alternately without $c$ occurring anywhere. The second conjunct asserts that if $p$ occurs then it occurs before $q$. Thus, the previous intuitive description of the behavior of the buffer of size two is captured by the formula *bv* given below.

$$bv = \text{alt}(W, R, l_2) \wedge \mathbf{G}[l_2 \supset \mathbf{X}\,\text{alt}(W, R, r_2)] \wedge \mathbf{G}[r_2 \supset \mathbf{X}\,\text{alt}(W, R, l_2)];$$

*bv* asserts that $l_2, r_2$ occur alternately with alternating *read* and *writes* occuring in between. Any *read* after $l_2$ but before any following $r_2$ is on a buffer containing one message. Any *read* after an $r_2$ but before any following $l_2$ and any *read* before the first $l_2$ are on a buffer containing one element. The formulae read-on-full, read-on-single given below characterize *reads* on a full buffer and *reads* on a buffer with one message, respectively.

$$\text{read-on-full} = R \wedge (\sim r_2 \,\mathbf{S}\, l_2)$$

$$\text{read-on-single} = R \wedge [(\sim l_2 \,\mathbf{S}\, r_2) \vee \sim(\text{True} \,\mathbf{S}\, l_2)].$$

For FIFO buffers, a *read* on a full buffer reads the message written by the *write* before the previous *write*.

$$fb_2 = I \wedge bv \wedge g \wedge h,$$

where

$$g = \mathbf{G}\left(\text{read-on-full} \supset \bigwedge_{\sigma} (R_\sigma \supset [\sim W\mathbf{S}\,(W \wedge \mathbf{Y}(\sim W\mathbf{S}\,W_\sigma))])\right),$$

$$h = \mathbf{G}\left(\text{read-on-single} \supset \bigwedge_{\sigma} [R_\sigma \supset (\sim W\mathbf{S}\,W_\sigma)]\right).$$

The formula on the left side of $\supset$ in $g$ is true when *reads* occur on a full buffer, while the formula on the right side asserts that the message read at these instances is the message written by the last but one *write* operation; $h$ asserts that *read* operations on a buffer containing a single message, read the message written by the previous *write* operation.

THEOREM 4.1. *For any $\omega$-sequence of states $t$, $t, 0 \models fb_2$ iff $t \in \mathrm{FIFO}_{\Sigma,2}$.*

Let $t \in \mathrm{LIFO}_{\Sigma,2}$. If $t, i \models r_2$, then there exists $j < i$ such that $t, j \models l_2$. The message read at the instance $i$ is the message written at the instance $j$. If $t, i \models R$ and $t, i \models \sim r_2$, then the message read at the instance $i$ is the message written in the previous write operation. These properties are expressed by $g'$ and $h'$, respectively,

$$g' = \mathbf{G}(r_2 \supset \bigwedge_{\sigma \in \Sigma} [R_\sigma \leftrightarrow \sim l_2\mathbf{S}\,(l_2 \wedge W_\sigma)])$$

$$h' = \mathbf{G}\left((\sim r_2 \wedge R) \supset \bigwedge_{\sigma \in \Sigma} [R_\sigma \leftrightarrow \sim W\mathbf{S}\,W_\sigma]\right).$$

Let $lb_2 = I \wedge bv \wedge g' \wedge h'$.

THEOREM 4.2. *For any $\omega$-sequence of states $t$, $t, 0 \models lb_2$ iff $t \in \mathrm{LIFO}_{\Sigma,2}$.*

Let $t \in \mathrm{UNOR}_{\Sigma,2}$. Then for every $\sigma \in \Sigma$, for all $i \geq 0$ the number of messages of value $\sigma$-written into the buffer up to the instance $i$ is greater than or equal to the number of messages of value $\sigma$ read from the buffer up to the instance $i$, and they do not differ by more than 2. For a given $\sigma$, we can obtain a formula $bv_\sigma$ asserting the above property by replacing $R$ by $R_\sigma$, $W$ by $W_\sigma$ in $bv$. Let $ub_2 = I \wedge bv \wedge \bigwedge_{\sigma \in \Sigma} bv_\sigma$.

The following theorem can be easily proved:

THEOREM 4.3. *For any $\omega$-sequence of states $t$, $t, 0 \models ub_2$ iff $t \in \mathrm{UNOR}_{\Sigma,2}$.*

To characterize bounded buffers with the liveness property for buffers of sizes one and two, we add a conjunct to the formula $bv$ by asserting that the buffer becomes empty infinitely often; this can be done by asserting that whenever $l_2$ holds then there is a future instance where $r_2$ holds, and whenever there is a write operation it is eventually followed by a read operation.

The above approach can be extended to characterize bounded buffers of arbitrary size. However, this approach turns out to be complex and cumbersome. For an account of this the interested reader is referred to [9]. Below we take a different approach to prove that all bounded buffers are characterizable in PTL.

### 4.2. *Star-free Regular Sets and Bounded Buffers*

We assume that the reader is familiar with regular sets. Let $\Delta$ be a finite alphabet. A *star-free* set $U \subseteq \Delta^*$ is inductively defined as follows:

The singleton $\{\delta\}$ is a star-free set over $\Delta$ where $\delta \in \Delta$ or $\delta$ is the empty string. If $V$, $W$ are star-free sets over $\Delta$, then $V \cup W$, $\bar{V}$ (the complement of $V$ relative to $\Delta^*$), and $V \cdot W$ (the concatenation of $V$ and $W$) are star-free sets over $\Delta$. A set is star-free over $\Delta$ only by implication from the preceding clauses.

A regular set $V \subseteq \Delta^*$ is a *non-counting* regular set over $\Delta$ iff there is an integer $l \geqq 0$ such that for all $x$, $y$, $z \in \Delta^*$,

$$xy^lz \in V \qquad \text{iff } xy^{l+1}z \in V.$$

It is proved in [7] that star-free sets are exactly the non-counting regular sets.

Let $FT_{\Sigma,k} = \{s \mid s$ is a prefix of some $t \in \mathrm{FIFO}_{\Sigma,k}$, i.e., $t$ is a model of the FIFO buffer$\}$. Similarly let $LT_{\Sigma,k}$, $UT_{\Sigma,k}$, respectively, be the sets of prefixes of sequences in $\mathrm{LIFO}_{\Sigma,k}$, $\mathrm{UNOR}_{\Sigma,k}$. We first prove that $FT_{\Sigma,k}$, $LT_{\Sigma,k}$, $UT_{\Sigma,k}$ are star-free regular sets by proving that they are non-counting regular sets. Remember that $\mathscr{M}$ is the set of states and is finite.

LEMMA 4.4. *For $k \neq \infty$, $FT_{\Sigma,k}$, $LT_{\Sigma,k}$, $UT_{\Sigma,k}$ are non-counting regular sets over $\mathscr{M}$.*

*Proof.* Clearly all the above sets are regular sets since the corresponding buffers have a finite number of possible states. First we prove that $FT_{\Sigma,k}$ is a non-counting set. We want to prove that for all $x$, $y$, $z \in \mathscr{M}^*$,

$$xy^{k+1}z \in FT_{\Sigma,k} \qquad \text{iff } xy^{k+2}z \in FT_{\Sigma,k}.$$

For any $\sigma \in \Sigma$ if $y$ contains more reads of $\sigma$ than writes of $\sigma$ or vice versa, then neither $xy^{k+1}z$ nor $xy^{k+2}z$ is in $FT_{\Sigma,k}$. Assume $y$ has equal number of reads and writes for each $\sigma \in \Sigma$. Let $xy^{k+1}z \in FT_{\Sigma,k}$, $n$ be the number of messages in the buffer after $x$, and $\alpha$, $\beta$, respectively, be the sequences of messages written and read from the buffer in $y$. Let $|\alpha| = |\beta| = m$. The nontrivial case occurs when $m$, $n > 0$. Let $p = n \bmod m$, $\alpha = \alpha_1\alpha_2$, $\beta = \beta_2\beta_1$, where $|\beta_2| = |\alpha_2| = p$. Clearly $n \leqslant k$. We see that all the messages in the buffer after $xy^k$ are those written in $y^k$ and it is easily seen that after $xy^k$ the buffer contains the sequence $(\alpha_2\alpha^q)$ for some $q \geqq 0$. Since $xy^{k+1}$ is a valid sequence of operations we see that $\alpha_1 = \beta_1$, $\alpha_2 = \beta_2$. Thus the buffer contains the same sequence after $xy^k$ and after $xy^{k+1}$. From this it follows that $xy^{k+2}$ is a valid sequence of operations and the contents of the buffer after $xy^{k+2}$ is same as that after $xy^{k+1}$. Hence $xy^{k+2}z \in FT_{\Sigma,k}$. The same argument proves that if $xy^{k+2}z \in FT_{\Sigma,k}$ then $xy^{k+1}z \in FT_{\Sigma,k}$.

As above, if $y$ contains unequal number of reads and writes for any $\sigma \in \Sigma$, then neither $xy^{k+1}z$ nor $xy^{k+2}z$ is in $LT_{\Sigma,k}$ or in $UT_{\Sigma,k}$. So assume $y$ has equal number of reads and writes for each $\sigma \in \Sigma$. Let $xy^{k+1}z \in LT_{\Sigma,k}$. Since $k \geqq 1$, it is seen that the messages read in $y$ correspond with the messages written in $y$. Hence $xy^{k+2}z \in LT_{\Sigma,k}$. Similarly it is seen that if $xy^{k+2}z \in LT_{\Sigma,k}$ then $xy^{k+1}z \in LT_{\Sigma,k}$. Since $y$ has equal number of reads and writes for every $\sigma \in \Sigma$, it easily follows that $xy^{k+1}z \in UT_{\Sigma,k}$ iff $xy^{k+2}z \in UT_{\Sigma,k}$. ▮

From the results of [7] and the above lemma it follows that $FT_{\Sigma,k}$, $LT_{\Sigma,k}$, $UT_{\Sigma,k}$ are star-free regular sets.

Let $L$ be the first-order languages of $(N, <)$ with monadic predicates, where $N$ is the set of non-negative integers and $<$ is the usual less than relation. This language is shown to be expressively equivalent to PTL in [10]. Hence it is enough if we show that all bounded buffers are characterizable in $L$.

We assume that the predicate symbols in $L$ are same as the propositional symbols in PTL, i.e., the elements of $\mathscr{P}$. Now we want to prove that for every star-free regular set, there is a formula in $L$ which defines the star-free regular set. From this it follows that $FT_{\Sigma,k}$, $LT_{\Sigma,k}$, $UT_{\Sigma,k}$ are definable in $L$. Using this result we will prove that all bounded buffers are characterizable in $L$.

Let $f(x, y)$ be a formula in $L$ with free integer variables $x, y$. Now we associate a language with $f$. Let $M$ be an interpretation of $x, y$ and the predicate symbols, that is, $M$ is a function which associates non-negative integers with $x, y$ and a subset of $N$ with each predicate symbol. Let $m = M(x)$, $n = M(y)$. For any $i \geqslant 0$, let $\phi_i \in \mathscr{M}$ be the unique state such that for all $P_j \in \mathscr{P}$, $\phi_i(P_j) = \text{True}$ iff $i \in M(P_j)$. With $M$ we associate a $t(M) \in \mathscr{M}^*$ defined as follows:

$t(M)$ is undefined if $n < m$,

$t(M)$ is the empty word if $n = m$,

For $m < n$, $t(M) = (\phi_m, \phi_{m+1}, \ldots, \phi_{n-1})$.

Define $\mathscr{L}(f) = \{t(M) \mid M \models f\}$. $\mathscr{L}(f)$ is called the language associated with $f$. It can easily be shown that the successor function given by $y = x + 1$ is definable in $L$.

LEMMA 4.5. *For any star-free regular set $E$ over $\mathscr{M}$ there is a formula $f_E$ in $L$ such that $E = \mathscr{L}(f_E)$.*

*Proof.* By structural induction on $E$:

if $E$ is a singleton say $\{t_0\}$ then

$$f_E = (y = x + 1) \wedge \left(\bigwedge_i P_i\right) \wedge \left(\bigwedge_j \sim P_j\right),$$

where the first conjunct is over all $P_i$ such that $t_0(P_i) = \text{True}$, and the second conjunct is over all $P_j$ such that $t_0(P_j) = \text{False}$;

If $E = V \cdot W$ then

$$f_E = \exists z(f_V(x, z) \wedge f_W(z, y));$$

If $E = V \cup W$ then $f_E = (f_V \vee f_W)$;

If $E = \bar{V}$ then $f_E = \sim (f_V)$.

It can easily be proved by induction that $E = \mathscr{L}(f_E)$. ∎

THEOREM 4.6. *All bounded buffers with or without the liveness property are characterizable in PTL.*

*Proof.* First we prove that all bounded buffers without the liveness property are characterizable in PTL. It is proved in [7] that the non-counting regular sets are exactly the star-free sets. From this and Lemma 4.4 it follows that $FT_{\Sigma,k}$, $LT_{\Sigma,k}$, $UT_{\Sigma,k}$ are star-free sets. Using Lemma 4.5, we see that there is a formula $f(x, y)$ in $L$ such that $\mathscr{L}(f) = FT_{\Sigma,k}$. Let $g(x) = \forall y f(x, y)$, and $M$ be any interpretation of predicate symbols in $L$ and the variable $x$ such that $M(x) = 0$. Let $t_M = (t_0, t_1, ...)$ be the model of PTL such that $t_i(P_j) = \text{True}$ iff $i \in M(P_j)$. It is easily seen that $M \models g$ iff $t_M \in \text{FIFO}_{\Sigma,k}$. This is because $g$ asserts that every finite prefix of $t_M$ is in $FT_{\Sigma,k}$. It is proved in [10] that $L$ and PTL are expressively equivalent. From this it follows that there is a PTL formula, say $h$, such that for all interpretations $M$, such that $M(x) = 0$, $M \models g$ iff $(t_M, 0) \models h$. Hence for any $t \in \mathscr{M}^\omega$, $(t, 0) \models h$ iff $t \in \text{FIFO}_{\Sigma,k}$. Thus $h$ characterizes bounded FIFO buffers of size $k$ over the message alphabet $\Sigma$. Similarly it follows that bounded LIFO and unordered buffers are characterizable in PTL.

It can easily be seen that Lemma 4.4 also holds for bounded buffers with the liveness property. From this it follows that bounded buffers with the liveness property are characterizable in PTL. The details are left to the reader. ▌

In the Appendix we give a constructive method for obtaining a characterization of bounded buffers. This method also uses the translation from $L$ to PTL. However it is known that this translation may cause a non-elementary blow up in the size of the formula. Thus the length of PTL formulae obtained may be non-elementary in the size of the buffer. Obtaining a characterization directly in PTL (i.e., without going through $L$) as given in [9], gives formulae of length $\exp(p(k))$, where $p$ is a fixed polynomial in $k$.

Using first order temporal logic, a uniform characterization of bounded buffers can be given, that is, we can give a formula in first-order temporal logic characterizing a bounded buffer of size $k$, which is independent of the message alphabet $\Sigma$.

### 4.3. *Using Auxilary Propositions in PTL*

Below we show that by introducing auxiliary propositions we can characterize bounded message buffers more elegantly and succinctly. The syntax of the well-formed formulae in this new logic is exactly the same as in PTL, except that some propositions are designated as auxiliary propositions and are not interpreted. Thus let $\mathscr{P} = \mathscr{P}_A \cup \mathscr{P}_1$ be the set of atomic propositions, where $\mathscr{P}_A$ is the set of auxiliary propositions. As usual, an interpretation is a pair $\langle s', i \rangle$, where $s'$ is an $\omega$-sequence of states $(s'_0, s'_1, ...)$, each state being a mapping from $\mathscr{P}_1$ into {True, False}, and $i \geq 0$ designates the present state. We define truth of a formula $f$ in an interpretation $\langle s', i \rangle$ (denoted by $s', i \mathrel{|}\!\approx f$) as follows:

$s', i \mathrel{|}\!\approx f$ iff there exists a sequence $s = (s_0, s_1, ...)$ such that $s, i \models f$, where for all $j \geq 0$, $s_j\colon \mathscr{P} \rightarrow$ {True, False} is an extension of $s'_j$.

In this new logic we can characterize bounded buffers more concisely. We show this for an FIFO buffer of size 2. An FIFO buffer of size 2 can be considered as two FIFO buffers each of size 1 in tandem as shown in Fig. 1.

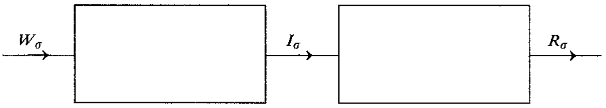

FIGURE 1

External writes come into the left buffer while external reads are from the right buffer. Whenever the left buffer is full and the right buffer is empty the message in the left buffer is internally read and is written into the right buffer. We consider this internal reading and writing to be occuring simultaneously and capture it by the auxiliary propositions $I_\sigma$ for $\sigma \in \Sigma$. Let $fb_1(\mathbf{W}_\sigma, \mathbf{R}_\sigma)$ be the formula characterizing a buffer of size 1, where $\mathbf{W}_\sigma$, $\mathbf{R}_\sigma$ indicate vectors of propositions. The sequence of operations on the left buffer is characterized by $fb_1(\mathbf{W}_\sigma, \mathbf{I}_\sigma)$, and the sequence of operations on the right buffer is characterized by $fb_1(\mathbf{I}_\sigma, \mathbf{R}_\sigma)$. Let

$$fb_2 = fb_1(\mathbf{W}_\sigma, \mathbf{I}_\sigma) \wedge fb_1(\mathbf{I}_\sigma, \mathbf{R}_\sigma),$$

where the propositions in $\mathbf{I}_\sigma$ are the auxiliary propositions.

Lemma 4.7. *$s, 0 \mathrel{|\!\approx} fb_2$ iff $s \in \mathrm{FIFO}_{\Sigma,2}$.*

It is easily seen how we can extend the above approach to characterize bounded FIFO buffers of size $k$. For characterizing LIFO and unordered buffers we take a different approach. We use auxiliary propositions $P_0, P_1, ..., P_k$. We will assert that $P_j$ is true at an instance $i$ iff the buffer has $j$ messages before the operation of the $i$th instance:

$$h' = \mathbf{G}\left[\bigwedge_{0 \leq l < m \leq k} \sim(P_l \wedge P_m) \wedge \bigwedge_{0 \leq l < k} ((P_l \wedge W) \supset \mathbf{X}P_{l+1}) \wedge \right.$$
$$\left. \bigwedge_{0 < l \leq k} ((P_l \wedge R) \supset \mathbf{X}P_{l-1}) \wedge (P_0 \supset \sim R) \wedge (P_k \supset \sim W)\right] \wedge P_0 \wedge g,$$

where $g = \mathbf{G}\,[\sim(R \vee W) \supset (\bigwedge_{0 \leqslant l \leqslant k} (P_l \leftrightarrow \mathbf{X}P_l))]$.

The first clause asserts that no more than one $P_l$ is true at any instance, the second clause asserts that if $P_l$ is true at an instance and the operation is a write operation then at the next instance $P_{l+1}$ is true, the third clause asserts similar property for read operation, the last two clauses assert that there are no writes on a full buffer and no reads on an empty buffer. $g$ asserts that if both $R$ and $W$ do not hold in the present state then the set of all $P_l$ true in the present and next states are the same. Let

$$lb_k = I \wedge h' \wedge \mathbf{G}\left(\bigwedge_{0 < l \leq k} \left(P_l \supset \bigwedge_{\sigma} [R_\sigma \supset (\sim P_{l-1} \mathbf{S}(W_\sigma \wedge P_{l-1}))]\right)\right).$$

The last clause asserts that the message read at any instance when the buffer has $l$ messages is same as the message written at the last instance when the buffer has $l-1$ messages. The following theorem can be easily proved.

Theorem 4.8. *$t, 0 \mathrel{|\!\approx} lb_k$ iff $t \in LS_{\Sigma,k}$.*

Similarly we can obtain a formula for unordered buffers. Note that the lengths of the formulae obtained above are only linear in $k$.

### 4.4. *Characterizing Unbounded Buffers*

Let $\mathscr{P}$ be a finite set of atomic propositions and $s = (s_0, s_1, ...)$ be an infinite sequence of states where each state is a mapping from $s$ into {True, False}. Let $f$ be a formula in PTL and SF($f$) denote the set of subformulae of $f$. It is easily seen that card($SF(f)$)$\leq$length ($f$). For $i \geqq 0$ let $[i]_{s,f} = \{g \in SF(f) \mid s, i \models g\}$.

LEMMA 4.9. *Let* $0 \leqq i \leqq j$ *be such that* $[i]_{s,f} = [j]_{s,f}$. Then $s, 0 \models f$ *iff* $s', 0 \models f$, *where* $s' = (s_0, s_1, ..., s_i, s_{j+1}, s_{j+2}, ...)$.

THEOREM 4.10. *Unbounded message buffers* (*unordered*, FIFO, *or* LIFO) *cannot be characterized in* PTL.

*Proof.* All the above unbounded buffers are identical in the case when the message alphabet is a singleton. We prove that there is no formula in PTL which characterizes an unbounded buffer over a message alphabet which is a singleton. Assume to the contrary $f$ is a such formula. Let $s$ be a model in which $n$ messages are written in the first $n$ states and $n$ messages are read in the next $n$ states. Choose $n > 2^{\text{length}(f)}$. Now it is easily seen that there are two integers $i, j$ such that $0 \leqslant i < j < n$ such that $[i]_{s,f} = [j]_{s,f}$. Now by applying Lemma 4.9 we see that there is a sequence $s'$ which has fewer than $n$ writes followed by $n$ reads such that $(s', 0) \models f$. Clearly $s'$ is not a model of an unbounded buffer. This is a contradiction. ▌

Indeed, we can show that there is no uniform characterization of unbounded buffers in first-order temporal logic. It can easily be seen that there are *partially interpreted* first-order temporal logics in which unbounded buffers can be characterized. For example, we can characterize unbounded FIFO buffers in a first order temporal logic that uses history variables ranging over sequences of messages, and the *prefix* relation ($\leqq$) among the sequences.

## 5. AXIOMATIZING MESSAGE BUFFERS

Axiomatization of message buffers in PTL is a weaker notion than expressiveness. We show that in general unbounded FIFO buffers are not axiomatizable, while unbounded LIFO buffers and unbounded unordered buffers are axiomatizable even though they are not characterizable in PTL.

THEOREM 5.1. *Bounded* FIFO, LIFO, *and unordered buffers over any finite alphabet* $\Sigma$ *are axiomatizable in* PTL.

*Proof.* Let $fb_k$ be the formula in PTL characterizing the FIFO buffer of size $k$ over a finite alphabet $\Sigma$. Consider a consistent and complete axiomatization for PTL as given in [14]. Let $A$ be the set of axioms and $I$ be the set of inference rules in this system. Then the system with $A \cup \{fb_k\}$ as the set of axioms and $I$ as the set of inference rules forms a complete axiomatization for FIFO buffers of size $k$ over $\Sigma$. Similarly, we can give an axiomatization for bounded LIFO and unordered buffers over a finite alphabet. ∎

A finite state automaton $M$ on infinite strings over an alphabet $\Delta$ is a 4-tuple $(Q, \delta, s_0, F)$, where $Q$ is the set of states, $\delta$ is the transition function, i.e., $\delta\colon Q \times \Delta \to 2^Q$, $s_0 \in Q$ is the initial state, $F \subseteq Q$. The states in $F$ are called final states. A run $\gamma$ of $M$ on an input $t \in \Delta^\omega$, is an $\omega$-sequence of states $(\gamma_0, \gamma_1, ...)$ such that $\gamma_0 = s_0$, for all $i \geq 0$, $\gamma_{i+1} \in \delta\ (\gamma_i, t_i)$; $\gamma$ is an accepting run iff some final state apears infinitely often in $\gamma$. An input $t$ is said to be accepted by $M$ iff there is an accepting run of $M$ on $t$. For every PTL formula $f$ there is a finite state automaton $M_f$ over the alphabet $\mathscr{M}$ such that $(t, 0) \models f$ iff $t$ is accepted by $M_f$, where $\mathscr{M}$ is the set of all mappings from $\mathscr{P}$ into {True, False} and $\mathscr{P}$ is the set of all propositions appearing in $f$. Indeed, we can obtain such an automaton with number of states exponential in the length of $f$. A procedure to obtain such an automaton is given by [9]. We will be using this automaton frequently in our proofs.

We assume familiarity with the hierarchy notation of [11]. $\Sigma_1^0$ is the class of recursively enumerable sets and $\Pi_1^0$ is the class of sets which are complements of recursively enumerable sets. The class $\Sigma_1^1$ and its complement $\Pi_1^1$ reside low in the analytical hierarchy [11].

THEOREM 5.2. *For any $\Sigma$ such that* $\mathrm{card}(\Sigma) \geq 2$, *the theory of unbounded* FIFO *buffers over $\Sigma$ is $\Pi_1^1$-complete, the theory of unbounded* FIFO *buffers with liveness property is $\Pi_1^0$-complete, and hence neither of the theories is axiomatizable.*

*Proof.* First we prove below that for $\Sigma = \{0, 1\}$ the theory of unbounded FIFO buffers is $\Pi_1^1$-complete. From this result it automatically follows that the above theory is not axiomatizable.

We first prove that the set of PTL formulae satisfiable over some model of an unbounded FIFO buffer over $\{0, 1\}$ is $\Sigma_1^1$-complete. We consider deterministic Turing machines on infinite strings with one read-only infinite input tape and one work tape. The set of encodings of all TMs on infinite strings which accept at least one input is known to be $\Sigma_1^1$-complete. We will reduce this set to the set of PTL formulae which are true on some model of an unbounded FIFO buffer over $\Sigma$. A Turing machine on infinite strings works exactly like an ordinary Turing machine, but it takes infinite input strings and it never halts. It accepts an input string by going through a final state infinitely often. Let $M = (\Delta, Q, \delta, H)$ be such a Turing machine, where $\Delta$ is the alphabet (including both the input alphabet and the tape alphabet), $Q$ is the set of states, $\delta: Q \times \Delta \times \Delta \rightarrow Q \times \Delta \times \{\text{left, right}\}$, $H$ is the set of final states. After each step the input head of $M$ moves right by one cell. If $\delta(q, \sigma_1, \sigma_2) = (q', \sigma_2', \text{left})$, then whenever $M$ is in state $q$ and sees the symbol $\sigma_1, \sigma_2$ on the input and work tapes, respectively, then $M$ moves into state $q'$, writes $\sigma_2'$ on the work tape and moves its head left, and it moves its input head right by one cell. We show below that given the encoding of $M$ we can recursively obtain a formula $f_M$ in PTL such that $f_M$ is satisfiable on an unbounded FIFO buffer over $\{0, 1\}$ iff $M$ accepts at least one input.

Let $C = (Q \times \Delta) \cup \Delta$. A partial id of $M$, is a sequence of values from $C$, containing exactly one symbol from $(Q \times \Delta)$. A partial id denotes the contents of the work tape and the head position on the work tape and the state of finite control in the usual way. The formula $f_M$ will be satisfied iff there is a $\omega$-sequence of partial ids such that each succeeding partial id is obtained from the previous partial id by one move of $M$ reading some input character, and there are infintely many partial ids in this sequence containing a symbol of the form $(q_f, \sigma)$, where $q_f$ is a final state. We call such a sequence an accepting sequence. Any such sequence denotes an accepting computation of $M$, and for every accepting computation of $M$ there is such a sequence.

We fix a unary encoding of symbols from $C$ using the character $0 \in \Sigma$. An encoding of a partial id is a sequence of encodings of the symbols in it separated by the symbol 1. An encoding of a sequence of partial ids is the sequence of encodings of the partial ids separated by two consecutive 1's. The formula $f_M$ can easily be constructed from the following description.

The initial buffer history consists of a sequence of writes which places the encoding of the initial partial id followed by two consecutive 1's. After writing of the initial id, reading and writing of symbols occurs alternately (thus whenever a symbol is read, it is the symbol of the previous id). Each symbol written into the buffer is the value of the symbol in the new id assuming some input symbol on the input tape ($f_M$ can express this because the value of a symbol in a new id depends only on the contents of that cell and its neighbors in the previous id, and the assumed value of input character). Two consecutive 1's are written at the end of each id.

Finally, $f_M$ also asserts that there are infinitely many places where a symbol of the form $(q_f, \sigma)$ is written into the buffer for some final state $q_f$. It is clearly seen that $f_M$ is satisfiable on a model of an unbounded FIFO buffer over $\{0, 1\}$ iff $M$ accepts at least one input.

Now we give a reduction in the other direction. Let $f$ be a PTL formula and $M_f$ be the finite state automaton on infinite strings corresponding to $f$.

From $M_f$ we give a Turing machine $M$ which operates as follows. $M$ takes each symbol in its input to be an encoding of a function assigning truth values to the set of atomic propositions. $M$ simulates $M_f$ on the input, and at the same time it makes sure that the values of the propositions $R_0$, $R_1$, $W_0$, $W_1$ denote a valid FIFO buffer behavior. $M$ accepts an input iff $M_f$ accepts it and the input sequence denotes a valid FIFO buffer behavior. It is easily seen that $M$ accepts at least one input iff $f$ is satisfiable at the beginning of a model of an unbounded FIFO buffer over $\{0, 1\}$.

It can easily be shown that the set of encodings of TMs on infinite strings that accept at least one input is $\Sigma_1^1$-complete. Hence the set of formulae in PTL that are satisfiable on a model of an unbounded FIFO buffer is $\Sigma_1^1$-complete. From this it follows that the set of formulae, not satisfiable on any model of an unbounded FIFO buffer over $\{0, 1\}$ is $\Pi_1^1$-complete. Hence the set of valid formulae is $\Pi_1^1$-complete.

Now we prove that the set of PTL formulae satisfiable on a model of an FIFO buffer with liveness property is $\Sigma_1^0$-complete. From this it follows that the set of PTL formulae valid on all models of FIFO buffers with liveness property is $\Pi_1^0$-complete.

Given a Turing machine $M$ on *finite* strings, analogous to before we can obtain a PTL formula $f_M$ such that $M$ accepts at least one input iff $f_M$ is satisfiable on a model of an FIFO buffer with liveness property. Now we show that the set of PTL formulae satisfiable on an FIFO buffer with liveness property is in $\Sigma_1^0$. Let $f$ be satisfiable on a model of an FIFO buffer with liveness property and $M_f$ be the automaton associated with $f$. Let $t \in \mathscr{M}^\omega$ be such a model. Let $\gamma = (\gamma_0, \gamma_1, ...)$ be an accepting run of $M_f$ on $t$. There are infinitely many values of $i \geqq 0$, such that the buffer is empty before $t_i$ in $t$. From this it follows that there are two instance $i, j$ such that (1) $i < j$ and the buffer is empty at these instances in $t$, (2) $\gamma_i = \gamma_j$, and (3) there exists a $k$ such that $i \leqq k < j$ and $\gamma_k$ is a final state of $M_f$. Now let $t' = \alpha \cdot \beta^\omega$, where $\alpha = (t_0, ..., t_{i-1})$, $\beta = (t_i, t_{i+1}, ..., t_{j-1})$. It is easily seen that $M_f$ accepts $t'$, and hence $f$ is true at the beginning of $t'$. Now we can easily give a Turing machine $M$ on finite strings which takes a PTL formula $f$ and checks if $f$ is satisfiable on a model of an FIFO buffer with the liveness property. $M$ guesses $\alpha$, $\beta$ and verifies that $M_f$ accepts $t'$ as given above. $M$ halts iff $M_f$ accepts $t'$. The details of $M$ are left to the reader. ∎

THEOREM 5.3. *The theory of unbounded* LIFO *buffers over a finite alphabet is decidable. Similarly the theory of* LIFO *buffers with the liveness property is decidable.*

*Proof.* Let $f$ be a formula in PTL and $M_f$ be the automaton associated with $f$ as defined previously. From $M_f$ we can obtain a push down automaton $P_f$ operating on infinite strings. $P_f$ uses its stack to make sure that the sequence of read/write operations represented by the input string is a legal series of read/write operations on the buffer, while at the same time the finite state control of $P_f$ makes state transitions exactly as $M_f$. $P_f$ accepts an infinite string iff its finite state control goes through any of a set of final states infinitely often. $P_f$ accepts an input $t$ iff $t \in LS_{\Sigma,\infty}$ and $t, 0 \models f$. Thus $f$ is satisfiable on a $t \in LS_{\Sigma,\infty}$ iff $P_f$ accepts some input.

We now need to show that the question of whether $P_f$ accepts some input is decidable. Assume that $t$ is an input accepted by $P_f$. Consider an accepting computation $C$ of $P_f$ on the input. Let $nb(l)$, $r(l)$, respectively, be the number of messages in the stack and the state of the finite state control of $P_f$ just before the $l$th step in $C$. It can easily be seen that there exists an infinite sequence of integers $i_0 < i_1 < i_2 < \cdots < i_j < \cdots$, such that for all $j \geqq 0$, for all $m \geqq i_j$, $nb(m) \geqq nb(i_j)$, that is, from the $i_j$th step onwards the height of the stack never drops below $nb(i_j)$. Let $l$, $m$ be integers such that $l < m$, $r(i_l) = r(i_m) = q$ and there exists a $p$ such that $i_l \leqq p \leqq i_m$ and $r(p)$ is a final state of $P_f$. Clearly $l$, $m$ exist since $P_f$ goes through a final state infinitely often in the computation $C$. Note that $P_f$ reads one input symbol on each step. Let $u = i_l$, $v = i_m$ and $\alpha = (t_0, t_1, ..., t_{u-1})$, $\beta = (t_u, t_{u+1}, ..., t_{v-1})$. We can also consider $P_f$ as a push down automaton on finite strings.

*Claim.* $P_f$ accepts some infinite string iff there exists a state $q$ such that

(a) there is a finite string $\alpha$ and there is a computation of $P_f$ on input $\alpha$ reaching state $q$; and

(b) there is a finite string $\beta$ and there is a computation of $P_f$ starting with initial state $q$ and empty stack and reaching $q$ at the end but going through a final state during this computation.

*Proof of claim.* Assume $P_f$ accepts an infinite input string. Then $q$, $\alpha$, $\beta$ as defined before satisfy (a) and (b). Assume (a) and (b) hold. Then it is easily seen that $P_f$ accepts the infinite input $\alpha \cdot \beta^\omega$. ∎

Now it is easily seen how to decide if $P_f$ accepts at least one infinite input. That is check if there is a state $q$ for which (a) and (b) of the above claim hold: (a) is the problem of deciding if a push down automaton on finite strings accepts at least one input which is known to be decidable; (b) can also be posed as such a problem and so is decidable. The above argument also shows that the theory of unbounded LIFO buffers with liveness property is decidable. ∎

We will show that the problem of deciding if a given PTL formula is satisfiable on a model of an unbounded unordered buffer is reducible to certain problems in vector addition systems. A *vector addition system with states of dimension* $k$ is a triple $G = (V, E, L)$, where $(V, E)$ is a directed graph and $L: E \rightarrow N^k$, where $N$ is the set of integers. A configuration is a pair $(s, a)$, where $s \in V$, $a \in N^k$. A path is a finite or infinite sequence of length $\geqq 2$ of configurations $c_0, c_1, ..., c_i$, where $c_i = (s_i, a_i)$ and for all $i \geqq 0$ $(s_i, s_{i+1}) \in E$, $a_{i+1} = a_i + L(s_i, s_{i+1})$. Let $A \subseteq \{1, ..., k\}$. A path is said to be positive with respect to $A$ if for all $i \geqq 0$, for all $j \in A$ $(a_i)_j \geqq 0$. We say that a path is positive if it is positive with respect to $\{1, ..., k\}$. We say that a configuration $d$ is positively reachable with respect to $A$ from the configuration $c$ if there is a positive path with respect to $A$ with initial configuration $c$ and final configuration $d$; and if $A = \{1, ..., k\}$ then we simply say that $d$ is positively reachable from $c$.

Let $f$ be a PTL formula and $M_f$ be the automaton associated with $f$. Let $G_f = (V, E, L)$ be the vector addition system associated with $f$ defined as follows: $V$ is the set of states of $M_f$ and $(s_1, s_2) \in E$ iff there is an $\alpha$ ($\alpha$ is a function that assigns truth values to propositions) such that there is transition in $M_f$ from $s_1$ to $s_2$ on input $\alpha$, and $L(s_1, s_2) = (a_1, a_2, ..., a_k)$, where for all $i$, $1 \leqq i \leqq k$,

$$\begin{aligned} a_i &= \phantom{-}1 && \text{if } \alpha(W_{\sigma_i}) = \text{True}, \\ a_i &= -1 && \text{if } \alpha(R_{\sigma_i}) = \text{True}, \\ a_i &= \phantom{-}0 && \text{otherwise.} \end{aligned}$$

It is easily seen that $f$ is satisfiable on a model of an unordered buffer iff there exists an infinite positive path in $G_f$ with initial configuration $(q_I, \bar{0})$ and containing infinitely many configurations of the form $(q_f, a)$, where $q_I$ is the initial state and $q_f$ is a final state in $M_f$.

Let $M = N \cup \{\omega\}$ and $<$ be the extension of "less than" relation to elements in $M$ so that for each $i \in N$, $i < \omega$. Also we extend the usual $+$ operation so that for any $i \in M$, $i + \omega = \omega$. Let $a$, $b \in M^k$. Then $a \leqq b$ iff $a_i \leqq b_i$ for each $i$ such that $1 \leqq i \leqq k$. Let $c$ be any configuration. We define a labelled tree $T_c$ as in [6]. Each node $x$ in $T_c$ is labelled with $l(x) \in M^k$ and is recursively defined as follows:

The root $r$ of $T_c$ is labelled with $c$, that is, $l(r) = c$. Let $y$ be a node in $T_c$ with $l(y) = (s, a)$:

(a) If there is a proper ancestor $x$ of $y$ in $T_c$ such that $l(y) = l(x)$ then $y$ is a leaf, that is, $y$ does not have any children; otherwise

(b) For each $(s, s') \in E$ such that $L(s, s') + a = \gamma \geqq \bar{0}$, there is a son $z$ with $l(z) = (s', b)$, where $b_i$ is given as follows:

If there is an ancestor $x$ of $y$ such that $l(x) = (s', d)$, where $d \leqq \gamma$ and $d_i < \gamma_i$ then $b_i = \omega$, otherwise $b_i = \gamma_i$.

In [6] it is proved that $T_c$ is finite.

LEMMA 5.4. *The following are equivalent*:

(a) *There is an infinite positive path in G with initial configuration c* *and containing infinitely many configurations of the form $(q, d)$, where $q$ is a fixed state in $G$.*

(b) *There is a node in $T_c$ with label $(q, a)$ satisfying the following property*:

*Let $A$ be the set of all $i$ such that $a_i \neq \omega$ and $a' \in N^k$ be such that for all $i \in A$, $a_i' = a_i$ and for all $i$ not in $A$, $a_i' = 0$. Then there exists $b \in N^k$ such that $a' \leqq b$ and $(q, b)$ is positively reachable with respect to $A$ from $(q, a')$.*

*Proof.* (b $\Rightarrow$ a) Let $q$, $a$, $b$, $A$, $a'$ be as given in (b). Let $e_0, e_1, \ldots, e_p$ be a positive path with respect to $A$, with $e_0 = (q, a')$, $e_p = (q, b)$. Let $e_i = (s_i, d_i)$, and $-(\Delta)$ be the lower bound for all $(d_i)_j$ for $0 \leqq i \leqq p$ and for all $j$ not in $A$. Since $(q, a)$ is a node in $T_c$ with $a_j = \omega$ for all $j$ not in $A$, it can easily be shown that there is a configuration $(q, f)$ such that $f_j = a_j$ for all $j \in A$, and $f_j \geqq \Delta$ for all $j$ not in $A$, and $(q, f)$ is positively reachable from $c$, i.e., there is a positive path $t$ with $c$, $(q, f)$ being the initial and final configurations, respectively. Let $g$ be the sequence of states corresponding to $t$, and $h$ be the sequence of states corresponding to $e_1, e_2, \ldots, e_p$. Now consider the infinite sequence of states $g \cdot h^{\omega}$. It is clearly seen that this sequence gives us an infinite positive path satisfying the property given in (a).

(a $\Rightarrow$ b) Let $C = c_0, c_1, \ldots,$ be an infinite positive path satisfying the property given in (a). Let $c_i = (s_i, a_i)$ for all $i \geqq 0$. We define an infinite sequence $d_0 d_1 \ldots d_i \ldots,$ as follows:

For all $i \geqslant 0$ $d_i = (s_i, b_i)$, where $b_i \in M^k$ is defined as follows: $b_0 = a_0$. For $i > 0$ we inductively define $b_i$, $\gamma_i \in M^k$ as follows:

For any $j$, $(\gamma_i)_j = \omega$ if $(b_{i-1})_j = \omega$, otherwise $(\gamma_i)_j = (a_i)_j$. For all $j$, $1 \leqq j \leqq k$, $(b_i)_j = \omega$ if there is an $l < i$ with $s_l = s_i$, $b_l \leqq \gamma_i$, and $(b_l)_j < (\gamma_i)_j$; otherwise $(b_i)_j = (\gamma_i)_j$.

It is clearly seen that for each $i \geqq 0$, $d_i$ is the label of a node in $T_c$. Now the above sequence consists of an infinite subsequence $d_{j_0} d_{j_1} \ldots,$ such that for all $l \geqq 0$, $s_{j_l} = q$. From this subsequence pick up an infinite subsequence with non-decreasing first coordinates of $b_i$. By repeatedly doing this for all coordinates, it can easily be shown that there is an infinite subsequence $d_{i_0}$, $d_{i_1}, \ldots,$ such that for all $l \geqq 0$, $s_{i_l} = q$, $b_{i_l} \leqq b_{i_{(l+1)}}$. But from the way we defined $d_i$, it has to be the case that there is an $n$, with the property that for all $l \geqq n$, $b_{i_l} = b_{i_{l+1}}$. Since $C$ is a positive path there exists $l$, $m$ such that $l < m$ and $a_{i_l} \leqq a_{i_m}$. Let $A$ be the set of all $j$ such that the $j$th coordinate of $b_{i_l}$ is not $\omega$. Clearly for all $j \in A$, the $j$th coordinate of $a_{i_l}$, $a_{i_m}$, $b_{i_l}$, $b_{i_m}$ are all equal.

Let $\alpha = b_{i_l}$ and $\beta$, $\gamma \in N^k$ be such that $\beta_j = \gamma_j = \alpha_j$ for all $j \in A$, and for all $j$ not in $A$, $\beta_j = 0$, $\gamma_j =$ ($j$th coordinate of $a_{i_m}$ $-$ $j$th coordinate of $a_{i_l}$). Now it is clearly seen that the configuration $(q, \gamma)$ is positively reachable with respect to $A$ from the configuration $(q, \beta)$. Also it is clear that there is a node in $T_c$ with label $(q, \alpha)$. Hence (b) is true. ∎

Lemma 5.5 follows directly from the results of [6].

LEMMA 5.5. *Given a configuration $c=(q, a)$ it is decidable if there is a configuration $d=(q, b)$ with $a \leqq b$ such that $d$ is positively reachable from $c$ in vector addition system $G$.*

The proof of the above lemma can easily be extended to the following lemma.

LEMMA 5.6. *Given $c=(q, a)$ and a set $A \subseteq \{1, 2, \ldots, k\}$ it is decidable if there exists a configuration $d=(q, b)$ such that $a \leqq b$ and $d$ is positively reachable with respect to $A$ from $c$.* ▌

THEOREM 5.7. *The theory of unbounded unordered buffers over any alphabet $\Sigma$ is decidable.*

*Proof.* It is enough if we show that the satisfiability problem is decidable. Let $f$ be a PTL formula and $M_f$ be the automaton associated with $f$, and $G_f$ be the vector addition system with states of dimension $k$ associated with $f$. Let $c=(s_0, \bar{0})$, where $s_0$ is the initial state in $M_f$. Then $f$ is satisfiable iff there exists an infinite positive path in $G_f$ with initial configuration $c$ and containing infinitely many configurations of the form $(q, d)$, where $q$ is a final state in $M_f$. Now to check this condition we use Lemma 5.4. We construct $T_c$ and for each pair $(q, a)$, where $q$ is a final state and some node in $T_c$ is labelled with this pair, we do the following:

Let $A$ be the set of all $i$ such that $a_i \neq \omega$ and $a' \in N^k$ be such that for all $i \in A$, $a'_i = a_i$, and for all $i$ not in $A$, $a'_i = 0$. Verify if there is some $b \geqq a'$ such that $(q, b)$ is positively reachable with respect to $A$ from $(q, a')$. This can be done due to Lemma 5.6.

Now from Lemma 5.4, $f$ is satisfiable on a model of an unbounded unordered buffer iff there exists a pair $(q, a)$ which satisfies the above condition. Thus satisfiability is decidable and hence the theorem follows. ▌

The following problem is known as the *reachability problem* for vector addition systems with states: Given a vector addition system $G$ with states, and two configurations $a$, $b$ of $G$, is $b$ positively reachable from $a$ in $G$?

THEOREM 5.8. *The theory of unbounded unordered buffers with the liveness property over a finite alphabet $\Sigma$ is axiomatizable (and also decidable) iff the reachability problem for vector addition systems with states of dimension card($\Sigma$) is decidable.*

*Proof.* First we want to show that the set of formulae satisfiable on a model of an unbounded unordered buffer with the liveness property is recursively enumerable. To show this it is enough if we give a partial decision procedure to check if $f$ is satisfiable on such a model. Let $t$ be such a model at the beginning of which $f$ is true, and let $M_f$ be the automaton associated with $f$. Since the buffer becomes empty infinitely often in $t$, it easily follows that there is a model of an unbounded unordered buffer of the form $\alpha \cdot \beta^{\omega}$, where $\alpha$, $\beta$ are finite strings such that the buffer is empty after $\alpha \cdot \beta^i$ for all $i \geqslant 0$ and $f$ is true at the beginning of the above model. Our partial decision procedure guesses $\alpha$, $\beta$ and verifies that they have the above properties. It is easily seen that we can easily verify if $\alpha \cdot \beta^{\omega}$ is accepted by $M_f$.

Now it easily follows that the set of valid PTL formulae in all models of unbounded unordered buffers with the liveness property is axiomatizable iff the set of satisfiable formulae over such models is decidable. We show that the set of satisfiable formulae is decidable iff reachability problem for vector addition systems with states is decidable.

First we reduce reachability problem to satisfiability problem. Let $G = (V, E, L)$ be a vector addition system with states of dimension $k$ and it is required to determine if $(t, b)$ is positively reachable from $(s, a)$. Let $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_k\}$. We give a formula $f$ such that there is a history $h \in \text{UNOR}_{\Sigma,\infty}$, in which the buffer becomes empty infinitely often and such that $(h, 0) \models f$ iff $(t, b)$ is positively reachable from $(s, a)$ in $G$. We use a proposition $P_u$ for each $u \in V$. The formula $f$ asserts the following:

(i) For each $i$, $1 \leqq i \leqq k$, initially $a_i$ messages of value $\sigma_i$ are written into the buffer; immediately after this $P_s$ is true.

(ii) The propositions $P_u$ (for $u \in V$) are mutually exclusive. For $u \neq t$ if $P_u$ is true at any instance $i$ then the next proposition to be true in future at instance $j$ will be $P_v$, where $(u, v) \in E$, and if $(c_1, c_2, ..., c_k) = L(u, v)$ then between $i$ and $j$, for all $l$ such that $1 \leqq l \leqq k$ if $c_l$ is positive (negative) then $|c_l|$ number of messages of value $\sigma_l$ are written into (read from) the buffer.

(iii) If $P_t$ is true at any instance, either (ii) holds or the following condition is satisfied. Immediately after $P_t$ is true, for all $l$ such that $1 \leqq l \leqq k$, $b_l$ messages of value $\sigma_l$ are read from the buffer, and after this all propositions are false forever.

(iv) There is a future instance from which point all propositions will be false forever.

It is easily seen how to obtain $f$. Thus if satisfiability problem is decidable then reachability problem is also decidable.

Now assume that the reachability problem is decidable. Let $f$ be a PTL formula, and $M_f$, $G_f$ be as defined before where $G_f = (V, E, L)$. Let $q_I$ be the initial state in $M_f$. The following is easily seen:

There is a $t \in \text{UNOR}_{\Sigma,\infty}$ such that the buffer becomes empty infinitely often in $t$ and $(t, 0) \models f$ iff there is a $q \in V$ and a $q_f \in V$ which is a final state satisfying

(i) $(q, \bar{0})$ is reachable from $(q_I, \bar{0})$ in $G_f$ and,

(ii) $(q, \bar{0})$ is reachable from $(q, \bar{0})$ by passing through $q_f$.

(ii) is not a direct reachability problem; however, we can put it as a reachability problem as follows: Introduce another copy of $G_f$, call it $G'_f$, and introduce a transition from $q_f$ in $G$ to $q'_f$ in $G'_f$, which is labelled with $\bar{0}$. Now (ii) is satisfied in $G_f$ iff $(q', \bar{0})$ is reachable from $(q, \bar{0})$ in the new vector addition system.

Since we assumed reachability is decidable, we can easily decide if there is a $q$ satisfying (i) and (ii). ▌

## 6. Conclusion

We have examined the possibility of using linear temporal logic to express the semantics of different message buffering systems. We have shown that it is possible to characterize bounded message buffers in PTL and that no such characterization is possible for unbounded buffers. We have also considered the possibility of axiomatization of the theory of various message buffer systems. We have shown that unbounded FIFO buffers are not axiomatizable in PTL. However it may be possible to give an axiomatization of these buffers in a logic weaker than PTL. This is still an open problem. We have also shown that unbounded LIFO and unordered buffers are axiomatizable in PTL. However to give a complete axiom system for these buffers in PTL is still an open problem. Some of these problems will be addressed in a future paper.

## Appendix

Below we give a constructive method for characterizing bounded buffers.

A *Semi-automaton* $A$ is a triple $(Q_A, \Delta_A, M_A)$ where $Q_A$ is a finite set of states, $\Delta_A$ is a finite alphabet, and $M_A\colon Q_A \times \Delta_A \to Q_A$. Let $A = (Q_A, \Delta_A, M_A)$ and $B = (Q_B, \Delta_B, M_B)$, and $\delta\colon Q_A \times \Delta_A \to \Delta_B$. The *cascade product* of $A$, $B$ with mapping $\delta$ is the semi-automaton $C = (Q_c, \Delta_c, M_c)$, where $Q_c = Q_A \times Q_B$, $\Delta_c = \Delta_A$ and for all $p \in Q_A$, $q \in Q_B$, $\sigma \in \Delta_c$, $M_c((p, q), \sigma) = (p', q')$, where $p' = M_A(p, \sigma)$, $q' = M_B(q, \delta(p, \sigma))$. The cascade product of three or more automatons is defined by association to the left.

A *reset* is a semi-automaton $A = (Q, \Delta, M)$ where $Q = \{0, 1\}$, $\Delta$ is a disjoint union of 3 sets $\Delta_0, \Delta_1, \Delta_I$ such that for all $p \in Q$, $\sigma \in \Delta_0$, $M(p, \sigma) = 0$, for all $\sigma \in \Delta_1$, $M(p, \sigma) = 1$, and for all $\sigma \in \Delta_I$, $M(p, \sigma) = p$.

Let $A = (Q, \Delta, M)$ be a semi-automaton and let $M$ be extended in a natural way to the domain $\Delta^*$. Now let $A_{p,q} = \{\sigma \in \Delta^* \mid M(p, \sigma) = q\}$. It is proved in ([7, 8]) that if $A$ is a cascade product of resets then $A_{p,q}$ is a star-free regular set. Indeed in [8] a constructive method is given to generate such a star-free regular set. We give a construction below to simulate bounded buffers by a cascade product of resets.

THEOREM. *Bounded* FIFO, LIFO, *and unordered buffers can be simulated by a cascade product of resets.*

*Proof.* Assume the message alphabet $\Sigma = \{0, 1\}$. It is easy to see how our construction can be extended to any finite $\Sigma$. Let $k$ be the size of the buffer. We informally describe the construction. We use a cascade product of $2k$ resets. The number of resets in state 1 among the first $k$ resets gives the number of messages in the buffer at any instance. The last $k$ resets contain the contents of the buffer right justified. Number the resets left to right starting from 1. If the number of messages in the buffer is $m$, then the resets $2k - m + 1$ through $2k$ contain the buffer contents, the latest being contained in $(2k - m + 1)$th reset and the oldest in $2k$th reset.

Whenever a write operation occurs, then the first reset in state 0 among the first $k$ resets is changed to state 1. Whenever a read operation occurs then the first reset in state 1 among the first $k$ resets is changed to state 0. Overflow and underflow can be easily detected.

In addition to the above update, a write operation places the new message in $(2k - m)$th reset if there are $m$ messages already in the buffer.

In case of FIFO buffers, a read operation transfers the state of $i$th reset to $(i + 1)$th reset for all $i$ such that $2k - m + 1 \leqq i < 2k$. In case of LIFO buffers decrementing of the counter in the first $k$ resets is enough. In case of unordered buffers, a read operation on a message $\sigma$ transfers the state of $i$th reset to $(i + 1)$th reset for all $i$ such that $2k - m + 1 \leqq i < j$, where $j$ is the first reset with state $\sigma$. The detailed designs of the resets are left to the reader. Finally, we can use another reset and handle error conditions. In this case if there is an error operation on the buffer then the $(2k + 1)$th reset goes into state 1 indicating an error state. ▌

Let $A$ be the cascade product of resets as constructed in the above theorem. Let $\bar{0}$ be the state of $A$ with all resets being in state 0. Let $B =$ union of all $A_{p,q}$, where $q$ is a non-error state of $A$ and $p = \bar{0}$. Using the construction given in [8] we can obtain a star-free regular event corresponding to $B$, and using the translation given in Lemma 4.5 we can obtain a characterization of bounded buffers in the language $L$. From this and using the translation given in [10] we can obtain a characterization of bounded buffers in PTL.

## ACKNOWLEDGMENT

The authors acknowledge the help of Professor Yuri Gurevich on an early version of Theorem 4.10.

RECEIVED: February 17, 1984; ACCEPTED: January 15, 1985

## REFERENCES

1. SISTLA, A. P., CLARKE, E. M., FRANCEZ, N., AND GUREVICH, Y. (1982), Can message buffers be characterized in linear temporal logic? *in* "Proc. ACM Sympos. on Principles of Distributed Systems," Ottawa, Canada, August.
2. MANNA, Z., AND PNUELI, A. (1981), Verification of concurrent programs, *in* "The Correctness Problem in Computer Science," International Lecture Series in Computer Science, London.
3. OWICKI, S., AND LAMPORT. L. (1982), Proving liveness properties of concurrent programs, TOPLAS **4**, No. 3.
4. WOLPER, P. (1983), Temporal logic can be more expressive, *in* "Proc. 22nd IEEE Sympos. on Foundations of Computer Science, Nashville, Tenn. October, 1981; *Inform. and Control* **56**, 72–99.
5. KOSARAJU, S. R. (1982), Decidability of reachability in vector addition systems, *in* "Proc. 14th Annu. ACM Sympos. on Theory of Computing," May.
6. KARP, R. M., AND MILLER, R. E. (1969), Parallel program schemata, *J. Comput. System Sci.* **3**, No. 2, 147–195.
7. PAPERT, S., AND MCNAUGHTON, R. (1966), On topological events, *in* "Theory of Automata," University of Michigan Engineering Summer Conferences.
8. MEYER, A. R. (1969), A note on starfree events, *J. Assoc. Comput. Mach.* **16**, No. 2.
9. SISTLA, A. P., "Theoretical Issues in the Design and Verification of Distributed Systems, Ph.D. thesis, Harvard; Tech. Report CMU-CS-83-146, Carnegie–Mellon University.
10. GABBAY, D., PNUELI, A., SHEALAH, S., AND STAVI, J. (1969), Temporal analysis of fairness, "Seventh ACM Symposium on Principles of Programming Languages," Las Vegas, December.
11. ROGERS, H., (1967), "Theory of Recursive Functions and Effective Computability," McGraw–Hill, New York.
12. HAREL, D. (1984), Recurring dominoes: Making highly undecidable highly understandable, *Ann. Discrete Math.*
13. SISTLA, A. P., AND CLARKE, E. M. (1982), Complexity of propositional linear temporal logics, *in* "ACM Sympos. on Theory of Computing," San Francisco.
14. KAMP, H. (1968), "Tense Logic and Theory of Linear Order," Ph.D. Thesis, UCLA.