CAN MESSAGE BUFFERS BE CHARACTERIZED IN LINEAR TEMPORAL LOGIC?

A. P. Sistla
E. M. Clarke
Harvard University
Cambridge, Massachusetts

N. Francez
The Technion
Haifa, Israel

Y. Gurevich
University of Michigan
Ann Arbor, Michigan

I. 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 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 the semantics of 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 [MP81] and Owicki and Lamport [OL80] 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 time. Moreover, important correctness properties such as mutual exclusion, deadlock freedom, and absence of starvation can be elegantly expressed in this system.

This research was supported by NSF Grant MCS79-08365.

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 as the set of all valid infinite input/output message sequences. Thus, characterizing a message buffer in temporal logic consists of obtaining a formula that is true exactly on these sequences. For *unbounded* buffers, we show that it is *impossible* to obtain such a formula in first order linear temporal logic that is *independent* of the underlying interpretation (i.e. message alphabet). Nor is it possible to obtain such a formula in propositional linear temporal logic (PTL) when the message alphabet is finite. It is possible, however, to give a formula in first order linear temporal logic that gives a domain-independent characterization of bounded buffers. In fact, if the message alphabet is finite, then such a formula can be expressed in PTL. Although such bounded message buffers can be characterized using ω-regular expressions (or monadic second order theory of one successor), it is not obvious that they can be expressed in PTL since this logic is provably less expressive than ω-regular expressions [WO81].

Since the formula we obtain 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 formulae in the extended logic which characterize message buffers of fixed size over a finite alphabet.

We also consider the problem of axiomatizing the various types of message buffers described above. 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. Since bounded buffers over finite alphabet can be characterized in PTL and since PTL has a complete axiom system it can easily be shown that bounded buffers are axiomatizable in PTL. We show that, in general, unbounded FIFO buffers are not axiomatizable. Surprisingly, it is possible to axiomatize unbounded LIFO buffers and unbounded unordered buffers; in fact, the theories of these buffers are decidable.

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. In Section 5 we prove that it is impossible to give a domain independent characterization of unbounded message buffers in first order temporal logic. We also 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

A *well-formed formula* in PTL is either an atomic proposition or is on the form $\neg f_1$, $f_1 \wedge f_2$, Xf_1, $f_1 \,U\, f_2$, Yf_1, $f_1 \,S\, f_2$ where f_1, f_2 are well-formed formulae. In addition, the following abbreviations will be used:

$$f_1 \vee f_2 \equiv \neg(\neg f_1 \wedge \neg f_2), \quad f_1 \supset f_2 \equiv \neg f_1 \vee f_2,$$
$$Ff \equiv \text{True } U\, f, \quad Gf \equiv \neg F \neg f.$$

A *state* is a mapping from the set of atomic propositions into the set {True, False}. An *interpretation* is an ordered pair <t,i> where t is an infinite ω-sequence of states and $i \geq 0$ 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 $t,i \models f_2$;
$t,i \models \neg f_1$	iff $t,i \not\models f_1$;
$t,i \models Xf_1$	iff $t,i+1 \models f_1$;
$t,i \models f_1 \,U\, f_2$	iff $\exists k \geq i$ such that $t,k \models f_2$ and $\forall j$ such that $i \leq j < k$, $t,j \models f_1$;
$t,i \models Yf_1$	iff $i > 0$ and $t,i-1 \models f_1$;
$t,i \models f_1 \,S\, f_2$	iff $\exists k \leq i$ $S,k \models f_2$ and $\forall j$ such that $k < j \leq i$ $t,j \models f_1$;

X,U,Y,S are the "next-time", "until", "last-time", and "since" operators respectively.

We will also consider a restricted version of the first order temporal logic. The language of this logic includes variables, function symbols, relation symbols and the universal quantifier in addition to the symbols in the propositional version of the logic. The *type* of the language is a tuple which gives the function symbols, the relation symbols with their arities. The variables are partitioned into two groups: local variables whose values depend on the current state and global variables whose values are state independent. Atomic formulae have the same syntax as in the usual first order case. The set of well formed formulae is the smallest set containing the atomic formulae and closed under universal quantification over global variables, boolean connectives, and the above temporal operators.

A *model* T is a triple (Δ,α,s) where Δ is the *domain*; α assigns meanings to the function symbols, relation symbols, and global variables; and s is a ω-*sequence* of states. A *state* assigns values from Δ to the local variables and truth values to the atomic propositions. An interpretation in this case is a pair <T,i> where T is a model and $i \geq 0$ specifies the present state. Truth of an atomic formula in an interpretation is defined as in the usual first order case; truth of a composite formula is defined as in the case of propositional temporal logic with following addition: $T,i \models \forall x f$ iff for each $c \in \Delta$ $T_c,i \models f$ where T_c is T with the meaning assigned to the global variable x changed to the value c.

3. What Are Message Buffers?

We characterize a message buffer by the set of legal *read/write* sequences allowed on the buffer. 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 in the buffer is the output for a read request; with LIFO buffers the latest written message in the buffer is used; and with unordered buffers any message present in the buffer is output. We also require that each physical message written into the buffer is ultimately read; this is the *liveness property of buffer behavior*.

Let Σ be the message alphabet and $\mathscr{P}_\Sigma$ be the set of atomic propositions $\{R_\sigma \mid \sigma \in \Sigma\} \cup \{W_\sigma \mid \sigma \in \Sigma\}$. Let $\mathscr{P} \supseteq \mathscr{P}_\Sigma$ be the set of atomic propositions in the language.

$$ST = \{\phi \mid \phi: \mathscr{P} \rightarrow \{\text{True},\text{False}\} \text{ such that } \phi(P) = \text{True for at most one } P \text{ in } \mathscr{P}_\Sigma\}$$

We consider each member of ST to be a state; if $R_\sigma(W_\sigma)$ is true in a state, then it indicates that the message σ is read (written) from (into) the buffer in that state.

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 $k \geq 0$. Then $\pi_r(t)$ denotes the sequence $(\sigma_0,\sigma_1,\ldots)$. Similarly, we define $\pi_w(t)$. Let $t^{(i)}$ denote the sequence $(t_0,t_1,\ldots,t_i)$, then $nb(i) = \text{length}(\pi_w(t^{(i)})) - \text{length}(\pi_r(t^{(i)}))$ is the number of messages in the buffer at the instance i.

$FS_{\Sigma,k}$ is the set of all infinite sequences of states which denote legal series of read/write operations on a FIFO buffer of size k. $LS_{\Sigma,k}$ and $US_{\Sigma,k}$ are 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 $k \geq 0$ and $k = \infty$

$$FS_{\Sigma,k} = \{t \in ST^\omega \mid \text{for all } i \geq 0 \quad 0 \leq nb(i) \leq k \text{ and } \pi_r(t^{(i)}) \text{ is a prefix of } \pi_w(t^{(i)}) \text{ and } \pi_r(t) = \pi_w(t)\}.$$

$$LS_{\Sigma,k} = \{t \in ST^\omega \mid \text{for all } i \geq 0 \quad 0 \leq nb(i) \leq k \text{ and if for some } \sigma \in \Sigma,\ t,i \models W_\sigma \text{ then there exists } j > i \text{ such that } t,j \models R_\sigma,\ nb(j-1) = nb(i) \text{ and } \forall \ell\ i \leq \ell \leq j-1\ nb(\ell) \geq nb(i)\}$$

$$US_{\Sigma,k} = \{t \in ST^\omega \mid \text{for all } i \geq 0 \quad 0 \leq nb(i) \leq k \text{ and for all } \sigma \in \Sigma, \text{ the number of writes of the message } \sigma \text{ upto } i \geq \text{ the number of reads of the message } \sigma \text{ upto } i, \text{ and for infinitely many } i,\ nb(i) = 0\}$$

In the case of both LIFO and unordered buffers we require that the buffer should become empty infinitely often, in order to satisfy the liveness requirement.

For a finite alphabet Σ, a formula f in PTL *characterizes* a FIFO message buffer of size k (unbounded FIFO buffer) if

$$\forall t \in ST^\omega \quad t,0 \models f \text{ iff } t \in FS_{\Sigma,k} \quad (t \in FS_{\Sigma,\infty}).$$

Similarly we define what it means to characterize LIFO and unordered buffers in PTL.

Let L be a language of first order linear temporal logic of type τ with local variables read_val, write_val and with atomic propositions R,W. Let $T = (\Sigma,\alpha,s)$ be a model of type τ. In any state s_i, if $s_i(R) = \text{True}$ then it signifies the reading of message s_i(read_val) in that state, and if $s_i(W) = \text{True}$ then it signifies the writing of message s_i(write_val) in that state. With s, we associate any sequence t defined as follows:

For all $i \geq 0$ $\quad t_i: \mathscr{P} \to \{\text{True, False}\}$ such that

for all $\sigma \in \Sigma$,

$t_i(R_\sigma)$ = True iff $s_i(R)$ = True and $s_i(\text{read\_val}) = \sigma$;

$t_i(W_\sigma)$ = True iff $s_i(W)$ = True and $s_i(\text{write\_val}) = \sigma$.

A first order linear temporal formula f of type τ is a *(domain independent) characterization* of a FIFO buffer of size k (unbounded FIFO buffer) if for all $T = (\Sigma,\alpha,s)$ of type τ $T,0 \models f$ iff $t \in FS_{\Sigma,k}$ ($t \in FS_{\Sigma,\infty}$). Similar definitions hold for LIFO and unordered buffers.

A model of a message buffer is an infinite sequence of states denoting a legal series of read/write operations on the buffer, as given above. The theory of a message buffer is the set of all PTL formulae which are true in all interpretations (t,i) where t is a model of the buffer. We say that a message buffer is axiomatizable if there exists a recursive set of axioms from which the formulae in the theory of the buffer can be deduced using some inference rules.

4. Characterizing Bounded Buffers

In this section we characterize bounded buffers over a finite alphabet using propositional linear temporal logic; we also give domain independent characterizations in first order linear temporal logic. We let fb_k, ℓb_k, ub_k denote formulae in propositional temporal logic characterizing FIFO, LIFO, and unordered message buffers of size k over the finite message alphabet Σ. Here we give formulae for buffer size = 1 and 2; in the full paper we show how to obtain these formulae for an arbitrary buffer size k.

Let Σ 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:

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

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

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

$$I = G(Ex)$$

'I' asserts that at any instant at most one operation occurs on the buffer, and *reads*, *writes* are mutually exclusive.

In the case of buffer size = 1 the buffer behavior is as follows:

(a) The *writes* and *reads* occur alternately;

(b) The message read in each *read* operation is the message written by the previous *write* operation. Thus, $f_{b_1} = I \wedge f_a \wedge f_b$ where

$$f_a = G(W \supset X(\neg W \;U\; R)) \wedge G((R \wedge X(F\; R)) \supset X(\neg R \;U\; W));$$

$$f_b = G(\bigwedge_{\sigma\in\Sigma} (R_\sigma \supset (\neg W \;S\; W_\sigma))).$$

It is easily seen that f_a and f_b assert properties (a) and (b), respectively.

Intuitively, the operation of a buffer of size = 2 can be described as follows. Initially, *writes* and *reads* occur alternately; whenever a *read* occurs the buffer becomes empty, and after each *write* the buffer will have exactly one message. 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 = 2. The formulae ℓ_2, r_2 are given below:

$$\ell_2 = W \wedge (\neg R \;U\; W)$$
$$r_2 = R \wedge (\neg W \;S\; R)$$

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

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

where $g = (p \supset X(\neg p \;U\; q)) \wedge (q \supset [X(\neg q \;U\; p) \vee X(\neg q \;U\; c)])$

The first conjunct in $alt(p,q,c)$ asserts that either there is a future instant at which c occurs and until this instant p,q occur alternately, or throughout the future p,q occur alternately without c occuring 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 2 is captured by the formula bv given below.

$$bv = alt(W,R,\ell_2) \wedge G[\ell_2 \supset X\ alt(W,R,r_2)] \wedge G[r_2 \supset X\ alt(W,R,\ell_2)]$$

bv asserts that ℓ_2, r_2 occur alternately with alternating *read* and *writes* occurring in between. Any *read* after ℓ_2 but before the next r_2 is on a full buffer, while any *read* after an r_2 but before the next ℓ_2 is on a buffer containing one message. The formulas read-on-full, read-on-single given below characterize *reads* on a full buffer and *reads* on a buffer with one message, respectively.

$$\text{read-on-full} = R \wedge (\neg r_2\ S\ \ell_2)$$
$$\text{read-on-single} = R \wedge [(\neg \ell_2\ S\ r_2) \vee \neg(\text{True}\ S\ \ell_2)]$$

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

$$fb_2 = I \wedge bv \wedge g \wedge h \quad \text{where}$$
$$g = G(\text{read-on-full} \supset \bigwedge_{\sigma} R_\sigma \supset [\neg W\ S\ (W \wedge Y(\neg W\ S\ W_\sigma))]),$$
$$h = G(\text{read-on-single} \supset \bigwedge_{\sigma} [R_\sigma \supset (\neg W\ S\ W_\sigma)]).$$

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 infinite sequence of states* t, $t,0 \models fb_2$ *iff* $t \in FS_{\Sigma,2}$.

Let $t \in LS_{\Sigma,2}$. If $t,i \models r_2$, then there exists $j < i$ such that $t,j \models \ell_2$. The message read at the instance i is the message written at the instance j. If $t,i \models R$ and $t,i \not\models r_2$, 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.

$$g' = G(r_2 \supset \bigwedge_{\sigma\in\Sigma} [R_\sigma \equiv \neg \ell_2\ S\ (\ell_2 \wedge W_\sigma)])$$
$$h' = G((\neg r_2 \wedge R) \supset \bigwedge_{\sigma\in\Sigma} [R_\sigma \equiv \neg W\ S\ W_\sigma])$$

Let $b_2 = I \wedge bv \wedge g' \wedge h'$

THEOREM 4.2. *For any infinite sequence of states* t, $t,0 \models \ell b_2$ *iff* $t \in LS_{\Sigma,2}$. □

Let $t \in US_{\Sigma,2}$. Then for every $\sigma \in \Sigma$, for all $i \geq 0$ the number of messages of value σ written into the buffer upto the instance i is greater than or equal to the number of messages of value σ read from the buffer upto the instance i, and they do not differ by more than 2. For a given σ, we can obtain a formula bv_σ asserting the above property by replacing R by R_σ, W by W_σ 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 infinite sequences of states* t, $t,0 \models ub_2$ *iff* $t \in US_{\Sigma,2}$. □

All the formulae fb_2, ℓb_2, ub_2 are in propositional linear temporal logic and are dependent on the message alphabet Σ. By making the following changes we can convert them into formulae in first order linear temporal logic that give domain independent characterizations of buffers of size 2.

(i) Replace all R_σ by $((read\_val = \sigma) \wedge R)$ and W_σ by $((write\_val = \sigma) \wedge W)$

(ii) Replace all $\bigwedge_\sigma$ (conjunctions over σ) by $\forall\sigma$.

It can easily be proved that the resulting formulae give domain independent characterizations of buffers of size 2.

Below we show that by introducing auxiliary propositions we can characterize bounded message buffers more elegantly. The syntax of the well formed formulae in this new logic is exactly the same as in the propositional linear temporal logic, except that some propositions are designated as auxiliary propositions and are not interpreted. Thus let $\mathcal{P} = \mathcal{P}_A \cup \mathcal{P}_1$ be the set of atomic propositions where $\mathcal{P}_A$ is the set of auxiliary propositions. As usual, an interpretation S' is a pair $\langle s',i \rangle$ where s' is an infinite sequence of states $(s'_0, s'_1, \ldots)$, each state being a mapping from $\mathcal{P}_1$ 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 $s',i \models f$) as follows:

> $s',i \models f$ iff there exists a sequence $s = (s_0,s_1,\ldots)$ such that $s,i \models f$ where for all $j \geq 0$ $s_j : \mathcal{P} \to \{\text{True},\text{False}\}$ is an extension of s'_j.

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

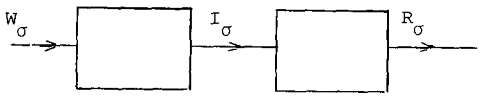

Figure 1

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 occurring simultaneously and capture it by the auxiliary propositions I_σ for $\sigma \in \Sigma$. Let $fb_1(\vec{W}_\sigma,\vec{R}_\sigma)$ be the formula characterizing a buffer of size 1, where $\vec{W}_\sigma,\vec{R}_\sigma$ indicate vectors of propositions. The sequence of operations on the left buffer is characterized by $fb_1(\vec{W}_\sigma,\vec{I}_\sigma)$, and the sequence of operations on the right buffer is characterized by $fb_1(\vec{I}_\sigma,\vec{R}_\sigma)$. Let

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

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

LEMMA 4.2. $s,0 \models fb_2$ *iff* $s \in FS_{\Sigma,2}$. □

For the general case of a buffer of size k we use a somewhat more complicated approach with k auxiliary propositions $P_0,P_1,\ldots,P_k$. 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.

$$\begin{aligned} h' = G\,[& \bigwedge_{0 \leq \ell < m \leq k} \neg(P_\ell \wedge P_m) \wedge \\ & \bigwedge_{0 \leq \ell < k} ((P_\ell \wedge W) \supset X\ P_{\ell+1}) \wedge \\ & \bigwedge_{0 < \ell \leq k} ((P_\ell \wedge R) \supset X\ P_{\ell-1}) \wedge (P_0 \supset \neg R) \wedge \\ & (P_k \supset \neg W)\,] \wedge P_0 \end{aligned}$$

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 $P_{\ell+1}$ 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.

Let

$$fb_k = I \wedge h' \wedge G(\bigwedge_{0 < \ell \leq k} (P_\ell \supset \bigwedge_\sigma (R_\sigma \supset DC(\sigma,\ell))))$$

where $DC(\sigma,\ell)$ asserts that the ℓ^{th} previous write is the message σ. It is easily seen that fb_k characterizes FIFO buffers of size k.

THEOREM 4.4. $t,0 \models fb_k$ *iff* $t \in FS_{\Sigma,k}$ □

Let $\ell b_k = I \wedge h' \wedge$

$$G(\bigwedge_{0 < \ell \leq k} (P_\ell \supset \bigwedge_\sigma [R_\sigma \supset (\neg P_{\ell-1}\ S(W_\sigma \wedge P_{\ell-1}))]))$$

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 4.5. $t,0 \models \ell b_k$ *iff* $t \in LS_{\Sigma,k}$.

Similarly we can obtain a formula for unordered buffers.

5. Characterizing Unbounded Buffers

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

LEMMA 5.1. *Let* $0 \leq i \leq j$ *be such that* $[i]_{s,f} = [j]_{s,f}$. *Then* $s,o \models f$ *iff* $s',0 \models f$ *where* $s' = (s_0, s_1, \ldots, s_i, s_{j+1}, s_{j+2}, \ldots)$. □

THEOREM 5.2. *Unbounded message buffers (unordered, FIFO or LIFO) cannot be characterized in propositional linear temporal logic.*

The above theorem can be proved by a simple argument using the previous lemma. □

THEOREM 5.3. *There is no domain independent characterization of unbounded message buffers (unordered, FIFO or LIFO) in first order linear temporal logic.*

Proof. Suppose there is a formula f of type τ in first order temporal logic, which is a domain independent characterization of an unbounded buffer on models of type τ. Consider any model of type τ with finite domain. Then f characterizes unbounded message buffers in this model. Since the domain of this model is finite, we can replace all universal quantifiers by finite conjunctions, and by some other trivial changes we can obtain a formula f' in propositional temporal logic characterizing unbounded buffers over this domain. But this contradicts the Theorem 5.3 □

We have proved that it is impossible to give a *domain independent* characterization of unbounded message buffers. However, there are *partially interpreted temporal logics* in which unbounded message buffers can be characterized. Assume that there are two local variables write-history, read-history such that at any instance write-history contains the sequence of messages written into the buffer, while read-history contains the sequence of messages read from the buffer. Then the following formula fb_∞ characterizes the behavior of an unbounded FIFO buffer:

$$\begin{aligned} fb_\infty &= g \wedge h \quad \text{where} \\ g &= G(\text{read-history} \leq \text{write-history}), \\ h &= G(\forall\, \text{hist}\ (\text{hist} = \text{write-history} \supset \\ &\qquad F\ \text{read-history} = \text{hist})) \end{aligned}$$

where $\leq$ is interpreted as the prefix relation, $\forall$ is interpreted as quantification over the set of all finite sequences of the message alphabet. 'g' asserts that the sequence of messages read from the buffer is a prefix of the sequence of messages written into the buffer; 'h' asserts that each message written into the buffer is ultimately read from the buffer. It can easily be shown that the above logic is undecidable.

Axiomatization of message buffers in PTL is a weaker notion than expressiveness. We show below that in general unbounded FIFO buffers are not axiomatizable. We also show that unbounded LIFO buffers and unbounded unordered buffers are axiomatizable though they are not expressible in PTL.

THEOREM 5.4. *Bounded FIFO, LIFO and unordered buffers over any finite alphabet* Σ *are axiomatizable in PTL.*

Proof. Let fb_k be the formula in PTL characterizing the FIFO buffer of size k over a finite alphabet Σ.

Let $\tilde{fb}_k = \text{True}\ S\ (fb_k \wedge \neg Y\ \text{True})$.
For any t and $i \geq 0$, $t,i \models \neg Y(\text{True})$ iff $i = 0$
Hence for any t and $i \geq 0$, $t,i \models \tilde{fb}_k$ iff $t,0 \models fb_k$, i.e. iff $t \in FS_{\Sigma,k}$. Let A be any consistent and complete axiomatization for PTL. Then $A \cup \{\tilde{fb}_k\}$ is a consistent and complete axiomatization for FIFO buffers of size k over Σ. Similarly we can give an axiomatization for bounded LIFO and unordered buffers over a finite alphabet. □

THEOREM 5.5. *For any* Σ *such that* $\text{card}(\Sigma) \geq 2$, *the theory of unbounded FIFO buffers over* Σ *is not axiomatizable.* (This result was suggested by Albert Meyer.)

Proof. Given a Turing machine M we can recursively obtain a formula f such that f is in the theory of unbounded FIFO buffer over $\Sigma = \{0,1\}$ iff M halts on all inputs. Thus $\Pi_2^0 \leq$ theory of unbounded FIFO buffers over any Σ with $\text{card}(\Sigma) \geq 2$. Hence theory of unbounded FIFO buffer over Σ is not axiomatizable for any Σ with $\text{card}(\Sigma) \geq 2$. In fact, it can be shown that this theory is Π_1^1-complete. □

Let $FS = \bigcup_{k \geq 1} FS_{\Sigma,k}$. Then the theory of finite FIFO message buffers is the set of all PTL formulae true in all interpretations over the models in FS. It can easily be shown that this theory is also not axiomatizable and that it is Π_1^0-complete.

Sometimes it is more realistic to consider models of FIFO buffers which are recursive; i.e. models for which we can recursively determine the truth value of an atomic proposition at any point on the model. For this case also, it can be shown that the theory of these models is Π_3^0-complete.

The degenerate case in which the message alphabet has a single element is also interesting since it corresponds to processes that communicate using signals.

THEOREM 5.6. *The theory of unbounded FIFO buffers over* Σ *where* Σ *has a single element, is decidable.*

Proof. We say that an infinite sequence of states t is ultimately periodic with starting index ℓ and period p if $\forall i \geq \ell \quad t_i = t_{i+p}$. We can easily prove that a formula f is satisfiable on a $t \in FS_{\Sigma,\infty}$ iff there exists a $t' \in FS_{\Sigma,\infty}$ such that t' is ultimately periodic with starting index $2^{c\cdot|f|}$ and period $2^{c\cdot|f|}$ for some constant c and f is satisfiable on t'. From this we can easily get a decision procedure for satisfiability of f in $FS_{\Sigma,\infty}$. Thus validity of f in $FS_{\Sigma,\infty}$ is decidable. The details will be given in the full paper. □

THOEREM 5.7. *The theory of unbounded LIFO buffers over a finite alphabet is decidable.*

Proof. For each formula f in PTL we can obtain a finite state automaton M_f on infinite strings such that M_f accepts exactly those sequences t such that $t,0 \models f$ (each state in t is a mapping from the set of atomic proposition appearing in f into {True, False}). From M_f we can obtain a pushdown automata 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. The latter problem can be reduced to a finite set of questions regarding whether an ordinary pushdown automaton (on finite strings) accepts any string. Hence the problem of satisfiability of a formula on a sequence in $LS_{\Sigma,\infty}$ is decidable. □

THEOREM 5.8. *For a finite* Σ, *satisfiability of a formula on a model of an unbounded unordered message buffer over* Σ *is decidable iff reachability problem for vector addition systems of dimension* $\text{card}(\Sigma)$ *is decidable.* □

The proof of the above theorem will be given in the full paper. Hence if the reachability problem for vector addition systems of finite dimension is decidable (as claimed by many researchers), then the theory of unbounded unordered buffers over a finite message alphabet is decidable.

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 but not unbounded ones. We have also considered axiomatization of the theory of various message buffer systems; unbounded FIFO buffers are, in general, not axiomatizable, while unbounded LIFO and unordered buffers are axiomatizable.

The techniques we have used to obtain our impossibility results should also be useful in showing that other properties are not expressible (not axiomatizable) in linear temporal logic. In addition we also believe that our extension of propositional linear temporal logic to include auxiliary propositions is quite natural and may prove useful in verifying concurrent systems. We can show that our logic is at least as powerful as Wolper's extensions of linear time logic [WO81] and is exactly as expressive (although not as concise) as the monadic second order theory of one successor.

Acknowledgement

The authors wish to acknowledge Albert Meyer's insightful suggestions; he suggested that we should look at the problem from a model theoretic point of view and provided the proof of Theorem 5.5.

Bibliography

[GPSS80] D. Gabbay, A. Pnueli, S. Shealah, J. Stavi, "Temporal analysis of fairness," Seventh ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, Jan. 1980.

[MP81] Z. Manna, A. Pnueli, "Verification of concurrent programs," The Correctness Problem in Computer Science, International Lecture Series in Computer Science, Academic Press, London, 1981.

[Ow76] S. Owicki, "A consistent and complete deductive system for verification of parallel programs," 8th Annual Symposium on Theory of Computing, 1976.

[OL80] S. Owicki, L. Lamport, "Proving liveness properties of concurrent programs," Unpublished report, October 1980.

[Pn77] A. Pnueli, "The temporal logic of programs," Proceedings of the 18th Symposium on Foundations of Computer Science, Providence, RI, Nov. 1977.

[Wo81] P. Wolper, "Temporal logic can be more expressive," Proceedings of the 22nd Symposium on Foundations of Computer Science, Nashville, TN, Oct. 1981.