Abstract—We present an operator for a propositional linear temporal logic over infinite schedules of iterated transactions, which, when applied to a formula, asserts that any schedule satisfying the formula is serializable. The resulting logic is suitable for specifying and verifying consistency properties of concurrent transaction management systems, that can be defined in terms of serializability, as well as other general safety and liveness properties. A strict form of serializability is used requiring that, whenever the read and write steps of a transaction occurrence precede the read and write steps of another transaction occurrence in a schedule, the first transaction must precede the second transaction in an equivalent serial schedule. This work improves on previous work in providing a propositional temporal logic with a serializability operator that is of the same PSPACE-complete computational complexity as standard propositional linear temporal logic without a serializability operator.

Index Terms—Temporal Logic, Iterated Transactions, Serializability.

I. INTRODUCTION

The model of concurrent iterated transactions, where transactions repeat infinitely often, was originally considered in [4] because of its applicability to the scheduling problem of service processes in operating systems. The behavior of such systems is an infinite schedule and the consistency condition a generalization of the familiar serializability condition for finite schedules of database transactions [3]. In fact, even in the case of concurrent transaction management for standard database systems, it is more accurate and assumption-free to model the output of schedulers as sets of infinite schedules. Infinite schedules have also acquired a greater significance with the advent of the newer technologies of web and mobile transactions in which transactions are continuously accessing data items. Despite this, there have been only a few attempts to address the problem of proving serializability of infinite schedules. Existing approaches advocate the use of temporal logic [12] for specifying infinite schedules generated by a scheduler, as models of temporal logic formulae. For example, the work [11] defines a partial-order temporal logic over trace models for specifying properties of schedules such as serializability. Also, the work [7] allows infinite schedules to be specified in a linear temporal logic. The problem with both of these approaches is that the only viable method of proof of conditions such as serializability is one using proof rules. In the work [11] an axiomatization is given for this purpose. Although no explicit axiomatization is given in the work [7], serializability is encoded into the Quantified Propositional Temporal Logic (QPTL) which is axiomatizable - however, no practical alternative based on a decision procedure is possible as QPTL has non-elementary computational complexity. The drawback of conducting proofs using proof rules is that they require considerable expertise by the person who is to carry out the proof manually, perhaps with the help of a 'proof assistant' tool. One of the attractions of certain temporal logics in computer science is their favorable computational complexity as compared to classical (non-temporal) logics that have the same expressiveness. For example, the validity problem for Propositional Linear Temporal Logic (PLTL) is PSPACE-complete whereas the validity problem for a classical equivalent is non-elementary. This has led to the development of industrial-strength fully automatic theorem provers, such as NuSMV [1] and SPIN [6], for commonly used such temporal logics. With this in mind, the ideal solution to proving serializability of infinite schedules would be one that could utilize these logics efficiently.

Numerous variants of serializability have been proposed as the appropriate consistency condition in various circumstances and for various reasons in the case of finite schedules of concurrent transactions, for example [10], [15], [13] and [8]. In the case of the infinite schedules that result from concurrent iterated transactions an extension of conflict serializability to unbounded schedules, based on that used for the case of finite schedules of fixed length, is defined in [4] and weaker versions given in the work [5]. Conflict serializability is characterized by the commutativity of non-conflicting operations and forms of commutativity-based serializability are discussed in [11] and [9] with regard to the partial-order temporal logics that can be used to specify them. Some non-commutative forms of serializability for infinite schedules are specified in [7] making use of propositional quantification which, however, is responsible for the non-elementary complexity of the logic. In this paper, we seek a notion of serializability for infinite schedules, that can be expressed easily and efficiently in a temporal logic for which fully automatic theorem provers exist. To this end, we will consider the notion of 'serializability in the strict sense’ from [10] or 'strict serializability' as we shall refer to it. Strict serializability has the following motivation. It is observed in [10], that certain schedules have a curious, maybe undesirable, property. Consider the following schedule:


where the R’s denote read steps, the W’s write steps, subscripts identify transactions and the brackets denote the data items.

Walter Hussak is a lecturer in the Department of Computer Science, Loughborough University, UK (Email: W.Hussak@lboro.ac.uk, Tel.: +44(0)1509 222937, Fax: +44(0)1509 211586).
accessed. This schedule serializes to the schedule:

$$R_2[x, z]W_2[z]R_1[x, y]W_1[x]R_2[y]W_2[y]$$

In the first schedule, transaction 2 has completed execution before transaction 3 has even started execution, yet the only serialized order has transaction 3 appearing before transaction 2. This undesirable property could be compounded in the case of an infinite schedule where any number of iterations of transaction 2 could execute before an occurrence of transaction 3, yet the only serialized order would have all those occurrences of transaction 2 coming after that single occurrence of transaction 3. Strict serializability does not allow such a serialization.

This paper is structured as follows. In section II, we extend strict serializability to the case of infinite schedules of infinitely repeating 2-step transactions and we give a test for a schedule to be strictly serializable that involves selecting an occurrence of each of the iterating transactions. This test is improved in section III by showing that only occurrences of a bounded subset of the iterating transactions have to be considered. A strict serializability operator is then defined for propositional linear temporal logic in section IV and the extended logic is shown to be PSPACE-complete. We give concluding remarks in section V.

II. STRICT SERIALIZABILITY

In this section strict serializability is defined (Definitions 1-4), a condition that provides a test for strict serializability is given (Definitions 5,6), and then this condition is proved to correspond to strict serializability (Lemma 7 and Theorem 8).

The assumptions and notation for our 2-step transaction model are largely as in [7]. We assume $n$ transactions $T_1, \ldots, T_n$ where each $T_i$ comprises a read step and a write step accessing finite sets of data items or variables denoted by $S(R_i)$ and $S(W_i)$ such that $S(W_i) \subseteq S(R_i)$, i.e. the write set is a subset of the read set. If $S(R_i) = \{y_1, \ldots, y_p\}$ and $S(W_i) = \{y_1', \ldots, y_q\}$ we shall display the read and write steps as $R_i[\{y_1, \ldots, y_p\}]$ and $W_i[\{y_1', \ldots, y_q\}]$ respectively. We shall omit the $[\ ]$ brackets if the variables accessed are of no interest and use the notation $R_i[x]$ and $W_i[x]$ to indicate a step that accesses $x$ and may access other variables. The finite set of all variables accessed by the $T_j's$ will be $\{x_1, \ldots, x_m\}$. A schedule or history for $T_1, \ldots, T_n$ is an interleaved sequence $h$ of the read and write steps of infinitely many occurrences of the $T_i's$, such that the subsequence of $h$ comprising steps of $T_i$ is the infinitely repeating sequence $R_iW_iR_iW_i \ldots$

Different occurrences of steps will be labelled by adding an extra subscript as in the following history

$$R_{i1}R_{i2}W_{i1}W_{i2}R_{i2}R_{i2}W_{i2}W_{i2}W_{i2}W_{i2} \ldots$$

The occurrence $R_{ij}$ (respectively $W_{ij}$) will be called the read (respectively write) step of the $j$-th occurrence $T_{ij}$ of $T_i$. In a history $h$, for each $i$ there will be a positive integer $e$, not necessarily equal to 1, such that occurrences of $T_i$ in $h$ are labelled by consecutive integers starting at $e$. Then, $T_{ie}$ will be referred to as the earliest occurrence of $T_i$ in $h$. We shall write $T_{ij} \in h$ when occurrence $T_{ij}$ belongs to $h$. For a history $h$, $h \subseteq T_{ie}$ will be the (irreflexive) total order between all the read and write steps of $h$. If $T_{ie}$ is the earliest occurrence of $T_i$ in $h$, then $h = T_{ie}$ will denote the history with $R_{ie}$ and $W_{ie}$ removed. The history comprising $R_{ie}$ followed by $W_{ie}$ followed by the sequence $h - T_{ie}$ will be denoted $h\downarrow (h - T_{ie})$.

Strict serializability of an infinite history $h$ means that it is ‘equivalent’, i.e. its read steps read the same write steps, to a serial history $h_S$ such that, if the write step of a transaction occurrence precedes the read step of another transaction occurrence in $h$, those two transaction occurrences must be in the same order in $h_S$. We formalize this as follows.

Definition 1 Histories $h_1$ and $h_2$ are equivalent, written $h_1 \sim h_2$, iff for $x \in \{x_1, \ldots, x_m\}$ and read and write occurrences $R_{i1j_1}$ and $W_{i2j_2}$

$$\text{sees}^{h_1}_{h_2}(R_{i1j_1}, W_{i2j_2}) \iff \text{sees}^{h_2}_{h_1}(R_{i1j_1}, W_{i2j_2})$$

where $\text{sees}^{R}_{h}(R_{i1j_1}, W_{i2j_2})$ holds if $h$ is of the form

$$\ldots W_{i2j_2}[x], \ldots, R_{i1j_1}[x], \ldots$$

no writes to $x$.

Definition 2 A history $h_S$ is serial iff it is of the form

$$R_{i1j_1}W_{i1j_1}R_{i2j_2}W_{i2j_2}R_{i3j_3}W_{i3j_3} \ldots$$

Definition 3 A history $h_S$ is strictly serial with respect to $h$ iff:

(i) $h_S$ is serial
(ii) $h_S$ has the same occurrences as $h$
(iii) if $W_{i1j_1} \in h$ then $W_{i1j_1} \in h_S$ and $R_{i2j_2} \in h_S$

Definition 4 A history $h$ is strictly serializable iff there is a strictly serial history $h_S$ such that $h \sim h_S$. It is easy to show that

$$R_{i1j_1}[x] \in h_S$$

iff $W_{i2j_2}[x] \in h_S$ and $R_{i1j_1}[x] \in h_S$ (1)

and

$$W_{i1j_1}[x] \in h_S$$

iff $W_{i2j_2}[x] \in h_S$ and $W_{i1j_1}[x] \in h_S$ (2)

The test for serializability that is encoded into temporal logic in [7] requires that any chosen set of occurrences of transactions in the history $h$ has a ‘ detachable ’ occurrence. For strict serializability, the corresponding test requires an additional condition to produce a ‘strictly detachable’ occurrence, i.e. one whose read step cannot come after a write step in the chosen set of occurrences (see (iv) of Definition 5 below).

Definition 5 Let $h$ be a history, $p$ be an integer such that $1 \leq p \leq n$ and $\{T_{i1j_1}, \ldots, T_{i1j_p}\} \subseteq \{T_1, \ldots, T_n\}$. Then, (the sequence of read and write steps of) $T_{i1j_1}, \ldots, T_{i1j_p}$ is strictly detachable or s-detachable in $h$ iff one of the occurrences $T_{i1j_g}$, called a s-detachable occurrence in $T_{i1j_1}, \ldots, T_{i1j_p}$ is such that, for $1 \leq g \leq p$, $g \neq k$, $x \in \{x_1, \ldots, x_m\}$
Definition 6  The condition ssrcond(h) holds iff every sequence of occurrences \(T_{i_1,j_1}, \ldots, T_{i_n,j_n}\) as in Definition 5, such that \(\{T_{i_1,j_1}, \ldots, T_{i_m,j_m}\} = \{T_1, \ldots, T_n\}\), is s-detachable.

We show that ssrcond is indeed a necessary and sufficient condition for strict serializability to hold.

Lemma 7  Let \(h\) be a history with earliest occurrences \(T_{i_1,e_1}, \ldots, T_{i_n,e_n}\) such that ssrcond(h) holds. Then, for some \(k\) with \(1 \leq k \leq n\),

(i) \(T_{i_k,e_k}\) is a s-detachable occurrence

(ii) \(h \sim T_{i_k,e_k}(h - T_{i_k,e_k})\)

(iii) ssrcond(h - T_{i_k,e_k}) holds

Proof As ssrcond(h) holds, it is immediate from Definition 5 that \(T_{i_k,e_k}\) satisfying (i) can be chosen. Now, let \(h' = T_{i_k,e_k}(h - T_{i_k,e_k})\). To prove (ii) we show that \(\text{sees}^g_k(R_{i,j}, W_{i,j'})\) iff \(\text{sees}^g_k(R_{i,j}, W_{i,j'})\) for any read and write steps \(R_{i,j}\) and \(W_{i,j'}\) respectively. Consider the non-trivial case that \(x \in S(t_k,e_k)\). As \(T_{i_1,e_1}, \ldots, T_{i_n,e_n}\) are the earliest occurrences in \(h\) and \(T_{i_k,e_k}\) is s-detachable then, by Definition 5(ii) and (iii), \(h\) is of the form

\[
\ldots R_{i_k,e_k}[x] \ldots W_{i_k,e_k}[x] \ldots
\]

and, as \(h'\) only moves \(R_{i_k,e_k}[x]\) and \(W_{i_k,e_k}\) to the left, \(\text{sees}^g_k(R_{i,j}, W_{i,j'})\) will be the same as \(\text{sees}^g_k(R_{i,j}, W_{i,j'})\). If \((i,j) \neq (i_k,e_k)\) and \((i',j') \neq (i_k,e_k)\), then \(h\) cannot be of the form

\[
\ldots R_{i_k,e_k}[x] \ldots W_{i_k,e_k}[x] \ldots
\]

as \(T_{i_k,e_k}\) is detachable and Definition 5(ii) would be breached as the read step of the earliest occurrence of \(T_i\) would precede \(R_{i_k,e_k}[x]\) and therefore \(W_{i_k,e_k}[x]\). So, \(h\) is of the form

\[
\ldots \ldots \ldots \ldots \ldots \ldots W_{i_k,e_k}[x] \ldots R_{i_k,e_k}[x] \ldots
\]

in which case \(\text{sees}^g_k(R_{i,j}, W_{i,j'})\) iff \(\text{sees}^g_k(R_{i,j}, W_{i,j'})\) as \(h\) only moves \(R_{i_k,e_k}[x]\) and \(W_{i_k,e_k}[x]\) to the left.

For (iii), let \(T_{i_1,j_1}, \ldots, T_{i_m,j_m}\) be a sequence of (not necessarily the earliest) occurrences of \(T_1, \ldots, T_n\) in \(h'' = h - T_{i_k,e_k}\). As \(T_{i_k,e_k} \notin \{T_{i_1,j_1}, \ldots, T_{i_m,j_m}\}\) then, by the definition of \(h''\), for \(1 \leq r < g \leq n\), \(R_{i_r,j_r} \leq h R_{i_t,j_t}\) iff \(R_{i_r,j_r} \leq h'' R_{i_t,j_t}\).

\[
W_{i,j} \leq h W_{i,j} \text{ iff } W_{i,j} \leq h'' W_{i,j}.
\]

From this, it is clear that a s-detachable \(T_{i_k,j_k}\) of \(T_{i_1,j_1}, \ldots, T_{i_m,j_m}\) in \(h\) is also s-detachable in \(h''\). It follows that ssrcond(h) holds.

Theorem 8  A history \(h\) is strictly serializable iff ssrcond(h) holds.

Proof Let \(h\) be strictly serializable. Choose a strictly serial history \(h_S\) such that \(h \sim h_S\). Let \(T_{i_1,j_1}, \ldots, T_{i_m,j_m}\) be occurrences in \(h\). As \(h\) is strictly serializable then, by Definition 3, one of the occurrences \(T_{i_k,j_k}\) is such that, for \(1 \leq g \leq n\), \(g \neq k\), \(R_{i_k,j_k} < h W_{i_j,j_g}\) and \(h_S\) is of the form

\[
\ldots R_{i_k,j_k} W_{i_j,j_g} \ldots R_{i_k,j_k} W_{i_j,j_g} \ldots
\]

Thus, \(T_{i_k,j_k}\) satisfies Definition 5(iv). By (1) and (2), for \(x \in \{x_1, \ldots, x_m\}\)

\[
W_{i_k,j_k}[x] < h R_{i_k,j_k}[x],
\]

\[
R_{i_k,j_k}[x] < h W_{i_j,j_g}[x],
\]

\[
W_{i_j,j_g}[x] < h W_{i_j,j_g}[x].
\]

and so the conditions Definition 5(i), (ii) and (iii) are also satisfied. Therefore, \(T_{i_k,j_k}\) is s-detachable. It follows that ssrcond(h) holds.

Conversely, suppose that ssrcond(h) holds. We show that \(h\) is strictly serializable. Define a sequence \(h_{0,0}, \ldots, h_{m,m}\) of histories, inductively, as follows

\[
h_0 = h, \quad h_{m+1} = h_m - T_{i_m,j_m} \quad (m \geq 0)
\]

where \(T_{i_m,j_m}\) is defined to be a s-detachable member of the earliest occurrences of \(h_m\). Now define the sequence \(h_S\) whose \(2m\)-th and \(2m+1\)-th \((m \geq 0)\) steps are

\[
h_S(2m) = R_{i_m,j_m}, \quad h_S(2m+1) = W_{i_m,j_m}
\]

We show that \(h_S\) is strictly serial by showing that conditions (i), (ii) and (iii) of Definition 3 are satisfied. Condition (i) is satisfied as \(h_S\) is serial by construction. For condition (ii), we need to show that \(h_S\) has the same occurrences as \(h\). Assume, on the contrary, that there is an occurrence, \(T_{i_{j_1}\ldots}\), say, in \(h\) that is not in \(h_S\). Without loss of generality, we can choose \(j_1\) to be the smallest value for which \(T_{i_{j_1}}\) is in \(h\) but not in \(h_S\), i.e.

\[
T_{i_{j_1}} \in h \text{ and } T_{i_{j_1}} \notin h_S \implies j_1 \leq j_1'
\]

Now, as \(h_S\) is infinite, there is some transaction, \(T_{i_2}\), say, which has infinitely many occurrences in \(h_S\). Therefore, we can choose an occurrence \(T_{i_{j_2}}\) in \(h_S\) such that

\[
W_{i_{j_2}} < h_{i_{j_2}}
\]

By (3), \(T_{i_{j_2}}\) belongs to \(h_S\) because there is an integer \(l \geq 0\) such that

\[
h_{l+1} = h_l - T_{i_{j_2}}
\]

and \(T_{i_{j_2}}\) is a s-detachable member of the earliest occurrences of \(T_1, \ldots, T_n\) in \(h_l\). Consider the earliest occurrence of \(T_{i_{j_2}}\) in \(h_l\). As \(T_{i_{j_1}}\) is not in \(h_S\), by the inductive definition of \(h_S\)
III. REPRESENTATIVES OF REFUTATIONS

In view of Theorem 8, a history \( h \) is strictly serializable iff whenever occurrences of all of \( T_1, \ldots, T_n \) in \( h \) are selected, the resulting subsequence of \( h \) of \( 2n \) steps is s-detachable. Therefore, in order to prove that \( h \) is not strictly serializable, a sequence of matching read and write steps of all of \( T_1, \ldots, T_n \) in \( h \), that is not s-detachable, has to be found. The number of different sequences (permutations) of the \( 2n \) steps of \( T_1, \ldots, T_n \) such that a read step comes before a write step is \( (2n)!/2^n \) which is greater than \( 2^n \) for \( n > 1 \). Now, strict serializability can be encoded into temporal logic by locating all such possible sequences of steps occurring in \( h \) and asserting their s-detachability. However, if all possible sequences of \( 2n \) steps are encoded, the temporal logic formula is exponential in the number of transactions \( n \). This presents a major obstacle to proving strict serializability in the cases of large numbers of transactions. Fortunately, this problem can be overcome as the number of data items places a bound on the number of steps of sequences that have to be considered. In this section, we define a ‘representative’ to be a sequence of steps of transactions that occur in a history \( h \) and refute the strict serializability of \( h \).

Definition 9 Let \( h \) be a history, \( p \) be an integer such that \( 1 \leq p \leq n \), \( \{1, \ldots, i_p\} \subseteq \{1, \ldots, n\} \), and \( \rho \) be a bijection
\[
\rho : \{1, \ldots, 2p\} \to \{R_{i_1}, W_{i_1}, \ldots, R_{i_p}, W_{i_p}\}
\]
Then, a subsequence \( \Sigma \) of \( h \) comprising the steps \( R_{i_1}, W_{i_1}, \ldots, R_{i_p}, W_{i_p} \) occurring in the order
\[
\Sigma = \rho(1) \ldots \rho(2p)
\]
is a representative of (a refutation of strict serializability for) \( h \) iff there is a sequence of transaction occurrences \( T_{i_1 j_1}, \ldots, T_{i_p j_p} \), whose steps in \( h \) occur in the order of the steps in \( \Sigma \), that is not s-detachable. The following theorem places a bound on the number of steps of representatives that need to be considered to refute strict serializability, independent of \( n \) if \( n \) is sufficiently large.

Theorem 10 If \( n \geq 2^{m+2} \) then a history \( h \) has a representative with \( 2n \) steps iff \( h \) has a representative with \( 2^{m+2} \) steps.

Proof
If \( h \) suppose that \( h \) has a representative \( \Sigma \) of \( 2^{m+2} \) steps. Then, by Definition 9, there is a corresponding sequence of transaction occurrences \( T_{i_1 j_1}, \ldots, T_{i_p j_p} \), where \( p = 2^{m+1} \), that is not s-detachable and whose steps occur in \( h \) in the same order as in \( \Sigma \). Choose occurrences \( T_{j_{p+1} j_{p+1}}, \ldots, T_{j_n j_n} \) such that \( \{T_{1, \ldots, T_n}\} = T_{1, \ldots, T_n} \) and that
\[
W_{i_p j_p} < h R_{i_j j_f} (1 \leq g \leq p, p + 1 \leq f \leq n)
\]
We show that \( T_{i_1, \ldots, T_n} \) is not s-detachable. Now, no \( T_{i_k j_k} \) such that \( 1 \leq k \leq p \) is s-detachable in \( T_{i_1 j_1}, \ldots, T_{i_p j_p} \) as \( T_{i_1 j_1}, \ldots, T_{i_p j_p} \) is not s-detachable, and so \( T_{i_k j_k} \) will not
satisfy (i)-(iv) of Definition 5 for $1 \leq g \leq p$ let alone for $1 \leq g \leq n$. But, also, no $T_{ikj}$ with $p + 1 \leq k \leq n$ is s-detachable as, by (11), for $1 \leq g \leq p$,

$$W_{ikj} \leq_R R_{ikj}$$

and (iv) of Definition 5 cannot be satisfied by such a $k$. Thus, $T_{i1j}, \ldots, T_{inj}$ is not s-detachable and the representative whose steps occur in the order that the steps of the occurrences $T_{i1j1}, T_{i1j2}, \ldots, T_{i1jn}$ occur in $h$ is the required representative of $2n$ steps. **Only if**

Suppose that $h$ has a representative of $2n$ steps, corresponding to the occurrences $T_{i1j1}, \ldots, T_{injn}$. Put

$$S_R = \{S(R_i) | 1 \leq f \leq n\}, \quad S_W = \{S(W_i) | 1 \leq g \leq n\}$$

Clearly, $S_R$ and $S_W$ each have at most $2^m$ elements as there are only $2^m$ subsets of the set of data items $\{x_1, \ldots, x_m\}$. Choose

$$T_{i1j1}, T_{i1j2}, \ldots, T_{injn}$$

to be such that

$$\{S(R_{i1j1}), \ldots, S(R_{injn})\} = S_R$$

and that, for all $1 \leq f \leq n$, there is a $l$, with $1 \leq l \leq 2^m$, such that

$$S(R_{i1j1}) = S(R_{i1j}) \quad \text{and} \quad R_{i1j1} \leq_R R_{i1jl} \quad (13)$$

Basically, (12) states that the read sets of the chosen $2^m$ transaction occurrences $T_{i1j1}, \ldots, T_{injn}$ span all the read sets of the possibly greater number of transaction occurrences $T_{i1j1}, \ldots, T_{injn}$. The condition (13) states that the earliest occurrences spanning those read sets, should be chosen. In a similar way, we can choose

$$T_{i1j1}, T_{i2j2}, \ldots, T_{injn}$$

to be such that

$$\{S(W_{i1j1}), \ldots, S(W_{injn})\} = S_W$$

and that, for all $1 \leq g \leq n$, there is a $l$, with $1 \leq l \leq 2^m$, such that

$$S(W_{i1j1}) = S(W_{i1j}) \quad \text{and} \quad W_{i1j1} \leq_R W_{i1jl} \quad (15)$$

We show that the sequence of the $2(2^m + 2^m) = 2m^2 + 2m$ steps in $h$ of the occurrences

$$T_{i1j1}, T_{i2j2}, \ldots, T_{injn}$$

is a representative. This means showing that the sequence (16) is not s-detachable. Now, the sequence $T_{i1j1}, T_{i2j2}, \ldots, T_{injn}$ is certainly not s-detachable as its steps form a representative. Assume, on the contrary, that the sequence (16) is s-detachable. Then, one of its occurrences, $T_{ikj}$ say, satisfies (i)-(iv) of Definition 5. We derive the contradiction that $T_{ikj}$ is a s-detachable occurrence of $T_{i1j1}, \ldots, T_{injn}$. Let $1 \leq g \leq n$, $g \neq k$ and $x \in \{x_1, \ldots, x_m\}$. We have, by (13), that, for some $l$ with $1 \leq l \leq 2^m$,

$$S(R_{ikj}) = S(R_{i1j1}) \quad \text{and} \quad R_{ikj} \leq_R R_{ikjl} \quad (17)$$

Thus, Definition 5(i) is satisfied by $T_{ikj}$ for occurrences $T_{i1j1}, \ldots, T_{injn}$. Next, by (15), we have that, for some $l$ with $1 \leq l \leq 2^m$,

$$S(W_{ikj}) = S(W_{i1j}) \quad \text{and} \quad W_{ikj} \leq_R W_{ikjl} \quad (19)$$

As $T_{ikj}$ is a s-detachable occurrence in (16), then, by Definition 5(ii),

$$R_{ikj} \leq_R W_{ikjl} \quad (20)$$

By (17) and (18),

$$W_{ikj} \leq_R R_{ikjl} \quad (21)$$

Thus, Definition 5(iii) is satisfied by $T_{ikj}$ for occurrences $T_{i1j1}, \ldots, T_{injn}$. Finally, as $T_{ikj}$ is a s-detachable occurrence in (16), then, by Definition 5(iv),

$$R_{ikj} \leq_R W_{ikjl} \quad (22)$$

By (17) and (21),

$$W_{ikj} \leq_R W_{ikjl} \quad (23)$$

Thus, Definition 5(iv) is satisfied by $T_{ikj}$ for occurrences $T_{i1j1}, \ldots, T_{injn}$. We have now derived the contradiction that $T_{ikj}$ is a s-detachable occurrence of $T_{i1j1}, \ldots, T_{injn}$, therefore, the assumption that (16) is detachable is untenable and it follows that the sequence of $2m^2 + 2m$ steps of the occurrences (16) is a representative as required. 

**IV. A TEMPORAL LOGIC**

We define propositional linear temporal logic with a strict serializability operator, and denote the logic by \textsc{PTL+sser}. The alphabet of \textsc{PTL+sser} consists of a list of propositional symbols $P_0, P_1, \ldots$, a list of special read/write step propositional symbols $R_1, R_2, \ldots$, and $W_1, W_2, \ldots$, booleans $\lor, \land, \neg$, and temporal operators $\square$ and $\Diamond$. Formulas in \textsc{PTL+sser} are either `top-level' formulae $\tau$ or bottom-level formulae $\psi$ generated by:

$$\tau := \neg \tau \lor \tau_1 \land \tau_2 \mid \text{sser}(\psi)$$

$$\psi := P_1 \mid R_i \mid W_i \mid \neg \psi \mid \psi_1 \land \psi_2 \mid \tau \lor \bot \mid \square \psi \mid \psi_1 \Diamond \psi_2$$

We use the standard abbreviations for $\lor$, $\Rightarrow$ and $\Leftrightarrow$, and

$$\circ \psi = \tau \square \psi, \quad \Box \psi = \neg \circ \neg \psi$$
A. Semantics

The semantics for $\text{PTL+sser}$ is given with respect to a given set of data items $X = \{x_1, \ldots, x_m\}$ being accessed by transactions and, for each positive integer $i$, a given set of data items read by transaction $i$, $S(R_i)$, and a given set of data items written to by transaction $i$, $S(W_i)$, such that $X \supseteq S(R_i) \supseteq S(W_i)$.

A model for $\text{PTL+sser}$ is an assignment $M$ of the propositions that are true at each point in time $a \in \mathbb{N}$, i.e.

$$M : \mathbb{N} \rightarrow \varphi(\{P_0, P_1, \ldots, R_1, R_2, \ldots, W_1, W_2, \ldots\})$$

where $M(a)$ gives the set of propositions equal to $\top$ (true) at time $a \in \mathbb{N}$ ($\varphi$ is the powerset constructor) such that:

(i) for each $a \in \mathbb{N}$,

$$M(a) \cap \{R_1, R_2, \ldots, W_1, W_2, \ldots\} = \{Q_0\}$$

is a singleton

(ii) the sequence of (read/write) step propositions

$$Q_0, Q_1, \ldots, (23)$$

is a history for $T_1, \ldots, T_n$ where $T_i$ comprises the read and write steps $R_i$ and $W_i$ ($1 \leq i \leq n$)

A model $M$ is strictly serializable iff the history corresponding to the sequence of propositions (23) is strictly serializable. The semantics of bottom-level formulae is given as for standard propositional linear temporal logic, by the truth relations $(M, a) \models \psi$ ($a \in \mathbb{N}$) defined inductively on the construction of $\psi$ as follows:

$$(M, a) \models P_i \iff P_i \in M(a)$$

$$(M, a) \models R_i \iff R_i \in M(a)$$

$$(M, a) \models W_i \iff W_i \in M(a)$$

$$(M, a) \models \neg \psi \iff (M, a) \not\models \psi$$

$$(M, a) \models \psi_1 \land \psi_2 \iff (M, a) \models \psi_1 \land (M, a) \models \psi_2$$

$$(M, a) \models \Box \psi \iff (M, a+1) \models \psi$$

$$(M, a) \models \Diamond \psi \iff \text{for some } b \geq a, (M, b) \models \psi_2 \text{ and, for } a \leq c < b, (M, c) \models \psi_1$$

A formula $\psi$ is said to be satisfied by the model $M$ at $a$ iff $(M, a) \models \psi$. The semantics of top-level formulae is given by the truth relation $(M, 0) \models \tau$ defined as follows:

$$(M, a) \models \neg \tau \iff (M, a) \not\models \tau$$

$$(M, a) \models \tau_1 \land \tau_2 \iff (M, a) \models \tau_1 \land (M, a) \models \tau_2$$

$$(M, a) \models \text{sser}(\psi) \iff (M, 0) \models \psi \text{ implies that } M \text{ is strictly serializable}$$

A $\text{PTL+sser}$ formula $\phi$ is valid written

$$\models \phi$$

iff $(M, 0) \models \phi$ for all models $M$. It is clear that $\models \text{sser}(\psi)$ asserts that all models satisfying $\psi$ (at 0) are strictly serializable.

B. An encoding of the sserr operator

We encode the sserr operator into plain propositional linear temporal logic (PTL) without the sserr operator, by encoding the representatives of Theorem 10. We consider the interesting case when $n \geq 2^{m+2}$. Suppose that $\psi$ has the read/write propositions $R_1, W_1, \ldots, R_n, W_n$. Let $\rho$ be the set of bijections:

$$\rho : \{1, \ldots, 2^{m+2}\} \rightarrow \{R_1, W_1, \ldots, R_{2^{m+1}}, W_{2^{m+1}}\}$$

where $\{i_1, \ldots, i_{2^{m+1}}\} \subseteq \{1, \ldots, n\}$

$$\rho^{-1}(R_y) < \rho^{-1}(W_y) \ (1 \leq g \leq 2^{m+1})$$

and such that there is no $k$ with $1 \leq k \leq 2^{m+1}$ satisfying, for $1 \leq g \leq 2^{m+1}$, $g \neq k$ and $x \in \{x_1, \ldots, x_m\}$, the following:

$$(i') \neg(\rho^{-1}(R_y[x]) < \rho^{-1}(W_y[x]))$$

$$(ii') \neg(\rho^{-1}(W_y[x]) < \rho^{-1}(R_y[x]))$$

$$(iii') \neg(\rho^{-1}(W_y[x]) < \rho^{-1}(W_{y+k}[x]))$$

$$(iv') \neg(\rho^{-1}(W_y[x]) < \rho^{-1}(R_{y+k}))$$

As (i')-(iv') correspond to (i)-(iv) of Definition 5, it is clear that a model $M$ is not strictly serializable iff, for some $\rho \in \rho$, the read and write propositions of $M$ are of the form

$$\ldots, \rho(1), \ldots, \rho(2), \ldots, \rho(2^{m+2}), \ldots$$

and are not of the form, for any $1 \leq u < v \leq 2^{m+1}$ and $1 \leq i \leq n$,

$$\ldots, \rho(u) = R_i, \ldots, \rho(v) = W_i, \ldots$$

Condition (24) is essentially the condition that $\rho(1), \ldots, \rho(2^{m+2})$ is a representative, although we need the extra condition (25) to guarantee that if $\rho(u) = R_i$, then the later $\rho(v) = W_i$ is the write step for the same occurrence of $T_i$. We can now encode sserr($\psi$) as follows:

$$\text{sserr}(\psi) = \psi \rightarrow \neg \bigvee_{\rho \in \rho} \Box(\rho(1) \cup (\rho(2) \cup (\ldots (\rho(2^{m+2})) \ldots)) \land$$

$$(27) \bigwedge_{1 \leq u \leq \rho, 1 \leq v \leq 2^{m+2}} (\rho(u) \land \rho(v) \rightarrow \neg W_i \cup \rho(v) \land W_i))$$

Here, (27) and (28) encode the conditions (24) and (25) respectively.

Theorem 11 The validity problem for $\text{PTL+sser}$ is PSPACE-complete.

Proof As $\text{PTL+sser}$ contains $\text{PTL}$, and $\text{PTL}$ is PSPACE-hard [14], it follows that $\text{PTL+sser}$ is PSPACE-hard. On the other hand, $\text{PTL+sser}$ can be encoded into $\text{PTL}$ by encoding the sserr operator as above. This involves computing fewer than $(2n)!/(2n - 2^{m+2})!(2^{m+2})! \rho \in \rho$ satisfying (i')-(iv'), i.e. a polynomial in $n$ number of $\rho$. The encoding of sserr($\psi$) into $\text{PTL}$, as given by (26), (27) and (28) is therefore achieved with at most a polynomial increase in the size of $\psi$. Thus, any formula $\phi$ in $\text{PTL+sser}$ containing subformulae of the form sserr($\psi$) can be rewritten by a $\text{PTL}$ formula without any occurrences of the sserr operator, incurring at worst a polynomial increase in size of formula. Therefore, $\text{PTL+sser}$ is in PSPACE. It follows that $\text{PTL+sser}$ is PSPACE-complete.
V. CONCLUSIONS

The importance of modelling infinite schedules of concurrent transactions is growing with the appearance of new technologies such as mobile transactions. A natural way of modelling such schedules is to use temporal logic. The few existing approaches that have considered this problem, use temporal logics that rely on the manual use of proof rules to produce correctness proofs of the main consistency property of serializability. In this paper, we have presented a version of serializability that can be easily realized as an additional operator to one of the most common temporal logics of all - propositional linear temporal logic. We have shown that the validity problem of the resulting extended logic is of the same PSPACE-complete computational complexity. Further to this, regarding $PTL^{++ser}$ model-checking, we note from [2] that the algorithm that checks whether a finite state machine satisfies a $PTL$ formula has time complexity $O((|S| + |R|)2^{|f|})$ where $|S|$ is the number of states, $|R|$ is the number of transitions and $|f|$ is the length of the formula. Given a $PTL^{++ser}$ formula $g$, the encoding in section IV which removes instances of the $ser$ operator produces a $PTL$ formula $f$ whose length is a polynomial in the length of $g$. It follows that the model-checking algorithm for $PTL^{++ser}$ is of comparable time complexity to that for $PTL$. Therefore, in every respect, proofs in $PTL^{++ser}$ should be as efficient as proofs in plain $PTL$. So, the logic $PTL^{++ser}$ is suitable for use with the well-known powerful model-checkers [1] and [6]. This opens up the possibility of conducting proofs of correctness of infinite schedules using fully automated means avoiding the drawbacks of manual proofs.

Further work will look to extend these results to the case of infinite schedules of concurrent multi-step transactions.

REFERENCES