World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:3, 2009

A CTL Specification of Serializability for Transactions Accessing Uniform Data

Rafat Alshorman, Walter Hussak

Abstract—Existing work in temporal logic on representing the execution of infinitely many transactions, uses linear-time temporal logic (*LTL*) and only models two-step transactions. In this paper, we use the comparatively efficient branching-time computational tree logic *CTL* and extend the transaction model to a class of multi-step transactions, by introducing distinguished propositional variables to represent the read and write steps of n multi-step transactions accessing m data items infinitely many times. We prove that the well known correspondence between acyclicity of conflict graphs and serializability for finite schedules, extends to infinite schedules. Furthermore, in the case of transactions accessing the same set of data items in (possibly) different orders, serializability corresponds to the absence of cycles of length two. This result is used to give an efficient encoding of the serializability condition into CTL.

Keywords—computational tree logic, serializability, multi-step transactions.

I. Introduction

AS concurrent users access and update databases in terms of transactions, a reliable condition of correctness is needed for the execution of these transactions. The established correctness condition is that of serializability, where an interleaved schedule of concurrent transactions is equivalent to a serial schedule of the transactions. Most work on serializability has modelled histories to be finite with a known fixed bound [8], [9]. Recently, with the emergence of new techniques such as web transactions and mobile databases, where an unlimited number of transactions may be incoming and outgoing to the databases in continuous streams, the importance of representing infinite histories has been recognised [5], [6], [7].

One way of representing infinite histories is as models of temporal logic formulae. A benefit of using temporal logic is the availability of powerful model checkers such as NuSMV [2]. Model checkers can carry out exhaustive checks of a correctness criterion such as serializability, and are fully automatic and therefore require no special expertise to carry out the verification. The drawback with model checking is that even the most powerful model checkers cannot overcome the theoretical worst-case complexity of model checking inherent from the temporal logic being used. The most benign temporal logic in this respect is *CTL* which can check whether executions represented by a finite-state machine satisfy a specification with time complexity $O((|S| + |R|).|f|)$, where $|S|$ is the number of states in the finite state machine, $|R|$ the number of transitions, and $|f|$ is the length of the specification formula. This is marginally better than for *LTL* which has a corresponding time complexity of $O((|S| + |R|).2^{O(|f|)}$ [3]. However, the temporal logics that have been used to specify transactional concurrency include the partial-order temporal logic *ISTL* in [10], quantified propositional temporal logic *QPTL* in [5], *LTL* in [6], a first-order temporal logic in the first part of [11] and a monodic fragment of first-order temporal logic in [7]. With the exception of *LTL* these are, at best, of exponential space complexity, and, at worst, undecidable.

In this paper, we give a computationally efficient specification of serializability in *CTL*. The serializability condition expressed in *CTL* is based on acyclicity of conflict graphs. To be able to use such a condition, we prove that acyclicity of conflict graphs corresponds to serializability for *infinite* schedules. We then assume the further property for our transactions, that they access the same set of data items in different orders. We show that serializability then corresponds to the efficient condition where only cycles of length two need be checked, and this condition is used for the *CTL* specification. This work advances that of [5] and [6], which both deal with two-step transactions, to the more normal case of multi-step transactions. We also produces the specification in the slightly more efficient *CTL* rather than *LTL*. The paper is organized as follows. In Section II, we give a mathematical model of concurrent multi-step transactions. In Section III, the results on acyclicity of conflict graphs and serializability for infinite schedules are given. From these, serializability is characterized mathematically in a way to be encoded into *CTL*. The *CTL* specification is given in Section IV, and conclusions are drawn in Section V.

II. A model of concurrent multi-step transactions

A. Steps and histories

The model of concurrent two-step transactions in [5] comprises n transactions $\{T_1, \ldots, T_n\}$ occurring infinitely many times, with each transaction containing a read step and a write step each accessing a finite number of data items. In this paper, we define transactions as containing multiple alternate read and write steps, each accessing a single data item. We shall denote a read step and the corresponding write step on the data item x_j by transaction T_i, as $r_i(x_j)$ and $w_i(x_j)$, respectively, and the set of data items accessed by all transactions as D. We say that two steps are *conflicting* if they belong to different transactions, they access the same data item and at least one of them is a write step. Later in this paper, we shall assume that, given transactions T_i and $T_{i'}$, the data items accessed by both are the same, but that the order of access of data

R. Alshorman is with the Department of computer science, Loughborough University, Loughborough, LE11 3TU, UK, e-mail: R.alshorman@lboro.ac.uk.

W. Hussak is with the Department of computer science, Loughborough University, Loughborough, LE11 3TU, UK, e-mail: W.Hussak@lboro.ac.uk.

Manuscript received June 24, 2009; revised July 8, 2009.

items by transaction T_i is not necessarily the same as that by $T_{i'}$. Precisely, we will assume a finite set of data items $D = \{x_1, \ldots, x_m\}$, an infinite set of (multi-step) transactions $T = \{T_i : i \in \mathbb{N}_1\}$, where $\mathbb{N}_1$ is the set of positive integers, such that all $T_i \in T$ are of the form,

$$T_i = r_i(x_{i_1})w_i(x_{i_1}) \ldots r_i(x_{i_m})w_i(x_{i_m})$$

where $\{x_{i_1}, \ldots, x_{i_m}\} = D$.

A *history* h of T is an interleaved sequence of all the read and write steps, of all the transactions in T, such that, for each $i \geq 1$, the subsequence of h compromising the steps of T_i is exactly the sequence of steps of T_i occurring in the order that they do in T_i. For a history h, $<_h$ will denote the (irreflexive) total order between all the read and write steps of h. If $T' \subseteq T$, then the *projection* of h to T', denoted $h_{T'}$, is the history of T', obtained from h, by deleting all steps of transactions not in T'.

B. Serializability

The required correctness condition of 'serializability' is that concurrent multi-step transactions should execute in a history whose effect is 'equivalent' to a serial execution of all the $T_i \in T$. Our definitions of equivalence and serializability are based on those in [9].

Definition 1. *Histories h_1 and h_2 of $T = \{T_i : i \in \mathbb{N}_1\}$ are* equivalent*, written as $h_1 \sim h_2$, iff for all $i, i' \geq 1, i \neq i'$, and for all $x \in D$,*

1) *if $r_i(x) <_{h_1} w_{i'}(x)$, then $r_i(x) <_{h_2} w_{i'}(x)$,*
2) *if $w_i(x) <_{h_1} w_{i'}(x)$, then $w_i(x) <_{h_2} w_{i'}(x)$ and*
3) *if $w_i(x) <_{h_1} r_{i'}(x)$, then $w_i(x) <_{h_2} r_{i'}(x)$*

Definition 2. *A history h of $T = \{T_i : i \in \mathbb{N}_1\}$ is* serializable *iff there is a serial history h_S of T of the form, for each $i \in \mathbb{N}_1$,*

$$h_S \quad = \quad \ldots \quad \underbrace{\ldots r_i(x) \ldots w_i(y) \ldots}_{\text{only (all) steps of } T_i} \quad \ldots$$

such that $h \sim h_S$.

III. A condition for serializability of multi-step transactions

In [5], serializability of infinite histories is characterized in terms of 'detachable' steps for certain finite subsequences of steps. We shall determine serializability in terms of acyclicity of 'conflict graphs' - a technique widely used for finite histories [9]. We define conflict graphs in Definition 3 and in Theorem 4 give conditions for which acyclicity of conflict graphs correspond to serializability in the case of an infinite number of transactions. In Lemma 5, we give a simpler correspondence in the case where transactions access the same set of data items. This result is used to prove the main result, Theorem 7, which gives the conditions for serializability that will form the basis of the specification in *CTL* in Section IV.

Definition 3. *A* directed graph *is a pair $G = (V, A)$, where V is a set of elements called* nodes*, denoted nodes(G), and $A \subseteq V \times V$ is a set of elements called* arcs*, denoted arcs(G).*

A walk *in a directed graph $G = (V, A)$ is a sequence of nodes $(v_1, v_2, \ldots, v_n)$ such that $(v_i, v_{i+1}) \in A$ for $i = 1, \ldots, n-1$. A walk with no nodes repeated is called a* path*; it is a* cycle *when only the first and last node coincide. For each history h, there is a directed graph $G(h)$ called the* precedence graph *or* conflict graph *of h . This graph has the transactions of h as its nodes, and contains an arc $(T_i, T_{i'})$, where T_i and $T_{i'}$ are distinct transactions of h, whenever there is a step of T_i which conflicts with a subsequent (in h) step of $T_{i'}$.*

Theorem 4. *A history h of an infinite number of multi-step transactions $T = \{T_i : i \in \mathbb{N}_1\}$, accessing data items in some finite set D (though not necessarily accessing the same data items), is serializable iff the conflict graph $G(h)$ is acyclic.*

Proof:

If

Let h be a history of T such that $G(h)$ is acyclic. Assume that, for some $T_i \in T$, we have the following infinite regression of arcs:

$$\ldots, (T_{k^{n+1}}, T_{k^n}), \ldots, (T_{k^2}, T_{k^1}), (T_{k^1}, T_i) \tag{1}$$

where $\{T_{k^1}, \ldots, T_{k^n}, \ldots\} \subseteq T$. Then, as only finitely many data items are accessed by the transactions, and as each step may be preceded by only finitely many steps in h, there exist $l > j \geq 0$ such that we have the following order of steps in h (assuming, without loss of generality, that the arcs in (1) are as the result of write-read conflicts):

$$w_{k^{l+1}}(x) <_h r_{k^l}(x) <_h \ldots <_h w_{k^{j+1}}(x) <_h r_{k^j}(x) \tag{2}$$

and $w_{k^l}(x)$ does not precede $r_{k^j}(x)$ in h, i.e.

$$r_{k^j}(x) <_h w_{k^l}(x) \tag{3}$$

From (2) and (3), we produce the cycle

$$(T_{k^l}, T_{k^{l-1}}), \ldots, (T_{k^{j+1}}, T_{k^j}), (T_{k^j}, T_{k^l})$$

This contradiction shows that (1) cannot occur. It follows that we can define, inductively, the sequence $i^1, i^2, \ldots$ thus:

$$\begin{aligned} i^1 \quad &= \quad min\{k \in \mathbb{N}_1 \ : \ \text{for all } i \neq k, \\ &\qquad (T_i, T_k) \notin arcs(G(h))\} \\ &\qquad \ldots \\ i^n \quad &= \quad min\{k \in \mathbb{N}_1 \ : \ \text{for all } i \neq k, i^1, \ldots, i^{n-1}, \\ &\qquad (T_i, T_k) \notin arcs(G(h))\} \\ &\qquad \ldots \end{aligned}$$

Firstly, we show that $\{i^1, \ldots, i^n, \ldots\} = \mathbb{N}_1$. Suppose, on the contrary, that there is some $i' \in \mathbb{N}_1$ such that $i' \neq i^n$ for any (all) $n \in \mathbb{N}_1$. As the situation (1) cannot occur, we can choose i' to be such that

$$(T_i, T_{i'}) \text{ implies } T_i = i^n \text{ for some } n \in \mathbb{N}_1 \tag{4}$$

Intuitively, $T_{i'}$ is the 'earliest' transaction for which $T_{i'} \neq T_{i^n}$ for any $n \in \mathbb{N}_1$. Now choose $n' \in \mathbb{N}_1$ to be such that:

(a) $i^{n'+1} \geq i'$

(b) all the steps of any T_{i^n}, where $n \geq n'$, come after the steps of $T_{i'}$ in h.

Assume that we have that

$$(T_i, T_{i'}) \in arcs(G(h)) \text{ for some } i \in \mathbb{N}_1,\ i \neq i^1, \ldots, i^{n'} \quad (5)$$

This implies, by (4), that

$$T_i = T_{i^n} \text{ for some } n \in \mathbb{N}_1$$

which implies, by the assumption at (5) that $i \neq i^1, \ldots, i^{n'}$,

$$T_i = T_{i^n} \text{ for some } n > n'$$

This, in turn, implies, by (b), that all the steps of $T_i = T_{i^n}$ come after the steps of $T_{i'}$ in h and so $(T_i, T_{i'}) \notin arcs(G(h))$. This contradicts (5) and so the assumption at (5) cannot hold. But, then,

$$i^{n'+1} = min\{k \in \mathbb{N}_1 \ : \text{ for all } i \neq k, i^1, \ldots, i^{n'},$$
$$(T_i, T_k) \notin arcs(G(h))\} \quad (6)$$

As (5) cannot hold, by (6) and (a) we have that

$$i' > i^{n'+1} \geq i'$$

This last contradiction means that the assumption that $i' \neq i^n$ for all $n \in \mathbb{N}_1$ is false. It follows that $\{i^1, \ldots, i^n, \ldots\} = \mathbb{N}_1$.

We construct the serial history h_S of T given by $h_S =$

$$\underbrace{\ldots r_{i^1}(x) \ldots w_{i^1}(y) \ldots}_{\text{all steps of } T_{i1}} \quad \ldots \quad \underbrace{\ldots r_{i^n}(s) \ldots w_{i^n}(t) \ldots}_{\text{all steps of } T_{i^n}} \quad \ldots$$

and show that $h \sim h_S$ by showing that Definition 1(1), (2) and (3) hold. For Definition 1(1), suppose that $r_i(x) <_h w_{i'}(x)$ for transactions $T_i, T_{i'} \in T$. Then, $(T_i, T_{i'}) \in$ *arcs(G(h))*. In the sequence $i^1, i^2, \ldots$ above, we cannot have $i = i^n$ and $i' = i^{n'}$ for some $n' < n$ as, from the definition of n', that would imply that $(T_i, T_{i'}) \notin$ *arcs(G(h))*. Thus, h_S is of the form

$$\underbrace{\ldots r_i(c) \ldots w_i(d) \ldots}_{\text{all steps of } T_i} \quad \ldots \quad \underbrace{\ldots r_{i'}(e) \ldots w_{i'}(f) \ldots}_{\text{all steps of } T_{i'}} \quad \ldots$$

and Definition 1(1) holds as required. The proof of Definition 1(2) and (3) are similar.

Only if

Let h be a serializable history. This means that there is a serial history h_S such that $h \sim h_S$. This implies, by Definitions 1 and 3, that $G(h) = G(h_S)$. As $G(h_S)$ is necessarily acyclic, since it must be a subgraph of the total order under which the transactions occur in h_S, we conclude that $G(h)$ is acyclic. ■

In the case where all transactions access the same set of data items, serializability is guaranteed if *G(h)* has no cycle of length 2.

Lemma 5. *Let h be a history of multi-step transactions $T = \{T_i : i \in \mathbb{N}_1\}$ accessing the same set of data items D (in possibly different orders). Then, if $G(h)$ has a cycle, there are transactions $T_i, T_{i'}$ such that $G(h)$ has the cycle $(T_i, T_{i'}), (T_{i'}, T_i)$.*

Proof:

Assume that $G(h)$ has a cycle

$$(T_{i^1}, T_{i^2}), \ldots, (T_{i^{(n-1)}}, T_{i^n}), (T_{i^n}, T_{i^1}) \quad (7)$$

where $n > 2$, but no such cycle for $n = 2$. We will derive a contradiction. Choose any $x \in D$. Then, for $1 \leq j \leq n-1$,

$$w_{i^j}(x) <_h r_{i^{(j+1)}}(x) \quad (8)$$

otherwise $(T_{i^{(j+1)}}, T_{i^j})$ is an arc in $G(h)$ and, from (7), $(T_{i^j}, T_{i^{(j+1)}})$ is also an arc in $G(h)$ giving a cycle between T_{i^j} and $T_{i^{j+1}}$ contrary to our assumption that there are no cycles of length 2. From (8) we have that

$$r_{i^1}(x) <_h w_{i^1}(x) <_h \cdots <_h r_{i^n}(x) <_h w_{i^n}(x) <_h r_{i^1}(x) \quad (9)$$

The contradiction, from (9), that $r_{i^1}(x) <_h r_{i^1}(x)$, means that our assumption that there is no cycle between two transactions is incorrect. ■

Definition 6. *We say that T_i* comes before *$T_{i'}$ in h iff $w_i(x) <_h r_{i'}(y)$, where x and y are the first data items accessed by T_i and $T_{i'}$ respectively.*

Theorem 7. *A history h of multi-step transactions $T = \{T_i : i \in \mathbb{N}_1\}$ is serializable iff for any two distinct transactions T_i and $T_{i'}$, one of them, T_i say, is such that*

(i) *T_i comes before $T_{i'}$ in h, and*
(ii) *for all $x \in D$, $w_i(x) <_h r_{i'}(x)$*

Proof:

If

Let h be not serializable. We show that there are T_i and $T_{i'}$ such that the conditions (i) and (ii) do not both hold. To have h not serializable means, by Theorem 4 and Lemma 5, that there is a cycle in the precedence graph $G(h)$, $(T_i, T_{i'}), (T_{i'}, T_i)$. Assume that (i) holds for T_i and $T_{i'}$, i.e. T_i comes before $T_{i'}$. Here, letting x and y denote the first data items accessed by T_i and $T_{i'}$ respectively, there are a limited number of cases causing the cycle:

$$\ldots w_i(x) \ldots r_{i'}(y) \ldots w_{i'}(y) \ldots r_i(z) \ldots \underline{r_{i'}(z)} \ldots w_{i'}(z) \ldots$$
$$\ldots \underline{w_i(z)} \ldots \quad (10)$$

$$\ldots w_i(x) \ldots r_{i'}(y) \ldots w_{i'}(y) \ldots r_i(z) \ldots \underline{r_{i'}(z)} \ldots \underline{w_i(z)} \ldots$$
$$\ldots w_{i'}(z) \ldots \quad (11)$$

$$\ldots w_i(x) \ldots r_{i'}(y) \ldots w_{i'}(y) \ldots \underline{r_{i'}(z)} \ldots r_i(z) \ldots w_{i'}(z) \ldots$$
$$\ldots \underline{w_i(z)} \ldots \quad (12)$$

$$\ldots w_i(x) \ldots r_{i'}(y) \ldots w_{i'}(y) \ldots \underline{r_{i'}(z)} \ldots r_i(z) \ldots \underline{w_i(z)} \ldots$$
$$\ldots w_{i'}(z) \ldots \quad (13)$$

In the all cases (10)-(13) condition (ii) is breached because $r_{i'}(z) <_h w_i(z)$ (underlined).

Only if

Assume that we have that h does not satisfy conditions (i) and (ii) for all T_i, $T_{i'}$. We show that h is not serializable. Firstly, suppose that condition (i) holds, but that condition (ii) does not hold for some T_i and $T_{i'}$. Then, if x and y are the first data items accessed by T_i and $T_{i'}$ respectively,

$$w_i(x) <_h r_{i'}(y) \quad (14)$$

giving the arc $(T_i, T_{i'})$ in $G(h)$. As condition (ii) does not hold, there is $z \in D$ such that

$$r_{i'}(z) <_h w_i(z)$$

This gives the arc $(T_{i'}, T_i)$ and hence a cycle in $G(h)$. By Theorem 4, h is not serializable. Secondly, suppose condition (i) does not hold for some T_i, $T_{i'}$. Then, by Definition 6, if x and y are the first data items accessed as above,

$$r_i(x) <_h w_{i'}(y) \tag{15}$$

and

$$r_{i'}(y) <_h w_i(x) \tag{16}$$

From (15), if $x = y$, $(T_i, T_{i'})$ is an arc in $G(h)$, and from (16) $(T_{i'}, T_i)$ is an arc in $G(h)$. This gives a cycle and shows, by Theorem 4, that h is not serializable. But, if $x \neq y$, T_i accesses y later, and $T_{i'}$ accesses x later. Thus, by (16),

$$r_{i'}(y) <_h w_i(x) <_h r_i(y) <_h w_i(y) \tag{17}$$

and, by (15),

$$r_i(x) <_h w_{i'}(y) <_h r_{i'}(x) <_h w_{i'}(x) \tag{18}$$

From (17), $(T_{i'}, T_i)$ is an arc in $G(h)$ and, from (18), $(T_i, T_{i'})$ is an arc in $G(h)$ giving a cycle. ■

IV. Specification of serializability in CTL

We present a *CTL* specification of infinite histories composed of n transactions each accessing all of m data items, and repeating infinitely often. The aggregate of all the repetitions of the n transactions will constitute the infinite number of transactions $\{T_i : i \in \mathbb{N}_1\}$ of the previous section. Such concurrent repeating or 'iterating' transactions were originally investigated in [4] and temporal logic models have been given in [5] and [6]. In [5] and [6] each iteration of a transaction is called an *occurrence*, and every occurrence of a particular transaction comprises the same two (read and write) steps. We improve this to a case of multi-step transactions where, different occurrences of particular transactions access the same data items, but in possibly different orders. So, the order of access of data items may be different between different transactions *and* between different occurrences of the 'same' transaction. Actually, in our model here, different occurrences of the 'same' transaction bear no relation to each other. As such, we model, not so much the same n transactions iterating, but a more general case of an infinite number of (possibly totally unrelated) transactions where there is a limit of n on how many are active at any given time.

The syntax for *CTL* is given in Section IV.A and the semantics in Section IV.B. The specification of the multi-step transactions model is in Section IV.C and serializability is specified in IV.D.

A. Syntax

The alphabet of *CTL* consists of a set of propositions symbols $p_0, p_1, \ldots$, distinguished read/write step propositional symbols $r_i(x_j), w_i(x_j)$ $(1 \leq i \leq n, 1 \leq j \leq m)$, booleans $\neg, \vee, \wedge, \top, \bot$, quantifiers **E**, **A**, and temporal operators **X**, **F**, **G** and **U**. Formulae in *CTL* are those generated by:

$$\phi ::= p_i \mid r_i(x_j) \mid w_i(x_j) \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \mathbf{AX}\phi \mid$$

$$\mathbf{EX}\phi \mid \mathbf{AF}\phi \mid \mathbf{EF}\phi \mid \mathbf{AG}\phi \mid \mathbf{EG}\phi \mid \mathbf{A}[\phi_1 \mathbf{U} \phi_2] \mid \mathbf{E}[\phi_1 \mathbf{U} \phi_2]$$

Note that, despite their appearance, $r_i(x_j)$ and $w_i(x_j)$ are propositions, and not predicates, in the logic. The symbols $\bot$ and $\top$ will also be used to denote the truth values false and true respectively and the abbreviations $\Rightarrow$ and $\Leftrightarrow$ will have their usual logical meaning.

B. Semantics of CTL

An *interpretation* for *CTL*, $I(s_a)$, at a given state $s_a \in S$, where S is a set of states, assigns truth values $p_i^{I(s_a)}$, $r_i(x_j)^{I(s_a)}$ and $w_i(x_j)^{I(s_a)}$ $(\in \{\bot, \top\})$ to propositional symbols p_i, $r_i(x_j)$ and $w_i(x_j)$, respectively. A *interpretation* I over S, is a set of interpretations $I = \{I(s_a) : s_a \in S\}$. A *Kripke structure* M is a triple $< S,\ R,\ I >$, where S is a set of states, $R \subseteq S \times S$ a transition relation such that, for all $s \in S$, there exists $s' \in S$ with $(s, s') \in R$, and I is an interpretation over S. A *path* in M is an infinite sequence of states, $\pi = s_a, s_{a+1}, \ldots$, such that, for every $b \geq a, (s_b, s_{b+1}) \in R$. The set of paths that start in state s_a is denoted $Paths(s_a)$. As each state in a Kripke structure is required to have at least one successor, it follows that $Paths(s_a) \neq \{\}$ for any state s_a. The *semantics* of a *CTL* formula ϕ is given by the truth relation $M, s_a \vDash \phi$ which means that ϕ holds at state s_a in the Kripke structure M. The relation $\vDash$ is defined inductively as follows

$M, s_a \vDash p_i$ iff $p_i^{I(s_a)} = \top$
$M, s_a \vDash r_i(x_j)$ iff $r_i(x_j)^{I(s_a)} = \top$
$M, s_a \vDash w_i(x_j)$ iff $w_i(x_j)^{I(s_a)} = \top$
$M, s_a \vDash \neg\phi$ iff $M, s_a \nvDash \phi$
$M, s_a \vDash \phi_1 \vee \phi_2$ iff $M, s_a \vDash \phi_1$ or $M, s_a \vDash \phi_2$
$M, s_a \vDash \phi_1 \wedge \phi_2$ iff $M, s_a \vDash \phi_1$ and $M, s_a \vDash \phi_2$
$M, s_a \vDash \mathbf{AX}\phi$ iff, for all $\pi \in Paths(s_a),\ M, s_{a+1} \vDash \phi$
$M, s_a \vDash \mathbf{EX}\phi$ iff there exists $\pi \in Paths(s_a)$ such that $M, s_{a+1} \vDash \phi$
$M, s_a \vDash \mathbf{AF}\phi$ iff, for all $\pi \in Paths(s_a)$, there exists $b \geq a$ such that $M, s_b \vDash \phi$
$M, s_a \vDash \mathbf{EF}\phi$ iff there exists $\pi \in Paths(s_a)$ and $b \geq a$ such that $M, s_b \vDash \phi$
$M, s_a \vDash \mathbf{AG}\phi$ iff, for all $\pi \in Paths(s_a)$, and, for all $b \geq a$, $M, s_b \vDash \phi$
$M, s_a \vDash \mathbf{EG}\phi$ iff there exists $\pi \in Paths(s_a)$ such that, for all $b \geq a$, $M, s_b \vDash \phi$
$M, s_a \vDash \mathbf{A}[\phi_1 \mathbf{U} \phi_2]$ iff, for all $\pi \in Paths(s_a)$, there is some $c \geq a$ such that $M, s_c \vDash \phi_2$ and, for all $a \leq b < c$, $M, s_b \vDash \phi_1$
$M, s_a \vDash \mathbf{E}[\phi_1 \mathbf{U} \phi_2]$ iff there exists $\pi \in Paths(s_a)$ such that, for some $c \geq a$, $M, s_c \vDash \phi_2$ and, for all $a \leq b < c$, $M, s_b \vDash \phi_1$

C. Specification of multi-step transactions model

The read and write step propositions have the following intuitive meanings:

$r_i(x_j) \sim$ active transaction T_i has read data item x_j
$w_i(x_j) \sim$ active transaction T_i has written to data item x_j

The multi-step transactions model is characterized by the following properties:

(C1) Read/write alternation
A transaction T_i cannot have read two distinct data items without having written to one of them, i.e. $r_i(x_j)$ and $r_i(x_{j'})$ cannot both be true if $w_i(x_j)$ and $w_i(x_{j'})$ are both false.

(C2) Write implies read
A transaction T_i can only have written to x_j if it has read x_j, i.e. if $w_i(x_j)$ is true, then $r_i(x_j)$ must be true.

(C3) Read/write steps remain true to transaction end
If a read/write step has taken place, the corresponding propositions remain true until the transaction ends, i.e. $r_i(x_j)/w_i(x_j)$ once true, remain true until all other steps $r_i(x'_j)$ and $w_i(x'_j)$ $(x'_j \in D)$ are true.

(C4) End of transaction occurrence
After a transaction occurrence ends, at most one read step $r_i(x_j)$ and no write steps $w_i(x_j)$ can be true in any next state.

(C5) At most one step occurs at each successive state
No two distinct steps can both be false in a state, and then both true in a next state.

Given a state s_a, and a path $\pi \in Paths(s_a)$, there corresponds a sequence of read and write step propositions that become true in $s_a, s_{a+1}, \ldots$. In this way, π yields a history of infinitely many occurrences of the transactions $T_1, \ldots, T_n$. We illustrate this correspondence between paths and histories in Figure 1. In Figure 1, we have $D = \{x, y, z\}$ and transactions

$$T_1 = r_1(x)w_1(x)r_1(y)w_1(y)r_1(z)w_1(z)$$

and

$$T_2 = r_2(x)w_2(x)r_2(z)w_2(z)r_2(y)w_2(y)$$

Interpretations for read and write step propositions are given for successive states, and the top of each column displays the unique proposition that becomes true in the particular state. The corresponding history h is:

$$h = r_1(x)w_1(x)r_1(y)r_2(x)w_2(x)w_1(y)r_1(z)$$

$$w_1(z)r_2(z)w_2(z)r_2(y)w_2(y)r_1(x)\ldots.$$

We encode the conditions (C1)-(C5) as $\sigma_1, \sigma_2, \sigma_3$, σ_4 and σ_5 respectively, below. We use an extra proposition $endT_i$ to mark the states at which an occurrence of T_i ends, i.e. the states at which $r_i(x_j)$ and $w_i(x_j)$ are true for all x_j. This is defined in σ_0 as follows:

$$\sigma_0 = \bigwedge_{1 \le i \le n} \mathbf{AG}(endT_i \Leftrightarrow \bigwedge_{1 \le j \le m} (r_i(x_j) \wedge w_i(x_j)))$$

Conditions (C1)-(C5) are given below:

(C1) Read/write alternation
A transaction T_i cannot have read two distinct data items without having written to one of them, i.e. $r_i(x_j)$ and $r_i(x_{j'})$ cannot both be true if $w_i(x_j)$ and $w_i(x_{j'})$ are both false.

$$\sigma_1 = \bigwedge_{1 \le i \le n} \bigwedge_{1 \le j \ne j' \le m} \neg\mathbf{EF}(r_i(x_j) \wedge r_i(x_{j'}) \wedge \neg w_i(x_j) \wedge \neg w_i(x_{j'}))$$

$\swarrow$	$\swarrow$	$\swarrow$	$\swarrow$	$\swarrow$	$\swarrow$
$r_1(x)$	$w_1(x)$	$r_1(y)$	$r_2(x)$	$w_2(x)$	$w_1(y)$
s_0	s_1	s_2	s_3	s_4	s_5
$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$
$r_1(x)$	$r_1(x)$	$r_1(x)$	$r_1(x)$	$r_1(x)$	$r_1(x)$
$\neg w_1(x)$	$w_1(x)$	$w_1(x)$	$w_1(x)$	$w_1(x)$	$w_1(x)$
$\neg r_1(y)$	$\neg r_1(y)$	$r_1(y)$	$r_1(y)$	$r_1(y)$	$r_1(y)$
$\neg r_2(x)$	$\neg r_2(x)$	$\neg r_2(x)$	$r_2(x)$	$r_2(x)$	$r_2(x)$
$\neg w_2(x)$	$\neg w_2(x)$	$\neg w_2(x)$	$\neg w_2(x)$	$w_2(x)$	$w_2(x)$
$\neg w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$	$w_1(y)$
$\neg r_1(z)$	$\neg r_1(z)$	$\neg r_1(z)$	$\neg r_1(z)$	$\neg r_1(z)$	$\neg r_1(z)$
$\neg w_1(z)$	$\neg w_1(z)$	$\neg w_1(z)$	$\neg w_1(z)$	$\neg w_1(z)$	$\neg w_1(z)$
$\neg r_2(z)$	$\neg r_2(z)$	$\neg r_2(z)$	$\neg r_2(z)$	$\neg r_2(z)$	$\neg r_2(z)$
$\neg w_2(z)$	$\neg w_2(z)$	$\neg w_2(z)$	$\neg w_2(z)$	$\neg w_2(z)$	$\neg w_2(z)$
$\neg r_2(y)$	$\neg r_2(y)$	$\neg r_2(y)$	$\neg r_2(y)$	$\neg r_2(y)$	$\neg r_2(y)$
$\neg w_2(y)$	$\neg w_2(y)$	$\neg w_2(y)$	$\neg w_2(y)$	$\neg w_2(y)$	$\neg w_2(y)$

$\swarrow$	$\swarrow$	$\swarrow$	$\swarrow$	$\swarrow$	$\swarrow$
$r_1(z)$	$w_1(z)$	$r_2(z)$	$w_2(z)$	$r_2(y)$	$w_2(y)$
s_6	s_7	s_8	s_9	s_{10}	s_{11}
$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$
$r_1(x)$	$r_1(x)$	$\neg r_1(x)$	$\neg r_1(x)$	$\neg r_1(x)$	$\neg r_1(x)$
$w_1(x)$	$w_1(x)$	$\neg w_1(x)$	$\neg w_1(x)$	$\neg w_1(x)$	$\neg w_1(x)$
$r_1(y)$	$r_1(y)$	$\neg r_1(y)$	$\neg r_1(y)$	$\neg r_1(y)$	$\neg r_1(y)$
$r_2(x)$	$r_2(x)$	$r_2(x)$	$r_2(x)$	$r_2(x)$	$r_2(x)$
$w_2(x)$	$w_2(x)$	$w_2(x)$	$w_2(x)$	$w_2(x)$	$w_2(x)$
$w_1(y)$	$w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$
$r_1(z)$	$r_1(z)$	$\neg r_1(z)$	$\neg r_1(z)$	$\neg r_1(z)$	$\neg r_1(z)$
$\neg w_1(z)$	$w_1(z)$	$\neg w_1(z)$	$\neg w_1(z)$	$\neg w_1(z)$	$\neg w_1(z)$
$\neg r_2(z)$	$\neg r_2(z)$	$r_2(z)$	$r_2(z)$	$r_2(z)$	$r_2(z)$
$\neg w_2(z)$	$\neg w_2(z)$	$\neg w_2(z)$	$w_2(z)$	$w_2(z)$	$w_2(z)$
$\neg r_2(y)$	$\neg r_2(y)$	$\neg r_2(y)$	$\neg r_2(y)$	$r_2(y)$	$r_2(y)$
$\neg w_2(y)$	$\neg w_2(y)$	$\neg w_2(y)$	$\neg w_2(y)$	$\neg w_2(y)$	$w_2(y)$

$\swarrow$	
$r_1(x)$	$\ldots$
s_{12}	
$\bullet$	$\ldots$
$r_1(x)$	
$\neg w_1(x)$	
$\neg r_1(y)$	
$\neg r_2(x)$	.
$\neg w_2(x)$	.
$\neg w_1(y)$	.
$\neg r_1(z)$	
$\neg w_1(z)$	
$\neg r_2(z)$	
$\neg w_2(z)$	
$\neg r_2(y)$	
$\neg w_2(y)$	

Fig. 1. Correspondence between paths and histories

(C2) Write implies read
A transaction T_i can only have written to x_j if it has read x_j, i.e. if $w_i(x_j)$ is true, then $r_i(x_j)$ must be true.

$$\sigma_2 = \bigwedge_{1\leq i\leq n}\bigwedge_{1\leq j\leq m} \mathbf{AG}(w_i(x_j) \Rightarrow r_i(x_j))$$

(C3) Read/write steps remain true to transaction end
If a read/write step has taken place, the corresponding propositions remain true until the transaction ends, i.e. $r_i(x_j)/w_i(x_j)$ once true, remain true until all other steps $r_i(x'_j)$ and $w_i(x'_j)$ $(x'_j \in D)$ are true.

$$\sigma_3 = \bigwedge_{\substack{1\leq i\leq n\\ 1\leq j\leq m}} \mathbf{AG}((r_i(x_j) \wedge \neg endT_i \Rightarrow \mathbf{AX} r_i(x_j)) \wedge (w_i(x_j) \wedge \neg endT_i \Rightarrow \mathbf{AX} w_i(x_j)))$$

(C4) End of transaction occurrence
After a transaction occurrence ends, at most one read step $r_i(x_j)$ and no write steps $w_i(x_j)$ can be true in any next state.

$$\sigma_4 = \bigwedge_{1\leq i\leq n} \mathbf{AG}(endT_i \Rightarrow \mathbf{AX} \bigvee_{1\leq i\leq m}\bigwedge_{1\leq j'\neq j\leq m} (\neg r_i(x_{j'}) \wedge \neg w_i(x_{j'}) \wedge \neg w_i(x_j)))$$

(C5) At most one step occurs at each successive state
No two distinct steps can both be false in a state, and then both true in a next state.

$$\sigma_5 = \bigwedge_{\substack{1\leq i,i'\leq n\\ 1\leq j,j'\leq m\\ i\neq i' \text{ or } j\neq j'}}$$
$$\mathbf{AG}\ [\ \neg((\neg r_i(x_j) \wedge \neg r_{i'}(x_{j'})) \wedge \mathbf{EX}(r_i(x_j) \wedge r_{i'}(x_{j'}))) \wedge$$
$$\neg((\neg r_i(x_j) \wedge \neg w_{i'}(x_{j'})) \wedge \mathbf{EX}(r_i(x_i) \wedge w_{i'}(x_{j'}))) \wedge$$
$$\neg((\neg w_i(x_j) \wedge \neg w_{i'}(x_{j'})) \wedge \mathbf{EX}(w_i(x_j) \wedge w_{i'}(x_{j'})))\ ].$$

We denote by σ_{trans} the specification of the transactions model, i.e.

$$\sigma_{trans} = \sigma_0 \wedge \sigma_1 \wedge \sigma_2 \wedge \sigma_3 \wedge \sigma_4 \wedge \sigma_5$$

D. Specification of serializability

We encode conditions (i) and (ii) of Theorem 7. We make use of additional propositions $before_{i,i'}$ $(1 \leq i \neq i' \leq n)$, each of which is true in a state if the current occurrence of T_i comes before the current occurrence of $T_{i'}$. We have that $before_{i,i'}$ becomes true either if T_i has performed a write step and $T_{i'}$ has not performed any read steps, or in a state which comes after a state in which the occurrence of $T_{i'}$ ended and T_i had previously performed a write step. This is specified as σ_6:

$$\sigma_6 = \bigwedge_{1\leq i\neq i'\leq n} \mathbf{AG}[\neg before_{i,i'} \Rightarrow \mathbf{A}(\neg before_{i,i'} \mathbf{U}$$
$$((\bigvee_{1\leq j\leq m} w_i(x_j) \wedge \bigwedge_{1\leq j'\leq m} (\neg r_{i'}(x_{j'}) \wedge w_{i'}(x_{j'})) \wedge before_{i,i'}) \vee$$
$$(\bigvee_{1\leq j\leq m} w_i(x_j) \wedge endT_{i'} \wedge \neg before_{i,i'} \wedge \mathbf{AX}\ before_{i,i'})))]$$

Also, we need to ensure that $before_{i,i'}$, once true, remains true until the end of the occurrence of T_i, and then becomes false. This is given by σ_7:

$$\sigma_7 = \bigwedge_{1\leq i,i'\leq n} \mathbf{AG}((before_{i,i'} \wedge \neg endT_i \Rightarrow \mathbf{AX} before_{i,i'}) \wedge (endT_i \Rightarrow \mathbf{AX} \neg before_{i,i'}))$$

Theorem 7 condition (i) can then be encoded as σ_8 which states that, if T_i and $T_{i'}$ are active, one of them must come before the other:

$$\sigma_8 = \bigwedge_{1\leq i,i'\leq n} \mathbf{AG}\ ((\bigvee_{1\leq j,j'\leq m} r_i(x_j) \wedge r_{i'}(x_{j'})) \Rightarrow (before_{i,i'} \vee before_{i',i}))$$

Theorem 7 condition (ii) is encoded as σ_9:

$$\sigma_9 = \bigwedge_{1\leq i,i'\leq n}\bigwedge_{1\leq j\leq m} \mathbf{AG}(before_{i,i'} \Rightarrow \neg(r_{i'}(x_j) \wedge \neg w_i(x_j)))$$

We denote by σ_{sz} the specification of the serializability condition, i.e.

$$\sigma_{sz} = \sigma_6 \wedge \sigma_7 \wedge \sigma_8 \wedge \sigma_9$$

V. Conclusions

We have given a method using CTL for specifying and verifying the correctness of concurrent executions of multi-step transactions produced by schedulers. For example, a scheduler might be specified as a finite-state machine in NuSMV, corresponding to a structure $Sched$ for CTL. The specification would then be checked to see that the transactions model had been specified in the correct way. This would mean running the NuSMV model checker to show that

$$Sched, s_a \vDash \sigma_{trans}$$

Serializability could then be verified by using the NuSMV model checker to show that

$$Sched, s_a \vDash \sigma_{sz}$$

A preliminary case study of the use of *CTL* to verify serializability of mobile transactions in this way has been conducted in [1]. The CTL method here improves on previous work in two respects - there is a slight efficiency gain in using *CTL* as opposed to *LTL* and the more usual case of multi-step transactions can be modelled.

We have defined a serializability condition that scales well with increasing numbers of transactions and data items. However, this has come at a price as we have added the assumption that transactions access the same set of data items, albeit in different orders. In fact, there are many applications where this assumption is realistic. For example, people booking meals at restaurants over mobile phones. Some may book the main course first, then maybe dessert, then starters, and finally tea or coffee. Others may choose to book in a different order.

The availability of one course may influence the choice of another course and serializability of the booking transactions for the whole meals would be the appropriate correctness condition. Furthermore, we have investigated other different assumptions on the transactions model, where there is an order on the accessed set of data items and transactions may access different subsets of data items, that result in a similar serializability condition that only needs to check for cyclicity between pairs of transactions and would have a similar efficient encoding into *CTL*.

REFERENCES

[1] R.Alshorman and W.Hussak, *Multi-step transactions specification and verification in a mobile database community*, in *3rd IEEE International Conference on Information Technologies: from Theory to Applications, IEEE, ICTTA 08*, Damacus, Syria, IEEE Computer Society Press, 2008, pp. 1407-1412.

[2] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, *NuSMV: a new symbolic model verifier*, Lecture Notes in Computer Science 1633 (1999), pp. 495-499.

[3] E. Clarke, O. Grumberg and D. Peled, *Model checking*, MIT Press, 1999.

[4] M.P. Fle and G. Roucairol, *On serializability of iterated transactions*, *Proc, 1st ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, 1982, pp. 194-200.

[5] W. Hussak, *Serializable histories in Quantified Propositional Temporal Logic*, International Journal of Computer Mathematics, vol. 81, issue 10 (2004), pp. 1203-1211.

[6] W. Hussak, *Specifying strict serializability of iterated transactions in Propositional Temporal Logic*, International Journal of Computer Science, vol. 2, issue 2 (2007), pp. 150-156 (at `www.waset.org/ijcs`)

[7] W.Hussak, *The serializability problem for a temporal logic of transaction queries*, Journal of Applied Non-Classical Logics, vol. 18, issue 1 (2008), pp. 67-78.

[8] C.H. Papadimitriou, *The serializability of concurrent database updates*, Journal of the ACM, vol. 26 (1979), pp. 631-653.

[9] C.H. Papadimitriou, *The Theory of Database Concurrency Control*, Computer Science Press, Pockville, Maryland, 1986.

[10] D. Peled and A. Pnueli, *Proving partial order properties*, Theoretical Computer Science, vol. 126 (1994), pp. 143-182.

[11] D.Peled, S.Katz, and A.Pnueli. *Specifying and proving serializability in temporal logic* in *Proceedings LICS 1991*, IEEE Computer Society Press, 1991, pp. 232-245.