I.J. Information Technology and Computer Science, **2015, 05, 10-17**
Published Online April 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.05.02

Temporal Logics Specifications for Debit and Credit Transactions

Rafat M. Alshorman
Zarqa University/Computer science, Zarqa, Jordan
Email: rafat_sh@zu.edu.jo

***Abstract*—** Recently, with the emergence of mobile technology and mobile banking, debit and credit transactions have been the most common transactions that are widely spreading, using such technologies. In this research, we specify the concurrent debit and credit transactions in temporal logics such as CTL (Computational Tree Logic) and LTL (Linear-Time Temporal Logic). These specifications describe the infinite histories that may be produced by the iterations of such concurrent transactions infinitely many times. We represent the infinite histories as a model of temporal logics formulae. Then, model checkers, such as NuSMV or SPIN, can carry out exhaustive checks of the correctness of the concurrent debit and credit transactions. Moreover, in this paper, we presume that the serializability condition is too strict. Therefore, a relaxed condition has been suggested to keep the database consistent. Moreover, the relaxed condition is easier to encode into temporal logics formulae.

***Index Terms*—** Debit And Credit Transactions, Temporal Logics Specifications, Model Checking, Serializability of Transactions.

I. Introduction

In recent times, temporal logic stands out as one of the tools that is useful to specify and reason about concurrent and reactive systems because it provides a natural way to describe the temporal behavior of these kinds of systems [1]. It is possible to represent the systems and their properties by using temporal logics formulae. Also, we can express the implementations and specifications of the system as two formulae written using temporal logics, and then, verify whether the implementations imply the specifications. Modern operating systems and most of DBMS's extensively make use of concurrent algorithms [2], [3]. Hence, the correctness of these algorithms is very important to achieve system reliability. Now, the wide use of mobile and banking technologies has led to a huge number of concurrent users, may be, processing their database transactions simultaneously. In this case, infinite histories will be produced. The importance of representing such infinite histories has been considered [4], [5] and [6]. Usually, database techniques deal with a finite number of transactions concurrently executing [7] and [8].

Our research issue, in this paper, is to specify an infinite history of the debit and credit transactions in term of serializability, as a correctness criterion, using temporal logics formulae. The availability of model checkers gives importance to the temporal logics specifications. In this context, model checkers can carry out exhaustive checks for a correctness criterion of concurrent debit and credit transactions automatically with no need to the expertise in carrying out the verification [9] and [10].

Some researchers, in general, have taken into their accounts representing infinite histories in temporal logics [11] and [12]. And, they presumed that the serailizabiity is the correctness condition. In this research, we will introduce a computationally efficient condition of serializability that can be used to specify the correctness of concurrent transactions in temporal logics such as CTL and LTL. The serializability condition is relaxed in a way that keeps database in a consistent state. This condition is based on the nature of debit and credit transactions.

This paper is organized as follows. In Section II, we shall discuss the debit and credit transactions, conflict serializability condition and the relaxed condition of serializability. The syntaxes and the semantics of LTL and CTL are introduced in Section III . In Section IV, the properties of transition structure for read and write operations and their interpretations on LTL and CTL paths are depicted. Furthermore, The encoding of debit and credit transactions into LTL and CTL and the relaxed serializability condition are also given in Section IV. The conclusions are drawn in Section V.

II. Debit and Credit Transactions Model

A. Debit and Credit Transactions Model

In general, transaction is a collection of one or more operations on one or more databases. Formally as in [9], [4], [11] and [12], a transaction is a sequence of read/write operations partially ordered such that:

A transaction T_i is a partial order with ordering relation $<_i$, such that if $r_i(x), w_i(x) \in T_i$ then either $r_i(x) <_i w_i(x)$ or $w_i(x) <_i r_i(x)$ $\forall x. x \in D = \{x_1, x_2, \ldots x_m\}$. In this paper, we shall denote to the set of data items that are accessed by all transactions by D.

Definition 1:

A debit or credit transaction T_i, accesses a set of data items $D_i = \{x_1, x_2, \ldots, x_k\} \subseteq D$, is a sequence of (totally ordered) of read and write operations, where

every read operation $r_i(x)$ *precedes write operation* $w_i(x), \forall x.x \in D_i$, *such that*

$$T_i = r_i(x_1)w_i(x_1)\ldots r_i(x_k)w_i(x_k).$$

As in [11],[9] and [13], a set of debit and credit transactions is denoted by $T = \{T_i : i = 1,2,..\}$. A history h is an interleaving sequence of read and write operations belonging to different transactions in T. Hence, a transaction $T_i \in T$ participating, in a history h, is a subsequence of operations where every read and write operations occurring in a history h in the same order as they do in T_i. We shall denote to the operation o_i (where o_i is a read or write operation in a transaction T_i) occurs in a history h before operation $o_{i'}$ by $o_i <_h o_{i'}$. In this paper, we assume that history h is considered to be serializable or correct (preserve the database in a consistence state) if it is equivalent to a serial execution of all transactions in T [14]. We formally define that two histories are equivalent as follows:

Definition 2:

Histories h_1 *and* h_2 *of* $T = \{ T_i : i = 1,2,..\}$ *are equivalent, written as* $h_1 \sim h_2$, *iff for all* $i_1, i_2 \geq 1, i_1 \neq i_2$, *and for all* $x \in D$,

1) If $r_{i_1}(x) <_{h_1} w_{i_2}(x)$, *then* $r_{i_1}(x) <_{h_2} w_{i_2}(x)$,

2) If $w_{i_1}(x) <_{h_1} w_{i_2}(x)$, *then* $w_{i_1}(x) <_{h_2} w_{i_2}(x)$ *and*

3) If $w_{i_1}(x) <_{h_1} r_{i_2}(x)$, *then* $w_{i_1}(x) <_{h_2} r_{i_2}(x)$.

We say that the history h is serializable if h is equivalent to a serial history h_S, as in the next definition.

Definition 3:

A history h *of* $T = \{ T_i : i = 1,2,..\}$ *is serializable iff there is a serial history* h_S *of* T *of the form, for each* $i = 1,2,..$,

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

such that $h \sim h_S$.

B. Conflict graph and serilizability

Conflict graph is a directed graph that is built and used to test whether a history h, of the concurrent transactions, is serializable, and subsequently is a correct history. We consider that the history h is serializable if there is no cycle in the corresponding conflict graph. The importance of this graph is that the test of serializability can be done in a polynomial time [14]. We shall consider that two operations are conflicting, if belonging to different transactions, accessing the same data item and one of them is a write operation. Next, we shall define how we can build a conflict graph of concurrent transactions participating in a history h.

Definition 4:

For each history h, *there is a directed graph* $CG(h)$ *called the conflict graph of* h. *This graph has the transactions of* h *as its nodes, and contains an arc* (T_{i_1}, T_{i_2}), *where* T_{i_1} *and* T_{i_2} *are distinct transactions of* h, *whenever there is a operation of* T_{i_1} *which conflicts with a subsequent (in* h *) operation of* T_{i_2}.

C. Serilizability of Debit and Credit Transactions

Usually, bank customers are interacting with bank database by invoking debit and credit transactions. Debit and credit transactions are representing the deposit and withdrawal to and from current balance of a bank account [15]. So, to understand the serializabiity of debit and credit transactions that are concurrently executing in a database, we shall give the following example:

Suppose that we have two data items x and y which are representing two bank accounts in a bank database, two transactions such that:

1) $T_1 : r_1(x); x = x - 100; w_1(x); r_1(y); y = y + 100; w_1(y)$

2) $T_2 : r_2(y); y = y - 200; w_2(y); r_2(x); x = x + 200; w_2(x)$

and assume that the concurrent execution of the transactions as follows:

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

Now, suppose the initial value of x is 1000 ($x = 1000$) and the initial value of y is 500 ($y = 500$). After execution above the history h, the final values of x and y are 1100 and 400, respectively. But, the serializable execution of the two transactions T_1 and T_2 is such that:

$$h_s = r_1(x)w_1(x)r_1(y)w_1(y)r_2(y)w_2(y)r_2(x)w_2(x).$$

Suppose that we have that the same initial values for x and y ($x = 1000, y = 500$), then the final values of x and y, after execution of the history h_s , are 1100 and 400, respectively. This means that the final values of the concurrent transactions in the history h is correct.

But, according to the ***Definition 2*** and ***Definition 3***, h is not seilaizable because it is not equivalent to the serial history h_s (h - h_s) and does not leave the database in a consistent state. Moreover, if we build the conflict graph that is corresponding to the history h as in Fig. 1, then we notice that the graph contains a cycle. This means that the history h is not serializable and subsequently it is not a correct history.

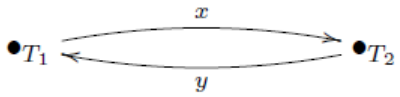


Fig. 1. Conflict graph of the history h.

Now, the above demonstration shows that the history h is not serializable but, at the same time, it is correct. The reason is that the addition and subtraction operations that are applied on debit and credit transactions are commutative and can be applied in any order [15]. This means that the condition of serializability in ***Definition 2*** and ***Definition 3*** is too restrictive. So, the relaxed condition of serializability of debit and credit transactions is defined formally as follows:

Definition 5:

A history hr *of debit and credit transactions* $T=\{T_i : i=1,2,..\}$ *is serializable iff, for any transaction* $T_i \in T$ *and data item* $x \in D$, *the read and write (* $r_i(x)$ *and* $w_i(x)$*) are occurring in the history* hr *without interleaving with any other operation(s) from different transactions* $T_j \in T$ *of the same data item* x. *This will be of the form, for each* $i = 1,2,..$

$$hr = \ldots \underbrace{r_i(x)\ldots w_i(x)}_{\text{no } r_j(x) \text{ or } w_j(x)}\ldots$$

To demonstrate the above definition, consider the transactions that are in the above example such that

1) $T_1 : r_1(x); x = x-100; w_1(x); r_1(y); y = y+100; w_1(y)$

2) $T_2 : r_2(y); y = y-200; w_2(y); r_2(x); x = x+200; w_2(x)$

and the following history

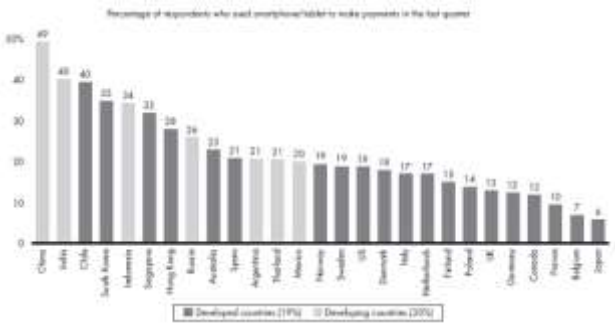
$$h_d = r_1(x)r_2(y)w_2(y)w_1(x)r_1(y)w_1(y)r_2(x)w_2(x).$$

Now, suppose that we have that the same initial values for x and y ($x=1000, y=500$) as in the above example, then the final values of x and y, after execution of the history h_d, are 1100 and 400, respectively. This means that the final values of the concurrent transactions in the history h_d are correct. Moreover, the ***Definition 5*** allows the operations from different transactions which are accessing different data items to be interleaved. This will relax the serializability condition in ***Definition 2*** and ***Definition 3*** to a new one which can be encoded into temporal logics in an easier way as we shall see later in this paper.

D. Infinite History of Debit and Credit Transactions

For the last decade, most people around the world have had smart mobile phones. Accordingly, a huge number of people access the Internet for shopping. Bank transactions involve deposit and withdraw to/form bank accounts. These are called debit and credit transactions. In 2015, the expectations say that over 900 million people are expected to transact $1 trillion in the global mobile market [16]. So, we can expect that the number of debit and credit transactions is huge and the transactions are non-stopping. This means that millions of people are constantly depositing and withdrawing to/from bank accounts. Also, the statistics show that the use of mobile transactions for debit and credit in the developing countries has excessively increased, see Fig. 2. Such situation will produce infinite histories of debit and credit transactions.

Most database management systems consider that the histories are finite but such applications signify the need to deal with infinite histories [17]. One of the most techniques that can deal with modeling of infinite and finite behavior is temporal logics [18]. These histories will be encoded in temporal logics formulae as we will see in the next sections.



Source: Bain/Research Now and Bain/ GMI NPS Surveys, 2013.

Fig. 2 Mobile Payment in 2013.

III. Temporal Logics

In this section, we will introduce two famous types of temporal logics: Linear-Time Temporal Logic (LTL) and Computational Tree Logic (CTL).

A. LTL Syntax and Semantics

LTL is a logic that can be used to specify infinite histories composed of n transactions repeating infinitely many times. The compilation of all the iterations of the n transactions gives an infinite number of transactions $T=\{T_i : i=1,2,..\}$. The reason for using LTL as a

specification language is that the LTL formulae can be interpreted over infinite sequence of states which are useful for the histories that are produced in this context [19], [20]. Furthermore, LTL is accepted as a specification language in modern model checkers such as NuSMV.

B. Syntax of LTL

The alphabet of LTL consists of a set of propositions symbols $p_i, i = 0,1,2,\ldots$, read/write step propositional symbols $r_i(x_j), w_i(x_j)$, where $i \geq 1$ and $1 \leq j \leq |D|$, boolean operations $\neg, \vee, \wedge, \bullet, \perp$, and temporal operators $\mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U}$. Formulae in LTL are of the form:

$$\phi ::= p_i \mid r_i(x_j) \mid w_i(x_j) \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \mathbf{X}\phi \mid \mathbf{F}\phi \mid \mathbf{G}\phi \mid \phi_1 \mathbf{U} \phi_2.$$

The symbols $\perp$ and $\bullet$ denote the truth values false and true respectively and the abbreviations $\Rightarrow$ and $\Leftrightarrow$ will denote usual implication and equivalency in logic, respectively.

C. Semantics of LTL

An interpretation for LTL, $I(s_i)$, at a given state $s_i \in S$, where S is a set of states, assigns truth values $p_j^{I(s_i)}$, $r_i(x_j)^{I(s_i)}$ and $w_i(x_j)^{I(s_i)}$ ($\in \{\perp, \bullet\}$) to propositional symbol p_j, $r_i(x_j)$ and $w_i(x_j)$, respectively. The model M, of the system to be specified, is represented by a structure of transition system called kripke structure see [4]. The *semantics* of a LTL formula ϕ is interpreted by the truth relation $M, s_i \text{ ‘ } \phi$ which means that ϕ holds at state s_i in the model structure M. If ϕ is a path formula, $M, \pi \text{ ‘ } \phi$ means that ϕ holds along path π in the Kripke structure M. The relation ‘ is defined as follows:

$M, s_i \text{ ‘ } p_j$ iff $p_j^{I(s_i)} = \bullet$

$M, s_i \text{ ‘ } r_i(x_j)$ iff $r_i(x_j)^{I(s_i)} = \bullet$

$M, s_i \text{ ‘ } w_i(x_j)$ iff $w_i(x_j)^{I(s_i)} = \bullet$

$M, s_i \text{ ‘ } \neg\phi$ iff $M, s_i \text{ —} \phi$

$M, s_i \text{ ‘ } \phi_1 \vee \phi_2$ iff $M, s_i \text{ ‘ } \phi_1$ or $M, s_i \text{ ‘ } \phi_2$

$M, s_i \text{ ‘ } \phi_1 \wedge \phi_2$ iff $M, s_i \text{ ‘ } \phi_1$ and $M, s_i \text{ ‘ } \phi_2$

$M, s_i \text{ ‘ } \mathbf{X}\phi$ iff $M, s_{i+1} \text{ ‘ } \phi$

$M, s_i \text{ ‘ } \mathbf{F}\phi$ iff there exists $k \geq i$ such that $M, s_k \text{ ‘ } \phi$

$M, s_i \text{ ‘ } \mathbf{G}\phi$ iff for all $k \geq i$ such that $M, s_k \text{ ‘ } \phi$

$M, s_i \text{ ‘ } \phi_1 \mathbf{U} \phi_2$ iff there exists $c \geq i$, $M, s_c \text{ ‘ } \phi_2$ and, for all $i \leq b < c, M, s_b \text{ ‘ } \phi_1$.

D. CTL syntax and semantics

Actually, LTL and CTL formulae are different in their interpretations. Therefore, some formulae in LTL cannot be specified in the CTL formulae and vice versa. LTL formula is considering each path isolated. Hence, if each individual path holds the path formula, then LTL formula is true. But, To interpret a CTL formula, we consider the alternative possibilities for each state in a path.

E. CTL syntax

As in Subsection *B*, the alphabet of CTL consists of a set of propositions symbol $p_0, p_1, \ldots$, read/write step propositional symbols $r_i(x_j), w_i(x_j)$ $(1 \geq 1, 1 \leq j \leq |D|)$, boolean operations $\neg, \vee, \wedge, \bullet, \perp$, quantifiers $\mathbf{E}$, $\mathbf{A}$, temporal operators $\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{U}$. Formulae in CTL are generated by:

$$\phi ::= p_i \mid r_i(x_j) \mid w_i(x_j) \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \mathbf{AX}\phi \mid \mathbf{EX}\phi \mid \mathbf{AF}\phi \mid \mathbf{EF}\phi \mid \mathbf{AG}\phi \mid \mathbf{EG}\phi \mid \mathbf{A}[\phi_1 \mathbf{U} \phi_2] \mid \mathbf{E}[\phi_1 \mathbf{U} \phi_2].$$

In this logic, $r_i(x_j)$ and $w_i(x_j)$ are propositions but not predicates. The symbols $\perp$, $\bullet$, $\Rightarrow$ and $\Leftrightarrow$ have the same logical meaning as in Subsection *B*.

F. Semantics of CTL

In this subsection, we shall use the same interpretation function $I(s_i)$ with the same meaning which has been introduced in Subsection *C*. in addition, the set of paths starting in a state s_i is denoted $Paths(s_i)$ with $Paths(s_i) \neq \{\}$. Therefore, the relation ‘ is defined inductively as follows:

$M, s_i \text{ ‘ } p_j$ iff $p_j^{I(s_i)} = ?$.

$M, s_i \text{ ‘ } r_i(x_j)$ iff $r_i(x_j)^{I(s_i)} = \bullet$.

$M, s_i \text{ ‘ } w_i(x_j)$ iff $w_i(x_j)^{I(s_i)} = \bullet$.

$M, s_i \text{ ‘ } \neg\phi$ iff $M, s_i \text{ —} \phi$.

$M, s_i \text{ ‘ } \phi_1 \vee \phi_2$ iff $M, s_i \text{ ‘ } \phi_1$ or $M, s_i \text{ ‘ } \phi_2$.

$M, s_i \text{ ‘ } \phi_1 \wedge \phi_2$ iff $M, s_i \text{ ‘ } \phi_1$ and $M, s_i \text{ ‘ } \phi_2$.

$M, s_i \text{ ‘ } \mathbf{AX}\phi$ iff, for all $\pi \in Paths(s_i), M, s_{i+1} \text{ ‘ } \phi$

$M, s_i \text{ ‘ } \mathbf{EX}\phi$ iff there exists $\pi \in Paths(s_i)$ such that $M, s_{i+1} \text{ ‘ } \phi$.

$M, s_i \text{ ‘ } \mathbf{AF}\phi$ iff, for all $\pi \in Paths(s_i)$, there exists $b \geq i$ such that $M, s_b \text{ ‘ } \phi$.

$M, s_i \text{ ‘ } \mathbf{EF}\phi$ iff there exists $\pi \in Paths(s_i)$ and $b \geq i$ such that $M, s_b \text{ ‘ } \phi$.

$M, s_i \; \text{‘} \; \mathbf{AG}\phi$ iff , for all $\pi \in Paths(s_a)$, and, for all, $b \geq i, M, s_b \; \text{‘} \; \phi$.

$M, s_i \; \text{‘} \; \mathbf{EG}\phi$ iff there exists $\pi \in Paths(s_i)$ such that, for all $b \geq i, M, s_b \; \text{‘} \; \phi$.

$M, s_i \; \text{‘} \; \mathbf{A}[\phi_1 \mathbf{U} \phi_2]$ iff, for all $\pi \in Paths(s_i)$, there is some $c \geq i$ such that $M, s_c \; \text{‘} \; \phi_2$ and, for all $a \leq b < c, M, s_b \; \text{‘} \; \phi_1$.

$M, s_i \; \text{‘} \; \mathbf{E}[\phi_1 \mathbf{U} \phi_2]$ iff there exists $\pi \in Paths(s_i)$ such that, for some $c \geq i, M, s_c \; \text{‘} \; \phi_2$ and, for all $i \leq b < c, M, s_b \; \text{‘} \; \phi_1$.

IV. Encoding Debit and Credit transactions into LTL and CTL

In this section, we shall give transition structure that has, at each state, a set of propositions which are either true or false. Therefore, as in [4], the set of propositions for each debit and credit transaction should satisfy the following properties:

(*C*1) *Write implies read*

A transaction T_i can only be written to x_j if it has read x_j, i.e. if $w_i(x_j)$ executes, then $r_i(x_j)$ must have been executed before.

(*C*2) *Read / write step proposition remains true until the transaction ends*

If a read/write step has taken place, the corresponding proposition remains true until the transaction ends, i.e. $r_i(x_j) / w_i(x_j)$ is true, remains true until all operations belonging to the same transactions become true.

(*C*3) *At most one step occurs at each successive state*

No two or more distinct steps can be false in a state, and then become true in a next state.

(*C*4) *Each read operation $r_i(x_j)$ should precede a write operation $w_i(x_j)$ without existence of any other read / write operation separate them.*

This property emphasizes that the transaction of the form

$$T_i = r_i(x_1) w_i(x_1) \ldots r_i(x_k) w_i(x_k)$$

as in ***Definition 1***.

The *semantics* of the temporal formula (in CTL or LTL) ϕ is given by a truth relation $M, s_i \; \text{‘} \; \phi$, where M is a structure that satisfies the additional conditions (C1)-(C4). Therefore, given a state s_i, and a path $\pi \in Paths(s_i)$, there corresponds a sequence of read and write step propositions that become true in $s_i, s_{i+1}, \ldots$. In this way, π yields a history of infinitely many occurrences (iterations) of the transactions $T_1, \ldots, T_n$ which are composing $T = \{T_i : i = 1, 2, ..\}$.

We illustrate this correspondence between paths and histories by the following example:

Assume that we have the set of data items $D = \{x, y\}$ and the debit and credit transactions are

$$T_1 = r_1(x) w_1(x) r_1(y) w_1(y) \text{ and}$$

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

The truth and the falsity for each read and write step propositions are given for successive states, and the top of each column, in Fig. 3, represents the unique proposition that becomes true in that state. The corresponding history h is:

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

$r_1(x)$	$r_2(y)$	$w_2(y)$	$w_1(x)$	$r_1(y)$	$w_1(y)$	$r_2(x)$	$w_2(x)$
s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
•	•	•	•	•	•	•	•
$r_1(x)$	$r_1(x)$	$r_1(x)$	$r_1(x)$	$r_1(x)$	$r_1(x)$	$r_1(x)$	$r_1(x)$
$\neg w_1(x)$	$\neg w_1(x)$	$\neg w_1(x)$	$w_1(x)$	$w_1(x)$	$w_1(x)$	$w_1(x)$	$w_1(x)$
$\neg r_1(y)$	$\neg r_1(y)$	$\neg r_1(y)$	$\neg r_1(y)$	$r_1(y)$	$r_1(y)$	$r_1(y)$	$r_1(y)$
$\neg w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$	$\neg w_1(y)$	$w_1(y)$	$w_1(y)$	$w_1(y)$
$\neg r_2(y)$	$r_2(y)$	$r_2(y)$	$r_2(y)$	$r_2(y)$	$r_2(y)$	$r_2(y)$	$r_2(y)$
$\neg w_2(y)$	$\neg w_2(y)$	$w_2(y)$	$w_2(y)$	$w_2(y)$	$w_2(y)$	$w_2(y)$	$w_2(y)$
$\neg r_2(x)$	$\neg r_2(x)$	$\neg r_2(x)$	$\neg r_2(x)$	$\neg r_2(x)$	$\neg r_2(x)$	$r_2(x)$	$r_2(x)$
$\neg w_2(x)$	$\neg w_2(x)$	$\neg w_2(x)$	$\neg w_2(x)$	$\neg w_2(x)$	$\neg w_2(x)$	$\neg w_2(x)$	$w_2(x)$
					$\ldots \infty$	$\ldots \infty$	$\ldots \infty$

$r_2(y)$	$r_2(y)$	$r_2(y)$
$w_2(y)$	$w_2(y)$	$w_2(y)$
$\neg r_2(x)$	$r_2(x)$	$r_2(x)$
$\neg w_2(x)$	$\neg w_2(x)$	$w_2(x)$

Fig. 3. The correspondence between path and history

A. CTL and LTL Specifications

Now, the transactions model properties (or conditions) (C1)-(C4) can be encoded as follows

(*C*1) *Write implies read*

A transaction T_i can only be written to x_j if it has read x_j, i.e. if $w_i(x_j)$ executes, then $r_i(x_j)$ must have been executed before.

1. CTL specification is:

$$\sigma_1 = \bigwedge_{1\le i\le n}\bigwedge_{1\le j\le m}\mathbf{AG}(w_i(x_j)\Rightarrow r_i(x_j)) \quad (1)$$

2. LTL specification is:

$$\sigma_1' = \bigwedge_{1\le i\le n}\bigwedge_{1\le j\le m}\mathbf{G}(w_i(x_j)\Rightarrow r_i(x_j)) \quad (2)$$

We shall add an extra proposition called end_i to indicate that the occurrence of T_i ends. This can be specified in CTL as follows

$$\sigma_0 = \bigwedge_{1\le i\le n}\mathbf{AG}(end_i \Leftrightarrow \bigwedge_{1\le j\le m}(r_i(x_j)\wedge w_i(x_j))) \quad (3)$$

also, σ_0 can be substituted in LTL by

$$\sigma_0' = \bigwedge_{1\le i\le n}\mathbf{G}(end_i \Leftrightarrow \bigwedge_{1\le j\le m}(r_i(x_j)\wedge w_i(x_j))) \quad (4)$$

(*C*2) *Read / write step proposition remains true until the transaction ends*

If a read/write step has taken place, the corresponding proposition remains true until the transaction ends, i.e. $r_i(x_j)/w_i(x_j)$ is true, remains true until all operations belonging to the same transactions become true.

1. CTL specification is:

$$\sigma_2 = \bigwedge_{\substack{1\le i\le n\\1\le j\le m}}\mathbf{AG}((r_i(x_j)\wedge\neg end_i \Rightarrow \mathbf{AX}r_i(x_j)) \wedge (w_i(x_j)\wedge\neg end_i \Rightarrow \mathbf{AX}w_i(x_j))) \quad (5)$$

2. LTL specification is:

$$\sigma_2' = \bigwedge_{\substack{1\le i\le n\\1\le j\le m}}\mathbf{G}((r_i(x_j)\wedge\neg end_i \Rightarrow \mathbf{X}r_i(x_j)) \wedge (w_i(x_j)\wedge\neg end_i \Rightarrow \mathbf{X}w_i(x_j))) \quad (6)$$

(*C*3) *At most one step occurs at each successive state*

No two or more distinct steps can be false in a state, and then become true in a next state.

1. CTL specification is:

$$\begin{aligned}\sigma_3 = \bigwedge_{\substack{1\le i,i'\le n\\1\le j,j'\le m\\ i\ne i' \text{ or } j\ne j'}}\mathbf{AG}[\neg((\neg r_i(x_j)\wedge\neg r_{i'}(x_{j'}))&\\ \wedge\mathbf{EX}(r_i(x_j)\wedge r_{i'}(x_{j'})))\wedge&\\ \neg((\neg r_i(x_j)\wedge\neg w_{i'}(x_{j'}))\wedge&\\ \mathbf{EX}(r_i(x_i)\wedge w_{i'}(x_{j'})))\wedge&\\ \neg((\neg w_i(x_j)\wedge\neg w_{i'}(x_{j'}))\wedge&\\ \mathbf{EX}(w_i(x_j)\wedge w_{i'}(x_{j'})))].\end{aligned} \quad (7)$$

2. LTL specification is:

In the CTL specification in (7) we use the operators $AGEX$. Likewise, there is no LTL specification that is equivalent to the CTL specification. So, we can say that, there are properties that can be expressed in CTL but cannot be expressed in LTL and vice versa.

(*C*4) *Each read operation $r_i(x_j)$ should precede a write operation $w_i(x_j)$ without existence of any other read / write operation separate them.*

1. CTL specification is:

$$\sigma_4 = \bigwedge_{1\le i\le n}\bigwedge_{1\le j\ne j'\le m}\neg\mathbf{EF}(r_i(x_j)\wedge r_i(x_{j'})\wedge\neg w_i(x_j)\wedge\neg w_i(x_{j'})) \quad (8)$$

2. LTL specification is:

$$\sigma_4' = \bigwedge_{1\le i\le n}\bigwedge_{1\le j\ne j'\le m}\neg\mathbf{F}(r_i(x_j)\wedge r_i(x_{j'})\wedge\neg w_i(x_j)\wedge\neg w_i(x_{j'})). \quad (9)$$

Now, to make sure that all read and write operations do not become true after the transaction T_i ends, we shall add the following specification:

$$\sigma_5 = \bigwedge_{1\le i\le n}\mathbf{AG}(end_i \Rightarrow \mathbf{AX}(\bigwedge_{1\le j\le m}(\neg r_i(x_j)\wedge\neg w_i(x_j))\wedge\neg end_i)). \quad (10)$$

or

$$\sigma_5' = \bigwedge_{1\le i\le n}\mathbf{G}(end_i \Rightarrow \mathbf{X}(\bigwedge_{1\le j\le m}(\neg r_i(x_j)\wedge\neg w_i(x_j))\wedge\neg end_i)). \quad (11)$$

We denote by σ_{dct} (as in (12)) the specification that the transition structure of the histories (of debit and credit transactions) should satisfy. i.e

$$\sigma_{dct} = (\sigma_0 \vee \sigma_0') \wedge (\sigma_1 \vee \sigma_1') \wedge (\sigma_2 \vee \sigma_2') \wedge \sigma_3 \wedge (\sigma_4 \vee \sigma_4') \wedge (\sigma_5 \vee \sigma_5'). \quad (12)$$

Next, we shall encode the relaxed condition of serializability for Debit and Credit transaction (see ***Definition 5***)

$$\sigma_{rcs} = \bigwedge_{1\le i\le n} \bigwedge_{1\le j\ne i\le n} \bigwedge_{1\le k\le m} \mathbf{AG}(r_i(x_k) \wedge r_j(x_k) \Rightarrow (w_i(x_k) \vee w_j(x_k))). \quad (13)$$

B. CTL Specifications Vs LTL Specifications

In specifying the property $C3$, we have seen that this property can be specified in CTL but cannot be specified in LTL. This could lead to a question: is CTL has more expressiveness power than LTL? the answer is no. To demonstrate this, we shall give a property to the system which can be expressed in LTL but cannot be expressed in CTL. *For example*: assume that we add a new property to the model such that

Every transaction started infinitely often is ended infinitely often

$$\bigwedge_{1\le i\le n} \mathbf{GF} r_i(x_1) \Rightarrow \mathbf{GF} end_i. \quad (14)$$

This means that the transaction that becomes started infinitely often (and may become ended) must occur infinitely often. This kind of properties is called fairness property. Furthermore, if we need to ensure that every transaction is executed infinitely often such that:

$$\bigwedge_{1\le i\le n} \mathbf{FG}\ end_i. \quad (15)$$

This property can be only specified in LTL.

V. Conclusion

In this research paper, We have given an LTL and CTL specifications for debit and credit transactions. These specifications can be used to verify a scheduler uses to schedule an unlimited number of debit and credit transactions that incoming and outgoing from a bank database system. we have assumed that the serializability is the correctness criterion for concurrent debit and credit transactions executing in a transactional processing system. We have shown that the transactional system and its properties can be specified and encoded using temporal logics. Also, in this paper, we have introduced a transition structure to model the transactional system and its properties in terms of propositions. This propositions represent the read and write operations from different transactions that become true (executed) at this state. The verification part can be executed by model checkers, such as NuSMV, to test whether the database scheduler satisfies the relaxed serializability condition and its properties or not. In case of no, counterexamples to show the errors are automatically given by the model checker.

We have found that temporal logics (LTL and CTL) are suitable to specify the relaxed serilaizablity condition. Actually, we encoded the (infinite or finite) histories of the debit and credit transactions in terms of read and write propositions in the transition structure that we defined. This means that we can prove that the histories, produced by the concurrent execution of debit and credit transactions in a scheduler, are serializable if

$$\sigma_{dct} \Rightarrow \sigma_{rcs}. \quad (16)$$

is satisfied. σ_{dct} represents the temporal logic formula that specifies the behavior of the history in transition structure, that we have given above, and σ_{rcs} represents the temporal logic formula that specifies the relaxed serializability condition. This approach gives a fully automatic verification method that can overcome the disadvantages of the traditional approaches such as the user should understand in detail why the system works correctly (deductive verification), human error (mathematical proofs) and may not cover all possible system behaviors as in the simulation [21].

Acknowledgment

This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan. Moreover, We would like to thank all members of faculty of information Technology and science at Zarqa university for their support to make these results possible.

References

[1] L. Lamport, *Secefying systems, the TLA+ language and tools for hardware and software engineers*, Microsoft Research. Addison-Wesley, 2002.
[2] S. Gnesi, "Formal Specification and Verification of Complex Systems", Electronic Notes in Theoretical Computer Science Netherlands, Vol. 80, pp. 294-298, 2003.
[3] Z. Manna and A. Pnueli, "Temporal verification of reactive systems: Safety", Springer-Verlag N.Y. Inc., 1995.
[4] R. Alshorman and W. Hussak, "A CTL Specification of Serializability for Transactions Accessing Uniform Data", International Journal of Computer Science and Engineering, Vol. 3, No. 1, pp. 26-32, 2009.
[5] Skype Heartbeats Archives, http://heartbeat.skype.com/2007/08/.
[6] D. Rossi, M. Mellia and M. Meo, "Evidences Behind Skype Outage", In proceedings of the IEEE International Conference on Communication (ICC'09), Dresde, Germany, June 2009. Link: http://www.tlc-networks.polito.it/mellia/papers/Skype_outage_icc09.pdf
[7] "NuSMV v2.4 Tutorial", NuSMV website. Link: http://nusmv.fbk.eu/NuSMV/tutorial/v24/tutorial.pdf
[8] Skype Web site, http://www.skype.com
[9] R. Alshorman and W. Hussak, "A Serializability Condition for Multi-step Transactions Accessing Ordered Data",

International Journal of Computer Science, Vol. 4, No. 1, pp. 13-20, 2009.
[10] A. Cimatti, E. Clarke, F. Giunchiglia and M. Roveri, "NuSMV: a new symbolic model verifier", In proceedings of the 11th International Conference on Computer Aided Verification, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1633, pp. 495-499, 1999.
[11] W. Hussak and J.A. Keane, "Algebraic Specification of Serializability for Partitioned Transactions", International Journal of Computer Systems Science and Engineering, Vol.1, No.1, pp. 60-67, 2007.
[12] W. Hussak, "Serializable Histories in Quantified Propositional Temporal Logic", International Journal of Computer Mathematics, Vol. 81, No. 10, pp. 1203-1211, 2004.
[13] R. Alshorman and W. Hussak, "Multi-step transactions specification and verification in a mobile database community", In proceedings of 3rd IEEE International Conference on Information Technologies: from Theory to Applications, IEEE, ICTTA 08, Damacus, Syria, IEEE Computer Society Press, pp. 1407-12, 2008.
[14] C.H. Papadimitriou, "The theory of database concurrency control". Computer Science Press, Pockville, Maryland, 1986.
[15] R. Elmasri, S. Navathe, "Fundamental of Database Systems", Addison-Wesley, Fourth Edition, 2004.
[16] "Mobile Payments: Three Winning Strategies for Banks," Swift White Paper, 2012.
[17] V.C.S. Lee, K-W. Lam, S.H. Son and E.Y.M. Chan, "On transaction processing with partial validation and timestamp ordering in mobile broadcast environments", IEEE Transactions on Computers, Vol. 51, No. 10, pp. 1196-1211, 2002.
[18] Ph. Schnoebelen, the complexity of temporal logic model checking, In AiML, 2002.
[19] R. Pucella, "The finite and the infinite in temporal logic", ACM SIGACT News, Vol. 36, No. 1, pp. 86-99, 2005.
[20] K. Sen, G. Rosu and G. Agha, "Generating Optimal Linear Temporal Logic Monitors by Coinduction", In proceedings of 8th Asian Computing Science Conference (ASIAN'03), Lecture Notes in Computer Science, Springer-Verlag, Vol. 2896, pp. 260-275, 2003.
[21] S. Koussoube, *R.* Noussi and B. O.Konfe *,"* A Formal Description of Problem Frames", International Journal of Information Technology and Computer Science, Vol. 6, No. 4, pp. PP.56-65, 2014. DOI: 10.5815/ijitcs.2014.04.07

Author's Profiles

Rafat M. Alshorman is an assistant professor in the department of computer science at Zarqa University/Jordan where he has been a faculty member since 2009. Dr. Alshorman completed his Ph.D. at Loughborough University/UK and his undergraduate studies at Yarmouk University/Jordan. His research interests lie in the area of formal methods, ranging from theory to implementation, with a focus on specifying and verifying transactions in mobile environments. In recent years, he has focused on theoretical computer science such as Graph theory and Numerical analysis. He has collaborated actively with researchers in several other disciplines of computer science. Dr. Alshorman research interests are: 1. Formal methods. 2. Temporal logics. 3. Concurrent Databases. 4. Serializability of Transactions. 5. Numerical analysis.

How to cite this paper: Rafat M. Alshorman,"Temporal Logics Specifications for Debit and Credit Transactions", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.5, pp.10-17, 2015. DOI: 10.5815/ijitcs.2015.05.02

