Tạp chí Tin học và Điều khiển học, T. 20, S. 1 (2004), 263–273

A FORMAL SPECIFICATION OF THE CORRECTNESS CRITERIA FOR CONCURRENT EXECUTIONS OF A TRANSACTION SYSTEM IN REAL TIME DATABASES

DOAN VAN BAN[1], NGUYEN HUU NGU[2], HO VAN HUONG[3]

[1]*Institute of Information Technology*
[2] *Vietnam National University, Hanoi*
[3]*Governmental Cipher Department, Hanoi*

Abstract. In this paper, we present the correctness criteria for concurrent executions of a transaction system and a formal specification of the temporal consistency in Real Time Databases using Duration Calculus (DC). We also give a formal verification of some conditions for maintaining the temporal consistency of the data.

Tóm tắt. Trong bài báo, chúng tôi sử dụng logic tính toán khoảng để đặc tả hình thức các điều kiện đúng cho thực hiện song song của hệ thống giao tác trong cơ sở dữ liệu thời gian thực. Sau đó, đặc tả và kiểm chứng hình thức một số điều kiện duy trì nhất quán thời gian của dữ liệu.

1. INTRODUCTION

In the past two decades, the research in RTDBS has received a lot of attention [6,11]. It consists of two different important areas in computer science: real time systems and database systems. Similar to conventional real time systems, transactions in RTDBS are usually associated with time constraint, e.g., deadline. On the other hand, RTDBS must maintain a database for useful information, support the manipulation of database, and process transactions [11]. RTDBS are used in a wide range of applications such as avionic and space, air traffic control systems, robotics, nuclear power plants, integrated manufacturing systems, programmed stock trading systems, and network management systems.

The main goal of this paper is to formalise some aspects of RTDBS, in particular the correctness criteria for concurrent executions of a transaction system using DC. This will allow us to verify the some conditions for maintaining the temporal consistency of the data using the proof system of the DC. The results of this paper is a part in our work. We refer interesting readers to [10] for details.

The paper is organized as follows. In the next section, we give an informal abstract description of RTDBS . Section 3 introduces a review of DC. Section 4 considers temporal consistency criteria. Section 5 presents some sufficient conditions for maintaining temporal consistency.

2. PRELIMINARIES

We briefly recall in this section the main concepts of RTDBS, which will justify our formal model given in later sections. We refer to [6,10,11] for more comprehensive introduction to RTDBS.

A real time database systems can be viewed as an amalgamation of conventional database management system and real time system [6]. In RTDB, the transactions not only have to meet their deadline, but also have to use the data that are valid during their execution.

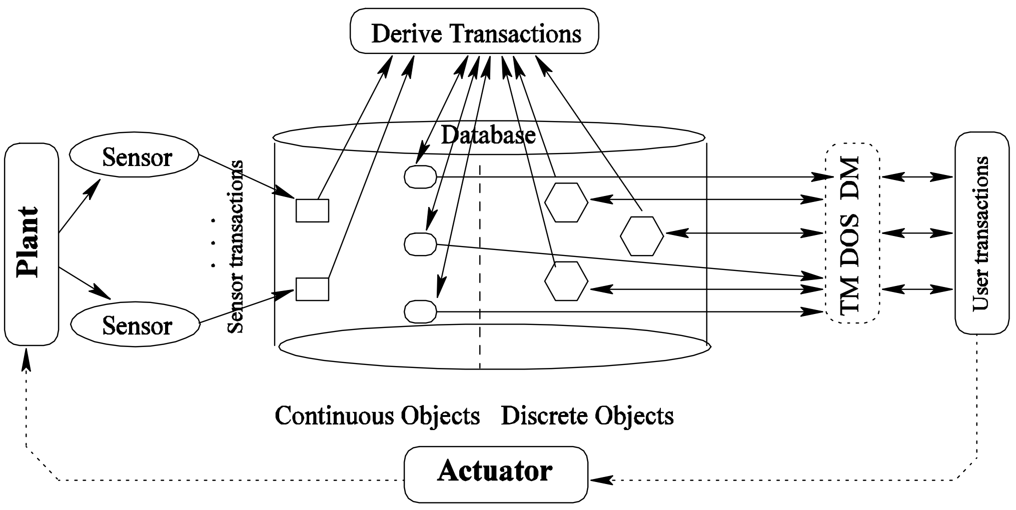

Figure 1. Real time database model

In RTDBS the set of data objects are divided into *continuous data objects* and *discrete data objects*. A graphical representation of the real time database model is shown in Figure 1. A value of continuous data object reflects the status of that object in the real world. Each value of continuous data objects may become invalid with the passage of time. Discrete data objects are static in the sense that their value do not become obsolete as time passes. Continuous data objects can be further divided into *base data objects* and *derived data objects*. The value of a base data object can be obtained directly from a sensor, while the value of the derived data objects is computed from the values of a set of base data objects.

In RTDB, the transactions have to meet their deadline as well. The deadline of a transaction may be hard deadline, firm deadline or soft deadline depending on its functional requirement. After a soft real time transaction misses its deadline, its value might decrease with time. A firm real time transaction loses its value after its deadline expires. When a hard real-time transaction misses its deadline, its value becomes negative. It means that a catastrophe might occur.

In RTDBS, beside the logical consistency as for the traditional databases, the data have to satisfy the *temporal consistency*. There are two different representations of data objects: an *external* representation (in the real world) and *internal* representation (in the DB). Two representations are temporally related with each other which are said to be temporal consistent. There are two types of temporal data consistency: absolute and relative temporal data consistency. The absolute data consistency says that the internal representation of the data is rather closed to their external representation at every moment of time. Each value of a data object is just valid for an interval of time. At any time, for any data object, there is a valid value of its. The relative temporal consistency says that a value of group of data can be used together only when they are approximately at the same age.

The absolute temporal consistency requires that transactions read temporally valid (i.e., recent enough) data objects and be committed before any of them becomes invalid. On the other hand, the relative temporal consistency requires that all the data read by a transaction must be relatively temporally consistent, i.e. having approximately the same age.

Since many users can access data in the database at the same time, we need to ensure that their concurrent execution always preserves the consistency of the DB.

In order to prevent transactions from interfering with each others, the execution of transactions must be controlled by a concurrency control protocol (CCP).

CCPs in RTDBS, apart from the fact that they have to guarantee the serializability and may have to follow priority policy as usual, they have to ensure the temporal consistency and that all transactions meet their deadline. Therefore, CCPs in RTDBS should be more complicated than CCPs in the traditional DBS, and more complicated than real-time schedulers. Some this concurrency control protocols formalised in our works [5,9,10].

In this paper, we will present the correctness criteria for concurrent executions of a transaction system and formalise the temporal consistency, we will present it more details.

3. DURATION CALCULUS

The Duration Calculus (DC) represents a logical approach to formal design of real time systems. DC is proposed by Zhou, Hoare, and Ravn, which is an extension of real arithmetic and interval temporal logic. We refer to [7] for more comprehensive introduction to Duration Calculus.

Time in DC is the set R^+ of non-negative real numbers. For $t, t' \in R^+$, $t \leq t'$, $[t, t']$ denotes the time interval from t to t'.

We assume a set E of boolean state variables. E includes the Boolean constants 0 and 1 denoting **false** and **true** respectively. State expressions, denoted by P, Q, P_1, Q_1, etc., are formed by the following rules:

1. Each state variable $P \in E$ is a state expression.
2. If P and Q are state expressions, then so are $\neg P$, $(P \wedge Q)$, $(P \vee Q)$, $(P \Rightarrow Q)$, $(P \Leftrightarrow Q)$.

A state variable P is interpreted as a function $I(P) : R^+ \rightarrow \{0, 1\}$ (a state). $I(P)(t) = 1$ means that state P is present at time instant t, and $I(P)(t) = 0$ means that state P is not present at time instant t. We assume that a state has finite variability in a finite time interval. A state expression is interpreted as a function which is defined by the interpretations for the state variables and Boolean operators.

For an arbitrary state expression P, its duration is denoted by $\int P$. Given an interpretation I of state variables and an interval, duration $\int P$ is interpreted as the accumulated length of time within the interval at which P is present. So for an arbitrary interval $[t, t']$, the interpretation $I(\int P)([t, t'])$ is defined as $\int_t^{t'} I(P)(t)dt$. Therefore, $\int 1$ always gives the length of the intervals and is denoted by ℓ. An arithmetic expression built from state durations and real constants is called a term.

We assume a set of temporal propositional letter $X, Y, \ldots$. Each temporal propositional letter is interpreted by I as truth-valued functions of time intervals.

A primitive duration formula is either a temporal propositional letter or a Boolean expression formed from terms by using the usual relational operations on the reals, such as equality $=$ and inequality $<$. A duration formula is either a primitive formula or an expression formed from other formulas by using the logical operators $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, the chop $\frown$.

A duration formula D is satisfied by an interpretation I in an interval $[t', t'']$ just when it evaluates to true for that interpretation over that time interval. This is written as

$$I,\ [t', t''] \models D,$$

where I assigns every state variable a finitely variable function from R^+ to $\{0, 1\}$, and $[t', t'']$ decides the observation window.

Given an interpretation I, the chop-formula $D_1 \frown D_2$ is true for $[t', t'']$ iff there exists a t such that $t' \leq t \leq t''$ and D_1 and D_2 are true for $[t', t]$ and $[t, t'']$ respectively.

We give now shorthands for some duration formulas which are often used. For an arbitrary state variable P, $\lceil\lceil P \rceil\rceil$ stands for $(\int P = \ell) \wedge (\ell > 0)$. This means that interval is a non-point interval and P holds almost everywhere in it. We use $\lceil\lceil\ \rceil\rceil$ to denote the predicate which is true only for point intervals.

Modalities $\Diamond$, $\Box$ are defined as: $\Diamond D \widehat{=}$ $\mathbf{true}^\frown D^\frown \mathbf{true}$, $\Box D \widehat{=} \neg\Diamond\neg D$ (we use $\widehat{=}$ as a define). This means that $\Diamond D$ is true for an interval iff D holds for some its subinterval, and $\Box D$ is true for an interval iff D holds for every its subintervals.

DC with abstract duration domain is a complete calculus, which has a powerful proof system.

4. CORRECTNESS CRITERIA OF CONCURRENT EXECUTION OF TRANSACTION SYSTEMS

In this section, we give a specification of the correctness criteria for concurrency control in RTDBS: serializability, temporal consistency criteria and timing constraints.

4.1. Serializability

As said in Section 2, the serializability is a basic criterion for the concurrency control. Now we give a characterisation of the serializability in our state model of the databases. The serializability of an execution of the transaction system says that the relation 'before' between the executions of transactions in this system execution defined by the order of the conflict operations in the execution is a partial ordering on the (infinite) set of transaction executions of the system execution. Given an execution of the transaction system. In our model, any transaction has its own period, and in each period, there is one execution of the transaction. As said before, we assume that $P_1 \leq P_2 \ \leq \cdots \leq P_n$. Since in a system execution, each transaction has the infinite number of repeated executions, and since the relation 'before' is over the set of executions of transactions which is infinite, it is not easy to describe a criterion for the relation 'before' to be acyclic by just a formula since the formula may have to capture the behaviour of the transaction system in a (potentially) infinite interval.

Fortunately, we do not have to consider the (potentially) infinite intervals. There is a nice characterisation for the relation 'before' of the transaction system to be acyclic which is about the behaviour of transaction system in an interval with the length $(n+1) * P_n$ only.

We have theorem as follows:

Theorem 1. *An execution of a transaction system modelled as above isserialisable if and only if in any time interval consisting of exactly $(n+1)$ consecutive periods of T_n, any n executions of different transactions is serialisable, i.e. the relation 'before' on them is acyclic.*

A proof this theorem be done in [10]. This theorem enable us to develop a DC formula to characterise the serialisability of an execution of a transaction system, even the number of transaction executions is infinite, and the time for the execution is also infinite.

The relation 'before' between the executions of different transactions are modelled as follows. The order between conflict operations on data object x in an interval with the length less than $a(= (n+1)P_n)$ is captured by

$$\begin{aligned}
WR_{ij}(x) &\widehat{=} (\Diamond(\lceil\lceil T_i.written(x) \rceil\rceil \wedge \lceil\lceil \neg T_j.read(x) \rceil\rceil)^\frown \ell < a^\frown \lceil\lceil T_j.read(x) \rceil\rceil) \\
RW_{ij}(x) &\widehat{=} (\Diamond(\lceil\lceil T_i.read(x) \rceil\rceil \wedge \lceil\lceil \neg T_j.written(x))^\frown \ell < a^\frown \lceil\lceil T_j.written(x) \rceil\rceil) \\
WW_{ij}(x) &\widehat{=} (\Diamond \lceil\lceil T_i.written(x) \rceil\rceil^\frown \ell < a^\frown \lceil\lceil T_j.written(x) \rceil\rceil)
\end{aligned}$$

Where, we used some state variables $T_i.written(x)$, $T_i.read(x)$, $T_i.period$, $T_i.run$, $T_i.arrived$ which be specifed in [10] for our model. With limmited space no detailed specify is included.

To express that the relation 'before' defined as above does not have a cycle longer than n, we first find an expression for its transitive closure. This is expresses by the following DC formula $\mathcal{C}_{ij}^n$ defined as:

$$\begin{aligned}
\mathcal{C}_{ij}^1 &\mathrel{\widehat{=}} (RW_{ij} \vee WR_{ij} \vee WW_{ij}) \\
\mathcal{C}_{ij}^2 &\mathrel{\widehat{=}} (\mathcal{C}_{ij}^1 \vee (\mathcal{C}_{ir}^1 \wedge \mathcal{C}_{rj}^1)) \\
&\vdots \\
\mathcal{C}_{ij}^n &\mathrel{\widehat{=}} (\mathcal{C}_{ij}^{n-1} \vee (\mathcal{C}_{ir}^{n-1} \wedge \mathcal{C}_{rj}^{n-1}))
\end{aligned}$$

Serializability Criterion

A concurrent execution of the set of transactions $\mathcal{T}$ is serializable iff it satisfies the following DC formula $SERIAL$ for any interval.

$$SERIAL \mathrel{\widehat{=}} (T_n.period^\frown \ell = n * P_n) \Rightarrow \bigwedge_{i,j \leq n, i \neq j} \neg(\mathcal{C}_{ij}^n \wedge \mathcal{C}_{ji}^n)$$

4.2. Temporal Consistency Criteria

As presented in Section 2, there are two kinds of data objects. Based on this classification, we will formalise some criteria for the transactions handling them. The set $\mathcal{O}$ of data objects in a RTDBS consists of:

Continuous data objects are related to external objects continuously changing with time. The value of a continuous data object can be obtained directly from a sensor (*base object*) or computed from the values of a set of base data objects (*derived objects*). So, the set of continuous data objects is classified into:

1 . A set of base objects X,

2 . A set of derived objects Y.

Discrete data objects are static in the sense that their values do not become obsolete as time passes. Let the set of discrete data objects Z.

For each data object $y \in Y$, the set of the data objects used to compute the value of y is denoted by $\Sigma_y, \Sigma_y \subseteq X \cup Z$.

Each transaction T_i has its own deadline D_i, a priority p_i, an execution time C_i, a period P_i, a data read set RO_i, a data write set WO_i (note that RO_i and WO_i may be empty). Our model of RTDBS is an extension of the Basic model which proposed by Ho Van Huong and Dang Van Hung. We refer interesting readers to [10] for details.

At each moment of time a continuous data object has a value represented by it's current version which is valid for some time interval. Note that at the same moment of time there may be several versions of the same continuous object in database that are valid.

The state variables to capture the behaviour of continuous data objects are as follows.

Let α be a continuous data object. For each $q \in N$ there is a state variable $validity_q(\alpha)$ to reflect the validity of q'th version for the value of α and a real state variable $value_q(\alpha)$ to reflect the value of α at time t is the q'th version. $validity_q(\alpha)$ holds at time t iff q'th version of a continuous data object α has been created (before time t) and is still valid at time t. For simplicity of the presentation, we assume that discrete data only have 0th version ($q = 0$) with the validity interval $[0, +\infty)$.

For all $\alpha \in \{X \cup Y\}$.

$$\begin{aligned}
&validity_q \in [(X \cup Y) \to Time \to \{0,1\}]\\
&validity_q(\alpha)(t) = 1 \text{ iff } t \text{ is in the valid interval of the } q\text{'th version of } \alpha\\
&value_q(\alpha) \in [\text{Time} \to \{0,1\}]\\
&value_q(\alpha)(t) = 1 \text{ iff at } t \text{ value of } \alpha \text{ is the } q\text{'th version}
\end{aligned}$$

There is a positive lower bound δ' for the valid interval (depending on the sampling periods), and each version may have only a single interval of validity. For a version q of the data object α, there is a predefined number $avi_q(\alpha)$ which is the maximal length of its validity interval. Namely, version q of α is valid for $avi_q(\alpha)$ $(\geq \delta')$ time units since the time it was created. Therefore,

$$\lceil\lceil \neg validity_q(\alpha) \rceil\rceil ^\frown \lceil\lceil validity_q(\alpha) \rceil\rceil ^\frown \lceil\lceil \neg validity_q(\alpha) \rceil\rceil \;\Rightarrow\; \ell \geq avi_q(\alpha) \tag{1}$$

$$(\lceil\lceil validity_q(\alpha) \rceil\rceil \Rightarrow \ell \leq avi_q(\alpha)) \tag{2}$$

$$\lceil\lceil validity_q(\alpha) \rceil\rceil ^\frown true \;\Rightarrow\; \lceil\lceil validity_q(\alpha) \rceil\rceil \vee \lceil\lceil validity_q(\alpha) \rceil\rceil ^\frown \lceil\lceil \neg validity_q(\alpha) \rceil\rceil \tag{3}$$

Recall that the absolute temporal consistency at a time t of a data object α means that the value of the internal representation of the data object at time t is closed to its external representation. More precisely, at time t, there is a version q of α which was borned at time $t_{(\alpha,q)}$ that is still valid, i.e. $t - t_{(\alpha,q)} \leq avi_q(\alpha)$. The absolute temporal consistency of the data in a RTDBS means that all data objects satisfy the absolute temporal consistency at any time. Since we have assumed that at any time, there should be a version q for a data object (normally, the version that was created most recently), the absolute temporal consistency is formalised simply as follows.

Absolute Temporal Consistency Criterion (for any RTDBS)

$$ACONS(\alpha, q) \;\widehat{=}\; \Box(\lceil\lceil value_q(\alpha) \rceil\rceil \Rightarrow \lceil\lceil validity_q(\alpha) \rceil\rceil)$$

Relative consistency says that data objects from some data set should be temporally correlated. Any set R of versions of continuous data objects, i.e. R is a set of pairs (α, q), is associated with a number called length of *relative validity interval* denoted by $rvi(R)$. The set R of the data read by a transaction during an execution must be relatively consistent, which means that the distance between their creation time is not more than $rvi(R)$.

The relative consistency of a set R of versions is now expressed by the following DC formula $RCONS(R)$, meaning that R is relatively consistent iff DC formula $RCONS(R)$ is true for all intervals.

$$RCONS(R) \;\widehat{=}\; \bigwedge_{(\alpha_1,q),(\alpha_2,r)\in R} \Box \left(\begin{array}{r} \lceil\lceil validity_q(\alpha_1) \wedge \neg validity_r(\alpha_2) \rceil\rceil \\ ^\frown \lceil\lceil validity_r(\alpha_2) \rceil\rceil \;\Rightarrow (\ell \leq rvi(R)) ^\frown \lceil\lceil validity_r(\alpha_2) \rceil\rceil \end{array} \right)$$

Recall that every transactions T_i is associated with a deadline D_i.

In our model, in each execution each transaction $T_i \in \mathcal{T}$ (in a period) is associated with a set of versions of continuous data objects which are read by it. A transaction in our real-time database model can commit only if

1 It meets its deadline, and

2 It reads temporally consistent sets of data, and the data it read are still valid when it commits.

As we have said earlier, for any data object α, at any time t, there is a version q for which $value_q(\alpha)$ is true. Normally, when a transaction reads α at time t, it will get the version q for which $value_q(\alpha)$. However, in some scheduler, they may give a different valid version. In order to be more general, we introduce the step function $T_i.readv$ to return the version number read by T_i for a value of data object.

$T_i.readv \in [\mathcal{O} \rightarrow \text{Time} \rightarrow N]$

$T_i.readv(\alpha)(t) = q$ iff at time t transaction T_i has performed a read operation on q'th version most recently of data object (α).

A transaction T_i can read a set of versions of data objects. Therefore, for each $i \leq n$ temporal variables $R\alpha_i$ is introduced to express that set of versions of data objects read by T_i.

$$R\alpha_i \in [Intv \rightarrow 2^{\mathcal{O}\times N}]$$

So, a transaction T_i can be committed if DC formulas DL_i, ATC_i, RTC_i is valid:

$$DL_i \mathrel{\widehat{=}} \left(\lceil\lceil T_i.period \rceil\rceil \;\Rightarrow ((\Box(\lceil\lceil T_i.arrived\rceil\rceil \Rightarrow \ell \leq D_i)) \wedge \int T_i.run = C_i) \right)$$

$$ATC_i \mathrel{\widehat{=}} \left(\begin{array}{r} \lceil\lceil T_i.period\rceil\rceil \;\Rightarrow \\ \Box \left(\lceil\lceil T_i.arrived\rceil\rceil \Rightarrow \left(\begin{array}{r} \bigwedge\limits_{(\alpha)\in RO_i, q\neq 0} \lceil\lceil T_i.read(\alpha)\rceil\rceil \wedge \\ \lceil\lceil T_i.readv(\alpha) = q\rceil\rceil \Rightarrow \lceil\lceil validity_q(\alpha)\rceil\rceil \end{array} \right) \right) \end{array} \right)$$

$$RTC_i \mathrel{\widehat{=}} \Diamond \left(\begin{array}{c} \lceil\lceil T_i.period\rceil\rceil \wedge ((\alpha, q) \in R\alpha_i, q \neq 0 \iff \\ \Diamond(\lceil\lceil T_i.read(\alpha)\rceil\rceil \wedge \lceil\lceil T_i.readv(\alpha) = q\rceil\rceil)) \end{array} \right) \Rightarrow \; RCONS(R\alpha_i)$$

Let $CM \mathrel{\widehat{=}} \bigwedge_{i\leq n} DL_i \wedge ATC_i \wedge RTC_i$.

Correctness criterion for the execution of transactions in RTDBS: an execution of set $\mathcal{T}$ of transactions is correct iff for any interval it satisfies the formula $SERIAL \wedge CM$.

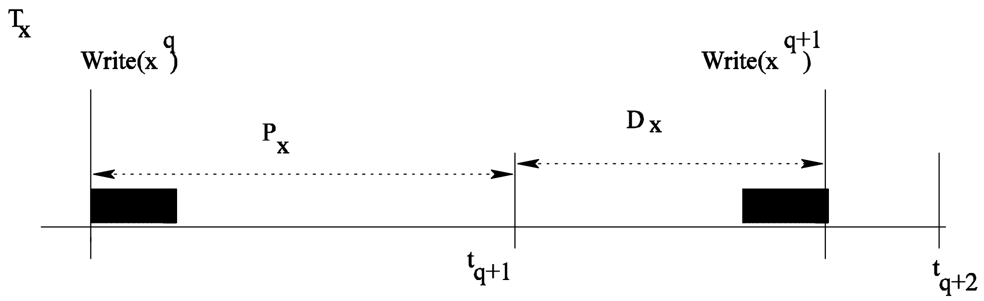

Figure 2. The worst-case update time for a sensor transaction

5. SOME SUFFICIENT CONDITIONS FOR MAINTAINING TEMPORAL CONSISTENCY

In this section, we will apply our formal model to specify and verify some conditions for maintaining the temporal consistency in RTDBS proposed in [6,11].

To get a deeper result, we will restrict ourselves to some special kinds of transactions. We classify the transactions in a RTDBS into three classes: Sensor Transactions, Derive Transactions, User Transactions. Sensor transactions update the based data objects, and Derive Transactions compute and update the derived objects. The user transactions are application programs of the users to access the database.

5.1. Sensor Transactions

A transaction of this class maintains the absolute temporal consistency of the database by writing a sampled value of an external object to the corresponding base object with a regular interval. It is a write-only transaction. It means that Sensor Transactions are responsible for maintaining the absolute temporal consistency of base objects. Let P_x be the period of sensor transaction T_x for maintaining the absolute temporal consistency of a base data object x, and D_x is the deadline of T_x. To guarantee the absolute temporal consistency of x, its period must satisfy the following condition in formula:

$$(P_x + D_x) \leq avi_q(x) \tag{4}$$

Informal justification for this fact is as follows. This is because the worst-case next update time for a base object written at the beginning of a certain period is the deadline of the next period.

If the deadline equals the period, a transaction's period must be less than or equal to half of the absolute validity interval of the related base object to maintain its absolute temporal consistency.

How to specify and verify this formally?

We have to verify that for a based object x, for all version q (each q corresponds to the execution of T_x in the qth period), $ACONS(x, q)$ is satisfied by all intervals, where:

$$ACONS(x, q) \mathrel{\widehat{=}} \Box(\lceil\lceil value_q(x)\rceil\rceil \Rightarrow \lceil\lceil validity_q(\alpha)\rceil\rceil)$$

The behaviour of the $value_q(x)$ is captured by: for all $q \neq q'$:

$$\begin{aligned}
&\lceil\lceil value_q(x)\rceil\rceil \Rightarrow \lceil\lceil \neg value_{q'}(x)\rceil\rceil\\
&\lceil\lceil value_q(x)\rceil\rceil^\frown\lceil\lceil \neg value_q(x)\rceil\rceil \Rightarrow \lceil\lceil value_q(x)\rceil\rceil^\frown\lceil\lceil value_{q+1}(x)\rceil\rceil^\frown true
\end{aligned}$$

The behavior of T_x is captured by the formula:

$$T_x.period \Rightarrow (\exists q((\lceil\lceil value_{q-1}(x)\rceil\rceil \wedge \ell \leq D_x)^\frown\lceil\lceil value_q(x)\rceil\rceil) \wedge \ell = P_x)$$

From these we can derive that

$$\begin{aligned}
T_x.period^\frown T_x.period \Rightarrow\ &\exists q((((\lceil\lceil \neg value_q(x)\rceil\rceil \wedge \ell \leq D_x)^\frown\lceil\lceil value_q(x)\rceil\rceil) \wedge \ell = P_x)\\
&^\frown(((\lceil\lceil value_q(x)\rceil\rceil \wedge \ell \leq D_x)^\frown\lceil\lceil \neg value_q(x)\rceil\rceil)) \wedge \ell = P_x)
\end{aligned}$$

This entails

$$T_x.period^\frown T_x.period \Rightarrow (\exists q((\lceil\lceil \neg value_q(x)\rceil\rceil^\frown(\lceil\lceil value_q(x)\rceil\rceil \wedge \ell \leq P_x + D_x)^\frown\lceil\lceil \neg value_q(x)\rceil\rceil))$$

From this, together with neighbourhood logics (to extend any interval satisfying $\lceil\lceil value_q(x)\rceil\rceil$ into an interval that satisfies $T_x.period^\frown T_x.period$), the absolute temporal consistency of x can be derived easily.

5.2. Derive Transactions

Transactions of this class read some data objects, compute new values of derived objects, and write them to the database. They are update transactions. As any other transaction in the system, they have to satisfy the temporal consistency ATC_i and RTC_i. For any data object $y \in Y$, there is a derive transaction T_y with the read set RO_y being Σ_y.

In the literature ([6]), the following conditions for a Derive Transaction's period in order to maintain temporal consistency of derived objects have been derived. Let Ru_i denote temporal variable to express the set of versions of data objects in Σ_y used to compute a new value of derive data objects read by T_y. P_y, P_u and D_y, D_u are period and deadline of T_y and T_u (transaction T_u is writing to a based data object u in $\sum_y$).

$$Ru_i \in [Intv \rightarrow 2^{\sum_y \times N}]$$

We assume that if $D_u = D_y$, then the priority of T_y is higher than that of T_u. Assume that the system has a single processor. The condition for maintaing the absolute temporal consistency for y and for T_y to satisfy the three criteria in the previous section is formulated as:

$$(P_u + D_u) \leq avi_q(u), \forall u \in \Sigma_y \cap X \qquad (5)$$

$$(P_y + D_y) \leq avi_q(y), \forall y \in Y \qquad (6)$$

$$D_y \leq D_u \qquad (7)$$

$$\left(\begin{array}{r} \lceil\lceil T_y.period\rceil\rceil \;\Rightarrow \\ \left(\begin{array}{r} (u,q) \in Ru_i, q \neq 0 \iff \;\; \Diamond\left(\lceil\lceil Ty.read(u)\rceil\rceil \wedge \lceil\lceil Ty.readv(u) = q\rceil\rceil\right) \\ \Rightarrow\;\; (P_u + D_u \leq rvi(Ru_i)) \end{array}\right) \end{array}\right) \qquad (8)$$

With the cases in Figure 3, we can justify the above conditions informally as follows:

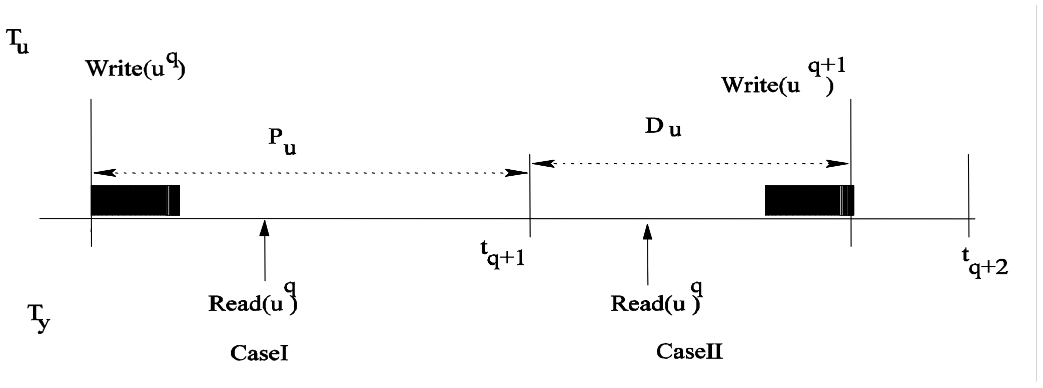

Figure 3. Maintaining Temporal Consistency

Case I: T_y read u after the current instance of T_u updates u (i.e., writes u^q). In this case, the value u^q will be valid at least until the time $t_{q+1} + D_u$ because of inequation $(P_u + D_u) \leq avi_q(u)$ and because the completion of T_y comes before that time (since $D_y \leq D_u$). Thus, the value u^q read by T_y, will be valid until T_y completes.

Case II: T_y read u before the current instance of T_u updates u (i.e., writes u^{q+1}). In this case, the value u^q read by T_y will be valid at least until the time $t_{q+1} + D_u$ from inequation

$P_u + D_u \leq avi_q(u)$. Furthermore, T_y should complete before that time (i.e., the deadline of T_u), since the priority of T_y is higher than the priority of T_u from inequation $D_y \leq D_u$. Thus, the value u^q read by T_y will be valid until the completion of T_y. Also, since the maximum temporal distance between the data object y and any data object in Σ_y is $max_{\forall u \in \Sigma_y}(P_u + D_u)$. Σ_y satisfies its relative temporal consistency requirement, as long as $P_u + D_u \leq rvi(Ru_i)$ for all u in $\sum_y \cap (X \cup Y)$.

This can be verified formally in our model will be done and omitted here.

5.3. User Transactions

A sufficient condition for a user transaction to satisfy the criteria in the previous section for user transaction T_i is as follows.

Let Rv_i denotes temporal variables to express that set of versions of data objects read by user transaction T_i. Transaction T_v is responsible for updating data objects v in the read set V of T_i. Let P_v, D_v be period and deadline of T_v.

$$Rv_i \in [Intv \to 2^{V \times N}]$$

We assume that if $D_v = D_i$, then the priority of T_i is be higher than that of Tv. The conditions to maintain the above temporal consistency requirements by a user transaction are formalised by the following DC formulas.

$$(P_v + D_v) \leq avi_q(v), \forall v \in V \cap (X \cup Y) \tag{9}$$

$$D_i \leq D_v \tag{10}$$

$$\left(\begin{array}{r} \lceil\lceil T_i.period \rceil\rceil \;\Rightarrow \\ \left(\begin{array}{l} (v,q) \in Rv_i, q \neq 0 \iff \Diamond \left(\lceil\lceil T_i.read(v) \rceil\rceil \wedge \lceil\lceil T_i.readv(v) = q \rceil\rceil \right) \\ \qquad \Rightarrow (P_v + D_v - R_i \leq rvi(Rv_i)) \end{array} \right) \end{array} \right) \tag{11}$$

We refer the readers to [6] for the justification of the correcness of this condition. The formal verification of this fact will be done and omitted here.

6. CONCLUSION

In this paper, we have presented the correctness criteria for concurrent executions of a transaction system and a formal specification of the temporal consistency in Real Time Databases using Duration Calculus. We also give a formal verification of some conditions for maintaining the temporal consistency of the data. These frameworks can be used in the future for specifying many other issues of RTDBS.

Acknowledgements. The authors would like to thank Dr. Dang Van Hung for his kind help.

REFERENCES

[1] Doan Van Ban, Ho Van Huong, Duration Calculus and Application, *Proccedings of Hanoi University of Sciences, National University of Vietnam,* Nov, 2000.

[2] Doan Van Ban, Ho Van Huong, A Formal Specification of the Read/Write Priority Ceiling Protocol in Real Time Databases, *Proccedings of National Information Technology,* Haiphong, June, 2001.

[3] Doan Van Ban, Ho Van Huong, Serializability of Two Phase Locking Concurrency Control Protocol in Real Time Database, *Journal of Computer Science and Cybernetics* **17** (3) (2001).

[4] Doan Van Ban, Nguyen Huu Ngu, Ho Van Huong, Concurrency control protocol in Real Time Databases, *Proccedings of Institute of Information Technology,* Nov, 2001.

[5] Doan Van Ban, Nguyen Huu Ngu, Ho Van Huong, Formalising Priority Ceiling Protocol with Dynamic Adjustment of Serialization Order in Real Time Databases, *Journal of Science* **19** (1) (2003) (National University, Hanoi).

[6] Azer Bestavros, Kwei-Jay Lin and Sang Hyuk Son. *Real-Time Database Systems: Issues and Applications.* Kluwer Academic Publishers, 1997.

[7] M.R. Hansen, Zhou Chaochen, Duration Calculus: Logical Foundations. *Formal Aspects of Computing* **9** (1997) 283–330.

[8] Dang Van Hung, Real-time Systems Development with Duration Calculus: an Overview, *UNU/IIST Report No. 255,* UNU/IIST, P.O. Box 3058, Macau, June, 2002.

[9] Ho Van Huong, A Formal Specification of The Abort-Oriented Concurrency Control for Real Time Database in Duration Calculus *Journal of Computer Science and Cybernetics* **16** (1) (2003).

[10] Ho Van Huong, Dang Van Hung, Modelling Real-Time Database Systems in Duration Calculus, *UNU/IIST Report No.260* , UNU/IIST, P.O. Box 3058, Macau, August, 2002.

[11] Kam-Yiu Lam, Tei-Wei Kuo, *Real-Time Database Systems: Architecture and Techniques.* Kluwer Academic Publishers, 2001.

[12] Ekaterina Pavlova, Dang Van Hung, A Formal Specification of the Concurrency Control in Real Time Database, *UNU/IIST Report No. 152,* UNU/IIST, P.O. Box 3058, Macau, January, 1999.

Received on October 6 - 2002