

# An Authorization Model for Temporal Data

Avigdor Gal  
Department of MSIS, Rutgers University  
avigal@rci.rutgers.edu

Vijayalakshmi Atluri  
Department of MSIS, Rutgers University  
atluri@andromeda.rutgers.edu

## ABSTRACT

The use of temporal data has become wide-spread in recent years, within applications such as data warehouses and spatiotemporal databases. In this paper, we extend the basic authorization model by facilitating it with the capability to express authorizations based on the temporal attributes associated with data, such as *transaction time* and *valid time*. In particular, a subject can specify authorizations based on data validity or data update time, using either absolute or relative time references. Such a specification is essential in providing access control for predictive data, or in constraining access to data based on currency considerations. We provide an expressive language for specifying such access control to temporal data, using a variation of temporal logic for specifying complex temporal constraints. We also introduce an easy-to-use access control mechanism for stream data.

## 1. INTRODUCTION

Temporal datum is a data value associated with information such as the time at which data has been *captured* and the time interval during which a data value is *valid*. Given a specific data model, temporal data can be collected into versions of objects, distinguished from one another on the basis of their temporal specifications. The use of temporal data has become widespread in recent years, within applications such as data warehouses [4] (*e.g.*, daily sales and seasonal catalogs), spatiotemporal databases [1], and electronic commerce [10] (with applications in finance and digital libraries). For example, a digital library should be able to enforce access control based on the time at which a document has been “checked-out,” (referred to as *replication time*) to allow flexible digital library policies for restricting access to a digital book after the expiry of the temporal-based authorization. A partial solution to this problem has been recently suggested by Authentica<sup>1</sup>, whose product (PageVault) enables

<sup>1</sup><http://www.authentica.com>

the user to select a specific time for a document to self deconstruct.

In this paper, we propose an authorization model for temporal data, called *Temporal Data Authorization Model* (TDAM), capable of expressing access control policies based on the temporal characteristics of data. TDAM extends existing authorization models to allow the specifications of temporal constraints on data, based on data validity, data capture time, and replication time, using either absolute or relative time references. We introduce a variation of temporal logic for specifying complex temporal constraints, which allow TDAM to express security policies such as “a subject is allowed to read financial data 15 minutes after it has been captured,” “only managers are allowed to read prospective product plans,” and “a digital book can be read for 21 days from time of download.” The ability to specify access control based on such temporal aspects were not supported before. The formulae are evaluated with respect to various temporal assignments to ensure the correctness of access control. To complement the specification language, we introduce a set of algorithms that handle access control based on temporal specifications. The algorithms use a simple interval algebra to ensure that access is allowed only during the interval in which the access granting formula is valid, and can handle situations where an access request is issued for an interval that is larger than the data validity. We facilitate *interval access request* to allow access for stream data, in which data is continuously updated. With an interval access request, a user may specify an ongoing request for data, for which our proposed algorithm continuously checks for the satisfiability of existing and new data against a given temporal formula. We provide an expressive specification language for temporal data. Also, a complementary access control mechanism is introduced, which allows a correct assessment of authorizations using temporal logic and an easy-to-use granting process for stream data.

To the best of our knowledge, no authorization model can be found in the literature that is capable of specifying access control policies based on temporal attributes of data. The only authorization model that allows temporal specification is the *Temporal Authorization Model* (TAM) proposed by Bertino et al. [2]. TAM enables temporal specifications, specification of derivation rules through which an authorization can be derived using other authorizations, and periodic authorizations [3]. TAM and TDAM are complementary models, since TAM allows specification of authorizations with a predefined time interval(s), yet does not derive them based on the temporal attributes of the data.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CCS '00, Athens, Greece.

Copyright 2000 ACM 1-58113-203-4/00/0011 ..\$5.00

The rest of the paper is organized as follows. In Section 2 we introduce the temporal data model. An authorization model, which is tightly coupled with the data model and extends existing authorization models to allow a refined access control based on the temporal characteristics of the data is presented in Section 3. Access control mechanisms are discussed in Section 4. Finally, Section 5 summarizes the contributions of the paper and offers directions for future work.

## 2. A TEMPORAL DATA MODEL

In this section we provide a temporal data model, based on existing works in the temporal database research area (e.g., [5], [6], [8], and [7]). We start with preliminary definitions of temporal concepts (Section 2.1) to be followed by the temporal data model in Section 2.2.

### 2.1 Preliminaries: Temporal concepts

A *temporal domain* is a pair  $(\mathcal{T}; \leq)$  where  $\mathcal{T}$  is a nonempty set and  $\leq$  is a total order on  $\mathcal{T}$ . Following previous works in the temporal database area, we adopt a discrete model of time [5], isomorphic to the natural numbers ( $\mathcal{T} \cong \mathbf{N}$ ). The model defines a *time point*<sup>2</sup>  $t \in \mathcal{T}$  to be a nondecomposable unit of time whose granularity is application-dependent. A *time interval* is designated as  $\bar{t} = [t_s, t_e)$ , the set of all time points  $t \in \mathcal{T}$  such that  $t_s \leq t < t_e$ . Clearly, a time point  $t \in \mathcal{T}$  can be represented as the time interval  $[t, t + 1)$ . For completeness, we define a time interval  $\bar{t} = [t_s, t_e)$  to be *empty* if  $t_s \geq t_e$ . We define  $\bar{\mathcal{T}}$ , the set of all possible time intervals, to be the *time interval domain*.

A time interval may be represented using a *symbolic time interval*, where  $t_e$ 's domain is  $\mathcal{T} \cup \{\text{now}, UC\}$ . The special purpose symbols specify time-varying constraints, as follows. *now* stands for the current time, and therefore, an interval  $[t, \text{now})$  is an ever-growing time interval starting from time point  $t$ . *UC* stands for ‘‘Until Change,’’ which reflects an upper bound set by the next relevant temporal datum.<sup>3</sup> A symbolic time interval can be mapped onto an *actual time interval*, with time points taken from the temporal domain.

Temporal database models associate one or more *temporal dimensions* with data, assigning different semantics to each dimension. For example, the temporal infrastructure document [9] advocates a bi-temporal database model, in which each datum is associated with two temporal dimensions, called valid time and transaction time. A *valid time* ( $t_v \in \bar{\mathcal{T}}_v$ ) is a time interval at which a datum is considered to be true in the modeled reality. A *transaction time* ( $t_x \in \mathcal{T}_x$ ), under a common interpretation, is a time point that designates the time in which the transaction that inserted the datum to the database was committed. Other temporal dimensions may be added, based on the application needs. For example, a digital library may attach a replication time ( $t_r \in \mathcal{T}_r$ ), reflecting the time at which an electronic copy of a book was replicated to a borrower's device.

We end this section with the syntax of a temporal logic to be used later in the paper. We use an instance of the temporal logic model suggested in [12]. We define *TC* to

<sup>2</sup>While the term *chronon* was coined as a standard term in temporal databases [8], we shall prefer the use of this more intuitive term.

<sup>3</sup>The semantics of ‘‘next relevant’’ will become clear in Section 2.2, with the introduction of discriminating temporal dimensions.

be a set of time point symbols, such as July30:1999:13:57 and 5minutes, and  $TV = \{t_x, t_s, t_e, t_r, t_{req}\}$ <sup>4</sup> be a set of temporal variables.  $TF = \{+, -, *, \div\}$  is a set of temporal function symbols, with a fixed arity of 2. Finally, we define  $R$  to be a set of relation symbols, each with a fixed arity of 1. Each relation shall correspond to a single property as will be explained shortly in Section 2.2. We use  $C$  to define a set of non-temporal constants and  $V$  to define a set of non-temporal variables.

**DEFINITION 1 (TEMPORAL TERMS).** A temporal term *trm* is defined inductively, as follows:

- If  $trm \in TC$ , then *trm* is a temporal term;
- If  $trm \in TV$ , then *trm* is a temporal term;
- If  $trm_1$  and  $trm_2$  are temporal terms, and  $f \in TF$  is a temporal function symbol, then  $f(trm_1, trm_2)$  is a temporal term.  $\square$

As examples of temporal terms, one may consider 5minutes,  $t_{req}$ , and  $+(t_x, 5minutes)$ . For the latter, we shall use the conventional writing of  $t_x + 5minutes$ .

**DEFINITION 2 (WELL-FORMED FORMULAE).** A well-formed formula (*wff*) is defined as follows:

- If  $trm_1$  and  $trm_2$  are temporal terms, then  $trm_1 = trm_2$  and  $trm_1 \preceq trm_2$  are wffs;
- If  $trm$  is a non-temporal term, and  $r \in R$  is a relation symbol, then  $r(trm)$  is a wff;
- If  $\varphi$  and  $\psi$  are wffs, then  $(\varphi \wedge \psi)$ , and  $(\neg\varphi)$  are wffs.  $\square$

As an example, one may consider the wff  $(t_x + 5minutes \preceq t_{req} \wedge t_{req} \preceq t_e + 5minutes)$ .

### 2.2 The data model

Let  $\bar{\mathcal{T}} = (\bar{\mathcal{T}}_1, \bar{\mathcal{T}}_2, \dots, \bar{\mathcal{T}}_n)$  be a sequence of  $n$  time interval domains. Using object-based notation,  $TS_{\bar{\mathcal{T}}} = \{C_1, C_2, \dots, C_m\}$  is a temporal database schema that consists of  $m$  classes. A class  $C$  has  $p$  properties  $p_1, p_2, \dots, p_p$ , each with a *property domain*  $Dom(p_j)$  ( $1 \leq j \leq p$ ), and a *temporal property domain*  $TDom(p_j) = Dom(p_j) \times \bar{\mathcal{T}}_1 \times \bar{\mathcal{T}}_2 \times \dots \times \bar{\mathcal{T}}_n$  ( $1 \leq j \leq p$ ), which is the Cartesian product of  $n + 1$  domains, one of which is the property domain ( $p_j.val$ ), while the other  $n$  domains are time interval domains, referred to as the *temporal dimensions* of  $p_j$  ( $p_j.t_k$  for  $1 \leq k \leq n$ ). For example, in a bi-temporal database, a temporal property domain is  $TDom(p_j) = Dom(p_j) \times \bar{\mathcal{T}}_v \times \mathcal{T}_x$ .

**EXAMPLE 1.** Consider the provision of stock exchange information using the principal of market segmentation, where some customers are willing to compromise the quality of retrieved information for a reduced cost.<sup>5</sup>

The stock market information service is partially represented using a class *MostActive* with the following six properties: *Market*, *Symbol*, *Name*, *LastSale*, *LastTradeSize*, and

<sup>4</sup>As will become evident shortly (see Section 3 and Section 4), we need to timestamp access requests ( $t_{req} \in TV$ ), in order to determine authorization granting.

<sup>5</sup>In fact, the public is entitled to get a time delayed data (e.g., 20 minutes for NYSE and AMEX) at no cost. However, this time delay may not be acceptable for some segments of the market.

LastTradeSize=	
$se_1$	600, $\underbrace{[July30:1999:13:57, UC], July30:1999:13:58}_{t_v}$
$se_2$	100, $\underbrace{[July30:1999:14:03, UC], July30:1999:14:04}_{t_x}$
$se_3$	500, $\underbrace{[July30:1999:15:55, UC], July30:1999:15:56}_{t_x}$

**Table 1: The LastTradeSize of the Dell share at NYSE**

**2ndActiveMarket.** The class `MostActive` describes the most active shares in various markets. Each industrial company is identified by both a share symbol and a market code, and has a name. In addition, information is provided regarding the share itself (last sale and last trade size) as well as information regarding other stock markets in which the same company is also active.  $\square$

**DEFINITION 3 (CLASS INSTANCE).** A class instance  $c$  of a class  $C$  is a sequence of sets  $\langle s_1, s_2, \dots, s_{n_i} \rangle$ , such that  $\forall i$  ( $1 \leq i \leq n_i$ ),  $s_i \subset TDom(p_i)$ .  $s_i$  ( $1 \leq i \leq n$ ) is termed a state of property  $p_i$  and each element  $se$  in  $s_i$  ( $1 \leq i \leq n$ ) is termed a state-element of property  $p_i$ .  $\square$

Definition 3 defines a class instance to be a sequence of sets, where the basic primitive (state-element) corresponds to a single value in a conventional database.<sup>6</sup> A state  $s_i$  is the set of state-elements of the property  $p_i$  of the class instance  $c$ . In further reference, we shall refer to the specific assignment of values to a state-element  $se$  as  $se.val$  for the non-temporal value and  $se.t_i$  ( $1 \leq i \leq n$ ) for the associated temporal values.

We term a set of temporal dimensions to be *discriminating temporal dimensions* iff it generates a partial order among the state-elements of a state. For example, transaction time is typically considered to be a discriminating temporal dimension in a bi-temporal database. Thus, given any two state-elements  $\{se_1, se_2\} \subseteq s$ ,  $se_1 \prec_{t_x} se_2$  iff  $se_1.t_x < se_2.t_x$ . We use  $t_x$  as a discriminating temporal dimension in the formal definition of the semantics of *UC*, as follows. Let  $se_1 \in s$  be a state-element of a state  $s$  such that  $se_1.t_v = [t_{s1}, UC)$ . The actual time interval of  $se_1.t_v$  is

$$[t_{s1}, \min_{se \in s | se.t_v = [t_s, t_e) \wedge se_1 \prec_{t_x} se \wedge t_{s1} < t_s} (se.t_s)).$$

The end time of the interval is defined by the minimal start time (still bigger than  $t_s$ ) of a state-element whose transaction time is greater than  $se_1.t_x$ .

In this paper, we assume the use of append-only databases, a special class of temporal databases, in which data can only be added to a database, but are never modified or deleted. For this class of databases, the symbol *UC* cannot be modified once inserted into the database and the derivation of an actual time interval from its symbolic counterpart is always evaluated at run-time.

**EXAMPLE 2.** The temporal aspect of the `MostActive` class is transparent to the user, as can be readily seen from the schema description in Example 1. Table 1 illustrates the changes to the state `LastTradeSize` of the `MostActive` class for the Dell share at NYSE. The state represents the changes of trade size for the share over time.

<sup>6</sup>Class instances are commonly termed *objects*. In this paper, however, the term object is reserved for the authorization model (see Section 3).

Table 1 consists of two temporal dimensions, namely valid time ( $t_v$ ) and transaction time ( $t_x$ ). We consider transaction time to be the discriminating temporal dimension. Therefore,  $se_1 \prec_{t_x} se_2 \prec_{t_x} se_3$ . Also, the actual time intervals in this example are  $se_1.t_v = [July30:1999:13:57, July30:1999:14:03)$  (as imposed by the valid time of  $se_2$ ) and  $se_2.t_v = [July30:1999:14:03, July30:1999:15:55)$  (as imposed by the valid time of  $se_3$ ). The actual valid time of  $se_3$  is yet to be determined by the next state-element to be added to the state `LastTradeSize` of the `MostActive` class for the Dell share at NYSE.  $\square$

Equipped with the data model, we are now ready to provide the semantics of wffs in this framework. We start by defining interpretation, to be followed by variable assignments.

**DEFINITION 4 (INTERPRETATION).** An interpretation is a quadruple  $S = \langle TW, \leq, W, TFN \rangle$  where  $TW$  is a nonempty universe of time points;  $\leq$  is a binary relation on  $TW$ ;  $W$  is a nonempty universe of individuals that is disjoint from  $TW$ ;  $TFN$  is a set of total functions in  $TW^2 \rightarrow TW$ . Let  $RL$  be a set of relations over  $W$ .  $M$  is a set of meaning functions where:

- $M_1 : TC \rightarrow TW$
- $M_2 : C \rightarrow W$
- $M_3 : TF \rightarrow TFN$
- $M_4 : TW \times TW \times TW \times R \rightarrow RL$   $\square$

Let  $TW$  be the integers.  $M_1$  assigns a temporal time point symbol with an integer, using some application-dependent granularity (e.g., minutes).  $M_2$  provides an interpretation of the non-temporal constants.  $M_3$  provides the common interpretation for addition, subtraction, multiplication, and division in the integers. Finally,  $M_4$  assigns a specific combination of temporal dimensions and a relation with a value. Follow some examples of the meaning functions  $M_1$  and  $M_4$ :

$$M_1(July30:1999:13:57) = 57,$$

$$M_1(5minutes) = 5,$$

$$M_4(58, 57, 63, LastTradeSize) = LastTradeSize(600)$$

In the first example, July30:1999:13:00 is taken as time 0. In the third example, the property `LastTradeSize` with a transaction time of 58 and a valid time of [57, 63) is mapped into a value 600 (based on Table 1).

For the purpose of variable assignments, we shall use four temporal assignments, as follows. In addition to the transaction time,  $t_x$ , we shall represent each valid time interval  $\bar{t}_v = [t_s, t_e)$  by its extreme time points.  $t_{req}$  is the time at which an access request was submitted to the database. Whenever additional temporal dimensions are needed (e.g., replication time in digital libraries), the variable assignment should include them as well.

**DEFINITION 5 (VARIABLE ASSIGNMENT).** A variable assignment is a pair  $VA = \langle \{VAT\}^4, VAV \rangle$  such that  $VAT : TV \rightarrow TW$  and  $VAV : V \rightarrow W$   $\square$

Let  $LTS$  be a non-temporal variable (representing the value of the `LastTradeSize` property of the Dell share at NYSE). An example of a variable assignment is  $VA(t_x, t_s, t_e, t_{req}, LTS)$   
 $= \langle VAT(t_x), VAT(t_s), VAT(t_e), VAT(t_{req}), VAV(LTS) \rangle$   
 $= \langle 58, 57, 63, 63, 600 \rangle$

DEFINITION 6 (TIME DEPENDENT MEANING). A time dependent meaning  $MVA$  is defined inductively according to the following rules:

- If  $tv \in TV$  then  $MVA(tv) = VAT(tv)$ .
- If  $tc \in TC$  then  $MVA(tc) = M_1(tc)$ .
- If  $f \in TF$  and  $trm = f(trm_1, trm_2)$  is a temporal term then  $MVA(trm) = M_3(f)(MVA(trm_1), MVA(trm_2))$ .  $\square$

### 3. THE TEMPORAL DATA AUTHORIZATION MODEL (TDAM)

In this section, we propose an authorization model, suitable for temporal databases, that is capable of providing access control based on the temporal characteristics of data, e.g., transaction time ( $t_x$ ), valid time ( $t_v$ ), and replication time ( $t_r$ ). We first introduce temporal authorization specifications (Section 3.1) followed by an analysis of possible preprocessing to the authorization base to expedite the access control process (Section 3.2).

#### 3.1 Temporal authorization specification

Let  $S = \{s_1, s_2 \dots\}$  denote a set of subjects,  $O = \{o_1, o_2 \dots\}$  a set of objects, and  $M = \{read, write, own, \dots\}$  a finite set of privilege modes, organized into a lattice  $\mu = (M, E_\mu)$ , with the following semantics. Let  $\{m_1, m_2\} \subseteq M$  such that there exists an edge  $m_1 \rightarrow m_2 \in E_\mu$ . Thus, any subject  $s$  who can gain access to an object  $o$  in mode  $m_1$ , can also gain access to  $o$  in mode  $m_2$ . For example, given two privilege modes, *read* and *write*, then  $write \rightarrow read \in E_\mu$  means that any subject  $s$  who can write to an object  $o$  can also read  $o$ . Users are classified into a set of *privilege groups* ( $PG$ ), e.g., all users who pay \$5 a month may be in one privilege group  $pg_i$  whereas all users who pay \$50 a month belong to another privilege group  $pg_j$ . The factors that determine how users are grouped into privilege groups depend on the administrative policies of the application. A privilege group can therefore be a single user or a group of users, where one user can possibly participate in several groups. Moreover, the participation of a user in a privilege group is dynamic. For example, it is not unusual for brokerage firms to provide a selected group of their customers with the recent stock quotes based on the amount of transactions they perform. Similar to privilege modes, we assume that the elements of  $PG$  are organized into a lattice  $\mu = (PG, E_\pi)$ , with the following semantics. Let  $\{pg_1, pg_2\} \subseteq PG$  such that exists an edge  $pg_1 \rightarrow pg_2 \in E_\pi$ . Thus, any subject  $s$  who is classified into a privilege group  $pg_1$  is also classified into a privilege group  $pg_2$ . For example, given two privilege groups, *salesperson* and *manager*, then  $manager \rightarrow salesperson \in E_\pi$  means that object  $o$  that is accessible to a salesperson, is also accessible to a manager.

The granularity of an object in TDAM may correspond to either a class or a property, depending on the needs. For

LastTradeSize=		
$se_1$	600, [ 57, UC),	58
$se_2$	100, [ 63, UC),	64
$se_3$	500, [175, UC),	176
	$t_v$	$t_x$

Table 2: The LastTradeSize of the Dell share at NYSE — a simplified version

simplicity sake, we shall assume in what follows that an authorization object corresponds to a single property. The model can easily be extended to either a set of properties or a class. The following definition allows specification of authorization based on the temporal attributes of objects.

DEFINITION 7 (AUTHORIZATION). An authorization  $\mathbf{a}$  is a quintuple  $(pg, o, m, sign, \tau)$ , where  $pg$  is a privilege group,  $o$  is an object,  $m$  is a privilege mode,  $sign \in \{+, -\}$  indicating access or denial, and  $\tau$  is a well-formed formula.  $\square$

According to Definition 7, a subject  $s$  that belongs to a privilege group  $pg$  can gain (or be denied) a privilege  $m$  on state-elements of  $o$ , as determined by  $\tau$ .  $\tau$  is a well-formed formula that may involve one or more temporal dimensions of the object. Without the  $\tau$  specification, either all state-elements of object  $o$  are accessible to a user or none are, which may be inappropriate, as will be demonstrated in the following examples.

EXAMPLE 3. Consider once again the stock exchange example and assume that users belonging to a specific privilege group (say  $pg$ ) are allowed access to a certain object ( $o$ ), say LastTradeSize, only five minutes after it has been written into the database.

The various state-elements of LastTradeSize, as introduced in Table 1, are simplified for presentation purposes in Table 2 (using July30:1999:13:00 as time 0) by replacing each of  $t_s$ ,  $t_e$ , and  $t_x$  with their  $M_1$  interpretation. The actual valid time intervals of  $se_1$  and  $se_2$  were computed in Example 2. The simplified version is  $se_1.t_v = [57, 63)$  and  $se_2.t_v = [63, 175)$ .

The authorization can be specified as follows:  $(pg, o, read, +, t_x + 5minutes \preceq t_{req})$ . The wff  $t_x + 5minutes \preceq t_{req}$  establishes the 5 minutes delay security policy. Thus, if  $t_{req} = 63$ ,  $se_1$  is selected, since  $se_1.t_x = 58$  and thus  $t_x + 5 = 58 + 5 = 63 \leq t_{req}$ . Both  $se_1$  and  $se_2$  are selected for  $t_{req} = 69$ , (but not earlier since only at time 69 the assignment  $se_2.t_x = 64$  is satisfied). It is worth noting that in the absence of any valid time constraint, a state-element will be presented to the user even if its valid time has already expired. Thus,  $se_1$  remains accessible by a subject from the  $pg$  group even after time 63.  $\square$

EXAMPLE 4. Consider a digital library setting, in which digital copies of books self-destruct 21 days from time of download. Each digital copy may be timestamped with a replication time  $t_r$ . The borrowing policy can thus be given as  $(borrower, o, read, t_r + 21days \preceq t_{req})$ .  $\square$

$\tau$  is utilized in identifying the state-elements for which an authorization may be granted, through the notion of formula satisfaction, as follows. Let  $S$  be an interpretation and  $VA$  a variable assignment.  $S$  satisfies a wff  $\tau$  with  $VA$  (written as  $S \models_{VA} \tau$ ) under the following inductively defined conditions:

- $S \models_{VA} trm_1 = trm_2$  iff  $MVA(trm_1) = MVA(trm_2)$ .
- $S \models_{VA} trm_1 \preceq trm_2$  iff  $MVA(trm_1) \leq MVA(trm_2)$ .
- $S \models_{VA} r(trm)$  iff  $r(VAV(trm)) \in M_4(VAT(t_x), VAT(t_s), VAT(t_e), r)$ .
- $S \models_{VA} (\tau_1 \wedge \tau_2)$  iff  $S \models_{VA} \tau_1$  and  $S \models_{VA} \tau_2$ .
- $S \models_{VA} (\neg \tau_1)$  iff  $S \not\models_{VA} \tau_1$ .

EXAMPLE 5. Let  $S$  be the interpretation as given in the stock exchange example, and let  $VA = \langle 58, 57, 63, 63, 600 \rangle$  be an assignment. Finally, let

$$\tau = ((t_x + 5minutes \preceq t_{req} \wedge t_{req} \preceq t_e + 5minutes)$$

$$\wedge \text{LastTradeSize}(LTS))$$

be a wff. We shall now prove that  $S \models_{VA} \tau$ .

- $\tau = \tau_1 \wedge \tau_2$ , where  $\tau_1 = (t_x + 5minutes \preceq t_{req} \wedge t_{req} \preceq t_e + 5minutes)$  and  $\tau_2 = \text{LastTradeSize}(LTS)$ .

- $S \models_{VA} \tau_2$  since

$$\text{LastTradeSize}(VAV(LTS))(600) \text{ and}$$

$$M_4(VAT(t_x), VAT(t_s), VAT(t_e), r) =$$

$$M_4(58, 57, 63, \text{LastTradeSize}) =$$

$$\text{LastTradeSize}(600).$$

- $\tau_1 = \tau_3 \wedge \tau_4$ , where  $\tau_3 = t_x + 5 \preceq t_{req}$  and  $\tau_4 = t_{req} \preceq t_e + 5$ .

- $S \models_{VA} \tau_3$  since

$$MVA(t_x + 5) = M_3(+)(MVA(t_x), MVA(5)) =$$

$$M_3(+)(58, 5) = 63,$$

$$MVA(t_{req}) = 63, \text{ and}$$

$$63 = MVA(t_x + 5) \leq MVA(t_{req}) = 63.$$

- $S \models_{VA} \tau_4$  since

$$MVA(t_{req}) = 63,$$

$$MVA(t_e + 5) =$$

$$M_3(+)(MVA(t_e), MVA(5)) =$$

$$M_3(+)(63, 5) = 68, \text{ and}$$

$$63 = MVA(t_{req}) \leq MVA(t_e + 5) = 68.$$

- $S \models_{VA} \tau_1$  since  $S \models_{VA} \tau_3$  and  $S \models_{VA} \tau_4$ .

- $S \models_{VA} \tau$  since  $S \models_{VA} \tau_1$  and  $S \models_{VA} \tau_2$ .  $\square$

A subject can be granted access to a state-element  $se$  under  $\mathbf{a} = (pg, o, m, sign, \tau)$  only if

$S \models_{VA(se.t_x, se.t_s, se.t_e, t_{req}, se.val)} \tau$ . Therefore, Definition 7 refines conventional authorization specifications in that it may restrict the access to some state-elements of the authorized object, based on  $\tau$ .

In what follows, we use  $pg_a, o_a, m_a, sign_a$ , and  $\tau_a$  to denote a privilege group, an object, a privilege mode, access or denial, and a constraining formula of  $\mathbf{a}$ , respectively. The set of all authorization specifications is termed the *authorization base*  $AB$ .

## 3.2 Preprocessing

To expedite the access control process, several preprocessing methods to  $AB$  can be performed. In this paper, we propose three such methods, namely unification, expansion, and segmentation. Preprocessing shortens the access control decision process, as it requires a single access to the authorization base to identify the existence of the needed authorization and the cumulative temporal wff. Without the preprocessing, the traversal through the lattices and the accumulation of temporal wffs, as discussed herein, need to be done at run-time, which may require multiple accesses to the authorization base.

### 3.2.1 Uni•cation

An initial unification process involves authorizations that differ solely in their temporal formula. Such authorizations are unified into a single authorization with a modified temporal formula, as follows. Let  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$  be  $n$  authorizations, such that for any  $i$  and  $j$  ( $1 < i < j < n$ ),  $pg_{a_i} = pg_{a_j}$ ,  $o_{a_i} = o_{a_j}$ ,  $m_{a_i} = m_{a_j}$ , and  $sign_{a_i} = sign_{a_j}$ . A unification of  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$  is a process in which  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$  are replaced with a single authorization

$$\mathbf{a} = \begin{cases} (pg_{a_1}, o_{a_1}, m_{a_1}, sign_{a_1}, \bigvee_{1 \leq i \leq n} \tau_i) & sign_{a_i} = + \\ (pg_{a_1}, o_{a_1}, m_{a_1}, sign_{a_1}, \bigwedge_{1 \leq i \leq n} \tau_i) & sign_{a_i} = - \end{cases}$$

where  $\bigvee_{1 \leq i \leq n} \tau_i$  and  $\bigwedge_{1 \leq i \leq n} \tau_i$  annotate the disjunction and conjunction of all temporal wffs, respectively. Performing unification can be performed in  $O(n \log n)$ , where  $n$  represents the number of authorizations in  $AB$ , which is the cost of sorting the authorizations in  $AB$ . We term an authorization base in which all possible unifications have been performed a *unified authorization base*  $UAB$ .

### 3.2.2 Expansion

The introduction of the lattices  $\mu$  and  $\pi$  in combination with  $AB$  bares impact on authorizations temporal formulae. By way of motivation, consider a privilege hierarchy  $\mu$  with  $write \rightarrow read \in E_\mu$ . Let  $\mathbf{a}_1$  and  $\mathbf{a}_2$  be the following two authorizations:

$$\begin{aligned} \mathbf{a}_1 &= (pg, o, read, +, t_x + 10minutes \preceq t_{req}) \\ \mathbf{a}_2 &= (pg, o, write, +, t_x + 5minutes \preceq t_{req}) \end{aligned}$$

The implication of  $\mathbf{a}_2$  on  $\mathbf{a}_1$  is that

$$\begin{aligned} \tau_{a_1} &= (\tau_{a_1} \vee \tau_{a_2}) \\ &= (t_x + 10minutes \preceq t_{req} \vee t_x + 5minutes \preceq t_{req}) \\ &= t_x + 5minutes \preceq t_{req} \end{aligned} \tag{1}$$

where Formula (1) is derived from the semantics of the  $write \rightarrow read$  relationship, and Formula (2) can be derived since  $S \models_{VA} t_x + 5minutes \preceq t_{req}$  entails  $S \models_{VA} t_x + 10minutes \preceq t_{req}$ . In this example, although a read authorization is explicitly given for 10 minutes old data, the write authorization extends it to include data that is not newer than 5 minutes, based on  $E_\mu$ .

We next provide an approach for generating an *expanded authorization base*  $EAB$  from a unified authorization base  $UAB$ , in which for any authorization  $\mathbf{a}$ ,  $\tau_a$  is modified to take into account the semantics of the privilege group and privilege mode lattices. The basic routine in this algorithm is the *Expand* recursive routine, which has four versions, namely, *Expand/m/+*, *Expand/m/-*, *Expand/pg/+*, and

*Expand/m/+*:  
 Input:  $(pg, o, m, +, \tau)$   
 Output:  $(pg, o, m, +, \tau')$   
 Procedure:  
 $\tau' := \emptyset$   
 If exists  $(pg, o, m, +, \tau^*) \in UAB$  then:  
 $\tau' := \tau^*$   
 For each  $m'$  such that  $m' \rightarrow m \in E_\mu$  do:  
 $(pg, o, m', +, \tau'') := \text{Expand}((pg, o, m', +, \tau))$   
 If  $\tau'' \neq \emptyset$  then:  
 $\tau' := \tau' \vee \tau''$   
 end {For}  
 If exists  $(pg, o, m, +, \tau^*) \in UAB$  do:  
 $UAB := UAB - (pg, o, m, +, \tau^*)$   
 $EAB := EAB + (pg, o, m, +, \tau')$   
 return  $(pg, o, m, +, \tau')$

Figure 1: Formulae Expansion/m/+

*Expand/pg/-*. Each of these versions handles a different combination of a hierarchy and a sign. *Expand/pg/+* is given in Algorithm 1 (Figure 1), and uses a Depth-First-Search (DFS) algorithm to scan the lattice. At each iteration, a single authorization is given as an input for the algorithm. The algorithm is recursively called for each preceding node in  $\mu$ . The returned temporal constraints are then combined and sent to the calling routine. The stop condition would be a root node in  $\mu$ . It is worth noting that the expanded authorization base consists of modified existing authorizations as well as newly introduced authorizations. Therefore, in the example given above,  $\tau_{a_1}$  will be modified into  $(t_x + 10\text{minutes} \preceq t_{req} \wedge t_x + 5\text{minutes} \preceq t_{req})$  (which is effectively  $t_x + 5\text{minutes} \preceq t_{req}$ ). Even had we had only  $\mathbf{a}_2$ , a new authorization  $((pg, o, read, +, t_x + 5\text{minutes} \preceq t_{req}))$  would have been created in *EAB*.

The routine is performed for each authorization in *UAB*. Given a lattice  $\mu$  with  $n$  nodes and a unified authorization base with  $m$  authorizations, computing *EAB* can be done in  $O(n \times m)$ . To optimize the process, authorizations should be evaluated from the leaf nodes up, ensuring a maximum number of deleted authorizations from *UAB* during the process.

There are two main differences between Algorithm 1 and the routine that handles the mode hierarchy and negative authorizations. The first is that  $\tau$  is further restricted whenever a negative authorization is involved (and therefore the individual formulae are combined using conjunction). The second difference lies in the ordering of the recursive calls. While in Algorithm 1, nodes are being scanned up the lattice, where each lower mode contributes an additional disjunctive term to  $\tau$ , in *Expand/m/-* nodes are being scanned down the lattice, where each higher mode contributes an additional conjunctive term to  $\tau$ . The differences among the four routines are minor, and we refrain from providing the pseudo-code for *Expand/m/-*, *Expand/pg/+*, and *Expand/pg/-* in this paper.

### 3.2.2.1 Maintaining the expanded authorization base.

Once the expanded authorization base has been created,

its maintenance can be performed in an incremental fashion. Therefore, for each newly inserted authorization  $\mathbf{a}$ , Algorithm 1 is performed, using  $\mathbf{a}$  as an input. Revocation of an authorization requires performing the following steps. Let  $(pg, o, m, +, \tau)$  be the revoked authorization:

1. If exists  $(pg, o, m, +, \tau') \in EAB$ , it should be replaced with  $(pg, o, m, +, \tau'')$ , where  $\tau' = \tau' \vee \tau$ . It is worth noting that we are guaranteed to have  $\tau''$ , due to the way the expansion algorithm was defined. Also, we assume that the authorization base *AB* cannot contain two identical authorizations, and therefore there is no “multiple support” for an authorization.
2. Using DFS, we identify any authorization on the mode/privilege group that may be affected by this change. Therefore, if exists  $(pg, o, m', +, \tau') \in EAB$  such that  $m' \rightarrow m \in E_\mu$ , it is modified to be  $(pg, o, m', +, \tau'')$ , where  $\tau' = \tau' \vee \tau$ , unless exists  $(pg, o, m', +, \tau^* \vee \tau) \in AB$ . In this case,  $\tau$  is explicitly supported by another authorization in *AB*, and therefore  $(pg, o, m', +, \tau')$  remains unchanged. The same holds for the  $\pi$  lattice.

Finally, whenever negative authorizations are deleted, we apply a similar approach, using conjunction instead of disjunction, and traversing the lattices from root to leaves.

### 3.2.3 Segmentation

We next present a practical approach towards validating whether given a state-element  $se$ ,

$$S \models_{VA(se.t_x, se.t_s, se.t_e, t_{req}, se.val)} \tau.$$

Such an approach takes an advantage of the natural indexing mechanism of temporal databases, where state-elements are stored in order of transaction time, and additional indices may exist on the start and end times of a valid time. Therefore, we establish separate constraints on  $t_s, t_e$ , and  $t_x$  ( $\tau_s, \tau_e$ , and  $\tau_x$ , respectively), using basic arithmetic manipulations. For example, given an authorization  $(pg, o, read, +, t_x + 5\text{minutes} \preceq t_{req})$ , the temporal expression is segmented, a process which results in a single constraint (no constraint is set on  $t_s$  or  $t_e$  since they are not present in  $\tau$ ):

$$\tau_x = t_x \leq t_{req} - 5$$

In the example given above,  $\tau_s, \tau_e$ , and  $\tau_x$  resulted in a constraint of the form  $t_o \diamond t$ , where  $t_o \in \{t_x, t_s, t_e\}$ ,  $\diamond$  is a comparison operator (*e.g.*,  $=$  and  $\geq$ ), and  $t$  is a time point. This type of constraint is definitely not the only possible constraint type. As an example, consider an authorization  $\mathbf{a}$  such that  $\tau_a = t_x \preceq t_s + 5$ , that is, all the state-elements which become valid only 5 minutes after being committed to the database. Applying segmentation to  $\tau_a$ , results in two constraints, as follows:

$$\begin{aligned} \tau_x &= (t_x \leq t_s + 5) \\ \tau_s &= (t_s \geq t_x - 5) \end{aligned}$$

Clearly, any state-element that satisfies  $\tau_x$  also satisfies  $\tau_s$ . Removing redundant constraints is based on efficiency considerations, *e.g.*, for which temporal dimension the database maintains indices, and whether a specific index is optimized for equality retrieval or a range retrieval.

We have omitted the details of the segmentation process. The process is simple in its logic, yet tedious in its structure, and introducing it does not contribute much to the essence of this paper. It is also worth noting that segmentation is performed on individual authorizations and needs to be performed for each modified authorization, *e.g.*, in case of revocation.

## 4. ACCESS CONTROL

An access request can be of two types. A user may follow the traditional access control method and request access for a specific object at a certain time point. This we refer to as a *point access request*. Alternatively, a user may request access for an object for a specific duration, which we refer to as an *interval access request*. The latter is useful, for example, in cases where information is provided continuously (as in data streaming) and the service is charged accordingly. Formally:

**DEFINITION 8** (ACCESS REQUEST). A point access request is a triple  $car = (s, o, m)$ . An interval access request is a quadruple  $iar = (d, s, o, m)$ , where  $d \in \mathbf{N}$ .  $\square$

Each access request is timestamped with the time for which access is requested by a subject  $s$  with privilege mode  $m$  on object  $o$  (annotated as  $t_{req} \in \mathcal{T}$ ).  $t_{req}$  serves as a decision factor whenever relative temporal expressions are involved, *e.g.*,  $t_x + 5\text{minutes} \preceq t_{req}$ . An interval access request is interpreted as an access request by a subject  $s$  with privilege mode  $m$  on object  $o$  for duration  $[t_{req}, t_{req} + d) \in \bar{\mathcal{T}}$ . It is worth noting that an access request in TDAM is a natural extension of an access request in authorization models for non-temporal data. Therefore, a point access request is interpreted the same way as in authorization models for non-temporal data whenever the authorization base does not enforce temporal constraints (effectively meaning,  $\tau = true$ ). Also, an interval access request, where  $d = 1$ , is equivalent to a point access request. It is also noteworthy that the access granting decision is binary with respect to any given time point, yet it may provide partial access for intervals, *i.e.*, access to a sub-interval only. A final note relates to the case where  $d = \infty$ . This case is interpreted as “the access request holds until it is explicitly terminated.” This type of authorization is useful whenever stream data is involved, as is the case with streaming stock exchange data to a user over the Web. In this case, the closing of a session (either in an orderly fashion or abruptly) serves as an explicit termination of the authorization.

Given an access request  $car = (s, o, m)$  at time  $t_{req}$ , we should first verify the *presence of authorization*, *i.e.*, to determine if there exists an authorization  $\mathbf{a} = (pg, o, m, +, \tau)$ , such that  $s \in pg$ . We next provide a *formula assessment*. This step involves evaluating the aggregated temporal impact of relevant authorizations (positive and negative) and generating an aggregated formula  $\tau'$  for which such an authorization can be granted. The next step involves *state-elements selection*, where a state-element  $se$  of  $o$  is selected only if  $S \models_{VA(se.t_x, se.t_s, se.t_e, t_{req}, se.val)} \tau$ .

An interval access request  $iar = (d, s, o, m)$  at time  $t_{req}$  can be conceptually viewed as  $d$  separate point access requests,  $\{car_i = (s, o, m)\}_{i=0}^d$ , timestamped with  $\{t_{req}^i = t_{req} + i\}_{i=0}^d$ . Each  $car_i$  is possibly granted with a different set of state-elements of  $o$ , such that  $\tau$  is satisfied with

respect to a  $t_{req}^i$  assignment. We shall refer to the process of substituting one set of state-elements with another as *version switching*. It is important to note at this point that since we assume the use of an append-only database, state-elements are never modified. Nevertheless, the access to a state-element may be revoked whenever the temporal characteristics of the state-element no longer satisfy  $\tau'$ . In most cases the actual revocation time can be computed as soon as access is granted. For example, consider an authorization

$$(pg, o, read, +, (t_x + 5\text{minutes} \preceq t_{req} \wedge t_{req} \preceq t_e + 5\text{minutes})). \quad (3)$$

For the state-element  $se_1$  in Example 3 above, a request time  $t_{req} = 63$ , and a request interval  $[t_{req}, t_{req} + 10)$ , a request can be granted for the interval  $[63, 68)$ . The lower bound of this interval (63) is computed by the assignment  $t_x + 5\text{minutes} = 58 + 5 = 63$  (see Table 2 for the transaction time of  $se_1$ ). The upper bound of the interval (68) is computed by the assignment  $t_e + 5\text{minutes} = 63 + 5 = 68$ . Whenever symbolic time intervals are involved, though, revocation decisions may need to be taken at run-time. To this end, consider once more Example 3, the authorization (3) given above, a request time  $t_{req} = 63$ , and a request interval  $[t_{req}, t_{req} + 150)$ . At the time of the request,  $se_3$  was not yet introduced (its transaction time is 176) and therefore  $se_2$  does not yet have an actual  $t_e$  and the access interval is set to be  $[63, 63 + 150) = [63, 213)$ . At time 176, the access request is still in effect, and the introduction of  $se_3$  requires re-evaluating  $se_2$  as well. The consequence of this process is an upper bound of  $t_e + 5 = 175 + 5 = 180$  for  $se_2$  (the reader should note that the actual valid time interval for  $se_2$  is  $[63, 175)$ ). The access interval for  $se_3$  is set to be  $[176, 213)$ .

### 4.1 Evaluating access requests

ALGORITHM 2.

```

Input:  $(s, o, m)$  or  $(d, s, o, m)$ ,  $t_{req}$ , and  $\{se\} \in o$ 
Output:  $\{(s, se, m, \bar{t})\}$ 
Procedure:
/* presence of authorization */
If there exists an authorization  $a = (pg, o, m, +, \tau) \in EAB$ 
such that  $s \in pg$ , then
 $\tau' = null$ 
/* formula assessment */
If there exists an authorization  $a = (pg, o, m, -, \tau) \in AB$ 
such that  $s \in pg$ , then
 $\tau' = \tau' \wedge \neg\tau$ 
/* state-elements selection */
If  $d = null$  then  $d = 1$ 
repeat
 $SE = \{se \in o \mid \tau_s \wedge \tau_e \wedge \tau_x\}$ 
/* version switching */
For each  $se_i \in SE$ 
 $\bar{t}_i = bind(se_i.t_s, se_i.t_e, se_i.t_x) \cap [t_{req}, t_{req} + d)$ 
If  $\bar{t}_i \neq \emptyset$  then output  $(s, se_i, m, \bar{t}_i)$ 
end For
until now  $\geq t_{req} + d$ 
end If

```

Figure 2: Access request evaluation

Figure 2 presents the algorithm for evaluating access requests. The algorithm accepts as an input an access request  $ar$  for an object  $o$  (either a point access request  $(s, o, m)$  or an interval access request  $(d, s, o, m)$ ), its timestamp  $(t_{req})$  and a continuous feed of state-elements of  $o$ . The output consists of a set of state-elements, authorized for access, and a time interval during which each state-element is authorized for  $ar$ . The algorithm is invoked once per access request and is terminated either when no authorization  $\mathbf{a} = (pg, o, m, +, \tau) \in EAB$  exists or when the request interval  $[t_{req}, t_{req} + d)$  has expired.

The algorithm uses the segmented constraints  $\tau_s, \tau_e$ , and  $\tau_x$ , as computed in the preprocessing phase. Consider an authorization request  $(s, o, read)$ , issued at time  $t_{req} = 63$  and assume that  $s \in pg$ . As a first step, assume the authorization  $(pg, o, read, +, t_x + 5minutes \preceq t_{req})$  is retrieved. Given  $t_{req} = 63$ , one has that

$$\begin{aligned} \tau_x &= t_x \leq t_{req} - 5 \\ &= t_x \leq 63 - 5 \\ &= t_x \leq 58 \end{aligned}$$

Once establishing constraints on the state-elements temporal dimensions, a set of state-elements that satisfy these constraints is computed. Using the example given above, and the set of state-elements from Table 2, the only state-element that qualifies for the set is  $se_1$ , with  $t_x = 58 \leq 58$ .  $se_2$ , although in the database, fails to satisfy the constraint with a transaction time of 64. Based on the  $t_{req}$  of this example,  $se_3$  is not yet in the database.

Consider now an interval authorization request of the form  $(10, s, o, read)$ . Again, the authorization  $(pg, o, read, +, t_x + 5minutes \preceq t_{req})$  is retrieved. Given an interval  $[t_{req} = 63, t_{req} + d = 63 + 10 = 73)$ ,  $\tau_x$  is computed as follows:

$$\begin{aligned} \tau_x &= (t_x \leq 58 \text{ (computed by substituting } t_{req}/63)) \\ &\cup t_x \leq 59 \text{ (computed by substituting } t_{req}/63 + 1) \\ &\dots \\ &\cup t_x \leq 68 \text{ (computed by substituting } t_{req}/63 + 10) \\ &= (t_x \leq 68) \end{aligned}$$

In this case, both  $se_1$  and  $se_2$  qualify for retrieval.

The next step involves the version switching. For each state-element in the selected set, an interval during which it can be presented to the user is computed, using  $bind(se_i.t_s, se_i.t_e, se_i.t_x)$ . Following the example given above, we next compute the access interval for  $se_1$ , by substituting  $t_x/58$  in  $t_x + 5minutes \preceq t_{req}$ , to result in  $[63, \infty)$  (since  $t_{req}$  only provide a lower bound in this case). For  $se_2$ , the substitution  $t_x/64$  results in an interval  $[69, \infty)$ . These intervals are further truncated by the request interval  $[63, 73)$ , since  $t_{req} = 63$  and  $t_{req} + d = 63 + 10 = 73$ . Therefore, the interval associated with  $se_1$  is  $[63, 73)$ , and that of  $se_2$  is  $[69, 73)$ . It is worth emphasizing that the combination of  $\tau$  and the interval  $[t_{req}, t_{req} + d)$  provides the needed mechanism to forbid access beyond that specified by  $\tau$  and the access request, even if the validity of the state-element goes beyond it.

## 4.2 Discussion

A technical issue involves the computation of actual valid times from symbolic valid times. Assuming actual valid

times are not materialized in the database (as with append-only databases), the computation of the actual valid time should be performed on-the-fly. For that purpose, it is beneficial to have an index on the transaction time dimension and perform the selection of state-elements by applying  $\tau_x$  first. This way, the number of state-elements for which the computation of actual valid time intervals is reduced and is easy to perform, since state-elements are accessed in transaction time order, where the transaction time serves as the discriminating temporal dimension. The same technique is applied whenever there is a need to recompute the access interval for state-elements with a symbolic valid time interval where  $t_e = UC$ . An example of such a situation was given earlier, where the introduction of  $se_3$  results in (virtually) replacing  $se_2.t_e = UC$  with  $se_2.t_e = 175$ . Indexing on the transaction time makes the identification of the existing state-element ( $se_2$  in this case) and thus, the re-evaluation process, an efficient task. In the algorithm, we have refrained from providing detailed computation of such situations, to avoid blurring the main elements of the access control process.

We are also concerned with the monitoring of new state-elements once an access is granted. Unless an access request is a point access request, the duration of the access spans into the future and therefore may include new state-elements as they come along. For example, consider the insertion of  $se_3$  of Table 2 at time 176. For an authorization of the form  $(pg, o, read, +, t_x + 5minutes \preceq t_{req})$  and an interval authorization request of the form  $(150, s, o, read)$ , such that  $t_{req} = 63$ ,  $se_3$  becomes eligible for this access request at time 181. Therefore, one may view Algorithm 2 as a generator of access authorizations, which are later being fed to a calendar-based utility that is responsible for the version switching process. For any new state-element, the algorithm identifies the time interval for which an access can be granted ( $\bar{t}$ ) and feeds the calendar-based utility with the state-elements identification and the start and end times of access granting. Whenever  $UC$  is involved, as was already discussed above, the end time may be changed upon the arrival of the next state-element, at which time the access interval of the previously current state-element may have to be changed. Consider, for example, the authorization (3) given above, and the state-elements of Table 2. An interval authorization request of the form  $(150, s, o, read)$  at time  $t_{req} = 63$  would first limit  $se_2$  to be presented during  $[69, 213)$ , where the terminating time point stems from the access request of 150 minutes. However, once  $se_3$  was introduced,  $se.t_e$  is computed to be 175, which in turn, limits the access interval to be  $[69, 181)$ , since after time 180, the formula  $t_{req} \preceq t_e + 5minutes$  can no longer be satisfied for  $se_2$ .

Both positive and negative authorizations for the same combination of privilege group, object, and access mode (with possibly different formulae) may coexist in  $EAB$ . In such a case, we take a conservative approach and assume that negative authorizations prevail whenever overlapping intervals are concerned. Unlike conventional authorization models, a careful analysis of the temporal expressions in both authorizations is required to identify conflicts. For example, consider the following two authorizations:

$$\begin{aligned} \mathbf{a}_1 &= (pg, o, write, +, t_x + 5minutes \preceq t_{req}) \\ \mathbf{a}_2 &= (pg, o, write, -, t_x + 9minutes \preceq t_{req}) \end{aligned}$$

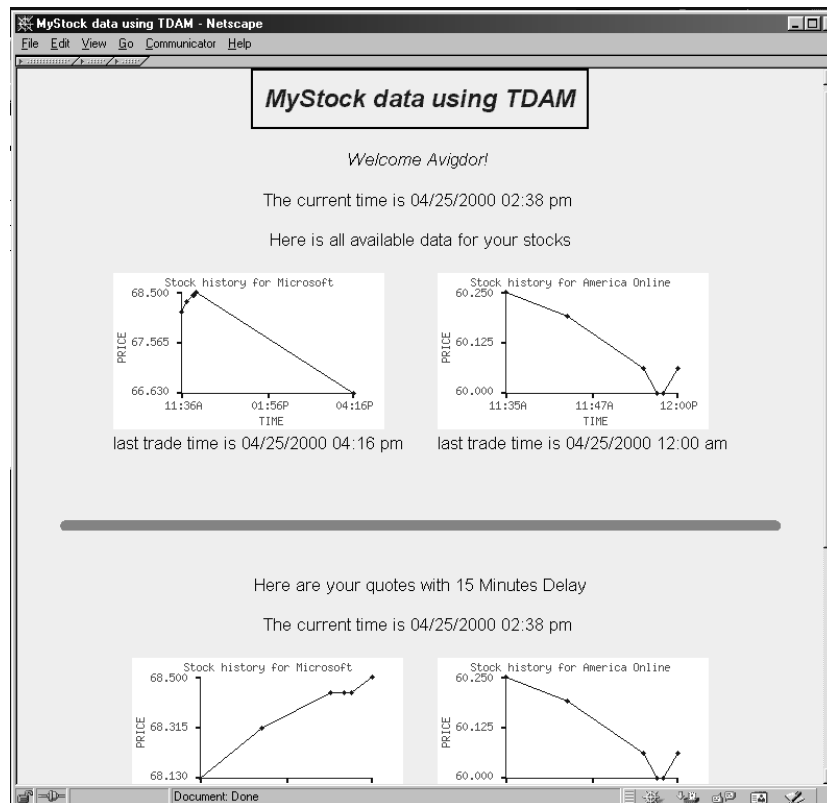


Figure 3: A snapshot of TDAM’s demo

A careful analysis concludes that the temporal expression of  $\tau_{a_1}$  is further restricted by  $\tau_{a_2}$  to be  $(t_x + 5\text{minutes} \preceq t_{req} \wedge \neg(t_{req} \preceq t_x + 9\text{minutes}))$ . For example, given the state-element  $se_1$  from Example 3 above with  $t_x = 58$ ,  $t_{req} = 63$ , and an interval access request for  $[t_{req}, t_{req} + 10)$ ,  $\mathbf{a}_1$  would allow access to  $se_1$  during  $[63, 73)$  and  $\mathbf{a}_2$  would prevent access to  $se_1$  during  $[68, 73)$  (in both cases 73 is taken from the access request terminating time point  $t_{req} + 10 = 63 + 10 = 73$ ). The combined impact of both authorizations would be an access granting during  $[63, 68)$ .

## 5. CONCLUSION

Temporal data is characterized by data values associated with such temporal attributes as the time at which data has been captured and the time interval during which a data value is valid. In this paper, we provide an extended authorization model, called Temporal Data Authorization Model (TDAM), to facilitate expression of authorizations based on the temporal attributes associated with data, such as transaction time and valid time. Such authorizations are essential in expressing security policies such as “a subject is allowed to read an object one month after it has been written.” Temporal constraints are defined in temporal logic to ensure clear specifications regarding the accessibility of state-elements and correct interpretation of such temporal constraints.

A prototype of the authorization model is currently under development. Figure 3 provides a screen snapshot, in which real-time data<sup>7</sup> is contrasted with delayed data, as specified

<sup>7</sup>The “real-time data” is actually a 20 minutes old data, as

provided by Yahoo. We are also working on extending this model to support the derivation of temporal authorizations in multi-tier databases [11], such as distributed databases with replicas and data warehouses. This way, a consistent authorization base can be constructed from the authorizations on the (possibly temporal) base data.

## 6. REFERENCES

- [1] K. Al-Taha, R. Snodgrass, and M. Soo. Bibliography on spatiotemporal databases. *SIGMOD Record*, 22(1):59–67, Mar. 1993.
- [2] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. A temporal access control mechanism for database systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):67–80, 1996.
- [3] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Database Systems (TODS)*, 23(3):231–285, 1998.
- [4] S. Chaudhuri and U. Dayal. An overview of data warehouses and OLAP technology. *SIGMOD Record*, 26(1):65–74, Mar. 1997.
- [5] J. Clifford and A. U. Tansel. On an algebra for historical relational databases: two views. In *Proceedings of the ACM SIGMOD*, pages 247–265, May 1985.

provided by Yahoo.

- [6] S. Gadia. The role of temporal elements in temporal databases. *Data Engineering Bulletin*, 7:197–203, 1988.
- [7] A. Gal, O. Etzion, and A. Segev. TALE — a Temporal Active Language and Execution model. In P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, editors, *Advanced Information Systems Engineering*, pages 60–81. Springer, May 1996.
- [8] C. Jensen, J. Clifford, S. Gadia, A. Segev, and R. Snodgrass. A glossary of temporal database concepts. *ACM SIGMOD Record*, 21(3):35–43, 1992.
- [9] N. Pissinou, R. Snodgrass, R. Elmasri, I. Mumick, M. Ozsu, B. Pernici, A. Segev, and B. Theodoulidis. Towards an infrastructure for temporal databases—A workshop report. *ACM SIGMOD Record*, 23(1):35, 1994.
- [10] A. Rajaraman. -commerce database issues and experience. In *Proceedings ACM SIGMOD International Conference on Management of Data*, page 531, 1999.
- [11] A. Rosenthal, V. Doshi, and E. Sciore. Security administration for federations, warehouses, and other derived data. In *Proceedings of the 13 Annual IFIP WG 11.3 Working Conference on Database Security*, 1999.
- [12] Y. Shoham. *Reasoning about change: Time and Causation from the standpoint of Artificial Intelligence*. MIT press, 1988.