University of Alberta

Modeling Time: Back to Basics

by

Iqbal A. Goralwalla, Yuri Leontiev, M. Tamer Özsu and Duane Szafron

Technical Report TR 96–03

February 1996

DEPARTMENT OF COMPUTING SCIENCE
The University of Alberta
Edmonton, Alberta, Canada

Modeling Time: Back to Basics[1]

Iqbal A. Goralwalla, Yuri Leontiev, M. Tamer Özsu and Duane Szafron
Laboratory for Database Systems Research
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
{iqbal,yuri,ozsu,duane}@cs.ualberta.ca

[1]This research has been supported by the Natural Sciences and Engineering Research Council of Canada under research grants OGP0951 and OGP8191. The second author's research is also supported by the Izaak Walton Killam Memorial Scholarship.

Contents

List of Figures

List of Tables

Abstract

Most of the work in modeling time in information systems has concentrated on issues such as support for historical information and providing query facilities to manipulate such information. In doing so, some simplistic view of the underling nature of time has been assumed. However, the domain of time is far from being simplistic. In this paper, we outline the various issues which arise in modeling basic temporal entities and propose solutions to these issues. More specifically, we note that the nature of temporal information can either be anchored (e.g., *October* 25, 1995) or unanchored (e.g., 1 *week*), and is usually available in multiple *granularities* (e.g., the airline flight departure and arrival times are usually given in *minutes*, while the history of the salary of an employee is usually recorded in *days*). Physical temporal information also needs to be represented in a manner so as to be human readable. This is achieved using *calendars*. In this work, we show how both anchored and unanchored temporal entities are represented within the context of calendars. We discuss how calendars provide relationships between multiple granularities and facilitate the conversion of anchored and unanchored times from one granularity to another. We also give the semantics of various operations on anchored and unanchored times.

Keywords: temporal databases, object models, calendars, granularity

1 Introduction

In the last decade there has been extensive research activity on temporal databases (see [Sno95a, ÖS95, Sno86, SS88, Soo91, Kli93, Sno90, Pea94, TCG+93]). Most of this research has concentrated on the definition of a particular temporal model and its incorporation into a (relational or object-oriented) database management system (DBMS). These temporal models are usually based on a simplistic view of the underlying structure of temporal entities. There is significant evidence, however, that suggests that many applications have varying requirements for the support of temporal entities. For example, in a university information system multiple time units need to be supported. These include *day*, *week*, *semester*, etc; in office information systems temporal information is usually available in different time units of the Gregorian calendar [BP85]; in real-time systems a process is usually composed of sub-processes that evolve according to different time scales [CMR91]; in financial trading multiple calendars with different time units and operations need to be available to capture the semantics of financial data [CS93, CSS94]. Therefore, it is necessary to be able to customize the temporal DBMS. In this paper we describe a novel approach to providing customizability. Instead of developing a temporal model that implements a particular notion of time, we model and implement, in an object-oriented system, temporal entities (primitives) on top of which various temporal models can be built. Our approach, as we discuss in more detail below, is based on using calendars to model these temporal primitives.

Modeling of basic temporal primitives requires the characterization of temporal data. A first order characterization is as follows:

- *Nature* – Temporal information can either be *anchored* (*absolute*) or *unanchored* (*relative*). For example, *July* 31, 1995 is an anchored time in that we know exactly where it is located on the time axis, whereas 31 *days* is unanchored in that we do not know where it is located on the time axis since it can stand for any block of 31 consecutive days on the time axis. Anchored temporal information can be specified using an *instant* (*moment*, *chronon*) and *interval* time primitives. An instant is a specific anchored moment in time, e.g., *July* 31, 1995. An interval is a duration of time between two specific anchor points (instants) which stand for the lower and upper bounds of the interval, e.g., [*June* 15, 1995, *July* 31, 1995]. Unanchored temporal information can be specified using the *span* time primitive. A span is an unanchored duration of time. It has a known length, but no specific starting and ending anchor points. Thus, it is independent of any instant or interval.

- *Structure* – Usually, temporal information is available in multiple *granularities*. For example, in a medical information system the history of an admitted patient would be kept on a daily basis whereas the history of the condition of the patient's body would be kept on an hourly basis.

For human readability, it is important to have a framework in which the above characteristics of temporal data can be represented. We propose to use a calendar as a framework for the representation of anchored and unanchored temporal primitives at various granularities. A calendar is a means by which physical time can be represented so as to be human readable. Calendars are comprised of different time units of varying granularities that enable the representation of different temporal primitives. Common calendars include the *Gregorian* and *Lunar* calendars. Educational institutions also use *Academic* calendars.

In many applications, it is desirable to have multiple calendars that have different calendric granularities. One way to meet this requirement is to provide system support for multiple calendars with a large number of calendric granularities. However, this has high overhead, and inevitably, there will be applications that will need calendars and calendric granularities beyond a reasonable set provided by the system. It is, therefore, important for the model to be extensible and not limited to a predefined set of calendars and calendric granularities. There are a number of important issues related to calendars that must be addressed:

1. How are calendars modeled? What are the components of a calendar? Are there any interactions between different calendars? Does a calendar provide relationships between granularities?

2. How is anchored and unanchored temporal information modeled within the context of calendars? Can instants be of mixed granularities? How about spans?

3. Can anchored time be converted from one granularity to another? What about unanchored time?

4. What are the semantics of operations between anchored times, anchored and unanchored time, unanchored times, where the finest granularities of the operands are different?

Table 1 shows the various works that have dealt with the above issues in one way or another. We note from Table 1 that not much work has been carried out in the temporal database research community towards comprehensively modeling the basic temporal entities. The few works that

Citation	Calendar(s)	Granularities	Time primitives	Granularity conversions
[CR87]	No support	Multiple	Anchored	Anchored
[WJL91], [WJS93]	No support	Multiple	Anchored	Anchored
[BP85], [MPB92]	No support	Multiple	Anchored	Anchored
[MMCR92]	No support	Multiple	Anchored	Anchored
[Sno95b]	Multiple	Multiple	Anchored & Unanchored	Anchored

Table 1: Temporal models supporting calendars and/or granularities

have appeared in this area have concentrated mainly on modeling anchored temporal entities. We contend that unanchored temporal information is equally important in information systems and the issues that arise in providing support for it should be addressed.

In this paper, we provide a model for supporting calendars and show how calendars provide relationships between multiple granularities. We further show how both anchored and unanchored temporal entities are represented in the context of calendars. We also show in detail how granularity conversions are carried out using both anchored and unanchored temporal entities. These conversions allow us to perform different kinds of operations between anchored and unanchored temporal entities.

The rest of the paper is organized as follows: in Section 2 we describe our model for supporting calendars and show how multiple granularities are accommodated. Sections 3 and 4 show how unanchored and anchored temporal entities are represented in the context of calendars, and how granularity conversions between temporal entities with different granularities are performed. In Section 5 we show how our model of calendars and temporal entities are mapped to an object model. Section 6 compares our research to the work given in Table 1. Finally, Section 7 presents conclusions.

2 Calendars

A calendar allows physical time to be represented in human readable form. Perhaps the most familiar calendar is the Gregorian calendar based on the revolution of the earth around the sun. Another familiar calendar is the Lunar calendar, based on the rotation of the moon around the earth. In general, calendars are based on the needs of different cultures or organizations. For example, the Lunar calendar is also commonly known as the Islamic calendar and is used by Muslims world wide. Academic calendars used by educational institutions are examples of organizational calendars.

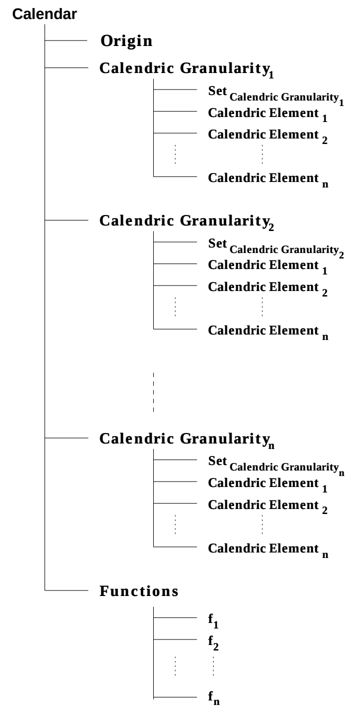

Figure 1: A hierarchical calendric structure.

Definition 2.1 *Calendar ($\mathcal{C}$):* A calendar $\mathcal{C}$ is a triplet $\langle \mathcal{O}, \{G\}, \{\mathcal{F}\}\rangle$, where $\mathcal{O}$ is the origin of $\mathcal{C}$, $\{G\}$ is the set of calendric granularities belonging to $\mathcal{C}$, and $\{\mathcal{F}\}$ is the set of conversion functions associated with $\mathcal{C}$.

A hierarchical structure of a calendar is given in Figure 1. The origin marks the start of a calendar. Calendric granularities define the reasonable time units (e.g., *minute*, *day*, *month*) that can be used in conjunction with this calendar. Calendric granularities within a calendar are counted from the origin of that calendar. The origin of a calendar is in essence a time span in calendric granularities of the calendar. The functions establish the conversion rules between calendric granularities of a calendar. In the remainder we discuss calendric granularities and functions in more detail.

2.1 Calendric Granularities

A calendar can be defined in terms of any reasonable time unit. For example, the Gregorian calendar has days and months as time units. However, the Academic calendar also adds semesters. More specifically, a calendar is comprised of a finite number of time units. We call these time units *calendric granularities*. In the Gregorian calendar, *years*, *months*, *days*, *hours*, *minutes*, *seconds*, etc. are the calendric granularities.

Definition 2.2 *Calendric granularity (G):* A calendric granularity is a special kind of determinate time span (duration) that can be used as a unit of time.

Generally speaking, a calendric granularity is a unit of measurement for time durations. For example, the calendric granularity of days (G_{day}) in the Gregorian calendar behaves similar to the time span 1 *day* (time spans are discussed in more detail in Section 3).

Calendric granularities are also commonly used to express a lack of information about the particular time of an event. For example, the statement "I am flying to Calgary on July 5th, 1996," implies that the flight will take place *some time* on July 5th. In this case the calendric granularity that is used to represent the time of the flight (1 day) is also a period of indeterminacy of an instant. Note that no indeterminacy arises about the duration of time. In other words, 1 day as a duration of time carries no indeterminacy. We do not deal with temporal indeterminacy in this paper. We refer the reader to [GLÖS95] for details on how we model indeterminate temporal information.

Since a calendric granularity is a special kind of a time span, it is meaningful to compare two calendric granularities with each other.

Definition 2.3 *Comparison between calendric granularities:* G_A is *coarser* than G_B if G_A is $>$ than G_B as a time span. Similarly, G_A is *finer* than G_A if G_A is $<$ than G_B as a time span.

Example 2.1 The span of 1 *day* is shorter ($<$) than the span of 1 *month* and therefore the calendric granularity of days (G_{day}) is finer than the calendric granularity of months (G_{month}) in the Gregorian calendar. Similarly, G_{month} is coarser than G_{day}.

We assume that we can always compare two calendric granularities belonging to the same calendar with each other. Thus,

Observation 2.1 The set of all possible calendric granularities in a given calendar ($\{G\}$) is totally ordered with respect to the comparison operators defined in Definition 2.3.

Observation 2.2 The set of all possible calendric granularities belonging to different calendars is partially ordered with respect to the comparison operators defined in Definitions 2.3.

Each calendric granularity in a calendar has a reference to a set of similar calendric granularities belonging to different calendars, hereafter referred to as Set_{G_A}, associated with it. Set_{G_A} contains calendric granularities which have the same time duration as G_A. For example, the calendric granularities G_{month} and $G_{academicMonth}$ have references to the set $\{month, academicMonth\}$. More specifically, $Set_{G_{month}} = Set_{G_{academicMonth}} = \{month, academicMonth\}$. Set_{G_A} is utilized when a calendric granularity of one calendar needs to be converted to a calendric granularity with the same time duration but belonging to a different calendar. For example, the span 2 *months* is equivalent to the span 2 *academicMonths* when converted to the calendric granularity of *academicMonths*.

A calendric granularity also has a list of *calendric elements*. For example in the Gregorian calendar, G_{day} has the calendric elements *Monday*, *Tuesday*, $\ldots$, *Sunday*. Similarly in the Academic calendar, $G_{semester}$ has the calendric elements *Fall*, *Winter*, *Spring*, and *Summer*.

Example 2.2 Figure 2 shows the hierarchical calendric structures of two real-world calendars namely, the Gregorian and Academic calendars. We use these two example calendars in the following discussion.

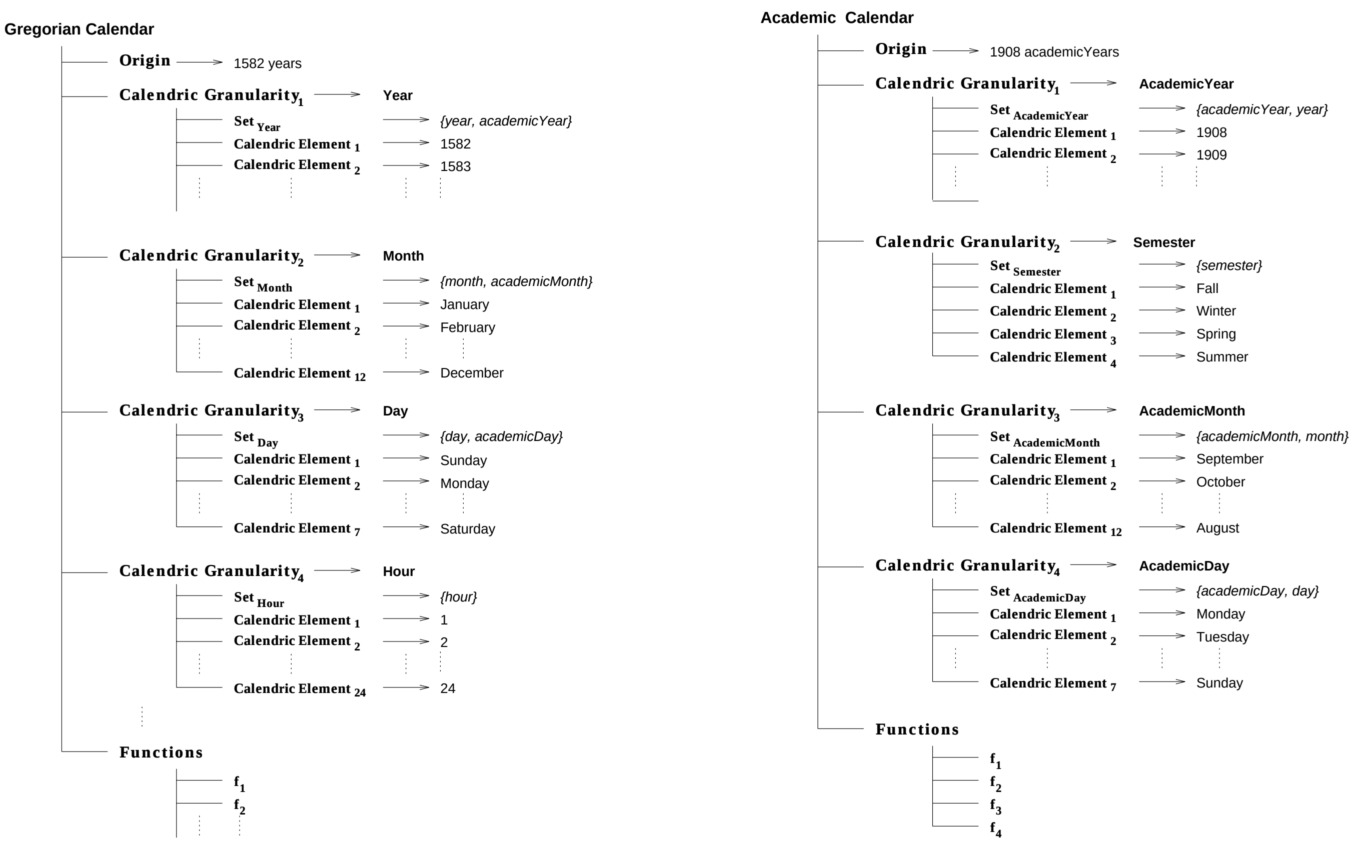

Figure 2: The Gregorian and Academic calendric structures.

The origin of the Gregorian calendar is given as the span 1582 *years* from the start of time. It was proclaimed in 1582 by Pope Gregory XIII as a reform of the Julian calendar. The calendric granularities in the Gregorian calendar are the standard ones, each having similar calendric granularities from the Academic and possibly other calendars, e.g., the Business calendar.

The origin of the Academic calendar shown in Figure **??** is assumed to be the span 1908 *academic-Years* having started in the year 1908, which is the establishment date of the University of Alberta. The Academic calendar has similar calendric granularities as the Gregorian calendar and defines a new calendric granularity of *semester*. It is worth noting that the calendric elements of *academic-Month* start from *September* as compared to those of the Gregorian calendar which start from *January*.

2.2 Functions

Associated with each calendar is a list of functions ($\{\mathcal{F}\}$) which determine the number of finer calendric elements in coarser calendric elements. For example, lets assume we have a calendar C which has the calendric granularities *year*, *month* and *day*. Then, three functions are defined. The first returns the number of months in a given year, the second returns the number of days in a given month of a given year. The third maps a given year, month, and day to a real number on a global timeline.

More generally, let C be a calendar with calendric granularities $G_1, G_2, \ldots, G_n$, where G_1 is the coarsest calendric granularity and G_n is the finest calendric granularity. Additionally, let the calendric elements of the calendric granularities be:

$$\begin{aligned}
G_1 &: ce_1^{G_1}, ce_2^{G_1}, \ldots, ce_{p_1}^{G_1} \\
G_2 &: ce_1^{G_2}, ce_2^{G_2}, \ldots, ce_{p_2}^{G_2} \\
&\vdots \\
G_n &: ce_1^{G_n}, ce_2^{G_n}, \ldots, ce_{p_n}^{G_n}.
\end{aligned}$$

where $ce_j^{G_i}$ stands for the j^{th} calendric element of the calendric granularity G_i. For example, $ce_2^{G_{month}}$ represents the second calendric element of the calendric granularity G_{month} in the Gregorian calendar. The following functions are then defined:

Definition 2.4 *Conversion functions:*

$$\begin{aligned}
f_C^1(i_1) &\rightarrow N_{G_2},\ 1 \leq i_1 \leq p_1 \\
f_C^2(i_1, i_2) &\rightarrow N_{G_3},\ 1 \leq i_1 \leq p_1, 1 \leq i_2 \leq p_2 \\
&\vdots \\
f_C^n(i_1, i_2, \ldots, i_n) &\rightarrow R,\ 1 \leq i_1 \leq p_1, 1 \leq i_2 \leq p_2, \ldots, 1 \leq i_n \leq p_n
\end{aligned}$$

where i_j $(1 \leq j \leq n)$ are natural numbers which correspond to the ordinal number of a calendric element of the j^{th} calendric granularity in calendar C. For example, the ordinal values of the year 1995 and the month September in the Gregorian calendar would be 1995-1582+1 and 9, respectively. N_{G_x} $(1 \leq x \leq n)$ is a natural number which stands for the number of G_x's. R is a real number.

The first function $(f_C^1(i_1))$ gives the number of G_2's in a given calendric element of G_1. The second function $(f_C^2(i_1, i_2))$ gives the number of G_3's in a given calendric element of G_1 and a calendric element of G_2. The last function $(f_C^n(i_1, i_2, \ldots, i_n))$ maps a calendric element of the finest calendric granularity (G_n) to a real number on an underlying global real timeline, hereafter referred to as G_{tl}. G_{tl} provides a homogeneous underlying platform whereby operations involving time instants belonging to different calendars (or the same calendar, but represented in different calendric granularities) can take place. More specifically, each time instant can be converted to its corresponding value on G_{tl}, and the operation can then be performed. The conversion functions and G_{tl} are used mainly in operations involving instants in which the operands could belong to different calendars. The role of the conversion functions and G_{tl} are discussed in detail later on in this paper. The scale of G_{tl} is dependent on the precision of the respective machine architecture. For simplicity and explanatory purposes, we assume the scale of G_{tl} to be *seconds* in this paper.

Example 2.3 To illustrate the workings of the above functions, lets suppose we interested in the number of months in 1995, the number of days in September 1995 and the number of seconds in September 12, 1995 in calendar C. The ordinal values corresponding to the year 1995, the month September, and the day 12, are 414 (1995-1582+1), 9, and 12, respectively. Then:

$$\begin{aligned} f_C^1(414) &\rightarrow 12 \\ f_C^2(414, 9) &\rightarrow 30 \\ f_C^3(414, 9, 12) &\rightarrow 86400.0 \end{aligned}$$

In this section we have described our model of calendars. In the next two sections we show how anchored and unanchored temporal primitives of different granularities are represented using the components of a calendar. Additionally, we show how operations between anchored and unanchored temporal primitives are carried out across different calendars.

3 Unanchored Temporal Entities

We identify a *time span* as being an unanchored, relative duration of time. Examples of time spans include 5 hours, 10 days, 2 to 3 months, etc. A time span is basically an atomic, cardinal quantity, independent of any time instant or time interval, with a number of operations defined on it:

1. A time span can be compared with another time span with the transitive comparison operators $<$ and $>$. This comparison operator forms a partial order between time spans.

2. A time span can be subtracted from or added to another time span to return a third time span.

Time spans can be further characterized as being determinate or indeterminate. A *determinate span* represents complete information about a duration of time. For example, the maximum time allowed for students to complete their *Introduction to Database Management Systems* examination is a determinate span. An *indeterminate span* represents incomplete information about a duration of time. It has lower and upper bounds that are determinate spans. $1\ day \sim 2\ days$, for example, is an indeterminate span that can be interpreted as "a time period between one and two days." Any determinate span can be represented as a special kind of indeterminate span with identical lower and upper bounds.

3.1 Representation of Spans

Since a calendric granularity is a unit measurement of a time span, we use calendric granularities to construct time spans. For example, to obtain a time span of 5 days we would multiply the calendric granularity of days by the integer 5: $5 \cdot G_{day}$. To obtain a time span of $2\ academicMonths\ and\ 5\ days$, we would add the span of $2\ academicMonths$ to the span of $5\ days$: $2 \cdot G_{academicMonth} + 5 \cdot G_{day}$. In general, a time span is made up of different calendric granularities, possibly belonging to different calendars.

Definition 3.1 *Discrete Determinate span:*

$$S_{discr} = \sum_{i=1}^{N} \sum_{j=1}^{M} (K_j^{C_i} \cdot G_j^{C_i}) \tag{1}$$

where $K_j^{C_i}$ is an integer coefficient of $G_j^{C_i}$, which is a distinct calendric granularity in calendar C_i.

Definition 3.2 *Continuous Determinate span:*

$$S_{cont} = \sum_{i=1}^{N}\sum_{j=1}^{M}(R_j^{C_i} \cdot G_j^{C_i}) \tag{2}$$

where $R_j^{C_i}$ is a real coefficient of $G_j^{C_i}$, which is a distinct calendric granularity in calendar C_i.

Basically, S_{discr} and S_{cont} are summations of distinct calendric granularities over different calendars. S_{cont} is a generalization of S_{discr} for the case of real coefficients.

In a temporal model where times with different calendars and calendric granularities are supported, the calendric granularities of a time span may belong to different calendars. Therefore, we need to be able to:

1. Convert one calendric granularity to another calendric granularity belonging to the same calendar.
2. Convert one calendric granularity to another calendric granularity belonging to another calendar.

We discuss these conversions below.

3.2 Conversions Between Calendric Granularities

The first question to answer is whether it is always possible to convert a time span from the coarser to the finer calendric granularity without loss of information. The answer, perhaps surprisingly, is no. To illustrate this point let us consider the following conversions. The conversion of the time span 1 *hour* to the calendric granularity of minutes is exact and will result in the time span of 60 *minutes*. However, the conversion of the time span 1 *month* to the finer calendric granularity of days can not possibly be an exact one. Should the resulting time span be 31, 30, 29 or 28 days? We cannot tell unless we know exactly which month is involved. Since a time span is *unanchored* this information is not available. Of course, we could convert 1 *month* to the indeterminate span 28 *days* $\sim$ 31 *days* but in this case the conversion is not exact and some information is lost. Therefore, the following observation is made:

Observation 3.1 The set of all calendric granularities is not totally ordered with respect to the binary relation "exactly convertible to."

In order to be able to carry out the conversion (whether exact or inexact) of a time span to a given calendric granularity, we define two functions.

Definition 3.3 *Lower bound factor ($lbf(G_A, G_B)$):* The lower bound factor of G_A and G_B is the minimum number of G_B units that can form 1 G_A unit.

Definition 3.4 *Upper bound factor ($ubf(G_A, G_B)$):* The upper bound factor of G_A and G_B is the maximum number of G_B units that can form 1 G_A unit.

Example 3.1 $lbf(G_{month}, G_{day}) = 28$ and $ubf(G_{month}, G_{day}) = 31$. Both factors coincide in the case of exact conversion. For instance, $lbf(G_{hour}, G_{minute}) = ubf(G_{hour}, G_{minute}) = 60$.

The user can define new calendric granularities in terms of existing ones. For example, the new calendric granularity *decade* could be defined in terms of the existing calendric granularity *year* using $lbf(G_{decade}, G_{year}) = ubf(G_{decade}, G_{year}) = 10$.

3.2.1 Derivation Procedure for $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$

We first show how $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ are derived from the conversion functions defined in Section 2, if G_A and G_B belong to the same calendar. To simplify the description, we first consider a simple calendar and then give the derivation for the general case which involves any calendar with any number of calendric granularities.

Derivation 3.1 *G_A is coarser than G_B – Simple calendar:* Let C be a calendar with the calendric granularities *year*, *month* and *day*. The following functions are defined in C:

$$\begin{aligned} f_C^1(y) &\rightarrow N_{months} \\ f_C^2(y, m) &\rightarrow N_{days} \\ f_C^3(y, m, d) &\rightarrow R \end{aligned}$$

where y, m, and d are ordinal values of calendric elements in the calendric granularities year, month, and day, respectively. Suppose we want to find $lbf(G_{year}, G_{day})$ and $ubf(G_{year}, G_{day})$. The number of days in any year y is given by the summation:

$$\sum_{m=1}^{f_C^1(y)} f_C^2(y, m)$$

The minimum (maximum) number of days in a year is then the minimum (maximum) of the above summation over all y. More specifically,

$$lbf(G_{year}, G_{day}) = \min_y \{ \sum_{m=1}^{f_C^1(y)} f_C^2(y, m) \}$$

$$ubf(G_{year}, G_{day}) \quad = \quad \max_y \{ \sum_{m=1}^{f_C^1(y)} f_C^2(y, m) \}$$

Derivation 3.2 *G_A is coarser than G_B – General calendar:*

We now consider the general case. Let $G_1, \ldots, G_A, \ldots, G_B, \ldots, G_n$ be the totally ordered calendric granularities of calendar C with G_1 being the coarsest calendric granularity and G_n the finest. The following functions are defined in C:

$$\begin{array}{rcl} & \vdots & \\ f_C^{k_A}(i_1, \ldots, i_A) & \rightarrow & N_{i_A+1} \\ & \vdots & \\ f_C^{k_B}(i_1, \ldots, i_B) & \rightarrow & N_{i_B+1} \\ & \vdots & \end{array}$$

Now, the number of G_B units in any given calendric element i_A is given by the following summation:

$$f_C^{k_A \rightarrow k_B}(i_1, \ldots, i_A) = \sum_{j_1=1}^{f_C^{k_A}(i_1,\ldots,i_A)} \; \sum_{j_2=1}^{f_C^{k_A+1}(i_1,\ldots,i_A,j_1)} \cdots \sum_{j_{k_B-k_A-1}=1}^{f_C^{k_B-2}(i_1,\ldots,i_A,j_1,\ldots,j_{k_B-k_A-2})} f_C^{k_B-1}(i_1, \ldots, i_A, j_1, \ldots, j_{k_B-k_A-1})$$

The minimum (maximum) number of G_B units in calendric element i_A is then the minimum (maximum) of the above formula over all $i_1, \ldots, i_A$. More specifically,

$$lbf(G_A, G_B) \quad = \quad \min_{(i_1,\ldots,i_A) \in C} \{ f_C^{k_A \rightarrow k_B}(i_1, \ldots, i_A) \} \tag{3}$$

and

$$ubf(G_A, G_B) \quad = \quad \max_{(i_1,\ldots,i_A) \in C} \{ f_C^{k_A \rightarrow k_B}(i_1, \ldots, i_A) \} \tag{4}$$

Derivation 3.3 *Minimum and maximum number of G_B in K units of G_A:* Formulas (3) and (4) calculate the minimum and maximum number of G_B in *one* unit of G_A, respectively. We now generalize formulas (3) and (4) to calculate the minimum and maximum number of G_B in K units of G_A, e.g., the minimum and maximum number of days in $2 \cdot G_{month}$ where $K = 2$.

$$lbf(K, G_A, G_B) \quad = \quad \min_{i_1,\ldots,i_A} \{ \sum_{0 \le dist_{k_A}((i'_1,\ldots,i'_A),(i_1,\ldots,i_A)) \le K-1} f_C^{k_A \rightarrow k_B}(i'_1, \ldots, i'_A) \} \tag{5}$$

and

$$ubf(K, G_A, G_B) = \max_{i_1,\ldots,i_A} \{ \sum_{0 \le dist_{k_A}((i'_1,\ldots,i'_A),(i_1,\ldots,i_A)) \le K-1} f_C^{k_A \to k_B}(i'_1, \ldots, i'_A) \} \quad (6)$$

The summation in formulas (5) and (6) is the number of k_B units in K consecutive k_A units starting with $(i_1, \ldots, i_A)$. The function $dist_{i_A}((i'_1, \ldots, i'_A), (i_1, \ldots, i_A))$ finds the number of k_A units elapsed between $(i'_1, \ldots, i'_A)$ and $(i_1, \ldots, i_A)$. For example, the number of months elapsed between (1996, February) and (1995, January) is 13. The lower and upper bound factors are then obtained by taking the minimum and maximum of the summation over all $(i_1, \ldots, i_A)$. Embedding the coefficient K within formulas (5) and (6) reduces the information lost in the process of calculating the number of G_B units in K units of G_A as compared to first finding the number of G_B units in one unit of G_A and then multiplying it by K to find the number of G_B in K units of G_A. For example, using formulas (3) and (4) to calculate the minimum and maximum number of days in $2 \cdot G_{month}$ gives us 56 and 62, respectively, while formulas (5) and (6) give us 59 and 62, respectively – thereby reducing the information lost by 3 days. Note that for exact conversions,

$$lbf(K, G_A, G_B) = ubf(K, G_A, G_B) = K \cdot lbf(G_A, G_B) = K \cdot ubf(G_A, G_B)$$

For example, $lbf(K, G_{days}, G_{hours}) = ubf(K, G_{days}, G_{hours}) = K \cdot 24$.

Derivation 3.4 *G_A is finer than G_B:* So far we have considered the case of G_A being coarser than G_B. If G_A is finer than G_B, then the lower and upper bound factors can be calculated using the formulas:

$$lbf_i(N, G_A, G_B) = \max_{K \in \mathbf{Z}} \{K \mid N \ge ubf(K, G_B, G_A)\} \quad (7)$$

$$ubf_i(N, G_A, G_B) = \min_{K \in \mathbf{Z}} \{K \mid N \le lbf(K, G_B, G_A)\} \quad (8)$$

Example 3.2 To illustrate the above formulas, suppose we want to find the number of months in 45 *days*. Then:

$$lbf_i(45, G_{day}, G_{month}) = \max_{K \in \mathbf{Z}} \{K \mid 45 \ge ubf(K, G_{month}, G_{day})\} = 1$$

$$ubf_i(45, G_{day}, G_{month}) = \min_{K \in \mathbf{Z}} \{K \mid 45 \le lbf(K, G_{month}, G_{day})\} = 2$$

Hence, the number of months in 45 *days* is $1 \sim 2$.

Note that it is not necessary that K be an integer. It can be a real number as well, in which case we reduce the amount of indeterminacy in finding the number of months in 45 *days*. Thus, the formulas in Derivation 3.4 become:

$$lbf_r(R, G_A, G_B) = \max_{K \in \mathbf{R}^+} \{K \mid R \geq ubf(K, G_B, G_A)\} \tag{9}$$

$$ubf_r(R, G_A, G_B) = \min_{K \in \mathbf{R}^+} \{K \mid R \leq lbf(K, G_B, G_A)\} \tag{10}$$

Example 3.3 We know that the number of days in 1 *month* is $28 \sim 31$ and the number of days in 2 *months* is $59 \sim 62$. Therefore, we can reasonably say that for $1 \leq K \leq 2$:

$$\begin{aligned} lbf(K, G_{month}, G_{day}) &= 28 + (59 - 28) \cdot (K - 1) = 31 \cdot K - 3 \\ ubf(K, G_{month}, G_{day}) &= 31 + (62 - 31) \cdot (K - 1) = 31 \cdot K \end{aligned}$$

Now:

$$\begin{aligned} lbf_r(45, G_{day}, G_{month}) &= \max_{K \in \mathbf{R}^+} \{K \mid 45 \geq ubf(K, G_{month}, G_{day})\} \\ &= \max_{K \in \mathbf{R}^+} \{K \mid 45 \geq 31 \cdot K\} = 45/31 \\ &= 1.45 \\ ubf_r(45, G_{day}, G_{month}) &= \min_{K \in \mathbf{R}^+} \{K \mid 45 \leq lbf(K, G_{month}, G_{day})\} \\ &= \min_{K \in \mathbf{R}^+} \{K \mid 45 \leq 31 \cdot K - 3\} = 48/31 \\ &= 1.55 \end{aligned}$$

In this case the number of months in 45 *days* is $1.45 \sim 1.55$, quite a contrast from what was obtained for K as an integer.

So far, we have considered derivations for $lbf(K, G_A, G_B)$ and $ubf(K, G_A, G_B)$ when G_A and G_B belong to the same calendar. We now consider the case when G_A and G_B belong to different calendars.

Derivation 3.5 *G_A and G_B belong to different calendars:*

Let G_A and G_B belong to calendars C_1 and C_2, respectively. The following procedure derives $lbf(K, G_A, G_B)$ and $ubf(K, G_A, G_B)$ for both K as an integer coefficient and K as a real coefficient:

if $\exists G_A', G_B' \mid G_A' \in Set_{G_A} \wedge G_B' \in Set_{G_B} \wedge G_A' \in C' \wedge G_B' \in C'$
{
  if G_A' is coarser than G_B'
    Derive $lbf(K, G_A', G_B')$ and $ubf(K, G_A', G_B')$ using Derivation 3.3
  else if G_A' is finer than G_B'

Derive $lbf(K, G'_A, G'_B)$ and $ubf(K, G'_A, G'_B)$ using Derivation 3.4

Use $lbf(K, G'_A, G'_B)$ for $lbf(K, G_A, G_B)$ and $ubf(K, G'_A, G'_B)$ for $ubf(K, G_A, G_B)$

}

else

$lbf(K, G_A, G_B)$ and $ubf(K, G_A, G_B)$ have to be explicitly specified

Since Set_{G_A} and Set_{G_B} contain calendric granularities which have the same time duration as G_A and G_B, respectively, the above procedure first checks whether there exists in Set_{G_A} and Set_{G_B} calendric granularities which belong to the same calendar. If such calendric granularities exist, then they are used instead of G_A and G_B in the derivations of $lbf(K, G_A, G_B)$ and $ubf(K, G_A, G_B)$. If no calendric granularities exist in Set_{G_A} and Set_{G_B} belonging to the same calendar, the $lbf(K, G_A, G_B)$ and $ubf(K, G_A, G_B)$ have to be explicitly provided.

Example 3.4 Suppose we want to calculate $lbf(G_{businessMonth}, G_{year})$ where $G_{businessMonth}$ belongs to the Business calendar and G_{year} belongs to the Gregorian calendar. Suppose also that $Set_{G_{businessMonth}} = \{G_{businessMonth}, G_{AcademicMonth}\}$ and $Set_{G_{year}} = \{G_{year}, G_{AcademicYear}\}$, where $G_{AcademicMonth}$ and $G_{AcademicYear}$ belong to the Academic calendar. Then, $lbf(G_{businessMonth}, G_{year}) \equiv lbf(G_{AcademicMonth}, G_{year}) \equiv lbf(G_{AcademicMonth}, G_{AcademicYear})$.

3.2.2 Span Conversion

Having discussed the derivation procedure of $lbf(K, G_A, G_B)$ and $ubf(K, G_A, G_B)$, we now define the conversion of a determinate span to any given calendric granularity G_A.

Definition 3.5 *Discrete span conversion:* The conversion of a span of the form depicted in formula (1) to a calendric granularity G_A results in an indeterminate span with lower bound

$$\lfloor \sum_{j=1}^{N} \sum_{i=1}^{M} L_i^{C_j} \rfloor \cdot G_A \tag{11}$$

and upper bound

$$\lceil \sum_{j=1}^{N} \sum_{i=1}^{M} U_i^{C_j} \rceil \cdot G_A \tag{12}$$

where

$$L_i^{C_j} = lbf_r(K_i^{C_j}, G_i^{C_j}, G_A) \tag{13}$$

and

$$U_i^{C_j} = ubf_r(K_i^{C_j}, G_i^{C_j}, G_A) \tag{14}$$

Definition 3.6 *Continuous span conversion:* The conversion of a span of the form depicted in formula (2) to a calendric granularity G_A results in an indeterminate span with lower bound

$$\sum_{j=1}^{N}\sum_{i=1}^{M} L_i^{C_j} \cdot G_A \tag{15}$$

and upper bound

$$\sum_{j=1}^{N}\sum_{i=1}^{M} U_i^{C_j} \cdot G_A \tag{16}$$

where

$$L_i^{C_j} = lbf_r(K_i^{C_j}, G_i^{C_j}, G_A) \tag{17}$$

and

$$U_i^{C_j} = ubf_r(K_i^{C_j}, G_i^{C_j}, G_A) \tag{18}$$

Example 3.5 To illustrate the conversion described above, let us convert the discrete time span 2 *months and* 45 *hours and* 3 *academicYears* to a discrete indeterminate span in the calendric granularity of days (G_{days}). First we represent the given span in the form given in formula (1):

$$2 \cdot G_{months} + 45 \cdot G_{hours} + 3 \cdot G_{academicYears}$$

In this span we have calendric granularities from two calendars, the Gregorian and Academic calendars. G_{months} and G_{hours} are members of the Gregorian calendar (C_1), while $G_{academicYears}$ is a member of the Academic calendar (C_2). Additionally, $K_1^{C_1} = 2$, $K_2^{C_1} = 45, K_1^{C_2} = 3, G_1^{C_1} = G_{months}$, $G_2^{C_1} = G_{hours}, G_1^{C_2} = G_{academicYears}$. We now use the formulas (13) and (14) to compute $L_1^{C_1}, L_2^{C_1}, L_1^{C_2}, U_1^{C_1}, U_2^{C_1}$, $U_1^{C_2}$:

$$\begin{aligned}
L_1^{C_1} &= lbf(K_1^{C_1}, G_1^{C_1}, G_{days}) \\
&= lbf(2, G_{months}, G_{days}) \\
&= 59 \\
U_1^{C_1} &= ubf(K_1^{C_1}, G_1^{C_1}, G_{days}) \\
&= 62 \\
L_2^{C_1} &= lbf(K_2^{C_1}, G_2^{C_1}, G_{days}) \\
&= lbf(45, G_{hours}, G_{days}) \\
&= max\{K \mid 45 \geq ubf(K, G_{days}, G_{hours})\} \\
&= max\{K \mid 45 \geq K \cdot 24\}
\end{aligned}$$

$$
\begin{aligned}
&= 45/24 \\
&= 1.875 \\
U_2^{C_1} &= ubf(K_2^{C_1}, G_2^{C_1}, G_{days}) \\
&= min\{K \mid 45 \leq lbf(K, G_{days}, G_{hours})\} \\
&= min\{K \mid 45 \leq K \cdot 24\} \\
&= 1.875 \\
L_1^{C_2} &= lbf(K_1^{C_2}, G_1^{C_2}, G_{days}) \\
&= lbf(3, G_{academicYears}, G_{days}) \\
&= lbf(3, G_{years}, G_{days}) \\
&= 1095 \\
U_1^{C_2} &= ubf(K_1^{C_2}, G_1^{C_2}, G_{days}) \\
&= ubf(3, G_{academicYears}, G_{days}) \\
&= ubf(3, G_{years}, G_{days}) \\
&= 1096
\end{aligned}
$$

$lbf(K, G_{months}, G_{days})$, $lbf(K, G_{days}, G_{hours})$, $ubf(K, G_{months}, G_{days})$, and $ubf(K, G_{days}, G_{hours})$ are calculated from the conversion functions in the Gregorian calendar. In deriving $lbf(K, G_{academicYears}, G_{days})$ and $ubf(K, G_{academicYears}, G_{days})$, since $G_{academicYears}$ and G_{days} belong to different calendars, we first make use of Derivation 3.5 and note that in $Set_{G_{AcademicYear}}$ and $Set_{G_{day}}$ there exist calendric granularities G_{year} and G_{day} which belong to the same calendar (the Gregorian calendar). Therefore, $lbf(K, G_{academicYears}, G_{days})$ is equivalent to $lbf(K, G_{years}, G_{days})$ which is then calculated from the conversion functions in the Gregorian calendar. The same holds true for $ubf(K, G_{academicYears}, G_{days})$. If $Set_{G_{AcademicYear}}$ and $Set_{G_{day}}$ did not have calendric granularities belonging to the same calendar, then $lbf(K, G_{academicYears}, G_{days})$ and $ubf(K, G_{academicYears}, G_{days})$ would have to be explicitly specified.

Lastly, we compute the lower and upper boundary of the resulting indeterminate span according to formulas (11) and (12), respectively:

$$
\begin{aligned}
lower\ bound &= \lfloor L_1^{C_1} + L_2^{C_1} + L_1^{C_2} \rfloor \cdot G_{days} \\
&= \lfloor 59 + 1.875 + 1095 \rfloor \cdot G_{days} \\
&= 1155 \cdot G_{days}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{upper\ bound} &= \lceil U_1^{C_1} + U_2^{C_1} + U_1^{C_2} \rceil \cdot G_{days} \\
&= \lceil 62 + 1.875 + 1096 \rceil \cdot G_{days} \\
&= 1160 \cdot G_{days}
\end{aligned}
$$

Hence, the result of our conversion is the indeterminate discrete time span 1155 $days \sim 1160\, days$.

In this section we have described in detail how conversion of a time span of mixed calendric granularities to another calendric granularity takes place. To the best of our knowledge, this feature of conversions between granularities based on unanchored time has not been considered by any of the previous models dealing with time granularities. In the following section we show, using examples, how mathematical operations between time spans take place.

3.3 Mathematical Operations between Spans

As described in Section 3.1, a span is represented as a summation of different calendric granularities. In this section we elaborate on the mathematical operations between spans using various examples. The semantics of adding (subtracting) two spans is to add (subtract) the components which have the same calendric granularity and concatenate the remaining components to the resulting span. For example, let us assume we have two calendars, the Gregorian calendar with calendric granularities *year*, *month*, and *day*, and the Academic calendar with calendric granularities *academicYear* and *semester*. Then:

Example 3.6

1. $(5\ years + 4\ months) + 2\ years \rightarrow (7\ years + 4\ months)$

2. $(5\ years + 4\ months) + (2\ years - 8\ months) \rightarrow (7\ years - 4\ months)$

3. $(5\ years + 4\ months) + 15\ days \rightarrow (5\ years + 4\ months + 15\ days)$

4. $(5\ years + 4\ months) + 2\ academicYears \rightarrow (5\ years + 4\ months + 2\ academicYears) \equiv 4\ months + 7\ years \equiv 4\ months + 7\ academicYears$

5. $(5\ years + 4\ months + 2\ academicYears) + (2\ academicYears + 1\ semester) \rightarrow (5\ years + 4\ months + 4\ academicYears + 1\ semester) \equiv 4\ months + 1\ semester + 9\ academicYears \equiv 4\ months + 1\ semester + 9\ years$

Similar semantics hold true for addition (subtraction) of determinate spans and indeterminate spans.

Example 3.7

1. $(5\ days \sim 7\ days) + 1\ day \rightarrow 6\ days \sim 8\ days$

2. $(5\ days \sim 1\ month) + 2\ days \rightarrow 7\ days \sim (1\ month + 2\ days)$

3. $(1\ month \sim 2\ months) - 30\ days \rightarrow (1\ month - 30\ days) \sim (2\ months - 30\ days)$

4. $((1\ month - 30\ days) \sim (2\ months - 30\ days)) - (1\ month - 30\ days + 5\ hours) \rightarrow -5\ hours \sim (1\ month - 5\ hours)$

5. $((1\ month - 30\ days) \sim 15\ days) + 5\ hours \rightarrow (1\ month - 30\ days + 5\ hours) \sim (15\ days + 5\ hours)$

Subtraction leads to the notion of negative spans. In our model, both positive and negative spans are allowed. Positive spans have the semantics of forward duration in time, whilst negative spans have the semantics of backward duration in time. Allowing positive and negative spans enables us to carry out the subtraction operation between spans of different calendric granularities which could result in either a positive or negative span, for example, $1\ month - 30\ days$. It is worth mentioning that mathematical operations between spans could result in spans which are composed of calendric granularities belonging to different calendars. In such a case, if human understandability becomes an issue, the span can be converted to a single calendric granularity using the conversion procedure described in Section 3.2.

4 Anchored Temporal Entities

We identify a *time interval* as the basic anchored specification of time; it is a duration of time between two specific anchor points which stand for the lower and upper bounds of the interval, e.g., $[June\ 15,\ 1995, July\ 31,\ 1995]$. A *time instant* is a specific anchored moment in time[1]. For example, the anchor points of the time interval $[June\ 15,\ 1995, July\ 31,\ 1995]$ are the time instants $June\ 15,\ 1995$ and $July\ 31,\ 1995$. We model a time instant as a special case of a (closed) time

[1]In this paper, a time instant refers to the beginning of the period it denotes. Therefore the time instants 1995, *January* 1995, and *January* 1, 1995 are equivalent and refer to the beginning of the year 1995. A discussion on time instants which refer to the whole period they denote is given in [GLÖS95].

interval which has the same lower and upper bounds. For example, the time instant *June* 15, 1995 is equivalent to the time interval [*June* 15, 1995, *June* 15, 1995]. Since a time interval is represented by two anchored instants, it is sufficient to show how a time instant is represented within the context of a calendar. The representation of time intervals is merely the representation of its two anchored time instants.

4.1 Representation of Time Instants

Figure 3 shows the structural representation of a time instant.

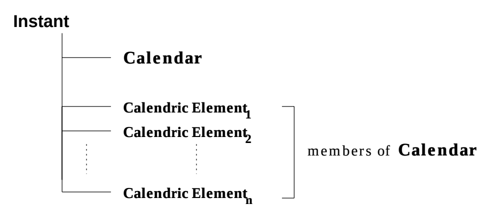

Figure 3: Structural representation of a time instant.

Every time instant belongs to a specific calendar and is composed of calendric elements which belong to different calendric granularities of the same calendar. Table 2 gives examples of time instants, the calendar they belong to, the calendric elements they are composed of and the respective calendric granularities.

Instant	Calendar	Calendric Elements	Calendric Granularities
June 15, 1995	Gregorian	15 June 1995	*Day* *Month* *Year*
Fall, 1995	Academic	Fall 1995	*Semester* *AcademicYear*

Table 2: Examples of time instants.

4.2 Operations on Time Instants

A wide range of operations can be performed on time instants:

1. A time instant can be compared with another time instant with the transitive comparison operators $<$ and $>$. This comparison operator forms a total order between time instants.

2. A time instant can be subtracted from another time instant to find the elapsed time between the two.

3. A time span can be added or subtracted to (from) a time instant to return another time instant.

Operands in the operations between time instants may belong to different calendars. Similarly, operations between spans and time instants may involve spans composed of calendric granularities belonging to different calendars. Therefore, to carry out operations on time instants, it may be necessary to convert time instants from one calendar to another. In the following section, we give the detailed conversion functions which enable an instant to be converted from one calendar to another.

4.2.1 Conversion of Time Instants

To convert a time instant from one calendar to another, the time instant is first mapped to a real value on the global time axis (G_{tl}). This value is then mapped to a time instant in the calendar of interest. Therefore, functions are defined to convert a time instant to its respective value on G_{tl}, and inverse functions are defined to convert a value on G_{tl} to an instant in a particular calendar. To simplify the description, we first give the functions for a simple calendar and then generalize them for any given calendar.

Derivation 4.1 *Mapping a time instant to G_{tl} – Simple calendar:* Let C be a calendar with the calendric granularities *year*, *month* and *day*. The following functions are defined in C:

$$\begin{aligned} f_C^1(y) &\rightarrow N_{months} \\ f_C^2(y,m) &\rightarrow N_{days} \\ f_C^3(y,m,d) &\rightarrow R \end{aligned}$$

where y, m, and d are ordinal values of calendric elements in the calendric granularities *year*, *month*, and *day*, respectively. Since a time instant is represented in terms of the calendric elements of a calendar, we can write any time instant in C using the ordinal values of calendric elements as (y, m, d). For example, the time instant September 12, 1995 is written as $(414, 9, 12)$.

Now, to map the time instant (y, m, d) to G_{tl} the R value for all days up to year y is first calculated. This is given by the summation:

$$\sum_{a_1=1}^{y-1} \sum_{a_2=1}^{f_C^1(a_1)} \sum_{a_3=1}^{f_C^2(a_1,a_2)} f_C^3(a_1,a_2,a_3) \tag{19}$$

Formula 19 calculates the R value up to year y by summing R for every day, in every month, of every year up to year y. Next, the R value for all days in all months up to month m in year y is calculated:

$$\sum_{a_1=1}^{m-1} \sum_{a_2=1}^{f_C^2(y,a_1)} f_C^3(y, a_1, a_2) \tag{20}$$

This formula calculates the R value in year y by summing R for every day, in every month up to month m of year y. Lastly, the R value for all days up to day d in month m is calculated:

$$\sum_{a_1=1}^{d-1} f_C^3(y, m, a_1) \tag{21}$$

Formula 21 calculates the R value in month m by summing R for every day up to day d in month m of year y. The R value corresponding to the time instant (y, m, d) is then obtained by summing Formulas 19, 20, and 21 to the origin of calendar C. We now consider the mapping of a time instant belonging to any general calendar to G_{tl}.

Derivation 4.2 *Mapping a time instant to G_{tl} – General calendar:* Let C be a calendar with origin $\mathcal{O}_C$, and conversion functions $f_C^1(i_1), f_C^2(i_1, i_2), \ldots, f_C^n(i_1, i_2, \ldots, i_n)$ (see Definition 2.4 in Section 2.2). Additionally, let $(i_1, \ldots, i_n)$ be a time instant in C. Then, $R(i_1, \ldots, i_n)$, the R value for the time instant $(i_1, \ldots, i_n)$ is given by:

$$\begin{aligned} R(i_1, \ldots, i_n) = {} & \mathcal{O}_C + \\ & \sum_{k=1}^{n} \left(\sum_{a_k=1}^{i_k-1} \sum_{a_{k+1}=1}^{f_C^k(i_1,\ldots,i_{k-1},a_k)} \cdots \sum_{a_n=1}^{f_C^{n-1}(i_1,\ldots,i_{k-1},a_k,\ldots,a_{n-1})} f_C^n(i_1, \ldots, i_{k-1}, a_k, \ldots, a_n) \right) \end{aligned} \tag{22}$$

Formula 22 first calculates the R value of the time instant up to the calendric element i_n followed by the R value in i_n, up to the calendric element i_{n-1}. This procedure is repeated up to the finest calendric granularity, i.e., up to calendric element i_1. We now show how a real value on G_{tl} is converted to a time instant in any given calendar.

Derivation 4.3 *Mapping a real value from G_{tl} to a time instant – Simple Calendar:* Let C be a calendar with origin $\mathcal{O}_C$, calendric granularities *year*, *month*, and *day*, and r be a real value on G_{tl}. Then the following formulas calculate a time instant in C corresponding to r:

$$\begin{aligned} Y &= \max_{y \in \mathbf{Z}} \{ y \mid R(y, 1, 1) \leq r - \mathcal{O}_C \} \\ M &= \max_{m \in \mathbf{Z}} \{ m \mid R(Y, m, 1) \leq r - \mathcal{O}_C \} \\ D &= \max_{d \in \mathbf{Z}} \{ d \mid R(Y, M, d) \leq r - \mathcal{O}_C \} \end{aligned}$$

The above formulas first find the maximum year Y which when mapped to G_{tl} gives a real value which is less than or equal to $r - \mathcal{O}_C$. The trick here is to vary the year value (y) in $R(y, m, d)$ (Formula 22) and keep the month (m) and day (d) values constant at 1. After having found Y, the maximum month in year Y which when mapped to G_{tl} gives a real value which is less than or equal to $r - \mathcal{O}_C$ is then calculated. In this case, the year value in $R(y, m, d)$ is kept fixed at Y, the month value is changed, and the day value is kept constant at 1. Finally, the maximum day in year Y and month M which when mapped to G_{tl} gives a real value which is less than or equal to $r - \mathcal{O}_C$ is calculated.

Derivation 4.4 *Mapping a real value from G_{tl} to a time instant – General Calendar:* Let C be a calendar with origin $\mathcal{O}_C$, and calendric granularities $G_1, G_2, \ldots, G_n$, where G_1 is the coarsest calendric granularity and G_n is the finest calendric granularity. Additionally, let r be a real value on G_{tl}. Then the following formulas calculate a time instant $(i_1, \ldots, i_n)$ in C corresponding to r:

$$\begin{aligned}
i_1 &= \max_{a \in \mathbf{Z}} \{a \mid R(a, \underbrace{1, \ldots, 1}_{n-1}) \leq r - \mathcal{O}_C\} \\
&\vdots \\
i_k &= \max_{a \in \mathbf{Z}} \{a \mid R(i_1, i_2, \ldots, i_{k-1}, a, \underbrace{1, \ldots, 1}_{n-k}) \leq r - \mathcal{O}_C\} \\
&\vdots \\
i_n &= \max_{a \in \mathbf{Z}} \{a \mid R(i_1, i_2, \ldots, i_{n-1}, a) \leq r - \mathcal{O}_C\}
\end{aligned}$$

Having defined the conversion functions necessary to convert a time instant from one calendar to another, in the following sections, we first show how the different operations on time instants are carried out when both operands belong to the same calendar, and then give algorithms to show how the operations are carried out when the operands belong to different calendars.

4.2.2 Comparison between Time Instants

We first assume that the instants belong to the same calendar. Let $I^1_{G_A} = (i_1, \ldots, i_m)$ and $I^2_{G_B} = (i'_1, \ldots, i'_n)$ be two time instants, with finest granularities G_A and G_B, respectively. We also assume without loss of generality that $m \geq n$. Then, the following algorithm checks if $I^1_{G_A} \leq I^2_{G_B}$:

Algorithm 4.1 *Comparison of instants belonging to the same calendar:*

$$I^2_{G_B} := (i'_1, \ldots, i'_n, \underbrace{1, \ldots, 1}_{m-n})$$
$$I^1_{G_A} \leq I^2_{G_B} \text{ iff } i_j \leq i'_j \ \forall j \mid 1 \leq j \leq m$$

The algorithm basically compares the time instants by comparing each of their calendric elements. The instant $I^2_{G_B}$ is adjusted by adding the calendric element with the ordinal number 1 until its finest granularity is the same as that of $I^1_{G_A}$. This is reasonable because a time instant refers to the beginning of the time period it denotes.

Example 4.1 Suppose we have the following time instants and the ordinal values of their respective calendric elements:

$June\ 5,\ 1990 \equiv (409, 6, 5)$

$June\ 15,\ 1990 \equiv (409, 6, 15)$

$June\ 1990 \equiv (409, 6)$

$1990 \equiv (409)$

Then,

$June\ 5,\ 1990 < June\ 15,\ 1990$ because $(409, 6, 5) < (409, 6, 15)$.

$June\ 1990 < June\ 15,\ 1990$ because $(409, 6) \equiv (409, 6, 1)$, and $(409, 6, 1) < (409, 6, 15)$.

$1990 < June\ 15,\ 1990$ because $(409) \equiv (409, 1, 1)$, and $(409, 1, 1) < (409, 6, 1)$.

We now look at the case when the two instants belong to different calendars. Let $(i_1, \ldots, i_n)$ and $(i'_1, \ldots, i'_m)$ be two time instants belonging to calendars C_1 and C_2, respectively. The algorithm to compare $(i_1, \ldots, i_n)$ and $(i'_1, \ldots, i'_m)$ is:

Algorithm 4.2 *Comparison of instants belonging to different calendars:*

$$r_1 := R(i_1, \ldots, i_n)$$
$$r_2 := R(i'_1, \ldots, i'_m)$$
$$\texttt{Compare } r_1 \text{ and } r_2$$

Algorithm 4.2 makes use of the global time axis which provides a homogeneous underlying platform on which time instants can be mapped. The two time instants are first converted to their respective real values on the global time axis using Derivation 4.2. These real values are then compared.

4.2.3 Elapsed Time between Time Instants

Let $(i_1, \ldots, i_n)$ and $(i'_1, \ldots, i'_n)$ be two time instants belonging to the same calendar. Then:

$$\texttt{Elapsed}((i_1, \ldots, i_n), (i'_1, \ldots, i'_n)) = \sum_{j=1}^{n} (K_j \cdot G_j), \text{where } K_j = i'_j - i_j$$

The following examples illustrate the various cases that can take place:

Example 4.2 $\texttt{Elapsed}((June\ 1,\ 1990),\ (July\ 31,\ 1995)) \Rightarrow (5\ years,\ 1\ month,\ 30\ days)$

This is the simplest case in which both instants have the same finest granularity. The calendric elements of the first time instant are simply subtracted from the corresponding calendric elements of the second time instant.

Example 4.3 $\texttt{Elapsed}((June\ 1990),\ (July\ 31,\ 1995)) \Rightarrow \texttt{Elapsed}((June\ 1,\ 1990),\ (July\ 31,\ 1995))$ $\Rightarrow (5\ years,\ 1\ month,\ 30\ days)$

Here, the finest calendric granularity of $June$ 1990 is coarser than that of $July$ 31, 1995. Thus, $June$ 1990 is first replaced by the time instant $June$ 1, 1990, its equivalent time instant with the finest granularity of *days*. The elapsed time between $June$ 1, 1990 and $July$ 31, 1995 is then calculated as shown in the previous example.

Example 4.4 $\texttt{Elapsed}((June\ 1990),\ (June\ 15,\ 1990)) \Rightarrow \texttt{Elapsed}((June\ 1,\ 1990),\ (June\ 15,\ 1990))$ $\Rightarrow 14\ days$

This example is similar to the one above in that the finest calendric granularity of the first instant is coarser than that of the second.

We now look at the case when the instants belong to different calendars. Let $(i_1, \ldots, i_n)$ and $(i'_1, \ldots, i'_m)$ be two time instants belonging to calendars C_1 and C_2, respectively. The algorithm to find the elapsed time between $(i_1, \ldots, i_n)$ and $(i'_1, \ldots, i'_m)$ is:

Algorithm 4.3 *Elapsed time between instants belonging to different calendars:*

$\texttt{Convert}$ $(i'_1, \ldots, i'_m)$ to $(i''_1, \ldots, i''_n)$ using Derivations 4.2 and 4.4

$S^N := \texttt{Elapsed}((i_1, \ldots, i_n),\ (i''_1, \ldots, i''_n))$

The algorithm first converts the time instant $(i'_1, \ldots, i'_m)$ to its equivalent counterpart in calendar C_1, $(i''_1, \ldots, i''_n)$ using Derivations 4.2 and 4.4. It then finds the elapsed time between $(i_1, \ldots, i_n)$ and $(i''_1, \ldots, i''_n)$. The result is a time span, S^N, which is of the form shown in formulas (1) or (2) (see Section 3.1).

4.2.4 Operations between Spans and Time Instants

In performing arithmetic operations that involve both spans and time instants[2], if all the calendric granularities of the span belong to the same calendar as the instant, then there are two cases to consider.

If the finest calendric granularity of the span is coarser than or the same as the finest calendric granularity of the instant, then each component of the span is simply added to the corresponding calendric element of the time instant.

Example 4.5 If the span is 2 *months* and the instant is *June* 15, 1995, then adding 2 *months* to *June* 15, 1995 simply results in the time instant *August* 15, 1995. Similarly, adding the span 16 *days* to *June* 15, 1995 results in the time instant *July* 1, 1995.

If the finest calendric granularity of the span is finer than the finest calendric granularity of the time instant, then the time instant is first replaced by an equivalent time instant whose finest granularity is the same as that of the span, and the addition is then carried out.

Example 4.6 If the span is 2 *months* + 5 *days* and the time instant is *June*, 1995, then the time instant is first replaced by its equivalent time instant *June* 1, 1995. The addition of the span 2 *months* + 5 *days* to this time instant results in the time instant *August* 6, 1995.

Now consider the situation when the calendric granularities of the span do not belong to the same calendar as the instant. Let S be a span of the form:

$$\begin{aligned} S &= \sum_{i=1}^{N}\sum_{j=1}^{M}(K_j^{C_i} \cdot G_j^{C_i}) \\ &= \sum_{j=1}^{M}(K_j^{C_1} \cdot G_j^{C_1}) + \ldots + \sum_{j=1}^{M}(K_j^{C_N} \cdot G_j^{C_N}) \\ &= S_{C_1} + S_{C_2} + \ldots + S_{C_N} \end{aligned}$$

Basically S_{C_i} is a span composed of calendric granularities belonging to calendar C_i. The algorithm for adding span S to a time instant I_{C_A} (a time instant belonging to calendar C_A) is as follows:

[2] We only consider operations in which the span is determinate. If the span is indeterminate, the arithmetic operation results in an indeterminate time instant. An example of an indeterminate time instant is $July, 1\ 1995 \sim July, 31\ 1995$ which denotes any time between $July, 1\ 1995$ and $July, 31\ 1995$. Indeterminate time instants are discussed in detail in [GLÖS95].

Algorithm 4.4 *Addition of a span to an instant:*

$I'_{C_A} := I_{C_A}$

repeat for $i := 1$ *to* N

  `Convert` I'_{C_A} to I_{C_i} using Derivations 4.2 and 4.4

  $I'_{C_i} := S_{C_i} + I_{C_i}$

  `Convert` I'_{C_i} to $I^i_{C_A}$ using Derivations 4.2 and 4.4

  $I'_{C_A} := I^i_{C_A}$

return I'_{C_A}

For each span component, S_{C_i}, the algorithm converts the time instant belonging to calendar C_A to a corresponding time instant in calendar C_i, adds it to S_{C_i} and converts the resulting instant back to an instant of calendar C_A. The algorithm for subtracting span S from a time instant I_{C_A} is similar. That is, $I_{C_A} - S = I_{C_A} + (-S)$.

5 Incorporating Calendars and Temporal Entities in an Object Model

In this section we describe how the calendar model, and the anchored and unanchored temporal primitives introduced in Sections 2–4 are incorporated into an object model. Our work is done within the framework of the TIGUKAT[3] [ÖPS+95] system which is under development at the University of Alberta; however, it is applicable to any object DBMS with similar characteristics. In the following sections, we first give a brief overview of TIGUKAT and then show the actual mapping between various temporal notions introduced so far and TIGUKAT types and behaviors. Types relevant to the representation of temporal information are depicted in Figures 4-6, along with their subtyping relationships. Likewise, every operation defined in this paper has a corresponding TIGUKAT behavior. These behaviors (along with their signatures) are given in Tables 3-5. The detailed time type system can be found in [GLÖS95].

[3]TIGUKAT (tee-goo-kat) is a term in the language of Canadian Inuit people meaning "objects." The Canadian Inuits, commonly known as Eskimos, are native to Canada with an ancestry originating in the Arctic regions of the country.

5.1 The TIGUKAT Object Model Overview

The TIGUKAT object model [Pet94] is purely *behavioral* with a *uniform* object semantics. The model is *behavioral* in the sense that all access and manipulation of objects is based on the application of behaviors to objects. The model is *uniform* in that every component of information, including its semantics, is modeled as a *first-class object* with well-defined behavior. Other typical object modeling features supported by TIGUKAT include strong object identity, abstract types, strong typing, complex objects, full encapsulation, multiple inheritance, and parametric types.

The primitive objects of the model include: *atomic entities* (reals, integers, strings, etc.); *types* for defining common features of objects; *behaviors* for specifying the semantics of operations that may be performed on objects; *functions* for specifying implementations of behaviors over types; *classes* for automatic classification of objects based on type[4]; and *collections* for supporting general heterogeneous groupings of objects. In this paper, a reference prefixed by "`T_`" refers to a type, "**C_**" to a class, "*B_*" to a behavior, and "`T_X`< `T_Y` >" to the type `T_X` parameterized by the type `T_Y`. For example, `T_person` refers to a type, **C_person** to its class, *B_age* to one of its behaviors and `T_collection`< `T_person` > to the type of collections of persons. A reference such as David, without a prefix, denotes some other application specific reference.

The access and manipulation of an object's state occurs exclusively through the application of behaviors. We clearly separate the definition of a behavior from its possible implementations (functions). The benefit of this approach is that common behaviors over different types can have a different implementation in each of the types. This provides direct support for behavior *overloading* and *late binding* of functions (implementations) to behaviors.

The model separates the definition of object characteristics (a *type*) from the mechanism for maintaining instances of a particular type (a *class*). A *type* defines behaviors and encapsulates behavior implementations and state representation for objects created using that type as a template. The behaviors defined by a type describe the *interface* to the objects of that type.

In addition to classes, a *collection* is defined as a general grouping construct. It is similar to a *class* in that it groups objects, but it differs in some respects. First, object creation cannot occur through a collection, only through classes. Second, an object may exist in any number of collections, but is a member of the shallow extent of only one class. Third, classes are automatically managed by the system based on the subtype lattice whereas the management of collections is *explicit*,

[4]Types and their extents are separate constructs in TIGUKAT.

meaning the user is responsible for their extents. Finally, the elements of a class are homogeneous up to inclusion polymorphism, while a collection may be heterogeneous in the sense that it may contain objects of types that are not in a subtype relationship with one another.

5.2 Calendars

We start with a description of how our model of calendars is incorporated into the TIGUKAT object model. The type `T_calendar` models different kinds of calendars. It is a direct subtype of the `T_object` type as shown in Figure 4. Behaviors defined on `T_calendar` are shown in Table 3.

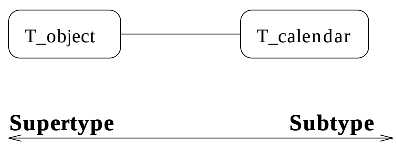

Figure 4: The calendar type.

Behavior *B_name* returns the name of a calendar e.g., *Gregorian*, *Academic*. *B_origin* returns the origin of the calendar in terms of a span. *B_calGranularities* returns a totally ordered collection of the calendric granularities of the calendar. For example, *B_calGranularities* of the Gregorian calendar shown in Figure 2 returns $\{G_{Year}, G_{Month}, G_{Day}, G_{Hour}\}$. Finally, behavior *B_convFunctions* returns a list of the conversion functions described in Section 2.

`T_calendar`	*B_name*:	`T_string`
	B_origin:	`T_span`
	B_calGranularities:	T_orderedColl⟨T_calGranularity⟩
	B_convFunctions:	T_list⟨T_function⟩

Table 3: Behaviors defined on calendars.

5.3 Spans

We now look at the types related to spans. These are shown in Figure 5. The various behaviors on time spans together with their signatures are shown in Table 4.

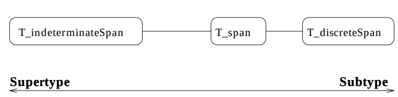

Figure 5: Span types.

The type T_indeterminateSpan is introduced to model indeterminate time spans. Behaviors defined on T_indeterminateSpan include *B_lessthan* and *B_greaterthan* which model the comparison operations on time spans. Behaviors *B_add* and *B_subtract* allow determinate spans to be added to and subtracted from indeterminate spans, respectively.

T_indeterminateSpan has the subtype T_span which models continuous determinate spans. This subtyping relationship has the following justification: Every determinate span can be treated as an indeterminate one (with identical lower and upper bounds).

T_indeterminateSpan	*B_lessthan*:	T_indeterminateSpan → T_boolean
	B_greaterthan:	T_indeterminateSpan → T_boolean
	B_add:	T_span → T_indeterminateSpan
	B_subtract:	T_span → T_indeterminateSpan
	B_lowerBound:	T_span
	B_upperBound:	T_span
T_span	*B_add*:	T_span → T_span
	B_subtract:	T_span → T_span
	B_calGranularities:	T_collection⟨T_calGranularity⟩
	B_coefficient:	T_calGranularity → T_real
	B_multiply:	T_real → T_span
	B_divide:	T_real → T_span
	B_convertTo:	T_calGranularity → T_indeterminateSpan
T_discreteSpan	*B_add*:	T_discreteSpan → T_discreteSpan
	B_subtract:	T_discreteSpan → T_discreteSpan
	B_coefficient:	T_calGranularity → T_integer
	B_multiply:	T_integer → T_discreteSpan
	B_succ:	T_span
	B_pred:	T_span

Table 4: Behaviors defined on time spans.

Behaviors *B_add* and *B_subtract* are refined in T_span to take a continuous determinate span as an argument and return a continuous determinate span as the result. Behaviors *B_calGranularities*, *B_coefficient*, *B_multiply*, *B_divide* and *B_convertTo* in T_span are used in the conversion process of a time span to a specific calendric granularity as shown in Section 3.1. *B_calGranularities* returns a collection of calendric granularities in a time span. For example, the behavior application $(1\ month + 5\ days)\cdot$ *B_calGranularities* returns $\{G_{day}, G_{month}\}$. The behavior *B_coefficient* returns the (real) coefficient of a time span given a specific calendric granularity. For example, $(1\ month + 5\ days)\cdot$ *B_coefficient*(G_{day}) returns 5.0. Behaviors *B_multiply* and *B_divide* are basically used in the conversion process. The *B_convertTo* behavior is derived from the rest of the behaviors in T_span and essentially converts a determinate time span to an indeterminate time span with the specified calendric granularity.

The type T_discreteSpan is defined as a subtype of the T_span type described above. Behaviors *B_add* and *B_subtract* are refined in T_discreteSpan to take a discrete determinate span as an argument and return a discrete determinate as a result. Behavior *B_coefficient* is refined to return the integer coefficient of a discrete time span and the *B_multiply* behavior is refined to multiply an integer by a discrete time span. Behaviors *B_succ* and *B_pred* are defined in `T_discreteSpan` to return the next or previous discrete time span of a particular discrete time span. For example, $(2\ months + 45\ hours)\cdot$ *B_succ* returns the time span $3\ months + 46\ hours$ while $(2\ months + 45\ hours)\cdot$ *B_pred* returns the time span $1\ month + 44\ hours$.

5.4 Calendric Granularities

Recall that a calendric granularity in our framework is a special kind of a determinate span. Therefore, we define the type `T_calGranularity` as a subtype of `T_discreteSpan` as shown in Figure 6.

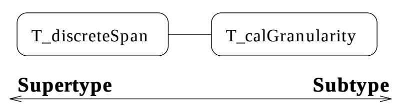

Figure 6: Calendric Granularity types.

Instances of `T_calGranularity` represent the different kinds of calendric granularities, e.g., *year*, *hour*, *semester*. Behaviors on calendric granularities are shown in Table 5.

`T_calGranularity`	*B_calendar*:	`T_calendar`
	B_similarCalGran:	T_collection⟨T_calGranularity⟩
	B_calElements:	T_list⟨T_calElement⟩
	B_lowerBound:	`T_calGranularity`
	B_upperBound:	`T_calGranularity`
	B_exactlyConvertibleTo:	`T_calGranularity` → `T_boolean`
	B_i-lbf:	`T_integer` → `T_calGranularity` → `T_integer`
	B_r-lbf:	`T_real` → `T_calGranularity` → `T_real`
	B_i-ubf:	`T_integer` → `T_calGranularity` → `T_integer`
	B_r-ubf:	`T_real` → `T_calGranularity` → `T_real`

Table 5: Behaviors defined on calendric granularity.

Behavior *B_calendar* in `T_calGranularity` returns the calendar which the calendric granularity belongs to. Behavior *B_similarCalGran* returns the set of calendric granularities that have similar duration as a particular calendric granularity. Behavior *B_calElements* returns an ordered collection of the calendric elements of a calendric granularity. For example *B_calElements* applied on the calendric granularity *semester* returns $\langle Fall, Winter, Spring, Summer\rangle$. The *B_lowerBound* and *B_upperBound* behaviors are refined accordingly to return a calendric granularity as the lower and upper bound of a calendric granularity. Since the calendric granularity is determinate, these behaviors return the same value. The behavior *B_exactlyConvertibleTo* checks if a calendric granularity is exactly convertible to another calendric granularity. For example, G_{day}. *B_exactlyConvertibleTo*(G_{hour}) returns `True`, while G_{month}. *B_exactlyConvertibleTo*(G_{day}) returns `False`. `T_calGranularity` also defines new behaviors, *B_i-lbf*, *B_r-lbf*, *B_i-ubf* and *B_r-ubf*. *B_i-lbf* and *B_i-ubf* return the lower and upper bound factors (see Section 3.1) of two calendric granularities with integer coefficients. Similarly, *B_r-lbf* and *B_r-ubf* return the lower and upper bound factors of two calendric granularities with real coefficients. For example, G_{month}. *B_i-lbf*$(1, G_{day})$ returns 28 while G_{month}. *B_i-ubf*$(1, G_{day})$ returns 31.

5.5 Time Instants

In our framework, every instant can be treated as an interval with identical lower and upper bounds. Therefore, the `T_instant` is defined to model time instants and is a direct subtype of the `T_interval` type which models time intervals.

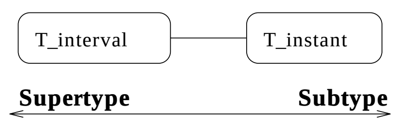

Figure 7: Instant types.

Behaviors defined in `T_instant` as shown in Table 6 are the comparison behaviors *B_less*, *B_greater*, *B_leq* and *B_geq* (these are essentially the $<, >, \leq$ and $\geq$ operators, respectively), the *B_elapsed* behavior which returns the elapsed time (duration) between two time instants, and the *B_add* and *B_subtract* which are used in arithmetic operations between time instants and time spans. Behavior *B_calendar* returns the calendar which the instant belongs to and behavior *B_calElements* returns a list of the calendric elements in a time instant. For example, *B_calElements* applied to the instant June 15, 1995 returns the list $(414, 6, 15)$.

6 Related Work

Although there have been a substantial number of proposals on adding time to object models [RS91, RS93, KS92, WD92, DW92, SC93, CG93], it is quite surprising to note that none of them provides comprehensive support for modeling multiple calendars and handling multiple granularities. Most of these models assume the presence of an underlying calendar (usually Gregorian) which has

T_instant	*B_less*:	T_instant → T_boolean
	B_greater:	T_instant → T_boolean
	B_leq:	T_instant → T_boolean
	B_geq:	T_instant → T_boolean
	B_elapsed:	T_instant → T_span
	B_add:	T_span → T_instant
	B_subtract:	T_span → T_instant
	B_calendar:	T_calendar
	B_calElements:	T_list⟨T_calElement⟩

Table 6: Behaviors defined on time instants.

a prefixed set of granularities. For example, in [RS91] the presence of the Gregorian calendar with granularities *Year*, *Month*, *Day*, *Week*, *Hour*, *Minute*, and *Second* is assumed. Translations between granularities in operations are automatically provided, with the default being to convert to the coarser granularity. It is not clear how these translations are carried out though.

Most of the research on temporal relational models has concentrated on modeling temporal information with a single underlying granularity. There have been some recent proposals however, that handle multiple granularities.

Clifford and Rao [CR87] introduce a general structure for time domains called a *temporal universe*. A temporal universe consists of a totally ordered set of granularities. Operations are defined on a temporal universe, which basically convert different *anchored* times to a (common) finer granularity before carrying out the operation. Wiederhold et al., [WJL91] also examine the issue of multiple granularities. An algebra is described that allows the conversion of event times to an interval representation. This usually involves converting the coarser granularity to the finer granularity in light of the semantics of the time varying domains. [WJS93] extend this work by providing semantics for moving up and down a granularity lattice. In [BP85] the issues of absolute, relative, imprecise and periodic times are discussed. Multiple granularities are supported for each time. Operands (which are anchored) in operations involving mixed granularities are converted to the coarser granularity to avoid indeterminacy. In a more recent work [MPB92], the existence of a minimum underlying granularity (quantum of time) to which time is mapped, is assumed. Montanari et al., [MMCR92] examined the issue of multiple granularities, but considered exact granularity conversions only. In [CSS94], structured collections of time intervals are defined and termed as "calendars." Corsetti et al., [CMR91] deal with different time granularities in specifications of real-time systems.

None of the above works considers granularity conversions in terms of unanchored durations of time (this includes TSQL2 [Sno95b], discussed below). To the best of our knowledge, this feature is novel to our work. In the above works, granularities are treated as partitionings on the timeline whereas in our work, granularities are treated as unit spans. Hence, while we consider both anchored and unanchored granularity conversions, other proposals only consider anchored granularity conversions. Furthermore, none of the above works address the issues of integration of granularities from multiple calendars.

Our work is closest to that of TSQL2 [Sno95b] in that TSQL2 supports multiple granularities and multiple calendars. However, there are some differences between our approach and theirs:

- A notable problem in TSQL2 is that a granularity is an anchored partitioning on the timeline, whereas a span is an unanchored duration of time. Consequently, conversions between spans is not possible. More specifically, all conversions of spans (from both finer to coarser granularity and vice-versa) gives rise to indeterminate spans as a default. For example, the span 1 *day* when converted to the granularity of hours results in the span $0 \sim 23$ *hours*. On the contrary, in our approach a granularity is a special kind of span with unit duration. Hence, we allow both exact and inexact conversions between spans (see Section 3). For example, the span 1 *day* when converted to the granularity of hours results in the span 24 *hours* in our approach.

- Before every binary operation involving spans they convert both operands to a common granularity. Our approach allows us to avoid these conversions, thereby preventing information loss. For example, TSQL2 cannot represent time spans like 1 *month and* 1 *day*. This would have to be represented as days (assuming the underlying base granularity is a day), necessarily leading to an indeterminate span and thereby losing information.

- TSQL2 treats all instants as indeterminate. In contrast, we treat our instants as indeterminate only in certain operations as shown in Section 4. In all other operations, especially those involving operands having granularities belonging to different calendars, we treat instants as determinate. This is because we use the calendric functions defined in Section 2 to map instants to their respective values on a global timeline when performing operations that have operands of differing granularities belonging to possibly different calendars.

7 Conclusion

In this paper we have gone back to the basic modeling of temporal entities. We have identified the various issues which arise when trying to accommodate the various characteristics of temporal entities and shown that they have not been resolved comprehensively. In light of this, we have provided a model for supporting calendars and have shown how multiple granularities are integrated within calendars. We have further shown how anchored and unanchored temporal entities are represented within the context of calendars. We described a calendric granularity as being part of a calendar and represented it as a special kind of span - a span with a unit duration. We then showed how conversions between spans of mixed granularities is carried out. The semantics of operations involving anchored and unanchored information were then given by utilizing appropriate algorithms. Finally, we showed how the model of calendars, and anchored and unanchored information can be implemented within an object model.

Modeling multiple granularities also results in temporal indeterminacy. For example, if the condition of a patient is checked on an hourly basis and it was noticed that a patient's condition was significantly worse at a particular hour, say $6am\ May$ 30, one can only reasonably conclude that his condition deteriorated sometime between $5:00am\ May$ 30 and $5:59am\ May$ 30.In this paper, we have concentrated on modeling multiple granularities and giving a comprehensive solution to the issues that arise therein. In [GLÖS95] we discuss our model of supporting indeterminate temporal entities.

In conclusion we emphasize that modeling of the basic anchored and unanchored temporal entities is an essential ingredient in the design of temporal models and temporal query languages. It is our position that assuming a simplistic view of the underlying temporal entities and thereby avoiding the inherent issues which arise will only make the resulting temporal model and temporal query language very restricted for real-world temporal data usage.

References

[BP85] F. Barbic and B. Pernici. Time Modeling in Office Information Systems. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pages 51–62, Austin, Texas, May 1985.

[CG93] T.S. Cheng and S.K. Gadia. An Object-Oriented Model for Temporal Databases. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, pages N1–N19, Arlington, Texas, June 1993.

[CMR91] E. Corsetti, A. Montanari, and E. Ratto. Dealing with Different Time Granularities in Formal Specifications of Real-Time Systems. *The Journal of Real-Time Systems*, 3:191–215, 1991.

[CR87] J. Clifford and A. Rao. A Simple, General Structure for Temporal Domains. In C. Rolland, F. Bodart, and M. Leonard, editors, *Temporal Aspects in Information Systems*, pages 17–30. North-Holland, 1987.

[CS93] R. Chandra and A. Segev. Managing Temporal Financial Data in an Extensible Database. In *Proc. 19th Int'l Conf. on Very Large Data Bases*, August 1993.

[CSS94] R. Chandra, A. Segev, and M. Stonebraker. Implementing Calendars and Temporal Rules in Next-Generation Databases. In *Proc. 10th Int'l. Conf. on Data Engineering*, February 1994.

[DW92] U. Dayal and G. Wuu. A Uniform Approach to Processing Temporal Queries. In *Proc. 18th Int'l Conf. on Very Large Data Bases*, pages 407–418, August 1992.

[GLÖS95] I.A. Goralwalla, Y. Leontiev, M.T. Özsu, and D. Szafron. A Uniform Behavioral Temporal Object Model. Technical Report TR-95-13, University of Alberta, May 1995.

[Kli93] N. Kline. An Update of the Temporal Database Bibliography. *ACM SIGMOD Record*, 22(4):66–80, December 1993.

[KS92] W. Kafer and H. Schoning. Realizing a Temporal Complex-Object Data Model. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pages 266–275, 1992.

[MMCR92] A. Montanari, E. Maim, E. Ciapessoni, and E. Ratto. Dealing with Time Granularity in Event Calculus. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 702–712, June 1992.

[MPB92] R. Maiocchi, B. Pernici, and F. Barbic. Automatic Deduction of Temporal Information. *ACM Transactions on Database Systems*, 17(4):647–688, 1992.

[ÖPS+95] M.T. Özsu, R.J. Peters, D. Szafron, B. Irani, A. Lipka, and A. Munoz. TIGUKAT: A Uniform Behavioral Objectbase Management System. *The VLDB Journal*, 4:100–147, August 1995.

[ÖS95] G. Özsoyoğlu and R. Snodgrass. Temporal and Real-Time Databases: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, August 1995.

[Pea94] N. Pissinou and et. al. Towards an Infrastructure for Temporal Databases: Report of an Invitational ARPA/NSF Workshop. *ACM SIGMOD Record*, 23(1):35–51, March 1994.

[Pet94] R.J. Peters. *TIGUKAT: A Uniform Behavioral Objectbase Management System.* PhD thesis, University of Alberta, 1994.

[RS91] E. Rose and A. Segev. TOODM - A Temporal Object-Oriented Data Model with Temporal Constraints. In *Proc. 10th Int'l Conf. on the Entity Relationship Approach*, pages 205–229, October 1991.

[RS93] E. Rose and A. Segev. TOOA - A Temporal Object-Oriented Algebra. In *Proc. European Conference on Object-Oriented Programming*, 1993.

[SC93] S.Y.W. Su and H.M. Chen. Modeling and Management of Temporal Data in Object-Oriented Knowledge Bases. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, pages HH1–HH18, Arlington, Texas, June 1993.

[Sno86] R. Snodgrass. Research Concerning Time in Databases: Project Summaries. *ACM SIGMOD Record*, 15(4), December 1986.

[Sno90] R. Snodgrass. Temporal Databases: Status and Research Directions. *ACM SIGMOD Record*, 19(4):83–89, December 1990.

[Sno95a] R. Snodgrass. Temporal Object-Oriented Databases: A Critical Comparison. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability and Beyond*, pages 386–408. Addison-Wesley/ACM Press, 1995.

[Sno95b] R. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.

[Soo91] M.D. Soo. Bibliography on Temporal Databases. *ACM SIGMOD Record*, 20(1):14–23, 1991.

[SS88] R. Stam and R. Snodgrass. A Bibliography on Temporal Databases1. *IEEE Database Engineering*, 7(4):231–239, December 1988.

[TCG+93] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993.

[WD92] G. Wuu and U. Dayal. A Uniform Model for Temporal Object-Oriented Databases. In *Proc. 8th Int'l. Conf. on Data Engineering*, pages 584–593, February 1992.

[WJL91] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with Granularity of Time in Temporal Databases. In R. Andersen, J.A. Bubenko Jr., and A. Solvberg, editors, *Advanced Information Systems Engineering, 3rd Int'l Conference CAiSE '91*, pages 124–140. Springer-Verlag, 1991.

[WJS93] X. Wang, S. Jajodia, and V. Subrahmanian. Temporal Modules: An Approach Toward Temporal Databases. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pages 227–236, 1993.