, , 1–24 ()

Temporal Granularity: Completing the Puzzle

IQBAL A. GORALWALLA*, YURI LEONTIEV, M. TAMER ÖZSU, DUANE SZAFRON

{iqbal,yuri,ozsu,duane}@cs.ualberta.ca

Laboratory for Database Systems Research
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1

CARLO COMBI (CORRESPONDING AUTHOR) combi@dimi.uniud.it

Laboratory of Database and Multimedia Systems
Department of Mathematics and Computer Science
University of Udine
via delle Scienze 206, 33100 Udine, Italy

Editor:

Abstract. *Granularity* is an integral feature of both *anchored* (e.g., 25 *October* 1995, *July* 1996) and *unanchored* (e.g., 3 *minutes*, 6 *hours* 20 *minutes*, 5 *days*) temporal data. In supporting temporal data that is specified in different granularities, numerous approaches have been proposed to deal with the issues of converting temporal data from one granularity to another. The emphasis, however, has only been on granularity conversions with respect to anchored temporal data. In this paper we provide a novel approach to the treatment of granularity in temporal data. A granularity is modeled as a special kind of unanchored temporal primitive that can be used as a unit of time. That is, a granularity is modeled as a *unit unanchored temporal primitive*. We show how unanchored temporal data is represented, give procedures for converting the data to a given granularity, provide canonical forms for the data, and describe how operations between the data are performed. We also show how anchored temporal data is represented at different granularities and give the semantics of operations on anchored temporal data.

Keywords: temporal databases, granularity, calendars

1. Introduction

We address the problem of modeling and managing, within a database management system (DBMS), anchored and unanchored temporal data of multiple granularities. Anchored data has a specific location on the time axis, while unanchored data has no specific location. For example, 31 *July* 1995 is an anchored temporal primitive in that we know exactly where it is located on the time axis, whereas 31 *days* is unanchored since we do not know where it is located on the time axis; it can stand for any block of 31 consecutive days. Anchored and unanchored temporal information is usually available in multiple granularities. Such information is prevalent in various sources. For example:

* currently with IBM Toronto Lab, IBM Software Solutions Division, North York, Ontario, Canada M3C 1H7, email: igoralwa@ca.ibm.com

- *clinical data* – Physicians usually specify temporal clinical information for patients with varying granularities [9, 10, 11]. For example, "the patient suffered from abdominal pain for 2 hours and 20 minutes on 15 June 1996," "in 1990, the patient took a calcium antagonist for 3 months," "in October 1993, the patient had a second heart seizure."

- *real-time systems* – A process is usually composed of sub-processes that evolve according to times that have different granularities [12]. For example, the temporal evolution of the basin in a hydroelectric plant depends on different subprocesses: the flow of water is measured daily; the opening and closing of radial gates is monitored every minute; and the electronic control which has a granularity of microsecond.

- *geographic information systems* – Geographic information is usually specified according to a varying time scale [14]. For example, vegetation fluctuates according to a seasonal cycle, while temperature varies daily.

- *office information systems* – temporal information is available in different time units of the Gregorian calendar [2, 8, 21]. For example, employee wages are usually recorded in the time unit of hours while the history of sales are categorized according to months.

Clearly many applications require support for both anchored and unanchored temporal primitives that are specified in different and mixed granularity. Without a sound and general treatment of temporal granularities, even if the choice of the most appropriate granularity (i.e., the finest) for a given application domain was possible (and this is not the case, for example, of medical applications), several problems would arise both in integrating data from heterogeneous systems based on different time units, and in converting a temporal primitive from some granularity to that of the system (e.g., *3 months* in days).

Although there have been various recent proposals that handle multiple granularities [2, 4, 7, 11, 21, 22, 23, 25, 26, 27], the focus has been mainly on representing anchored temporal primitives that are specified in different granularities. Granularity conversions are given for anchored temporal primitives only. However, supporting unanchored temporal primitives with different granularity is equally important [18]. All the proposals dealing with temporal granularity can be viewed as pieces of a puzzle: we contend that this puzzle will be completed only by supporting both anchored and unanchored temporal primitives with different granularities. In the next section we give a real-world example from clinical medicine that motivates our claim.

1.1. Motivation

We focus on a clinical example related to a patient with cardiological problems, particularly related to the widely known problem of diagnosing and following up unstable angina [1]. Unstable angina[1] is a transitory clinical syndrome usually

associated with an increased duration and/or intensity of symptoms related to coronary artery disease; risk of cardiac death and myocardial infarction increase. In this situation it is important to consider both the time when the symptoms (like chest pain) began and the time duration of these symptoms.

Let us consider the following sentences which are related to information contained in the cardiological medical record of a patient:

1. The patient suffered from chest pain at rest for 2 hours and 55 minutes on 13 December 1995.
2. The patient presented an episode of acute chest pain on 29 January 1996 from 13:20:15 to 13:56:23.
3. The patient has been admitted to an Intensive Care Unit from 21:00 29 January 1996, and he has undergone intensive medical management for 36 hours.
4. On 15 February 1996 the patient had myocardial infarction.
5. At 3 pm 12 April 1996 the patient presented a new episode of chest pain of 7 minutes and 35 seconds during a soft exertion.
6. From December 1994 to April 1996 the patient took aspirin.
7. From 30 January 1996 the patient had to take a thrombolytic therapy for 38 months.

We can observe from the above sentences that there are many different granularities for time instants (days in sentences 1, 4, and 7; seconds in sentence 2; minutes in sentence 3; hours in sentence 5; months in sentence 6) and different and mixed granularities for time spans (hours and minutes in sentence 1; hours in sentence 3; minutes and seconds in sentence 5; months in sentence 7). Moreover, in a single sentence there may be time instants and time spans having heterogeneous granularities (for example, in sentence 3 the time instant is specified at the granularity of minutes, while the time duration is specified at the granularity of hours).

In addition to the different and mixed granularities in the patient-related information, the definition of unstable angina itself involves time spans given at different granularities. Unstable angina is, in fact, defined as: (1) symptoms of angina at rest, for more than 20 *minutes*, or (2) new onset, within two *months*, of exertional angina, involving marked limitations of ordinary physical activity, or (3) increasing angina within two *months* from the initial presentation, or (4) post- myocardial infarction angina, i.e., angina occurring from 1 to 60 *days* after an acute myocardial infarction [1].

In a clinical setting, we need to be able to derive some extra information from the stored sentences about the patient. For example:

1. What is the time span between the myocardial infarction and the last episode of chest pain?

   To derive this, we need to compute the elapsed time (which is a time span) between the time instants 15 *February* 1996 and 3 *pm* 12 *April* 1996 (see sentences 4 and 5).

2. What is the global span of the symptoms of angina?

   To answer this question, the elapsed time between the time instants 13:20:15 29 *January* 1996 and 13:56:23 29 *January* 1996 has to be added to the time spans 7 *minutes and* 35 *seconds*, and 2 *hours and* 55 *minutes* (see sentences 1, 2, and 5).

3. When did the patient finish the intensive medical management and what is the time span between the end of the intensive medical management and the onset of the new angina episode?

   To answer the first part of the question, we need to add the time span 36 *hours* to the time instant 21:00 29 *January* 1996. The elapsed time between the resulting time instant and the time instant 3 *pm* 12 *April* 1996 gives the answer to the second part of the question (see sentences 3 and 5).

4. Was the patient taking aspirin when the past episode of chest pain happened?

   The answer to this question depends on what interpretation we choose to give to the temporal labels *December* 1994 and *April* 1996. If we consider that the patient took aspirin from *sometime* in *December* 1994 to *sometime* in *April* 1996, then we cannot give a definite answer to the question. However, if we interpret *December* 1994 and *April* 1996 to mean the entire specified months, then *December* 1994 means the entire period between 00:00:00 1 *December* 1994 and 23:59:59 31 *December* 1994. Similarly, *April* 1996 means the entire period between 00:00:00 1 *April* 1996 and 23:59:59 30 *April* 1996. In this case we are able to give a definite answer that the patient was taking aspirin when the episode of chest pain happened (see sentences 5 and 6).

5. When did the thrombolytic therapy end?

   In this case we have to add the time span 38 *months* to the time instant 30 *January* 1996 (see sentence 7).

These questions substantiate the need for a temporal DBMS to provide the means for (a) representing and storing time instants with different granularities, and time spans with different and mixed granularities, (b) handling granularity mismatches in operations between temporal primitives with different granularities, (c) converting a temporal primitive from one granularity to another, and (d) considering different interpretations for time labels. In the rest of the paper, we show how these issues can be supported in a temporal DBMS.

1.2. Paper Organization

The rest of the paper is organized as follows: in Section 2 we describe how multiple granularities are accommodated within the context of calendars and the derivation procedures of converting one granularity to another. In Section 3 we present our model for unanchored temporal primitives. In Section 4 anchored temporal primitives are presented along similar lines to that of Section 3. Section 5 sheds some

light on implementation issues. In Section 6, we compare our approach with the related work in literature. Finally, Section 7 presents conclusions and outlines future avenues of research.

2. Calendars

2.1. Modeling Issues

A calendar is a means by which physical time can be represented so as to be human readable. It is comprised of time units of varying granularities that enable the representation of different temporal primitives. Common calendars include *Gregorian* and *Lunar*. Educational institutions also use *Academic* calendars. In many applications, it is desirable to have multiple calendars that have different calendric granularities. In this paper, we base our work on a single calendar and refer the reader to [15] for details on multiple calendar support.

Definition 1. Calendar ($\mathcal{C}$). A calendar $\mathcal{C}$ is a triplet $\langle G_{tl}, \mathcal{G}, \mathcal{F} \rangle$, where G_{tl} is the global timeline of $\mathcal{C}$, $\mathcal{G}$ is the set of calendric granularities belonging to $\mathcal{C}$, and $\mathcal{F}$ is a list of conversion functions associated with $\mathcal{C}$. ■

Calendric granularities define the reasonable time units (e.g., *minute*, *day*, *month*) that can be used in conjunction with a calendar. Calendric granularities within a calendar are counted from the origin of its global timeline G_{tl}. The functions establish the conversion rules between the different granularities of a calendar.

2.2. Calendric Granularities

A calendar is comprised of a finite number of time units, called *calendric granularities*. In the Gregorian calendar the set of calendric granularities consists of *year*, *month*, *day*, *hour*, *minute*, and *second*. Generally speaking, a calendric granularity is a unit of measurement for time durations. For example, the calendric granularity of days (*day*) in the Gregorian calendar behaves similar to the unanchored duration 1 *day* (unanchored durations are discussed in more detail in Section 3.1).

Definition 2. Calendric granularity (G). A calendric granularity is a special kind of, possibly varying, unanchored duration that can be used as a unit of time. ■

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

Definition 3. Comparison between calendric granularities. G_A is *coarser* than G_B if $G_A > G_B$ as a time span. Similarly, G_A is *finer* than G_A if $G_A < G_B$ as a time span. ■

For example, the span of 1 *day* is shorter ($<$) than the span of 1 *month* and therefore the calendric granularity of days (*day*) is finer than the calendric granularity of months (*month*) in the Gregorian calendar. Similarly, *month* is coarser than *day*.

Because we must always be able to compare two calendric granularities with each other, we assume that the set of all possible calendric granularities is totally ordered with respect to the comparison operators given in Definition 3. This condition provides us with a sharp contrast from other work dealing with temporal granularity. While granularities are totally ordered in our framework, in many others they are typically only partially ordered [23, 26]. This allows us to carry out granularity conversions in terms of time spans.

2.3. Functions

Associated with a calendar is a list of functions ($\mathcal{F}$) which determine the number of finer calendric elements in coarser calendric elements. Notice that these functions depend on the particular value of a granularity and not just the granularity itself. For example, the number of days in a month depend on the month itself. More generally, let $\mathcal{C}$ 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. The following functions are then defined:

Definition 4. Conversion functions.

$$\begin{aligned}
f_{\mathcal{C}}^{G_1}(i_1) &\rightarrow N_{G_2}, lb_1 \leq i_1 \leq ub_1 \\
f_{\mathcal{C}}^{G_2}(i_1, i_2) &\rightarrow N_{G_3}, lb_1 \leq i_1 \leq ub_1, lb_2 \leq i_2 \leq f_{\mathcal{C}}^{G_1}(i_1) \\
&\vdots \\
f_{\mathcal{C}}^{G_n}(i_1, i_2, \ldots, i_n) &\rightarrow R_{G_{tl}}, lb_1 \leq i_1 \leq ub_1, lb_2 \leq i_2 \leq f_{\mathcal{C}}^{G_1}(i_1), \ldots, lb_n \leq i_n \leq f_{\mathcal{C}}^{G_{n-1}}(i_{n-1})
\end{aligned}$$

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 $\mathcal{C}$. N_{G_x} $(1 \leq x \leq n)$ is a natural number which stands for the number of G_x's. lb_i define the lower bound of the range of calendric elements for each considered granularity (e.g., months from 1, hours from 0). $R_{G_{tl}}$ is a real number. ■

The scale of G_{tl} is dependent on the precision of the respective machine architecture. For simplicity and explanatory purposes in this paper, we assume the scale of G_{tl} to be *seconds*. Just to illustrate how the conversion functions work, let us suppose we are interested in the number of months in 1995 and the number of days in September 1995 in calendar $\mathcal{C}$: the functions $f_{\mathcal{C}}^{year}(1995)$ and $f_{\mathcal{C}}^{month}(1995, 9)$ will return 12 and 30, respectively. More complicated cases will be discussed in the next section.

2.4. Conversions between Calendric Granularities

In a temporal model where times with different calendric granularities are supported, we need to be able to convert a finer calendric granularity to a coarser calendric granularity, and vice-versa. We discuss these conversions below by first defining two functions. These functions are necessary since the number of units of one granularity contained in a unit of another granularity is not fixed.

Definition 5. 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 6. 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. ■

Both factors coincide in the case of those granularities that have exact conversions (e.g., $lbf(hour, minute) = ubf(hour, minute) = 60$); in general, however, these factors can be different (e.g., $lbf(month, day) = 28$ and $ubf(month, day) = 31$).

We now show how $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ are derived from the conversion functions defined in Section 2.3 when G_A is coarser than G_B, and when G_A is finer than G_B.

DERIVATION 1 *G_A is coarser than G_B.* Let $G_1, \ldots, G_A, \ldots, G_B, \ldots, G_n$ be the totally ordered calendric granularities of calendar $\mathcal{C}$ with G_1 being the coarsest calendric granularity and G_n the finest. Now, the number of G_B units in any given calendric element $(i_1, \ldots, i_A)$ is given by the following summation:

$$f_{\mathcal{C}}^{G_A \to G_B}(i_1, \ldots, i_A) = \sum_{j_1 = lb_{A+1}}^{f_{\mathcal{C}}^{G_A}(i_1, \ldots, i_A)} \sum_{j_2 = lb_{A+2}}^{f_{\mathcal{C}}^{G_{A+1}}(i_1, \ldots, i_A, j_1)} \cdots$$

$$\cdots \sum_{j_{B-A-1} = lb_{B-1}}^{f_{\mathcal{C}}^{G_{B-2}}(i_1, \ldots, i_A, j_1, \ldots, j_{B-A-2})} f_{\mathcal{C}}^{G_{B-1}}(i_1, \ldots, i_A, j_1, \ldots, j_{B-A-1})$$

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

$$lbf(G_A, G_B) = \min_{(i_1, \ldots, i_A)} \{f_{\mathcal{C}}^{G_A \to G_B}(i_1, \ldots, i_A)\} \quad (1)$$

$$ubf(G_A, G_B) = \max_{(i_1, \ldots, i_A)} \{f_{\mathcal{C}}^{G_A \to G_B}(i_1, \ldots, i_A)\} \quad (2)$$

■

EXAMPLE: Let $\mathcal{C}$ be a calendar with the calendric granularities *year*, *month* and *day*. The following functions are defined in $\mathcal{C}$: $f_{\mathcal{C}}^{year}(y) \rightarrow N_{month}$, $f_{\mathcal{C}}^{month}(y,m) \rightarrow N_{day}$, $f_{\mathcal{C}}^{day}(y,m,d) \rightarrow R_{G_{tl}}$, 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 $lbf(year, day)$ and $ubf(year, day)$. The number of days in any year y is given by the summation: $\sum_{m=1}^{f_{\mathcal{C}}^{year}(y)} f_{\mathcal{C}}^{month}(y,m)$. The minimum (maximum) number of days in a year is then the minimum (maximum) of this summation over all y. More specifically,

$$lbf(year, day) = \min_{y}\{\sum_{m=1}^{f_{\mathcal{C}}^{year}(y)} f_{\mathcal{C}}^{month}(y,m)\}$$

$$ubf(year, day) = \max_{y}\{\sum_{m=1}^{f_{\mathcal{C}}^{year}(y)} f_{\mathcal{C}}^{month}(y,m)\}$$

□

DERIVATION 2 *Minimum and maximum number of G_B in K units of G_A.* Formulas (1) and (2) calculate the minimum and maximum number of G_B in *one* unit of G_A, respectively. We now generalize formulas (1) and (2) to calculate the minimum and maximum number of G_B in K units of G_A.

$$lbf(K, G_A, G_B) = \min_{i_1,\ldots,i_A}\{\sum_{0 \le dist_{G_A}((i_1',\ldots,i_A'),(i_1,\ldots,i_A)) \le K-1} f_{\mathcal{C}}^{G_A \rightarrow G_B}(i_1', \ldots, i_A')\} \quad (3)$$

$$ubf(K, G_A, G_B) = \max_{i_1,\ldots,i_A}\{\sum_{0 \le dist_{G_A}((i_1',\ldots,i_A'),(i_1,\ldots,i_A)) \le K-1} f_{\mathcal{C}}^{G_A \rightarrow G_B}(i_1', \ldots, i_A')\} \quad (4)$$

■

The summation in formulas (3) and (4) is the number of G_B units in K consecutive G_A units starting with $(i_1, \ldots, i_A)$. The function $dist_{G_A}((i_1', \ldots, i_A'), (i_1, \ldots, i_A))$ finds the number of G_A units elapsed between $(i_1', \ldots, i_A')$ and $(i_1, \ldots, i_A)$. For example, $dist_{month}((1996, 2), (1995, 1)) = 13$ (number of months elapsed between *February 1996* and *January 1995*). 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 (3) and (4) 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 (1) and (2) to calculate the minimum and maximum number of days in $2 \cdot month$ gives us 56 and 62, respectively, while formulas (3) and (4) 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, $lbf(K, day, hour) = ubf(K, day, hour) = K \cdot 24$.

DERIVATION 3 *G_A is finer than G_B.* If G_A is finer than G_B, then the lower and upper bound factors can be calculated using the formulas:

$$lbf(R, G_A, G_B) = \max_{K \in \mathbf{R}^+} \{K \mid R \geq ubf(K, G_B, G_A)\} \quad (5)$$

$$ubf(R, G_A, G_B) = \min_{K \in \mathbf{R}^+} \{K \mid R \leq lbf(K, G_B, G_A)\} \quad (6)$$

■

EXAMPLE: 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 $1 \leq K \leq 2$:

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

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

Hence, the number of months in 45 *days* is $1.45 \sim 1.55$. □

3. Unanchored Temporal Primitives

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. These operations include comparison with another time span with the transitive comparison operators $<$ and $>$ (which forms a partial order between time spans) and subtraction or addition of 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 an 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 Time Spans

Since a calendric granularity is a unit time span, we can use calendric granularities to construct time spans. For example, the time span of 36 *hours* which represents the duration of intensive medical management the patient underwent (see sentence 3 in Section 1.1), is obtained as $36 \cdot hour$. A time span of 2 *hours and* 55 *minutes*, which represents the duration of chest pain the patient suffered from (see sentence 1 in Section 1.1), can be obtained as $2 \cdot hour + 55 \cdot minute$. In general, a time span is made up of mixed calendric granularities and is defined as a finite sum:

Definition 7. Discrete determinate span. A discrete determinate span S_{discr} is given as as

$$S_{discr} = \sum_{i=1}^{n} (K_i \cdot G_i) \tag{7}$$

where K_i is an integer coefficient of G_i, which is a distinct calendric granularity in the calendar. ■

3.2. Conversion of Time Spans

The first question is whether it is always possible to convert a time span from a coarser to a finer calendric granularity without loss of information. The answer, perhaps surprisingly, is negative. To illustrate this point, consider the following: the conversion of the time span 1 *month* to the finer calendric granularity of days cannot possibly be an exact one. Should the resulting time span be 31, 30, 29 or 28 days? We cannot tell unless we know which month is involved. Since a time span is *unanchored* this information is not available. 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. We now define the conversion of a determinate time span to any given calendric granularity G_A.

Definition 8. Discrete time span conversion. The conversion of a time span of the form depicted in Definition 7 to a calendric granularity G_A results in a time span with lower bound

$$\lfloor \sum_{i=1}^{n} L_i \rfloor \cdot G_A \tag{8}$$

and upper bound

$$\lceil \sum_{i=1}^{n} U_i \rceil \cdot G_A \tag{9}$$

where

$$L_i = lbf(K_i, G_i, G_A) \text{ and } U_i = ubf(K_i, G_i, G_A) \tag{10}$$

■

For example, let us convert the duration of the chest pain in sentence 1 (see Section 1.1), which is the discrete time span 2 *hours and* 55 *minutes*, to a time span in the calendric granularity of minutes (*minute*). According to Definition 8, the time span $2 \cdot hour + 55 \cdot minute$ will be converted in the time span $175 \cdot minute$.

The more interesting case of inexact time span conversion is shown the next example.

EXAMPLE: Let us convert the time span $2 \cdot month + 45 \cdot hour$ to a time span in the calendric granularity of days (*day*). In this span, $K_1 = 2, K_2 = 45, G_1 = month, G_2 = hour$. We now use formula (10) to compute L_1, L_2, U_1, U_2:

$$\begin{array}{lll} L_1 & = & lbf(2, month, day) \\ & = & 59 \end{array} \qquad \begin{array}{lll} U_1 & = & ubf(2, month, day) \\ & = & 62 \end{array}$$

$$\begin{array}{lll} L_2 & = & lbf(45, hour, day) \\ & = & max\{K \mid 45 \geq ubf(K, day, hour)\} \\ & = & max\{K \mid 45 \geq K \cdot 24\} \\ & = & 1.875 \end{array} \qquad \begin{array}{lll} U_2 & = & ubf(45, hour, day) \\ & = & min\{K \mid 45 \leq lbf(K, day, hour)\} \\ & = & min\{K \mid 45 \leq K \cdot 24\} \\ & = & 1.875 \end{array}$$

$lbf(K, month, day), lbf(K, day, hour), ubf(K, month, day)$, and $ubf(K, day, hour)$ are calculated from the conversion functions in the Gregorian calendar. Lastly, we compute the lower and upper boundary of the resulting time span according to formulas (8) and (9), respectively:

$$\begin{array}{lll} lower\ bound & = & \lfloor L_1 + L_2 \rfloor \cdot day \\ & = & \lfloor 59 + 1.875 \rfloor \cdot day \\ & = & 60 \cdot day \end{array} \qquad \begin{array}{lll} upper\ bound & = & \lceil U_1 + U_2 \rceil \cdot day \\ & = & \lceil 62 + 1.875 \rceil \cdot day \\ & = & 64 \cdot day \end{array}$$

Hence, the result of our conversion is the indeterminate discrete time span $60 \cdot day \sim 64 \cdot day$. □

3.3. Canonical Forms for Time Spans

In addition to the set of granularities $G_1, \ldots, G_n$ and conversion functions discussed earlier, each calendar also implicitly defines the relation *exactly convertible to* between its granularities. We say that G_i is *exactly convertible to* G_j iff $ubf(k, G_i, G_j) = lbf(k, G_i, G_j) = k \cdot Q$, where Q is a natural number. Note that exact convertibility is a partial order on granularities which is a suborder of magnitude ordering. If G_i is exactly convertible to G_j, then $G_i = Q \cdot G_j$, where Q is a natural number. Since discrete determinate time spans have the form $S = \sum_{i=1}^{n} K_i \cdot G_i$, where K_i are integer numbers, the presence of the exact conversion rules implies

the existence of different forms of a time span. For example, 2 *hour* 55 *minutes* and 175 *minutes* are different forms of the same time span S'. To adhere as much as possible to human readability and user intuition, it is usually desirable to represent time spans in some canonical form. For example, when the time span 1 *hour* 30 *minutes* is added to the time span 35 *minutes*, the user would expect the time span 2 *hours* 5 *minutes* rather than the time span 1 *hour* 65 *minutes*. In this section, we define canonical forms for time spans. We begin by defining *representations* for time spans.

Definition 9. Span Representation. The n-tuple $r = \langle a_i \rangle_{i=1}^{n}$ (where a_i are integer numbers and n is the number of calendric granularities in a calendar) is called a *representation* of a span S (denoted $r \in Rep(S)$) iff $S = \sum_{i=1}^{n} a_i \cdot G_i$. ■

For example, let's assume that the Gregorian calendar has the calendric granularities *year*, *month*, *day*, *hour*, *minute* and *second*. Then $2 \cdot hour + 55 \cdot minute$ and $175 \cdot minute$, which are two forms of S', have the representations $r_1 = \langle 0, 0, 0, 2, 55, 0 \rangle$ and $r_2 = \langle 0, 0, 0, 0, 175, 0 \rangle$, respectively.

We will use span representations to define a *canonical form* for a time span. In order to do that, we introduce the notion of a *strictly non-negative span*.

Definition 10. Strictly Non-Negative Span. A span S is a *strictly non-negative span* (denoted $S >^{+} 0$) iff $\exists r = \langle a_i \rangle_{i=1}^{n} \in Rep(S) : a_i \geq 0$ for $i = 1, \ldots, n$. ■

The time span 2 *hour* 55 *minutes*, for example, is strictly non-negative while the time span 1 *month*$-$30 *days* is not strictly non-negative: no positive representations are possible since 1 *month* does not have an exact conversion to *days*.

Another definition that we need to define for a canonical form is a dominancy relation between span representations. The dominancy relation is in fact a lexicographical order on span representations, which is used in determining the canonical representation of a span.

Definition 11. Dominancy. A representation $r = \langle a_i \rangle_{i=1}^{n}$ *dominates* another representation $r' = \langle b_i \rangle_{i=1}^{n}$ (denoted $r \succ r'$), $r, r' \in Rep(S)$, iff $\exists k : a_k > b_k \wedge a_i = b_i$ for $i = 1, \ldots, (k-1)$. ■

According to this definition, given two representations $r_1 = \langle 0, 0, 0, 2, 55, 0 \rangle$ and $r_2 = \langle 0, 0, 0, 0, 175, 0 \rangle$, the dominancy relation $r_1 \succ r_2$ holds.

Having defined strictly non-negative spans and dominancy, we can now proceed to define the *canonical representation* and the *canonical form* for strictly non-negative spans.

Definition 12. Canonical Representation. A representation $r = \langle a_i \rangle_{i=1}^{n} \in Rep(S)$ is *the canonical representation* of span $S >^{+} 0$ iff $a_i \geq 0$ for $i = 1, \ldots, n \wedge \forall r' \in Rep(S) : r \succ r' \vee r = r'$. ■

For example, r_1, i.e., $\langle 0,0,0,2,55,0\rangle$ is the canonical representation of the time span S'.

Every strictly non-negative span has one and only one canonical representation. The canonical representation is the best representation of a given strictly non-negative span.

Definition 13. Canonical Form. A strictly non-negative span $S = \sum_{i=1}^{n} a_i \cdot G_i$ is in *canonical form* iff $r = \langle a_i \rangle_{i=1}^{n}$ is the canonical representation of S. ■

For example, the canonical form for the time span S' is $2 \cdot hour + 55 \cdot minute$.

3.4. Operations between Time Spans

In this section we give the semantics of arithmetic and comparison operations between time spans and show how some of the questions posed in Section 1.1 are answered.

3.4.1. Arithmetic Operations between Time Spans The semantics of adding (subtracting) two time spans is to add (subtract) the components which have the same calendric granularity, concatenate the remaining components to the resulting time span, and reduce the resulting time span to canonical form as described in Section 3.3.

For example, the global duration, in canonical form, of the symptoms of angina, described in sentences 1 and 5 in the motivating example (see Section 1.1), is obtained by the addition operation:
$(2 \cdot hour + 55 \cdot minute) + (7 \cdot minute + 35 \cdot second) \rightarrow (2 \cdot hour + 62 \cdot minute + 35 \cdot second)$
$\rightarrow (3 \cdot hour + 2 \cdot minute + 35 \cdot second)$

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, while negative spans have the semantics of backward duration in time.

3.4.2. Comparison Operations between Time Spans The semantics of comparing two time spans is to first convert each time span to the finest granularity that exists between the two time spans, and then carry out the comparison. As an example, let us compare the duration of the first symptom of angina (sentence 1) with the duration (20 *minutes*), for establishing if we have a case of unstable angina (see the motivating example in Section 1.1): being $(2 \cdot hour + 55 \cdot minute) = 175 \cdot minute > 20 \cdot minute$, we identify the patient as suffering from unstable angina.

Considering more complex situations, the comparison $1 \cdot month > 30 \cdot day$ will be translated $(28 \cdot day \sim 31 \cdot day) > 30 \cdot day$, which will return the value `Unknown`. We note from the above example that time spans which *overlap* (or even *meet* each other) cannot be compared. This follows from the fact that calendric granularities are partially ordered with respect to the binary relation "exactly convertible to."

4. Anchored Temporal Primitives

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., [15 *June* 1995, 31 *July* 1995]. A *time instant* is a specific anchored moment in time. For example, the anchor points of the time interval [15 *June* 1995, 31 *July* 1995] are represented by the time instants 15 *June* 1995 and 31 *July* 1995. In this paper we concentrate on time instants. The treatment of time intervals is a straightforward extension of the discussion in this section.

4.1. Representation of Anchored Times

In representing anchored time concepts, i.e., instants and intervals, we use a *time granule* [3]. In our approach, a time granule is an interval on the global timeline and can be easily identified by the functions defined for a calendar (see Section 2.3). Every time granule belongs to a specific calendar and is composed of calendric elements which belong to different calendric granularities of the same calendar.

We identify three possible interpretations of a time granule:

- *Beginning Instant* (I_{beg}). This type of instant refers to the *beginning* of the period the granule denotes. Examples of beginning instants would be the time of space launches, start of exams, process start and end times in real-time systems, etc. Therefore the time instants 1995_{beg}, *January* 1995_{beg}, and 1 *January* 1995_{beg} are equivalent and refer to the beginning of the year 1995.

- *Determinate Interval* (I_{det}). It refers to the *whole* period the granule denotes. The time granules denoting national holidays are examples of determinate intervals. For example, Victoria Day (a national holiday in Canada) occurs on 24 May each year. This means that the whole day of 24 May is a holiday. In this case, 24 May_{det} is a determinate interval.

- *Indeterminate Instant* (I_{indet}). This type of instant refers to *sometime* in the period the granule denotes, and is perhaps the most commonly used time instant in "real-world" temporal measurements. For example, in sentence 4 of the motivating example we would represent the time at which the patient had a myocardial infarction as 15 *February* 1996_{indet} (which means *the patient had a myocardial infarction some time on that day*).

Essentially, a determinate (indeterminate) time interval (instant) can be expressed by an interval whose lower and upper bounds are beginning time instants. For example, the time interval 5 *February* 1997_{det} is analogous to the interval [5 *February* 1997_{beg}, 6 *February* 1997_{beg}).

Every determinate (indeterminate) time interval (instant) has a granularity (G_i) associated with it. This granularity determines the mapping of the given determinate (indeterminate) time interval (instant) I_{det} (I_{indet}) to the domain of beginning time instants. The mapping is defined as follows:

$$I_{det} \mapsto [I_{beg}, I_{beg} + G_i)$$
$$I_{indet} \mapsto [I_{beg} \sim I_{beg} + G_i)$$

Here I_{beg} denotes the counterpart of I_{det} and I_{indet} in the domain of beginning time instants. The upper bound of the resulting interval is defined to be open to ensure that different time granules with the same granularity do not overlap. Table 1 gives an example of the granule 1995, interpreted respectively as a beginning instant, a determinate time interval, and an indeterminate time instant, at different granularities.

Table 1. Conversion of the different interpretations of a time granule (1995) on different granularities (*Year*, *Month*, and *Day*).

Beginning Instant	Determinate Interval	Indeterminate Instant
1995_{beg}	$[1995_{beg}, 1996_{beg})$	$[1995_{beg} \sim 1996_{beg})$
$Jan\ 1995_{beg}$	$[Jan\ 1995_{beg}, Jan\ 1996_{beg})$	$[Jan\ 1995_{beg} \sim Jan\ 1996_{beg})$
$1\ Jan\ 1995_{beg}$	$[1\ Jan\ 1995_{beg}, 1\ Jan\ 1996_{beg})$	$[1\ Jan\ 1995_{beg} \sim 1\ Jan\ 1996_{beg})$

A formal treatment of determinate intervals and indeterminate time instants and mappings is given in [16]; moreover, in handling indeterminacy, we have to consider modality, multiple-valued logics, and probabilistic approaches [6, 11, 13, 21] for dealing with uncertainty in relationships. In the rest of this section, since determinate intervals and indeterminate time instants can be mapped to the domain of beginning time instants, we will concentrate on beginning time instants (hereinafter simply *time instants*).

4.2. Operations on Time Instants

As with time spans, instants can be compared with each other, and subtracted from one another to find the elapsed time between them. Additionally, a time span can be added to, or subtracted from, a time instant to return another time instant.

4.2.1. Comparison between Time Instants Let $I^1_{G_m} = (i_1, \ldots, i_m)$ and $I^2_{G_p} = (i'_1, \ldots, i'_p)$ be two time instants, with finest granularities G_m and G_p, respectively. We assume that the instants belong to the same calendar. We also assume without loss of generality that $m \geq p$. Then, the following algorithm checks if $I^1_{G_m} \leq I^2_{G_p}$:

$Compare_1(I^1_{G_m}, I^2_{G_p})$
  $I^1_{G_m} = (i_1, \ldots, i_m)$: a time instant where $i_1, \ldots, i_m$ are ordinal numbers of its calendric elements;
  $I^2_{G_p} = (i'_1, \ldots, i'_p)$: a time instant where $i'_1, \ldots, i'_p$ are ordinal numbers of its calendric elements;
  {

$I^2_{G_p} := (i'_1, \ldots, i'_p, \underbrace{lb_{p+1}, \ldots, lb_m}_{m-p});$

for j **from** 1 **to** m {
  if $(i_j > i'_j)$
    return `False`
}
return `True`
}

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

Suppose we have the following time instants and the ordinal values of their respective calendric elements:
5 *June* 1990 $\equiv (1990, 6, 5)$; 15 *June* 1990 $\equiv (1990, 6, 15)$; *June* 1990 $\equiv (1990, 6)$; 1990 $\equiv (1990)$. Then, 5 *June* 1990 $<$ 15 *June* 1990 because $(1990, 6, 5) < (1990, 6, 15)$; *June* 1990 $<$ 15 *June* 1990 because $(1990, 6) \equiv (1990, 6, 1)$, and $(1990, 6, 1) < (1990, 6, 15)$; 1990 $<$ 15 *June* 1990 because $(1990) \equiv (1990, 1, 1)$, and $(1990, 1, 1) < (1990, 6, 1)$.

4.2.2. Elapsed Time between Time Instants Let $(i_1, \ldots, i_m)$ and $(i'_1, \ldots, i'_m)$ be two time instants belonging to the same calendar. Then:

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

In the simplest case, 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. This way, we can evaluate the duration of the acute chest pain in sentence 2 (see Section 1.1). Adding this time span to the time span, previously evaluated on the basis of sentences 2 and 5, enables us to determine the global span of the symptoms of angina (see question 2. in Section 1.1).

Let us consider a more complex case (see sentences 4 and 5, and question 1. in Section 1.1): we have to evaluate the time span between the myocardial infarction (happened on 15 *February* 1996) and the last episode of angina (happened at 15 *hour* 12 *April* 1996). Here, the finest calendric granularity of 15 *February* 1996 is coarser than that of 15 *hour* 12 *April* 1996. Thus, 15 *February* 1996 is first replaced by the time instant 0 *hour* 15 *February* 1996, its equivalent time instant with the finest granularity of *hour*: `Elapsed`((15 *February* 1996), (15 *hour* 12 *April* 1996)) $\Rightarrow$ `Elapsed`((0 *hour* 15 *February* 1996), (15 *hour* 12 *April* 1996)) $\Rightarrow (1 \cdot month + 27 \cdot day + 15 \cdot hour)$.

The `Elapsed` function can be extended to accept a third argument, namely, the granularity of the resulting span. If such an additional argument is omitted, the `Elapsed` function reverts to the default behavior described in the above example.

4.2.3. Operations between Spans and Time Instants In performing arithmetic operations that involve spans and time instants, 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. For example, adding the time span 38 *months* to the time instant 30 *January* 1996, in order to know when the thrombolytic therapy (see sentence 7 and question 5. in Section 1.1) will finish, results in the time instant 30 $March$ 1999.

- 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 carried out. For example, if we want to know when the the first episode of chest pain ended (see sentence 1 in Section 1.1), we have to add the time span 2 *hour and* 55 *min* to the time instant 13 *December* 1995. In this case the time instant is first replaced by its equivalent time instant 0 *hour* 0 *min* 13 *December* 1995. The addition of the time span to this time instant results in the time instant 2 *hour* 55 *min* 13 *December* 1995.

5. Implementation Issues

The formulae (3) and (4) for $lbf(K, G_A, G_B)$ and $ubf(K, G_A, G_B)$ (see Derivation 2) are computationally expensive. However, they are not designed for direct computation. These formulae are just mathematical definitions. A technique that can be used to make computations less expensive would be to simplify these formulae since they allow for many simplifications once a set of particular calendric functions is chosen. As an example, let us consider the Gregorian calendar. In this calendar,

$$
\begin{aligned}
f^{year}(y) &= 12 \text{ (months)} \\
f^{month}(y,m) &= \begin{cases} 30 & \text{if } m \text{ is 4, 6, 9, or 11} \\ 31 & \text{if } m \text{ is 1, 3, 5, 7, 8, 10, or 12} \\ 28 & \text{if } m = 2 \text{ and } y \text{ is not leap} \\ 29 & \text{if } m = 2 \text{ and } y \text{ is leap} \end{cases} \quad \text{(days)} \\
f^{day}(y,m,d) &= 24 \text{ (hours)}
\end{aligned}
$$

where y is leap when $y \bmod 400 = 0 \vee y \bmod 4 = 0 \wedge y \bmod 100 \neq 0$.

We will consider the conversions from years to months, from years to days, and from months to days. Then,

$$\begin{aligned} f^{year\to month}(y) &= f^{year}(y) = 12 \\ f^{year\to day}(y) &= \begin{cases} 366 & \text{if } y \text{ is leap} \\ 365 & \text{otherwise} \end{cases} \\ f^{month\to day}(y,m) &= f^{month}(y,m) \end{aligned}$$

Then, using formulae (3) and (4) we find that

$$\begin{aligned} lbf(K, year, month) &= \min_y\{ \sum_{0\le dist_{year}(y',y)\le K-1} f^{year\to month}(y')\} \\ &= \min_y\{12K\} \\ &= 12K \\ ubf(K, year, month) &= 12K \\ lbf(K, year, day) &= \min_y\{365(y+K) + \lfloor (y+K)/4\rfloor - \\ &\quad \lfloor (y+K)/100\rfloor + \lfloor (y+K)/400\rfloor - \\ &\quad 365y - \lfloor y/4\rfloor + \lfloor y/100\rfloor - \lfloor y/400\rfloor\} \\ &\ge 365K + \lfloor K/4\rfloor - \lfloor (K+96)/100\rfloor \\ ubf(K, year, day) &\le 365K + \lfloor (K+3)/4\rfloor \end{aligned}$$

The above lbf and ubf bounds can be used instead of exact formulae. These bounds are easily computable and introduce an error that is less than a day per century. Analogous methods can be used to find computationally cheap approximations for conversion of months to days; however, to obtain reasonable approximations, values for small K ($K \le 48$) have to be tabulated. Let $g_{min}(K)$ and $g_{max}(K)$ be such tabulations. Then we have:

$$\begin{aligned} lbf(K, month, day) &= g_{min}(K \bmod 48) + lbf(\lfloor K/48\rfloor \cdot 4, year, day) \\ ubf(K, month, day) &= g_{max}(K \bmod 48) + ubf(\lfloor K/48\rfloor \cdot 4, year, day) \end{aligned}$$

Using these formulae we can now make fast and quite precise conversions. For example, the number of days (d) in 100 months according to the above formulae is $120 + 2921 = 3041 \le d \le 3044 = 122 + 2922$, which is the correct estimate.

6. Related work

6.1. Representing time spans

Since a time span is independent of any time instant or time interval due to its relative nature, granularity conversions in the context of anchored temporal primitives cannot be used for unanchored temporal primitives. Hence, most of the

previously described temporal models [4, 7, 21, 22, 23, 25, 26, 27] cannot support completely the unanchored temporal information needs of an application like the clinical example given in Section 1.1.

Although the work of Lorentzos [20] does not explicitly deal with temporal granularity, it proposes a scheme for representing and operating on non-metric types. Mixed granularity time durations, with separate fields for their composite parts (e.g., hours, minutes, seconds) are one example of a non-metric data type. These can be represented as elements of sets of composite numbers which provide conversion relationships (mappings) between the composite fields. However, only exact (regular) mappings are discussed. Therefore time durations with composite parts having granularities of months and days cannot be modeled. In our approach, a time span is simply a summation of calendric granularities. Both exact and inexact mappings between granularities are provided (using the $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ functions). This allows time durations to be converted to any given calendric granularity. The conversion of a time duration to a particular granularity is possible in [20]. However, the target granularity is restricted to be one of the granularities of the composite parts of the time duration. We do not enforce such a restriction in our work. A time duration can be converted to any desired granularity in the calendar. In [20], addition between time durations is also possible. However, the operands have to be *addition compatible*. If S_1 and S_2 are time durations, then they are addition compatible if S_2 consists of at most as many composite parts as S_1, and for these composite parts, the granularities should be the same. For example, the time durations with composite granularities (days, hours, minutes, seconds), (hours, minutes, seconds), (minutes, seconds), and (seconds) are addition compatible, and thus can be added to each other. Our approach is more general in that time durations do not have to be addition compatible.

In TSQL2, a calendar has a specification file which provides regular and irregular mappings between granularities [23]. It is not clear however, how these mappings are derived. In Section 3, we gave detailed derivation procedures for the $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ functions which represent regular and irregular mappings between any two granularities in a calendar. Then, we described how the $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ functions are used in the conversion of unanchored temporal primitives to a given calendric granularity.

Moreover, in TSQL2 time spans (durations) which have mixed granularities cannot be represented [24]. For example, the duration of the chest pain in sentence 1 (see Section 1.1) would have to be represented in hours or in minutes. Since a time span is a summation of distinct granularities in our approach, representing symptom durations with mixed granularities is straightforward. Our approach of representing mixed granularity time spans is also more general than that used in SQL-92 in that we do not restrict time spans to only *year-month* or *day-time* combinations.

A time span in TSQL2 is necessarily indeterminate at both coarser and finer granularities. This is because a granularity is modeled as an anchored partitioning of the timeline, whereas a time span in unanchored. Therefore, all time span conversions in TSQL2 are treated as inexact, resulting in indeterminate time spans. In

our approach, a time span conversion can be exact or inexact. Consider the simple conversion of the time span 1 *hour* to the granularity of minutes. In TSQL2, this conversion results in the indeterminate span $1 \sim 119$ *minutes* – an indeterminacy of 120 minutes. In our approach however, the conversion is exact and results in the determinate span 60 *minutes*, which is what is expected in reality. Being based on the mentioned conversions, operations involving time spans in TSQL2 could give rise to ambiguities and even incorrect results. Details on results of operations involving time spans in TSQL2 are given in [18].

Comparison of time spans of different granularities in TSQL2 can also lead to incorrect results. Consider the comparison of the time span 30 *minutes* with the time span 1 *hour* in TSQL2, using left-operand semantics: 30 *minutes* $>$ 1 *hour* ? $\Leftrightarrow$ 30 *minutes* $>$ *cast*(1 $\sim$ 119 *minutes*) $\Leftrightarrow$ 30 *minutes* $>$ 1 *minute* $\Leftrightarrow$ `True`!

The time span 1 *hour* is first converted to the granularity of the leftmost operand. Since a time span is indeterminate at any finer or coarser granularity in TSQL2, the conversion of 1 *hour* to the granularity of minutes yields the indeterminate time span 1 $\sim$ 119 *minutes*. The cast operation then converts this to a determinate time span by arbitrarily choosing the lower bound. This leads to comparing the time span 30 *minutes* to the time span 1 *minute*, and subsequently returning `True` which is the opposite of what is expected. In our approach, the time span 1 *hour* would be converted exactly to the time span 60 *minutes*, and the comparison would then return `False`.

Some of the counter-intuitive results in TSQL2 may be avoided by defining duration literals for time spans with mixed granularities in the adopted calendar. The calendar would also manage different interpretations in mapping a time span from one granularity to another. Even though calendar definition in TSQL2 allows users to overcome some of the underlined limitations, this solution seems to be less general than our approach. Further problems in TSQL2 could arise in according different calendar-dependent interpretations with the adopted scaling/left operand semantics.

6.2. Representing time instants

Clifford and Rao [7] introduce a general structure for time domains called a *temporal universe* which 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, [27] also examine the issue of multiple granularities. An algebra is described that allows the conversion of event times to an interval representation. This involves converting the coarser granularity to the finer granularity in light of the semantics of the time varying domains. Wang et al, in [25], extend this work by providing semantics for moving up and down a granularity lattice. In [2], Barbic and Pernici discuss the issues of absolute, relative, imprecise, and periodic times. 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 [21], the existence of a minimum underlying

granularity (quantum of time) to which time is mapped, is assumed. Montanari et al, [22] examined the issue of multiple granularities, but considered exact granularity conversions only. Corsetti et al, [12] deal with different time granularities in specifications of real-time systems. The above works do not consider granularities for expressing unanchored time span; hence, while we can perform *both* anchored and unanchored granularity conversions, the proposals above consider *only* anchored granularity conversions. Furthermore, none of the models, with the exception of [21], explicitly address the notion of indeterminacy.

Bettini et al in [4] propose a general framework for defining time granularity systems and analyze different kinds of relationships between granularities. In comparison with this framework, it is worth noting that our proposal does not allow one to model granularities with gaps either inside a granule or between granules (e.g., *business-week* or *business-day*); a further limit is that incomparable granularities (e.g., *months* and *weeks*) are not allowed. Our proposal allows one to formally specify a granularity system having, according to the framework of Bettini and colleagues, comparable granularities possibly having nonuniform granules, which cover partially or completely the global time axis. Two main differences exist between the approach of Bettini and colleagues and our approach: they do not face the problem of managing different granularities for unanchored temporal primitives, focusing mainly on the formal specification of the features related to the mapping functions from different granularity to the global timeline; furthermore, they do not consider the problem of expressing anchored and unanchored time span at mixed granularities (e.g., *3 days and 20 minutes*) and of converting them to a specified granularity.

In [5], Bettini and De Sibi propose formal definitions and a mathematical characterization of finite and periodical time granularities: this kind of granularities (intuitively, all those granularities for which there is a periodic pattern for their granules: *years* and *days*, *years* and *months*, and so on), excluding the case of gaps between or inside a granule, can be represented in our approach by a suitable calendar. Moreover, our notion of calendar is able to represent no-gap, comparable granularities, which do not have periodic patterns, i.e., having completely irregular mappings between them.

TSQL2 treats all instants as indeterminate at finer granularities [23]. In contrast, our treatment of time instants depends on the interpretation given to the time instant. This is illustrated in Table 1. In our approach, indeterminacy in time instants is not in the conversion to finer granularities, but it is in the interpretation of the time instant. In TSQL2, the semantics of arithmetic operations which involve time spans and time instants are left to the calendar as calendar specific operations. These include $instant + span$, $span + instant$, and $instant - instant$. For example, consider the operation 1 $month$+15 $June$ 1995 in TSQL2. If the operation $month + DATE = DATE$ is not supported by the calendar, according to the [left operand] semantics of TSQL2, the following would happen: 1 $month$ + 15 $June$ 1995 $\rightarrow$ 1 $month$+$June$ 1995 $\rightarrow$ $July$ 1995 [24]. This result is not what one would intuitively expect. In our approach (see Section 4.2), the addition operation would return the expected time instant 15 $July$ 1995.

Combi et al in [9, 10, 11] propose an object-oriented data model and a related query language to deal with temporal information given at different granularities and/or with indeterminacy: they propose the adoption of a three-valued logic for managing uncertainty coming from relationships between intervals and/or instants given at different granularities or with indeterminacy. The proposed data model allows one the define durations at different calendric granularities: it considers only the Gregorian calendar and models granularities for unanchored time span by regular mappings (e.g., one unanchored year is composed by 365.2425 days). Irregular mappings and different interpretations for unanchored time spans are not considered: durations given at different granularities are regularly mapped on the global timeline as indeterminate time spans.

7. Conclusion

This paper is a further step in completing the puzzle on temporal granularity by providing support both for anchored and unanchored temporal primitives with different and mixed granularities and addressing the issues that arise therein. This will help temporal DBMSs to better support the various applications in which such primitives are inherent.

A model for supporting calendars is given and it is shown how multiple granularities and unanchored and anchored temporal entities are integrated within the context of calendars. A calendric granularity is described as being part of a calendar and represented as a special kind of span - one with a unit duration. The process of conversion between time spans of mixed granularities is then given and canonical forms for time spans defined. These forms are used by arithmetic operations on time spans to return time spans which are what a user intuitively expects. The arithmetic and comparison operations involving time spans were described and compared to similar ones proposed in literature: we show how our semantics of operations are more general and intuitive than that present in the considered proposals.

We also give a treatment of anchored temporal primitives and provide three different interpretations of time granules. We show how time instants are converted to different granularities and describe the various operations on time instants.

In Section 5 we show that it is possible to establish computationally inexpensive yet quite precise formulae for lower and upper bound coefficients. Currently, the derivation of these formulae has to be done by a database administrator; their automatic derivation is a topic for future research.

In conclusion, it is our position that assuming a simplistic view of unanchored temporal data 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.

Notes

1. Formally known as angina pectoris. A clinical syndrome typically characterized by a deep, poorly localized chest or arm discomfort that is reproducibly associated with physical exertion or emotional stress and relieved promptly by rest or sublingual nitroglycerine. The discomfort of angina is often hard for patients to describe, and many patients do not consider it to be "pain." In most, but not all patients, these symptoms reflect myocardial ischemia resulting from significant underlying coronary artery disease [1].

References

1. E. Braunwald, D.B. Mark, and R.H. Jones. Diagnosing and Managing Unstable Angina - Quick Reference Guide for Clinicians. (10. AHCPR Publication No. 94-0603), May 1994.
2. F. Barbic and B. Pernici. Time Modeling in Office Information Systems. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, ACM Press, New York, 51–62, 1985.
3. C. Bettini, C.E. Dyreson, W.S. Evans, R.T. Snodgrass, and X.S. Wang. A Glossary of Time Granularity Concepts. In O. Etzion, S. Jajodia, and S. Sripada (Eds.), *Temporal Databases - Research and Practice*, LNCS 1399, Springer–Verlag, Berlin Heidelberg, 406–413, 1998.
4. C. Bettini, X. Wang, and S. Jajodia. A general framework for time granularity and its application to temporal reasoning. *Annals of Mathematics and Artificial Intelligence*, 22:29–58, 1998.
5. C. Bettini and R. De Sibi. Symbolic Representation of User-Defined Time Granularities. In *Proceedings sixth international workshop on temporal representation and reasoning (TIME '99)*. IEEE Computer Society Press, Los Alamitos, 17–28, 1999.
6. L. Chittaro and C. Combi. Temporal Indeterminacy in Deductive Databases: an Approach Based on the Event Calculus. In *Proceedings of the Second International Workshop on Active, Real-time and Temporal Database Systems (ARTDB-97)*, LNCS 1553, Springer, Berlin Heidelberg, 212–227, 1998.
7. J. Clifford and A. Crocker. The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans. In *Proc. 3rd Int'l. Conf. on Data Engineering*, IEEE Computer Society Press, Los Alamitos, 528–537, 1987.
8. J. Clifford and A. Rao. A Simple, General Structure for Temporal Domains. In C. Rolland, F. Bodart, and M. Leonard, (Eds.), *Temporal Aspects in Information Systems*, North-Holland, Amsterdam, 17–30, 1988.
9. C. Combi, F. Pinciroli, and G. Pozzi. Managing Different Time Granularities of Clinical Information by an Interval-Based Temporal Data Model. *Methods of Information in Medicine*, 34(5):458–474, 1995.
10. C. Combi and C. Cucchi. GCH-OSQL: a Temporally-Oriented Object-Oriented Query Language Based on a Three-Valued Logic. In *Proceedings fourth international workshop on temporal representation and reasoning (TIME '97)*. IEEE Computer Society Press, Los Alamitos, 119–126, 1997.
11. C. Combi, C. Cucchi, and F. Pinciroli. Applying Object-Oriented Technologies in Modeling and Querying Temporally-Oriented Clinical Databases Dealing with Temporal Granularity and Indeterminacy. *IEEE Transactions on Information Technology in Biomedicine*, 1(2): 100–127, 1997.
12. 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(2):191–215, 1991.
13. C.E. Dyreson and R.T. Snodgrass. Valid-time Indeterminacy. In *Proc. 9th Int'l. Conf. on Data Engineering*, IEEE Computer Society Press, Los Alamitos, 335–343, 1993.
14. R. Flowerdew. *Geographical Information Systems.* Volume 1. John Wiley and Sons, New York, 1991.
15. I.A. Goralwalla, Y. Leontiev, M.T. Özsu, and D. Szafron. Modeling Time: Back to Basics. In *Proc. of 6th International Conference on Information and Knowledge Management (CIKM'97)*, ACM Press, New York, 24–31, 1997.

16. I.A. Goralwalla. Temporality in Object Database Management Systems. Ph.D. Thesis, University of Alberta, CA, 1998.
17. I.A. Goralwalla, Y. Leontiev, M.T. Özsu, and D. Szafron. An Object-Oriented Framework for Temporal Data Models. In *O. Etzion, S. Jajodia, and S. Sripada (Eds.), Temporal Databases - Research and Practice*, LNCS 1399, Springer-Verlag, Berlin Heidelberg, 1–35, 1998.
18. I.A. Goralwalla, Y. Leontiev, M.T. Özsu, D. Szafron, and C. Combi. Temporal Granularity for Unanchored Temporal Data. In *Proc. of 7th International Conference on Information and Knowledge Management (CIKM'98)*, ACM Press, New York, 414–423, 1998.
19. C. Jensen, C. Dyreson (Eds.) et al. The Consensus Glossary of Temporal Database Concepts - February 1998 Version. In *O. Etzion, S. Jajodia, and S. Sripada (Eds.), Temporal Databases - Research and Practice*, LNCS 1399, Springer-Verlag, Berlin Heidelberg, 367–405, 1998.
20. N. Lorentzos. DBMS Support for Non-Metric Measuring Systems. *IEEE Transactions on Knowledge and Data Engineering*, 6(6):945–953, 1994.
21. R. Maiocchi, B. Pernici, and F. Barbic. Automatic Deduction of Temporal Information. *ACM Transactions on Database Systems*, 17(4):647–688, 1992.
22. 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*, IOS Press, Tokyo, 702–712, 1992.
23. R. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, Dordrecht, 1995.
24. R. Snodgrass. May 1996. Private correspondence.
25. X.S. Wang, S. Jajodia, and V. Subrahmanian. Temporal Modules: An Approach Toward Temporal Databases. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, ACM Press, New York, 227–236, 1993.
26. X.S. Wang, C. Bettini, A. Brodsky, and S. Jajodia. Logical Design for Temporal Databases with Multiple Granularities. *ACM Transactions on Database Systems*, 22(2):115–170, 1997.
27. 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 (Eds.), *Advanced Information Systems Engineering, 3rd Int'l Conference CAiSE '91*, Springer-Verlag, Berlin Heidelberg, 124–140, 1991.