Symbolic User-Defined Periodicity in Temporal Relational Databases

Paolo Terenziani

Abstract—Calendars and periodicity play a fundamental role in many applications. Recently, some commercial databases started to support *user-defined* periodicity in the *queries* in order to provide "*a human-friendly way of handling time*" (see, e.g., TimeSeries in Oracle 8). On the other hand, only few *relational data models* support user-defined periodicity in the *data*, mostly using "*mathematical*" expressions to represent periodicity. In this paper, we propose a high-level "*symbolic*" language for representing user-defined periodicity which seems to us more *human-oriented* than mathematical ones, and we use the domain of Gadia's temporal elements in order to define its properties and its extensional semantics. We then propose a temporal relational model which supports user-defined "*symbolic*" *periodicity* (e.g., to express "on the second Monday of each month") in the validity time of tuples and also copes with *frame times* (e.g., "from 1/1/98 to 28/2/98"). We define the temporal counterpart of the standard operators of the relational algebra, and we introduce new temporal operators and functions. We also prove that our temporal algebra is a consistent extension of the classical (atemporal) one. Moreover, we define both a fully symbolic evaluation method for the operators on the periodicities in the validity times of tuples, which is correct but not complete, and semisymbolic one, which is correct and complete, and study their computational complexity.

Index Terms—Temporal relational model and algebra, user-defined symbolic periodicity in the validity time, high-level "symbolic" language, symbolic (intensional) evaluation method, semisymbolic evaluation method, user-friendly treatment of periodicity, integration and extension of artificial intelligence and temporal databases techniques.

1 Introduction

One challenge is for applications to provide a human-friendly way of handling time. Standard relational data is very different from the way people relate to time. People think in terms of calendars and clock units .... presentation of Time Series Cartridge in Oracle 8 (see www.oracle.com\databases\timeseries.html).

Time plays a fundamental role in many real-world applications; thus, many approaches extended relational databases to deal with the *transaction time* and the *validity time* [36] of tuple (see, e.g., [32], [39]). Moreover, many real-world applications, including planning, scheduling, process control, multimedia, active databases, banking, law, etc., need to deal with periodic events and the interest towards the treatment of periodic events is rapidly increasing. Thus, dealing with periodicity and user-defined calendars in the data and/or in the queries is a very important task, which has been faced by many approaches in the area of temporal databases (consider, e.g., [4], [5], [6], [7], [18], [22], [23], [29], [34], [45], [46], [49]). In particular, an *implicit* (i.e., not extensional) treatment is advantageous since it allows one to store data holding at periodic times in a compact way instead of explicitly listing all the instances of the given periodicity (e.g., all "days" in a possibly infinite frame of time). Baudinet et al. [5] distinguished between the approaches dealing in an implicit way with periodicity *in the data* using *deductive rules* and those using *constraints*. Regarding the approaches based on deductive rules, consider, e.g., [15], [16], which dealt with periodicity via the introduction of the successor function in Datalog. Kabanza et al. proposed a very influential approach using constraints [22], [23], that extended classical relational databases to deal with periodic data by representing infinite temporal information by generalized tuples defined by *linear repeating points* (points of the form c1+c2*X) and constraints on points (e.g., X1 $\leq$ X2+c3). Kabanza also defined a temporal algebra for this model and studied its complexity. Toman et al. [45] extended Datalog with integer periodic constraints. Tuzhilin and Clifford [46] proposed a survey of many approaches.

Constraint-based approaches usually represent periodicity by mathematical formulae based on a granularity fixed a priory. On the other hand, recently, many works have pointed out the importance of dealing with different (user-defined) *granularities* [7], [8], [18] and/or supporting multiple user-defined *calendric systems* (see, e.g., [31], [38]). In the meantime, many approaches (mainly in the area of artificial intelligence) developed user-friendly, high-level "*symbolic*" (i.e., not "mathematical," as in constraint-based approaches) languages to allow users to define calendars, periodicities, and granularities (consider, e.g., [9], [17], [27], [34]). High-level symbolic languages (like mathematical languages) provide big advantages with respect to the extensional approaches: Besides saving space, they may allow a more efficient treatment of time (see, e.g., [34]). Even more important, they provide a *human-oriented* way of handling time, allowing users to define periodicity (and possibly multiple calendars and granularities) in a natural, incremental, and compositional way. These advantages have been widely debated, e.g., in [27], [34]; in particular, the user-friendliness of symbolic languages with respect to

• *The author is with the Dipartimento di Scienze e Tecnologie Avanzate, Universitá del Piemonte Orientale "Amedeo Avogadro," Corso Borsalino 54, 15100 Alessandria, Italy. E-mail: terenz@di.unito.it.*

Manuscript received 8 July 1999; revised 2 Nov. 2000; accepted 3 Apr. 2001. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 110196.

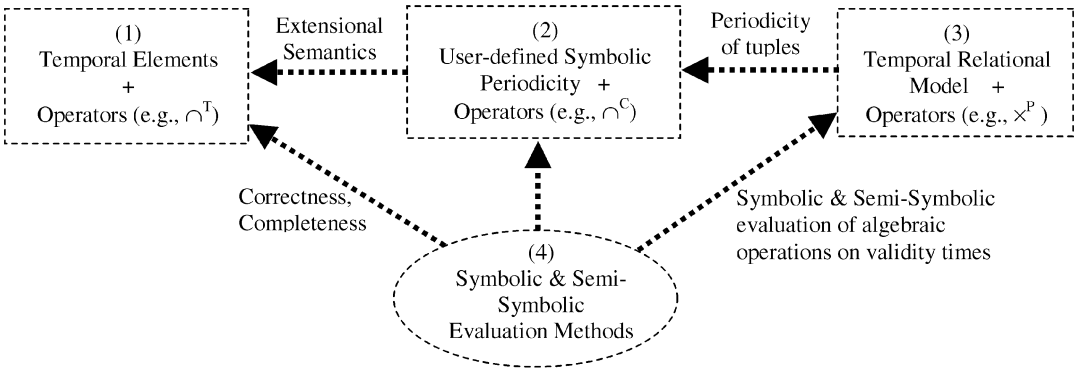

Fig. 1. Outline of our symbolic approach to periodicity in temporal relational DataBases.

mathematical ones will be fully discussed in Section 8.2 of this paper. However, no approach (except only [34], to the best of our knowledge) supports user-defined "symbolic" definitions of periodicity in the relational *data*. In this paper, we propose an approach dealing with user-defined "symbolic" periodicity in relational databases, both in the queries and in the data, thus overcoming such a limitation.

2 Outline of Our Overall Approach

The general outline of our work can be graphically represented as in Fig. 1. In a bottom up way, the approach presented in this paper can be summarized as follows:

1. We use Gadia's *temporal elements* [20] and set-operators (e.g., intersection $\cap^T$) on them.
2. We define a "symbolic" high-level language to describe user-defined periodicity. We give the extensional semantics to the language in terms of the domain of temporal elements, which is also used to demonstrate the properties of the operators in the language (e.g., intersection $\cap^C$ between periodicities).
3. We propose a relational temporal model in which periodicity (expressed using the language in point 2 can be stored as the validity time of tuples. We also define the temporal counterpart of standard algebraic operators (e.g., temporal Cartesian product $\times^P$) and introduce new temporal operators and functions. The properties of the resulting algebra are based on the properties of the operators of the symbolic language in point 2.[1]
4. We define a fully symbolic (intensional) and a semisymbolic implementation (evaluation method) of the operators in point 2. These evaluation methods support an *intensional* (implicit) treatment of user-defined periodicities in the validity time of tuples (see point 3 above). We prove the correctness and completeness of these methods on the basis of the extensional semantics of the operators in 2, i.e., on the domain of temporal elements. Notice, however, that points 1, 2, and 3 are independent of the evaluation method (e.g., fully symbolic vs. semisymbolic) one chooses to adopt.

The development of a temporal SQL based on the newly defined temporal algebra is outside the goals of this paper, as well as the definition of a Data Manipulation Language to insert, delete, and modify tuples in the temporal database.

The paper is organized as follows: Section 3 sketches the domain of temporal elements, introduces our high-level symbolic language to support user-defined periodicity (which is mainly a revision of Leban's language [27]), and studies its expressive power. Sections 4, 5, and 6 introduce our original approach to symbolic user-defined periodicity in relational databases. In particular, Section 4 introduces our temporal model, which allows one to express both the frame-time (e.g., "from 1/1/1998 to 30/3/1998") and the user-defined periodicity (e.g., "on the second Monday of each month") in the validity time of tuples. Moreover, Section 4 also presents a temporal relational algebra dealing with periodicity in the tuples and discusses its properties (equivalence and reduction to the standard atemporal algebra). Section 5 presents a fully symbolic evaluation method of the algebraic operators on periodicity, which is correct but not simplification-complete and analyzes its computational complexity. Section 6 briefly sketches a semisymbolic evaluation method which is both correct and simplification-complete. Section 7 shows some examples of query answering. The discussion of related approaches in the literature and comparisons are demanded to the last section (Section 8). Moreover, Section 8 also presents some final considerations, conclusions, and a sketch of our future work. Proofs are reported in the technical report RT58-00 [41] (the postscript RT58-00.ps of the report can be get at the URL: http://www.di.unito.it/~terenz/TECH-REP/).

3 A High-Level Symbolic Language for Representing User-Defined Periodicity

In this section, we define our high-level language to deal with *user-defined* periodicities and study its properties.

3.1 The Domain of Temporal Elements

At the bottom level, we use Gadia's temporal elements [20] (henceforth called: TE). Adopting TE is a quite standard way

1. For example, in our approach, temporal Cartesian Product $\times^P$ determines the intersection $\cap^C$ between the periodicities in the validity time of tuples. In turn, given two periodicities C1 and C2, C1 $\cap^C$ C2 corresponds (in the extensional semantics) to Gadia's intersection $\cap^T$ between the temporal elements [20] which constitute the extensions of C1 and C2, i.e., to $\mathrm{Extension(C1)} \cap^T \mathrm{Extension(C2)}$.

of dealing with time within the DB community, for which many implementations and optimizations have already been developed (see, e.g., [35], [37]). Gadia's definition of TE is based on the following notions. First, in Gadia [20], time is a linearly ordered set. For the sake of simplicity, in this paper, we assume also that time is *discrete*. A TE is a finite union of intervals $\mathrm{i_1} \cup \ldots \cup \mathrm{i_n}$. In turn, time intervals are *convex* sets of points between a starting and an ending point, and may be closed, open to the left, and/or to the right. Gadia also defined set-operators (i.e., $\cap$, $\cup$, $\neg$, and $-$) on TE's, which are basically standard set operators applied on the set of time points denoted by TE's. For example, given $\mathrm{TE1} = [3,5] \cup [8,12]$ and $\mathrm{TE2} = [4,6] \cup [15,18]$, we have, e.g., $\mathrm{TE1} \cup \mathrm{TE2} = [3,6] \cup [8,12] \cup [15,18]$ and $\mathrm{TE1} \cap \mathrm{TE2} = [4,5]$. TE is closed under these operators; TE with $\cap$, $\cup$, and $\neg$ is a Boolean algebra [20].

In the rest of the paper, we adopt the following notational variants. As in many other approaches (see, e.g., [32], [39]), we denote a $\mathrm{TE}\ \mathrm{i_1} \cup \ldots \cup \mathrm{i_n}$ by a set $\{\mathrm{i_1}, \ldots, \mathrm{i_n}\}$. We denote by i^- and i^+ the starting and ending points of an interval $\mathrm{i} =< i^-, i^+ >$ and indicate by $\cap^{\mathrm{T}}$, $\cup^{\mathrm{T}}$, $\neg^{\mathrm{T}}$, and $-^{\mathrm{T}}$ Gadia's set operators.

3.2 Symbolic Language for User-Defined Periodicity

In our approach, we aim at providing a "human-oriented" (and, thus, "user-friendly") way of dealing with time in temporal relational databases. In our opinion, considering the suggestions emerging from the analysis of *natural language* is an important step towards such a goal. In the linguistic analysis, several authors (see, e.g., Van Eynde [47]) pointed out the importance of distinguishing between the *frame time* in which the repeated event occurred (e.g., "from 1/1/98 to 31/3/98") and its *periodicity* (e.g., "on each Saturday"). We thus chose to represent a *periodic event* as a pair $<< d1, d2 >, p >$ and defined it as follows: $< d1, d2 >$ is a time interval representing the frame time, and d1 and d2 are two dates representing its starting and ending points (or $-\infty$ and/or $+\infty$ in case of infinite frame times); p represents a user-defined symbolic expression (in the language presented below) denoting a *periodicity*.

3.2.1 Semantics: Extension Function

As in [24], we choose to use the well-known and widely exploited domain of TE [20] in order to define the extensional semantics of periodicity definitions. We introduce the extension function *Ext*, which, given an expression C defining a periodicity, provides its extensional semantics in terms of the set of time intervals which constitute the extension of C. For example, if C is the definition of Mondays using our language, Ext(C) is the TE representing the set of all Mondays (each one represented by a time interval). Given a frame time $< \mathrm{x1}, \mathrm{x2} >$ and a periodicity C, the function BExt (Bounded EXTension) gives as output the extensions of C in the interval $< \mathrm{x1}, \mathrm{x2} >$ (i.e., the extension of the periodic event $<< \mathrm{x1}, \mathrm{x2} >, \mathrm{C} >$), and is defined by Definition 1 (notice that $\mathrm{BExt}((-\infty, +\infty), \mathrm{C})$ is equal to Ext(C)).

Definition 1. *Let Ext be the extension function.*

$$\mathrm{BExt}(< \mathrm{x1}, \mathrm{x2} >, \mathrm{C}) = \{\mathrm{i} \in \mathrm{Ext}(\mathrm{C}) \backslash \mathrm{Cut\_Intersects}(\mathrm{i}, < \mathrm{x1}, \mathrm{x2} >)\} \ \diamond,$$

where (in case i and $< \mathrm{x1}, \mathrm{x2} >$ *are closed intervals) Cut_Intersects gives as a result the interval* $[\mathrm{i}^-, \mathrm{i}^+]$ *if* $\mathrm{i}^- \geq \mathrm{x1} \wedge \mathrm{i}^+ \leq \mathrm{x2}$*; the interval* $[\mathrm{i}^-, \mathrm{x2}]$ *if* $\mathrm{i}^- \geq \mathrm{x1} \wedge \mathrm{i}^+ > \mathrm{x2} \wedge \mathrm{i}^- < \mathrm{x2}$*; the interval* $[\mathrm{x1}, \mathrm{i}^+]$ *if* $\mathrm{i}^- < \mathrm{x1} \wedge \mathrm{i}^+ \leq \mathrm{x2} \wedge \mathrm{i}^+ > \mathrm{x1}$ *and* $\oslash$ *otherwise.*

In some cases, the definition of a periodicity is such that its extension is a TE covering the whole time-line. We call these periodicities *convex periodicities* (which correspond, e.g., to *calendars* in [27]). In the following, we describe the operators of our symbolic language to define *user-defined periodicity*, showing their extensional semantics on TE's.

3.2.2 Convex Periodicity

The first step when defining a periodicity is the definition of a *basic periodicity* (the basic granularity—tick—of the system, e.g., "Days" in this section). Moreover, extensions of a periodicity can be more easily represented as distances from a given reference time point (e.g., 1/1/1998 in this section; we express dates as day/month/year). The approach we propose is independent of the basic periodicity and of the reference time point. In our approach, a convex periodicity C can be built on the basis of another convex periodicity C' using the operator Convex_Cal, whose type is the following:

$\mathrm{P} \times \mathrm{CPer} \times \mathrm{N} \times \ldots \times \mathrm{N} \rightarrow \mathrm{CPer}$ (N stands for natural numbers and CPer for a *convex periodicity*).

(UC1) $\quad \mathbf{C = Convex\_Cal(X_0; C'; (g_1, \ldots, g_k))}.$

UC1 defines a new periodicity C on the basis of anchor time point $\mathrm{X_0}$, of a convex periodicity C', and of a set of natural numbers $\mathrm{g_1}, \ldots, \mathrm{g_k}$, stating that C is obtained from C' by "merging together to form the smallest covering interval" sets of $\mathrm{g_1}, \ldots, \mathrm{g_k}$ consecutive extensions of C, using the anchor time point $\mathrm{X_0}$ as a synchronization point for the generation of extensions. For example, weeks and months[2] can be defined as in (ex.1) and (ex.2); 5/1/1998 is the start of a European week.

(ex.1) Weeks = Convex_Cal(5/1/1998; Days; (7))

(ex.2) Months = Convex_Cal(1/1/1998;Days; (31,28,31,30,31,30,31,31,30,31,30,31,31,28,31,30, 31,30,31,31,30,31,30,31,31,29,31,30,31,30,31,31, 30,31,30,31,31,28,31,30,31,30,31,31,30,31,30,31))

(E1) shows the extension of Weeks in the time from 1/2/1998 to 30/3/1998; this extension is a TE (where, e.g., [0,0] represents the first day of 1998 and [31,31] the first day of February, giving the distance of its starting and its ending points from the reference time):

2. Notice that, if we considered that years multiple of 100 have 365 days except multiple of 400 (which have 366 days), the list in the definition of months should have 400*12 entries. To the best of our knowledge, this limitation is shared by all the approaches to the definitions of periodicities and calendars in the artificial intelligence and in the temporal databases literature (of course, some syntactic sugar can be used to avoid the repetition of identical patterns).

(E1) BExt([1/2/1998,30/3/1998],Weeks)
=[[31,31],[32,38],[39,45],[46,52],[53,59],
[60,66],[67,73],[74,80],[81,87],[88,88]]

3.2.3 Nonconvex Periodicity

Nonconvex periodicities are periodicities whose extensions have "holes" on the timeline. For example, the extension of Mondays is a TE which is not convex. In our approach, nonconvex periodicities can be defined as the **Select_Periods** operator. The inputs of Select_Periods are a natural number *n*, a relational operator *rop*, and two periodicities (convex or not) *C1* and *C2*, and the output is a new periodicity *C*. The type of Select_Periods is the following: $\mathrm{N} \times \mathrm{Per} \times \mathrm{Rop} \times \mathrm{Per} \rightarrow \mathrm{Per}$ (Per stands for a periodicity in our language and Rop for a relational operator). In our approach, we consider the relational operators Meets, Met-by, Starts, During, Equal, Overlaps, Overlapped-by, and Finishes (as defined in Allen's Interval Algebra [3]) plus the operators Cut_Intersects above and NSDur (acronym for Non Strict DURing, denoting strict and nonstrict containment), which represent the following Allen's relation: $\mathrm{NSDur(i,j)} = \mathrm{Starts(i,j)}$, $\mathrm{During(i,j)}$, $\mathrm{Equal(i,j)}$, or $\mathrm{Finishes(i,j)}$. From the syntactic point of view, we represent an application of Select_Periods as in (UC2.1).

(UC2.1) **C = n / C1 rop C2**

For example, Select_Periods can be used in order to define Mondays as the set of the first days of the (European) weeks (see ex.3), and to define the second Mondays of each month as shown in (ex.4):

(ex.3) Mondays = 1/ Days NSDur Weeks
(ex.4) 2^{nd} Mondays-of-Months
= 2/ Mondays NSDur Months

The extensional semantics of the periodicities built using Select_Periods can be always given in terms of TE's. For example,

(E2) BExt([1/2/1998,30/3/1998], Mondays)
= {[32,32], [39,39], . . ., [81,81], [88,88]} and
(E3) BExt([1/2/1998,30/3/1998],
2^{nd}-Mondays-of-Months)= {[39,39], [67,67]}

We also introduce in our language other ways of selecting periods. For example, (UC2.2) selects the nth intervals starting from the last interval of each set $\mathrm{S_i}$ (i.e., the nth in the reverse order) and (UC2.3) allows to specify a range of selections.

(UC2.2) **C = -n / C1 rop C2** (where n is a natural number)
(UC2.3) **C = [n-m] / C1 rop C2** (where n and m are natural numbers, and $\mathrm{n} \leq \mathrm{m}$)

For example, the first three days of each week (considering each week as a unique three-days-long time interval) can be defined as in (ex.5) and the extensional semantics of (ex.5) from 1/2/98 to 30/3/98 is the TE shown in (E4):

(ex.5) 1^{st}3Days_of_Weeks =
[1-3] / Days NSDur Weeks
(E4) BExt([1/2/98,30/3/98],
1^{st}3Days_of_Weeks) = {[32,34], [39,41],
[46,49],. . . , [81,84], [88,88]}

Finally, we introduce the set-operators of **intersection** $\cap^{\mathbf{C}}$, **union** $\cup^{\mathbf{C}}$, **difference** $-^{\mathbf{C}}$, and **complement** $\neg^{\mathbf{C}}$, which perform the usual set-operations on *periodicities* and on *periodic events*. We give their extensional semantics in terms of union $\cup^{\mathrm{T}}$, intersection $\cap^{\mathrm{T}}$, difference $-^{\mathrm{T}}$, and complement $\neg^{\mathrm{T}}$ on TE's [20]. For example, given two periodic events P1 and P2, the extension of $\mathrm{P1} \cap^{\mathrm{C}} \mathrm{P2}$ is the intersection $\cap^{\mathrm{T}}$ between the extension of P1 and the extension of P2 (which are TE's). Formally,

$$\mathrm{BExt(P1} \cap^{\mathrm{C}} \mathrm{P2)} = \{\mathrm{BExt(P1)} \cap^{\mathrm{T}} \mathrm{BExt(P2)}\}$$
$$\mathrm{BExt(P1} \cup^{\mathrm{C}} \mathrm{P2)} = \{\mathrm{BExt(P1)} \cup^{\mathrm{T}} \mathrm{BExt(P2)}\}$$
$$\mathrm{BExt}(\neg^{\mathrm{C}}\mathrm{P1}) = \{\neg^{\mathrm{T}}\mathrm{BExt(P1)}\}$$
$$\mathrm{BExt(P1} -^{\mathrm{C}} \mathrm{P2)} = \{\mathrm{BExt(P1)} -^{\mathrm{T}} \mathrm{BExt(P2)}\}$$

The operators we introduced can be composed in order to define complex user-defined periodic events/periodicities. For example, one might incrementally define the set of "Mondays and Wednesdays which are the first days of the months in 1998" by defining *Weeks* (as in ex.1), *Mondays* (as in ex.3) and *Wednesdays* (analogous to ex.3), $1^{st}\mathit{Days\_of\_Months}$ $(1/\mathrm{Days\ NSDur\ Months} >)$ and using them in a definition like (ex.6).

(ex.6) $1^{\mathrm{st}}\mathrm{Mon\&Wed\_1998} = (\mathrm{Mondays} \cup^{\mathrm{C}} \mathrm{Wednesdays}) \cap^{\mathrm{C}} ([1/1/1998, 31/12/1998] 1^{\mathrm{st}}\mathrm{Days\_of\_Months})$

3.2.4 Properties

Property 1. $\mathrm{Convex\_Cal}$, $\mathrm{Select\_Periods}$, $\cap^{\mathrm{C}}$, $\cup^{\mathrm{C}}$, $\neg^{\mathrm{C}}$, *and* $-^{\mathrm{C}}$ *operate (by definition) on periodicities/periodic events whose extensions are TE's and give as output periodicities/periodic events whose extensions are still TE's.*

The semantics of $\cap^{\mathrm{C}}$, $\cup^{\mathrm{C}}$, $\neg^{\mathrm{C}}$, and $-^{\mathrm{C}}$ is based on the corresponding operators on TE's (i.e., $\cap^{\mathrm{T}}$, $\cup^{\mathrm{T}}$, $\neg^{\mathrm{T}}$, and $-^{\mathrm{T}}$, respectively). If we define ALL as the periodicity whose extension is the TE $\{(-\infty, +\infty)\}$, and "Ø" as the periodicity whose extension is {}, the algebraic properties of $\cap^{\mathrm{T}}$, $\cup^{\mathrm{T}}$, and $\neg^{\mathrm{T}}$ (see [20]) also hold for $\cap^{\mathrm{C}}$, $\cup^{\mathrm{C}}$, and $\neg^{\mathrm{C}}$.

Property 2. *The following properties hold for* $\cap^{\mathrm{C}}$, $\cup^{\mathrm{C}}$, *and* $\neg^{\mathrm{C}}$: *Commutativity of* $\cap^{\mathrm{C}}$ *and* $\cup^{\mathrm{C}}$; *Associativity of* $\cap^{\mathrm{C}}$ *and* $\cup^{\mathrm{C}}$; *Distributivity of* $\cap^{\mathrm{C}}$ *with respect to* $\cup^{\mathrm{C}}$ *and vice-versa; Absorption:* $\mathrm{E1} \cap^{\mathrm{C}} (\mathrm{E1} \cup^{\mathrm{C}} \mathrm{E2}) = \mathrm{E1}$ *and* $\mathrm{E1} \cup^{\mathrm{C}} (\mathrm{E1} \cap^{\mathrm{C}} \mathrm{E2}) = \mathrm{E1}$; *Existence of neutral elements* $\mathbf{0} = \oslash$ *and* $\mathbf{1} = \mathrm{ALL}$ *such that* $\mathrm{E} \cup^{\mathrm{C}} \oslash = \mathrm{E}$ *and* $\mathrm{E} \cap^{\mathrm{C}} \mathbf{ALL} = \mathrm{E}$; *Complementation:* $\mathrm{E} \cap^{\mathrm{C}} (\neg^{\mathrm{C}}\mathrm{E}) = \oslash$ *and* $\mathrm{E} \cup^{\mathrm{C}} (\neg^{\mathrm{C}}\mathrm{E}) = \mathbf{ALL}$.

Property 2 trivially follows from the definition of $\cap^{\mathrm{C}}$, $\cup^{\mathrm{C}}$, and $\neg^{\mathrm{C}}$ and from the properties of Gadia's set operators on TE [20]. As regards the operator $\mathrm{Select\_Periods}$, it is clearly not commutative; on the other hand, Property 3 holds.

Property 3. *The operator Select_Periods (applied using the relational operator NSDur) is associative.* $\mathrm{Select\_Periods}$ *does not distribute over Intersection* $\cap^{\mathrm{T}}$ *and Union* $\cup^{\mathrm{T}}$, *which, in turn, do not distribute over* $\mathrm{Select\_Periods}$.

3.3 Expressive Power

Let us call "**Cal**" our language for defining periodic events (i.e., a pair $<< \mathrm{s,e} >, \mathrm{C} >$—see Section 3.2). In order to

evaluate Cal's expressive power, we compare it with both a standard reference approach to periodicity (namely, Presburger arithmetic) and a very recent approach proposed within the DataBase community (see [8], [9]).

Presburger Arithmetic is the additive fragment of the arithmetic over Natural numbers and is decidable by quantifier elimination (see, e.g., [19]). Moreover, Enderton proved Property 4:

Property 4 [19: Theorem 32F]. *A set of natural numbers is* definable *in (N, 0, S, <, +) (also called Presburger Arithmetic) iff it is eventually periodic (where* eventually periodic numbers *are defined in Definition 2).*

Definition 2 [19]. *A set* D *of natural numbers is* periodic *if for some positive* p, *a number* n *is in* D *iff* $n + p$ *is in* D; D *is* eventually periodic *iff there exist positive numbers* M *and* p *such that for all* n *greater than* M, $n \in D$ *iff* $n + p \in D$.

We draw the comparison between Cal and Presburger arithmetic at the *extensional level*. We notice that each *periodic event* in Cal has a *temporal extent*, which is a set of *time intervals*. Moreover, since our time intervals are convex by definition, they can be compactly represented by a pair <*start point, end point*>. In turn, each time point can be represented by a *Natural number* (see Definition 3). Thus, we could compare the *sets of pairs of Natural numbers* defined by Cal with those defined by Presburger arithmetic. Such a comparison is carried on in a separate ongoing paper of ours. On the other hand, in this paper, we show that, even in case we restrict our attention to *sets of Natural numbers*, the expressiveness of Cal (or, better, of Cal^-; see below) and of Presburger arithmetic is the same. First of all, since Presburger arithmetic is over Natural numbers, we have to restrict our approach to deal only with finite and *right-infinite* periodic events. This can be obtained by restricting Cal, imposing that the starting point of each *frame time* cannot be less than the reference time point (RT, which is assumed to have the value "0"). Let Cal^- be the restricted language. We can now introduce our notion of *definability* for Cal^- and state Property 5 (the proof is in [41]).

Definition 3. *Let* $\mathrm{StartEndPoints}(<< s, e >, C >)$ *be the function that returns the set of starting and ending points of the time intervals that constitute the temporal extent of the periodic event* $<< s, e >, C >$ *in* Cal^-. *Moreover, given a set of time points* S, *let* NNum(S,RT,BP) *be the function that returns the set of Natural numbers obtained by evaluating the distance of each point in* S *from the reference time point* RT, *using the basic granularity* BP *(given a periodic event* $<<s, e>, C>$ *, the set of numbers* $\mathrm{NNum}(\mathrm{StartEndPoints}(<<s, e>, C >), \mathrm{RT}, \mathrm{BP})$ *can be evaluated as shown in [41]).*

Definition 4. *A periodic event* $<<s, e>, C>$ *expressed in* Cal^- defines *the set of natural numbers* $\mathrm{NNum}(\mathrm{StartEndPoints}(<< s, e >, C >), \mathrm{RT}, \mathrm{BP})$.

Property 5. *Given the Definition 4 above of* definability *for* Cal^-, *the expressions in* Cal^- define *the eventually periodic sets of natural numbers (i.e., the same set of numbers defined by Presburger Arithmetic).*

Recently, Bettini et al. [8] and Bettini and De Sibi [9] proposed a mathematical characterization of granularity, aiming at providing a standard reference in the area of temporal databases. Since Bettini's granularities are a proper superset of *periodic granularities* (corresponding to our *periodic events*), they constitute a natural reference to evaluate the expressiveness of Cal. Bettini et al. [8] defined granularities as mappings from integers to subsets of the time domains. They defined *periodical granularities* (which are those granularities which are periodical with respect to the basic granularity) and classified them on the basis of two main parameters: boundedness of the extensions and convexity of the time intervals in the extensions [8], [9]. Finally, Bettini and De Sibi [9] extended Leban's approach to deal also with *nonconvex* (called *gap*) *intervals*. On the basis of their definitions, Bettini and De Sibi showed that, given a *no-gap periodical granularity* G, there is an *equivalent periodicity C expressed using Leban's language, and vice-versa*. Since our language Cal is quite close to Leban's one, we use the same technique in order to prove that Cal covers *no-gap periodical granularities*[3] in [9], i.e., those periodical granularities in which each granule is a convex time interval (i.e., no gap is allowed in a single granule). We first define equivalence between a *periodical granularity* in [9] and a *periodic event* in Cal and then we prove Property 6 (the proof is in [41]).

Definition 5. *A periodical granularity in [9] is equivalent to a periodic event in Cal (see Section 3) if and only if they denote the same subsets of the time domain.*

Property 6. *Given a no-gap periodical granularity G (defined as in [9]), there is an equivalent periodic event* $<< s, e >, C >$ *expressed using Cal, and vice-versa.*

4 Temporal Model and Algebra

In this section, we extend the relational model and algebra to cope with *periodic events*.

4.1 Periodic Tables

In our approach, a temporal database consists of standard atemporal tables, bitemporal tables [36], and also of *periodic* tables, with a *data part*, a *validity time T*, and a *transaction time*. Additional system tables store the definitions of user-defined periodicities. The validity time T of each tuple in a periodic table is a pair consisting of a time interval (representing a *frame time*) and a symbolic expression (defined in the language in Section 3) representing a *periodicity* (e.g., $< [1/10/98 - 31/1/99], \mathrm{Mon} : 12\text{-}13 \cup^{C} \mathrm{Wed} : 10\text{-}12 >$ in the first tuple of COURSES in Fig. 2). In the following, we focus only on periodic tables and, for the sake of clarity and brevity, we do not consider the

3. Notice that the fact that we only consider "no-gap" periodical granularities does not mean that we only consider periodicities which cover the whole time line (i.e., convex periodicity in the terminology in Section 3.2.2). In fact, we also consider nonconvex periodicities such as, e.g., "Mondays" (see Section 3.2.3). We left out only the possibility of dealing with cases in which time intervals "per se" in the extensions of a periodicity are nonconvex. For example, let us consider the periodicity "Mondays plus Wednesdays." We can deal with it using union $\cup^{C}$ (i.e., MW = Mondays $\cup^{C}$ Wednesdays). However, the extension of MW is a set of convex time intervals covering each one a day, being it a Monday or a Wednesday. On the other hand, our language does not allow one to define a periodicity whose extension is a set of nonconvex time intervals in which the same interval covers a Monday and a Wednesdays (having a gap in it).

COURSES

Course-Cod	Teach-Cod	Room-Cod	T
Math1	P11	R01	<[1/10/98, 31/1/99], Mon:12-13 $\cup^{C}$ Wed:10-12>
Math1	A08	R01	<[1/10/98, 31/1/99], Thu:9-10>
Phys1	P07	R01	<[1/10/98, 31/1/99], Mon:9-10 $\cup^{C}$ Fri:9-10

OFFICE

Teach-Cod	T
P11	<[1/1/98, 31/12/98], Mon:11-13 $\cup^{C}$ Tue:9-11>
P07	<[1/10/98, 31/1/99], Tue:9-11>
A02	<[1/1/98, 31/1/99], Mon:14-16 >

MEET

Meet-Cod	Room-Cod	T
M03	R01	<[1/1/98, 31/12/98], 1stMon_of_Months:14-16>

Fig. 2. Timetable in a school.

transaction time. Data part is not compulsory in periodic tables, which may contain as validity time any periodicity defined using our language (provided that a definition has been stored in a dedicated system table). Tuples in the same table may hold at different periodicities and granularities (e.g., 11^{th} Hours of Mondays vs. first three Days of each Month). Fig. 2 shows a simplified example of a timetable in a school. COURSES states, for each course (attribute Course-Cod), teacher, classroom, and validity time. OFFICE describes the office-hours of teachers. MEET gives classroom and time of meetings (Meet-Cod). In the example, we take Hours as the *basic periodicity*. XXX:n-m is a mnemonic symbolic name for the interval consisting of the (n+1)-th, ..., (m)th hours of day XXX. For example, the user definition for "Wed:10-12" (stored in a separate dedicated table) can be given as shown by (ex.7) below, and has as extension Wednesdays between 10 and 12. Finally, $1^{\text{st}}\text{Mon\_of\_Months}$ can be defined as in (ex.8).

(ex.7) Wed:10-12 = [11-12] \ Hours NSDur Wednesdays
(ex.8) $1^{\text{st}}\text{Mon\_of\_Months} = 1$ \ Mondays NSDur Months

4.2 Algebraic Operators on Periodic Tables: Basic Issues

In our model, each *validity time* $<< \mathrm{d1}, \mathrm{d2} >, \mathrm{C} >$ is a symbolic (intensional) representation of a temporal element (i.e., $\mathrm{BExt}(< \mathrm{d1}, \mathrm{d2} >, \mathrm{C})$). Thus, we could operate at the extensional level, applying Gadia's operators [20]. On the other hand, we define the operators σ^{P}, π^{P}, $\times^{\mathrm{P}}$, $-^{\mathrm{P}}$, $\cup^{\mathrm{P}}$, which extend standard snapshot operators (i.e., σ, π, $\times$, $-$, $\cup$) to work on *periodic tables*, operating directly on our symbolic representation.[4] We denote by Sch(R) the schema of the data part of R and, by t(S), a tuple t restricted to the schema S. t(FT) denotes the *frame time* of a tuple t and t(PER) its *periodicity* (e.g., if t is the first tuple of COURSES, t(FT) = [1/10/98, 31/11/99] and $\mathrm{t(PER)} = \mathrm{Mon} : \text{12-13} \cup^{\mathrm{C}} \mathrm{Wed} :$ 10-12). Finally, $\cap^{\mathrm{I}}$ stands for standard intersection between two time intervals.

An important issue concerns the treatment of validity times of tuples with equal data part in Projection, Union, and Difference since one has to compose only the periodicities of those tuples whose frame times intersects in time. Let us consider, e.g., the example in Fig. 3, where we have two periodic tables R1 and R2 containing tuples with equal data part t (omitted in the figure), and different validity times, and we want to compute $\mathrm{R} = \mathrm{R1} \cup^{\mathrm{P}} \mathrm{R2}$ (with no loss of generality, in this example, we represent frame times as pair of numbers; Mon stands for Mondays, etc.).

The time interval [10,12) is covered by the frame time of the 1st tuple of R1 only, so that the tuple

$$\mathrm{t1} = \#\ \mathrm{t}| < [10, 12), \mathrm{Mon} \cup^{\mathrm{C}} \mathrm{Tue} \cup^{\mathrm{C}} \mathrm{Wed} > \#$$

belong to R (notation: we use "#" as tuple delimiter). On the other hand, [12,13) is covered by the 1st tuple of R1 and by the 1st tuple of R2. Thus, the union of the periodicities of the two tuples (i.e., $(\mathrm{Mon} \cup^{\mathrm{C}} \mathrm{Tue} \cup^{\mathrm{C}} \mathrm{Wed}) \cup^{\mathrm{C}} (\mathrm{Wed} \cup^{\mathrm{C}} \mathrm{Thu})$) must be given in output, i.e.,

$$\mathrm{t2} = \#\mathrm{t}| < [12, 13), \mathrm{Mon} \cup^{\mathrm{C}} \mathrm{Tue} \cup^{\mathrm{C}} \mathrm{Wed} \cup^{\mathrm{C}} \mathrm{Thu} > \# \in \mathrm{R}.$$

Since [13,13] is covered by the 1st tuple of R1 and by both tuples of R2,

$$\mathrm{t3} = \#\ \mathrm{t}| < [13, 13], \mathrm{Mon} \cup^{\mathrm{C}} \mathrm{Tue} \cup^{\mathrm{C}} \mathrm{Wed} \cup^{\mathrm{C}} \mathrm{Thu} \cup^{\mathrm{C}} \mathrm{Fri} > \#$$

belongs to R, etc. Thus, in general, the definition of $\cup^{\mathrm{P}}$ (and of π^{P} and $-^{\mathrm{P}}$) involve considering the *fragments* (e.g., [10,12), [12,13), [13,13], (13,15], [16,16], [19,20] in the example above) of frame times of the tuples with equal data part.

4.3 Defining Fragments

In the following, given a tuple t and a relation R, *EqData(t,R)* denotes the set of all (and only) the tuples in R having the same data part as t. Let t be a tuple in a periodic table R' and let $\mathrm{t}(\mathrm{R}')$ indicate the data part of t. Let R a periodic table

4. Notice that the very same table can contain different periodicities expressed at different user-defined granularities and that the operators of the relational algebra operate on tables regardless of whether the granularities are different or not.

R1

	Data	T
1		<[10,15], Mon $\cup^C$ Tue $\cup^C$ Wed>
2		<[19,20], Mon>

R2

	Data	T
3		<[12,13], Wed $\cup^C$ Thu >
4		<[13,16], Mon $\cup^C$ Fri>

R

Data	T
	<[10,12), Mon $\cup^C$ Tue $\cup^C$ Wed >
	<[12,13), Mon $\cup^C$ Tue $\cup^C$ Wed $\cup^C$ Thu >
	<[13,13], Mon $\cup^C$ Tue $\cup^C$ Wed $\cup^C$ Thu $\cup^C$ Fri>
	<(13,15], Mon $\cup^C$ Tue $\cup^C$ Wed $\cup^C$ Fri >
	<(15,16], Mon $\cup^C$ Fri >
	<[19,20], Mon>

Fig. 3. $\mathrm{R} = \mathrm{R1} \cup^{\mathrm{P}} \mathrm{R2}$ (in case tuples denoted by 1, 2, 3, and 4 have the same data part). Only the validity time T is shown.

(possibly R') such that $\mathrm{Sch(R)} = \mathrm{Sch(R')}$. We define EqData as follows:

Definition 6. $EqData(t, R) = \{\mathrm{s} \backslash \mathrm{s} \in \mathrm{R}, \mathrm{s(R)} = \mathrm{t(R)}\}$.

Projection, union, and difference should consider the set of all the frame times of tuples t′ in EqData(t,R). Let $FT's(\mathrm{S})$ denote the set of the frame times of a set S or tuples.

Definition 7. $FT's(S) = \{\mathrm{t(FT)} \backslash \mathrm{t} \in \mathrm{S}\}$.

Now, using all the starting and ending points of the intervals in FT's(EqData(t,R)), one can subdivide the parts of the timeline covered by FT's(EqData(t,R)) into smaller nonoverlapping subintervals, which cover all spans of time covered by intervals in FT's(EqData(t,R)) and have as endpoints the endpoints of such intervals (see Fig. 4). Let *Fragments*(FT's(EqData(t,R)) be the set of such subintervals. Given a set ISet of intervals representing frame times, let IStarts(ISet) and IEnds(Iset) be the sets of starting and ending points of the intervals in ISet, and let $\mathrm{IBounds(Iset)} = \mathrm{IStarts(Iset)} \cup \mathrm{IEnds(Iset)}$, and let us assume, for the sake of brevity, that all intervals in ISet are closed (only point 4 in the definition below is affected by this assumption). Fragments(Iset) can be defined as follows:

Definition 8. $Fragments(ISet) = \{\mathrm{i}, \text{time-interval(i)}\}$ *such that*

1. $\exists \mathrm{j} \in \mathrm{ISetNSDur(i,j)} \wedge$,
2. $\forall \mathrm{j}\ \text{time-interval(j)}, \mathrm{j} \in \mathrm{ISet}, \forall \mathrm{p}\ \text{time-point(p)}, \mathrm{p} \in \mathrm{j} \Rightarrow \exists \mathrm{j'} \text{time-interval(j')}, \mathrm{j'} \in \mathrm{Fragments(ISet)} \wedge \mathrm{p} \in \mathrm{j'} \wedge,$
3. $\forall \mathrm{p}\ \text{time-point(p)}, \mathrm{p} \in \mathrm{IBounds(ISet)} \Rightarrow$ not $(\exists \mathrm{j'} \text{time-interval(j')}, \mathrm{j'} \in \mathrm{Fragments(ISet)} \wedge \mathrm{During(p, j')}) \wedge$, and
4. $(\mathrm{i} = [\mathrm{i}^-, \mathrm{i}^+]$ if $\mathrm{i}^- \in \mathrm{IStarts(ISet)} \wedge \mathrm{i}^+ \in \mathrm{IEnds(ISet)}) \vee$
$(\mathrm{i} = [\mathrm{i}^-, \mathrm{i}^+)$ if $\mathrm{i}^- \in \mathrm{IStarts(ISet)} \wedge \mathrm{i}^+ \in \mathrm{IStarts(ISet)}) \vee$
$(\mathrm{i} = (\mathrm{i}^-, \mathrm{i}^+]$ if $\mathrm{i}^- \in \mathrm{IEnds(ISet)} \wedge \mathrm{i}^+ \in \mathrm{IEnds(ISet)}) \vee$
$(\mathrm{i} = (\mathrm{i}^-, \mathrm{i}^+)$ if $\mathrm{i}^- \in \mathrm{IEnds(ISet)} \wedge \mathrm{i}^+ \in \mathrm{IStarts(ISet)})$.

Intuitively, condition 1 states that each interval i in Fragments(ISet) must be contained (properly or not) into an interval in ISet. Condition 2 states that Fragments(ISet) covers all time points covered by ISet. Condition 3 states that no starting/ending point of ISet can be *properly* contained (predicate During(p,i) holds if the point p is such that $\mathrm{i}^- < \mathrm{p} < \mathrm{i}^+$) into any interval in Fragments(ISet); this grants that the intervals in Fragments(ISet) are nonoverlapping and are the smallest intervals one can build considering the points in IBounds as starting and ending points. Finally, condition 4 states how starting/ending points of i are obtained, given the time points in IStarts and IEnds. The graphical example in Fig. 4 better illustrates the intuition underlying the definition of Fragments(Iset). The frame times in Fig. 4 are those we used in the example in Fig. 3. In the example in Fig. 4, ISet = {[10,15],[12,13],[13,16],[19,20]}, IStarts = {10,12,13,19}, IEnds = {13,15,16,20}, IBounds = {10,12, 13,15,16,19,20}, and Fragments(ISet) = {[10,12),[12,13),[13,13], (13,15],(15,16],[19,20]}. For example, the first interval i = [10,12) in Fragments(ISet) is obtained by taking as starting point the first point in IBounds (i.e., 10). The ending point should be the next point in IBounds (i.e., 12). However, since $12 \in \mathrm{IStarts}$, the interval is open to the right. The next interval has 12 as a starting point, the next point in IBounds (i.e., 13) as an ending point, and so on.

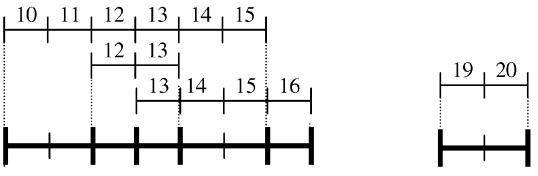

Fig. 4. Evaluation of fragments of the set of time intervals {[10,15], [12,13],[13,16],[19,20]}. The result (shown in bold in the figure) is the set {[10,12),[12,13),[13,13],(13,15],(15,16],[19,20]}.

4.4 Primitive Algebraic Temporal Operators

Now, we can define the operators σ^{P}, π^{P}, $\times^{P}$, $-^{P}$, $\cup^{P}$, which extend the standard snapshot algebraic operators (i.e., σ, π, $\times$, $-$, $\cup$) to operate on periodic tables. Notice that, in all the definitions, tables contain only those tuples t such that both t(FT) and t(PER) are not empty. Card(S) denotes the cardinality of a set S, and **Intersects(i,j)** holds if two time intervals i and j intersect in time (formally, it stands for the disjunction of Allen's relations [3] Overlaps(i,j), Overlapped-by(i,j), Starts(i,j), Started-by(i,j), Contains(i,j), During(i,j), Equal(i,j), Finishes(i,j), or Finished-by(i,j)).

Selection. (Atemporal) selection selects the tuples whose data part satisfies a condition ϕ (which is a condition on the data part only), regardless of its temporal part.

$$\mathrm{Sch}(\sigma^{P}{}_{\phi}(R)) = \mathrm{Sch}(R),$$
$$\sigma^{P}{}_{\phi}(R) = \{t | t \in R, \phi(t(R))\}.$$

Cartesian product. We chose to adopt the intersection semantics for the Cartesian product. Thus, the FT of the resulting tables is the (time interval) intersection $\cap^{I}$ of the frame times, and the periodicity is the intersection $\cap^{C}$ between periodicities.

$$\mathrm{Sch}(R1 \times^{P} R2) = \mathrm{Sch}(R1) \cup \mathrm{Sch}(R2),$$
$$R1 \times^{P} R2 = \{s | \exists t1 \in R1, \exists t2 \in R2, s(R1) = t1(R1),$$
$$s(R2) = t2(R2), s(FT) = t1(FT) \cap^{I} t2(FT),$$
$$s(PER) = t1(PER) \cap^{C} t2(PER), s(FT) \neq \oslash, s(PER) \neq \oslash\}.$$

Union. The union operator $\cup^{P}$ makes the union of two periodic tables R1 and R2. Tuples with different data part are put unchanged into the resulting table. On the other hand, the union of a set S of tuples with equal data part (e.g., of the tuples in $\mathrm{EqData}(t, R1) \cup \mathrm{EqData}(t, R2)$) is obtained as exemplified in Section 4.2. More formally, such a union is a set of j tuples, where j is the cardinality of Fragments(FT's(S)) (i.e., Card(Fragments(FT's(S))) = j). Each resulting tuple has the same data part, has as the frame time one of the fragments (say I), and as periodicity the union of the periodicities of all the tuples $t'_1, \ldots, t'_k$ in S such that $t'_i(FT)$ covers I, $1 \leq i \leq k, k \geq 1$ (i.e., $t'_1(PER) \cup^{C} \ldots \cup^{C} t'_k(PER)$). Formally, union is defined as follows:

$\mathrm{Sch}(R1 \cup^{P} R2) = \mathrm{Sch}(R1)(\mathrm{Sch}(R1) = \mathrm{Sch}(R2))$
$R1 \cup^{P} R2 = \{s | (\exists t1 \in R1,$
$\mathrm{Card}(\mathrm{EqData}(t1, R1) \cup \mathrm{EqData}(t1, R2)) = 1,$
$s(R1) = t1(R1), s(FT) = t1(FT), s(PER) = t1(PER)),$
OR $(\exists t1 \in R1, \mathrm{Card}(\mathrm{EqData}(t1, R1) \cup \mathrm{EqData}(t1, R2)) > 1,$
$\exists I, t'_1, \ldots, t'_k s(R1) = t1(R1), s(FT) = I,$
$s(PER) = t'_1(PER) \cup^{C} \ldots \cup^{C} t'_k(PER),$
where $I \in \mathrm{Fragments}(FT's(\mathrm{EqData}(t1, R1) \cup \mathrm{EqData}(t1, R2)))$ and $t'_1, \ldots, t'_k$ are all and only the tuples $t' \in (\mathrm{EqData}(t1, R1) \cup \mathrm{EqData}(t1, R2))$ such that $\mathrm{NSDur}(I, t'(FT))$ holds,
OR $(\exists t2 \in R2, \mathrm{Card}(\mathrm{EqData}(t2, R1) \cup \mathrm{EqData}(t2, R2)) = 1,$
$s(R1) = t2(R2), s(FT) = t2(FT), s(PER) = t2(PER)),$
OR $(\exists t2 \in R2, \mathrm{Card}(\mathrm{EqData}(t2, R1) \cup \mathrm{EqData}(t2, R2)) > 1,$
$\exists I, t'_1, \ldots, t'_k s(R1) = t2(R1), s(FT) = I,$
$s(PER) = t'_1(PER) \cup^{C} \ldots \cup^{C} t'_k(PER)$, where $I \in \mathrm{Fragments}(FT's(\mathrm{EqData}(t2, R1) \cup \mathrm{EqData}(t2, R2)))$ and $t'_1, \ldots, t'_k$ are all and only the tuples $t' \in (\mathrm{EqData}(t2, R1) \cup \mathrm{EqData}(t2, R2))$ such that $\mathrm{NSDur}(I, t'(FT))$ holds}.

Projection. Projection can be used to select data attributes. In case the data parts of the tuples obtained by projection on the attributes A is different, the frame time and the periodicity are not changed by Projection. On the other hand, for each tuple t such that $\mathrm{Card}(\mathrm{EqData}(t(A), R(A))) > 1$ (i.e., such that there is at least another tuple t' in R such that $t(A) = t'(A); R(A)$ indicates the restriction of a table R to the attributes in A) the union of periodicities must be performed in the intersecting parts of the frame times, as in the case of union above. More precisely, projection is defined as follows:

$\mathrm{Sch}(\pi^{P}{}_{A}(R)) = A(A \subseteq \mathrm{Sch}(R))$
$\pi^{P}{}_{A}(R) = \{s | (\exists t1 \in R, \mathrm{Card}(\mathrm{EqData}(t1(A), R(A))) = 1,$
$s(A) = t1(A), s(FT) = t1(FT), s(PER) = t1(PER))$ OR
$(\exists t, I, t'_1, \ldots, t'_k t \in R, \mathrm{Card}(\mathrm{EqData}(t(A), R(A))) > 1,$
$s(A) = t(A), s(FT) = I, s(PER) = t'_1(PER) \cup^{C} \ldots \cup^{C} t'_k(PER),$
where $I \in \mathrm{Fragments}(FT's(\mathrm{EqData}(t(A), R(A))))$ and
$t'_1, \ldots, t'_k$ are all and only the tuples
$t' \in (\mathrm{EqData}(t(A), R(A)))$ such that
$\mathrm{NSDur}(I, t'(FT))$ holds }.

Difference. Difference between two periodic tables R1 and R2 gives as result a table containing all the tuples of R1 which are different in the data part from all tuples in R2. Tuples with the same data part are dealt with considering the fragments of their frame times (as in the case of union). In particular, let S1 = EqData(t,R1) and S2 = EqData(t,R2). Then, the fragments $F = \mathrm{Fragments}(FT's(S1 \cup S2))$ must be considered. For each time interval (fragment) $I \in F$, let $\{t_1, \ldots t_k\}$ the set of all tuples in S1 whose frame time covers I, and let $\{t'_1, \ldots, t'_h\}$ the set of all tuples in S2 whose frame time covers I. Then, the tuple

$$t' = \# t | < I, (t_1(PER) \cup^{C} \ldots \cup^{C} t_k(PER)) -^{C}$$
$$(t'_1(PER) \cup^{C} \ldots \cup^{C} t'_h(PER)) > \#$$

belongs to R, if its periodicity is not empty. Formally,

$\mathrm{Sch}(R1 -^{P} R2) = \mathrm{Sch}(R1)(\mathrm{Sch}(R1) = \mathrm{Sch}(R2))$
$R1 -^{P} R2 = \{s | (\exists t1 \in R1, \mathrm{Card}(\mathrm{EqData}(t1, R1) \cup$
$\mathrm{EqData}(t1, R2)) = 1, s(R1) = t1(R1), s(FT) =$
$t1(FT), s(PER) = t1(PER))$ OR
$(\exists t, I, t'_1, \ldots, t'_k, t''_1, \ldots, t''_h t \in R1 \mathrm{Card}(\mathrm{EqData}(t, R1) \cup$
$\mathrm{EqData}(t, R2)) > 1, s(R1) = t(R1), s(FT) = I, s(PER) =$
$(t'_1(PER) \cup^{C} \ldots \cup^{C} t'_k(PER)) -^{C} (t''_1(PER) \cup^{C} \ldots \cup^{C} t''_j(PER))$
where $I \in \mathrm{Fragments}(FT's(\mathrm{EqData}(t, R1) \cup \mathrm{EqData}(t, R2)))$
and $t'_1, \ldots, t'_k$ $(k \geq 1)$ are all and only the tuples
$t' \in \mathrm{EqData}(t, R1)$ such that $\mathrm{NSDur}(I, t'(FT))$ holds and
$t''_1, \ldots, t''_j$ $(j \geq 0)$ are all and only the tuples
$t'' \in \mathrm{EqData}(t, R2)$ such that $\mathrm{NSDur}(I, t''(FT))$ holds, and
$s(PER) \neq \oslash$[5] }.

For example, given the tables R1 and R2 in Fig. 3, $R = R1 -^{P} R2$ is shown in Fig. 5.

5. Obviously, in case $|\mathrm{EqData}(t,R2)| = 0$, the difference is simply $t'_1(PER) \cup^{C} \ldots \cup^{C} t'_k(PER)$.

R

Data	T
	<[10,12), Mon $\cup^{C}$ Tue $\cup^{C}$ Wed >
	<[12,13), Mon $\cup^{C}$ Tue >
	<[13,13], Tue>
	<(13,15], Tue $\cup^{C}$ Wed >
	<[19,20], Mon >

Fig. 5. $R = R1 -^{P} R2$, where R1 and R2 are the periodic tables in Fig. 3. Only the validity time T is shown.

4.5 Temporal Operators and Functions

We now introduce new operators and functions that are unique to the temporal algebra, focusing only on the *Extension* function and on the temporal operators selecting tuples on the basis of *temporal conditions* involving only the validity time of tuples independently of each others. In our model, the validity time is composed by a frame time and by a periodicity. Thus, temporal selection can involve a frame time (SelectPt, SelectIn), a periodicity (SelectPer), or both components (SelectPtPer, SelectInPer). Moreover, as regards frame times, the selection condition may compare it with a time point (SelectPt, SelectPtPer) or with a time interval (SelectIn, SelectInPer).

- **SelectPt** (SelectPoint). SelectPt applies to a periodic table R, a relation Rel (Before or After) and a time point P and selects all tuples in R occurring in a time in relation Rel with the time point P. The frame time t(FT) of each tuple in R is restricted to the subpart of it which satisfies Rel, and the resulting table contains all those tuples whose frame time and periodicity are not empty. For example, the case where Rel = Before is shown below, where Cut_Before applies on an interval $< i^{-}, i^{+} >$ and a point p and gives as a result the interval $< \mathrm{i}^{-}, \mathrm{i}^{+} >$ if $\mathrm{i}^{+} < \mathrm{p}$; the interval $< \mathrm{i}^{-}, \mathrm{p})$ if $\mathrm{i}^{-} < \mathrm{p} \leq \mathrm{i}^{+}$, and $\oslash$ otherwise.

  $\mathrm{Sch}(\mathrm{SelectPt}_{\mathrm{Before,P}}(\mathrm{R})) = \mathrm{Sch}(\mathrm{R})$
  $\mathrm{SelectPt}_{\mathrm{Before,P}}(\mathrm{R})) = \{\mathrm{s} \mid \exists \mathrm{t1} \in \mathrm{R}, \mathrm{s}(\mathrm{R}) = \mathrm{t1}(\mathrm{R}),$
  $\mathrm{s}(\mathrm{PER}) = \mathrm{t1}(\mathrm{PER}),\ \mathrm{s}(\mathrm{FT}) = \mathrm{Cut\_Before}(\mathrm{t1}(\mathrm{FT}), \mathrm{P}),$
  $\mathrm{s}(\mathrm{FT}) \neq \oslash\ \}.$

  For example, given the table OFFICE in Fig. 2, $\mathrm{SelectPt}_{\mathrm{Before,1/9/98}}(\mathrm{OFFICE})$ selects tuples whose frame time is (at least partially) before 1/9/98, giving as result

  $$\mathrm{Res} = \{\#\mathrm{P11}| < [1/1/98, 1/9/98),$$
  $$\mathrm{Mon} : \text{11-13} \cup^{\mathrm{C}} \mathrm{Tue} : \text{9-11} > \#,$$
  $$\#\mathrm{A02}| < [1/1/98, 1/9/98), \mathrm{Mon} : \text{14-16} > \#\}.$$

- **SelectIn** (Select Interval). SelectIn is similar to SelectPt above, selecting (i.e., "cutting," via the operator Cut_Intersects introduced in Section 3.2.1) the part of the frame time nonstrict during the time interval I (see, e.g., query Q3 in Section 7).
- **SelectPer** (Select Periodicity). SelectPer applies to a periodic table R and to a user-defined periodicity C (whose definition is stored in a dedicated table), and provides as output all the tuples t of R occurring at the periodicity specified by C (i.e., tuples occurring in $\mathrm{t(PER)} \cap^{\mathrm{C}} \mathrm{C}$, such that $(\mathrm{t(PER)} \cap^{\mathrm{C}} \mathrm{C}) \neq \oslash$).

  $\mathrm{Sch}(\mathrm{SelectPer}_{\mathrm{C}}(\mathrm{R})) = \mathrm{Sch}(\mathrm{R})$
  $\mathrm{SelectPer}_{\mathrm{C}}(\mathrm{R})) = \{\mathrm{s}|\exists \mathrm{t1} \in \mathrm{R}, \mathrm{s}(\mathrm{R}) = \mathrm{t1}(\mathrm{R}),$
  $\mathrm{s}(\mathrm{FT}) = \mathrm{t1}(\mathrm{FT}), \mathrm{s}(\mathrm{PER}) = \mathrm{p}, \mathrm{p} = \mathrm{t1}(\mathrm{PER}) \cap^{\mathrm{C}} \mathrm{C}, \mathrm{p} \neq \oslash\}.$

  For example, $\mathrm{SelectPer}_{\mathrm{Mondays}}(\mathrm{OFFICE})$ selects tuples holding on Mondays, giving as result

  $$\mathrm{Res} = \{\#\mathrm{P11}| < [1/1/98, 31/12/98],$$
  $$\mathrm{Mon} : \text{11-13} > \#, \#\mathrm{A02}| < [1/1/98, 31/1/99],$$
  $$\mathrm{Mon} : \text{14-16} > \#\}.$$

- **SelectPtPer** (Select Point and Period), **SelectInPer** (Select Interval and Periodicity) are simply a combination of the types of selections above (see, e.g., queries Q1 and Q4 in Section 7).

  Further temporal operators could be introduced in order to select tuples on the basis of *temporal conditions* which *compare* the validity times of different tuples (similar, e.g., to left/right temporal join in [13]). For example, considering the tables in Fig. 2, these operators might be used in order to look for teachers who had their office hours *after* one of their lessons, on Mondays. Introducing such operators involves a careful analysis of the semantics of relational temporal predicates (e.g., "after") in the context of periodic validity times and is one of the goals of our future work.
- **Extension.** The Extension function operates on a time interval I and a periodic table and extends the schema of the table with a new attribute, Ext_Attr. For each tuple t in R, t(Ext_Attr) will contain the set of all time intervals (expressed as pairs of dates in the Gregorian calendar) which constitutes the extension of the period t(PER) in the frame time $\mathrm{t(FT)} \cap^{\mathrm{I}} \mathrm{I}$. As an example, see e.g., query Q2 in Section 7.

  $\mathrm{Sch}(\mathrm{F}_{\mathrm{Extension}}(\mathrm{R})) = \mathrm{Sch}(\mathrm{R}) \cup \{\mathrm{Ext\_Attr}\}$
  $\mathrm{F}_{\mathrm{Extension}}(\mathrm{R}) = \{\mathrm{s}|\exists \mathrm{t1} \in \mathrm{R}, \mathrm{s}(\mathrm{R}) = \mathrm{t1}(\mathrm{R}),$
  $\mathrm{s}(\mathrm{FT}) = \mathrm{t1}(\mathrm{FT}), \mathrm{s}(\mathrm{PER}) = \mathrm{t1}(\mathrm{PER}),$
  $\mathrm{s}(\mathrm{Ext\_Attr}) = \mathrm{BExt}(\mathrm{t1}(\mathrm{FT}) \cap^{\mathrm{I}} \mathrm{I}, \mathrm{t1}(\mathrm{PER}))\}.$

4.6 Properties of the Temporal Algebra

Different criteria can be used in order to evaluate temporal algebrae. In particular, the properties of *equivalence* and *reduction* are very relevant since they can be used to check whether a model is a well-formed temporal extension of the atemporal (standard) relational algebra (see [32]). In fact, the equivalence property grants that, with an appropriate definition of a transformation function that maps each snapshot (atemporal) table R into a corresponding periodic table R^{P}, the results that one obtain by first transforming snapshot tables into periodic ones, then applying the temporal operators of our algebra (e.g., $\times^{\mathrm{P}}$) are the same that would be obtained by first applying the corresponding standard operators of the atemporal algebra (e.g., $\times$), and then transforming the result. On the other hand, the reduction property grants that, with an appropriate definition of a transformation function that maps each periodic

table R^P into the corresponding snapshot (atemporal) table R, the results that one obtain by first applying the temporal operators of our algebra (e.g., $\times^P$) to periodic tables and then transforming the results into snapshot tables are the same that would be obtained by first transforming the periodic tables into snapshot ones and then applying the corresponding standard operators of the atemporal algebra (e.g., $\times$). A sketch of the proofs can be found in [41].

Property 7. *The equivalence property holds for our "periodic" temporal algebra.*

Property 8. *The reduction property holds for our "periodic" temporal algebra.*

5 Fully Symbolic Evaluation of Set-Operators on User-Defined Periodicities

In this section, we introduce a symbolic evaluation method for our extended algebra and study its properties.

5.1 Intensional (Symbolic) versus Extensional Evaluation

The temporal algebra in Section 4 operates in a standard way on the atemporal attributes (data part of tuples). Moreover, our temporal algebra is independent of how the operations on validity times are implemented. The frame time component in the validity time of derived tables can be easily computed by performing set-operations on the time intervals representing the frame times, and data attributes can be implemented in a standard way. On the other hand, the treatment periodicities in validity times can be either extensional or intensional (symbolic). In particular, the execution of the algebraic operators in Sections 4.4 and 4.5 on periodic tables involve the application of the set operators $\cap^C$, $-^C$, and $\cup^C$ on the periodicities in the validity times. In the *extensional treatment*, set-operators are performed on the extensions of periodicities; for example, given two periodicities C1 and C2 and a frame time I, $C1 \cap^C C2$ is implemented as $\mathrm{BExt}(I, C1) \cap^T \mathrm{BExt}(I, C2)$, i.e., as an operation on Gadia's temporal elements [20]. In the *intensional treatment*, each periodicity is represented by the symbolic definition given by the user and the algebraic temporal operators operate in a symbolic way on the periodicity in the validity times of tuples, in the sense that only symbolic string manipulation of expressions in our language in Section 3 is performed (extensions are computed only if they are explicitly required by the user—via the Extension function). An intensional treatment might be very advantageous from the point of view of storage allocation and from the computational point of view. Moreover, the extensional approach is *not* feasible in case of infinite frame times (infinite extensions). However, the main reason for which we believe that intensional evaluation is important is that it provides more perspicuous and high-level outputs to the users (see Section 8.2). For example, the extensional evaluation of the union of the periodicity "Mondays" and "Wednesdays" (e.g., in a one-year-long frame time) results in a long list of intervals (pair of dates), while the intensional (symbolic) one could provide as output the compact and "significant" symbolic expression "Mondays $\cup^C$ Wednesdays." In principle, one could devise a "hybrid" approach in which one 1) takes in input the symbolic expressions, 2) convert them into extensions, 3) performs operations on extensions, and 4) retransform the extensional results into "significant" output symbolic expressions. However, such an approach seems to us computationally expensive and quite problematic in practice. In fact, in general, symbolic expressions automatically generated from a set of extensions might be far from being "significant" for users (unless very skilful and complex *learning* techniques are devised and used to reach such a goal). Thus, we believe that, if one wants to give in output "perspicuous" symbolic results, the evaluation should be performed directly at the symbolic level. However, in order to provide "perspicuous" results, the symbolic evaluation cannot just give in output a string concatenation of input periodicities and operators. At least two types of simplifications can be (and need to be) performed:

1. **redundancy elimination**; e.g., the output of the union of "Working-Days" (i.e., days from Mondays to Fridays) and "Mondays" should be just "Working-Days" and not "Working-Days $\cup^C$ Mondays" and
2. **empty periodicity detection**; for instance, the output of the intersection of "Mondays" and "Wednesdays" should be the empty periodicity (henceforth represented by "Ø") and not "Mondays $\cap^C$ Wednesdays."

In this section, we present a *symbolic evaluation method* that performs these simplifications, and study its properties. It is important to remark that our method aims at performing simplifications *without generating the extensions* of the input periodicities. For example, in our symbolic evaluation method, the intersection of "1^{st}Days_of_Months" and "Mondays" is "1^{st}Days_of_Months $\cap^C$ Mondays" since some intersections between the two extensions are possible. Now, in certain frame times (such as, e.g., [1/4/1999, 30/7/1999]), the extension of "1^{st}Days_of_Months $\cap^C$ Mondays" is empty, but this can only be detected via a generation of the extensions. All simplifications that depend on the chosen *frame time* can be obtained by users by explicitly invoking the temporal function Extension (see Section 4.5). In Section 6, we will sketch a semisymbolic evaluation method that considers also the simplifications due to the frame time still providing a symbolic output and that exploits the fully symbolic evaluation method discussed in this section whenever possible.

5.2 Relations between User-Defined Periodicities

Since periodicity is user-defined, it is not possible to define all intersections, unions, etc. between all pairs of periodicities a priori. We thus pointed out a set of six possible relations between two user-defined periodicities which are exhaustive and mutually exclusive, and defined the symbolic evaluation of $\cup^C$, $\cap^C$, $-^C$, and $\neg^C$ on these basis. The relations are:

C1 $=^C$ C2 (C1 temporally equal to C2) holds if the extensions of C1 and C2 are the same (or, in other words, if they denote the same TE). For example, Mondays $=^C$ Mondays, and, considering the European week and the obvious definitions of Mondays and of First-Day-of-Weeks, we have Mondays $=^C$ First-Day-of-Weeks. More formally,

$$C1 =^C C2 \Leftrightarrow \mathrm{Ext}(C1) = \mathrm{Ext}(C2).$$

C1 $\supset^C$ C2 (C1 temporally contains C2) holds if, for each time interval in the extension of C2, there is a time interval

TABLE 1
Symbolic Evaluation of the Operators $\cup^C$, $\cap^C$, and $-^C$ on Base Periodicities

Periodicity Relation	C1 $\cap^C$ C2	C1 $\cup^C$ C2	C1 $-^C$ C2	C2 $-^C$ C1
C1 = C2	C1	C1	$\varnothing$	$\varnothing$
C1 $\subset^C$ C2	C1	C2	$\varnothing$	C2 $-^C$ C1
C1 $\supset^C$ C2	C2	C1	C1 $-^C$ C2	$\varnothing$
C1 $\neq_C$ C2	$\varnothing$	C1 $\cup^C$ C2	C1	C2
C1 $\neq_N$ C2	$\varnothing$	C1 $\cup^C$ C2	C1	C2
C1 Λ^C C2	C1 $\cap^C$ C2	C1 $\cup^C$ C2	C1 $-^C$ C2	C2 $-^C$ C1

in the extension of C1 containing (properly or not) it and the relation $=^C$ does not hold. For example, with usual definitions, Days $\supset^C$ Mondays, Weeks $\supset^C$ Days. More formally,

$$C1 \supset^C C2 \Leftrightarrow (\forall I \in Ext(C2) \Rightarrow \exists J \in Ext(C1) \wedge NSDur(I, J)) \wedge \neg(C1 =^C C2).$$

C1 $\subset^C$ C2 (C1 temporally contained into C2). The inverse of **C1 $\supset^C$ C2**.

C1 $\neq_C{}^C$ C2 (C1 covering temporally disjoint from C2) holds in case all the time intervals in the extensions of C1 and C2 are temporally disjoint (the predicate **Disjoint(i,j)** stands for Before(i,j), Meets(i,j), Met-by(i,j), or After(i,j) in Allen's Algebra [3]). Moreover, the set of all the extension covers the whole timeline. For example, the set of all days from Mondays to Fridays and the set of Saturdays and Sundays are covering temporally disjoint. More formally,

$$C1 \neq_C{}^C C2 \Leftrightarrow (\forall I \in Ext(C1), \forall J \in Ext(C2) \Rightarrow Disjoint(I, J)) \wedge (\forall P \text{ time-point}(P), \exists I(I \in Ext(C1) \vee I \in Ext(C2)) \wedge P \in I).$$

C1 $\neq_N{}^C$ C2 (C1 noncovering temporally disjoint from C2) holds in case all time intervals in the extensions of C1 and C2 are disjoint. Moreover, the set of all the extension does not cover the whole timeline. For example, with usual definitions, Mondays $\neq^C$ Tuesdays and Mondays $\neq^C$ Wednesdays. More formally,

$$C1 \neq_N{}^C C2 \Leftrightarrow (\forall I \in Ext(C1), \forall J \in Ext(C2) \Rightarrow Disjoint(I, J)) \wedge (\exists P \text{ time-point}(P), \forall I(I \in Ext(C1) \vee I \in Ext(C2)) \wedge P \notin I).$$

C1 Λ^C C2 (C1 temporally intersects C2) holds in case none of the above relations hold. We call this relation temporal intersection since, if none of the relations $=^C$, $\subset^C$, $\supset^C$, $\neq_N{}^C$, $\neq_C{}^C$ hold between C1 and C2, we have that there is an intersection between the extensions of C1 and of C2 (otherwise, $\neq_N{}^C$ or $\neq_C{}^C$ would hold), but there are some intervals or subintervals of intervals in Ext(C1) which are not contained into an interval in Ext(C2) (otherwise, $\subset^C$ or $=^C$ would hold), and vice-versa. For example, with usual definitions, Mondays Λ^C 1st Day-of-Months. More formally,

$$C1 \Lambda^C C2 \Leftrightarrow \neg(C1 =^C C2 \vee C1 \subset^C C2 \vee C1 \supset^C C2 \vee C1 \neq_N{}^C C2 \vee C1 \neq_C{}^C C2).$$

5.3 Symbolic Evaluation of $\cup^C$, $\cap^C$, and $-^C$ on Base User-Defined Periodicities

In the following, we divide the periodicities which can be defined using the constructs in our high-level symbolic language in Section 3 into two subclasses: *base periodicities* and *composed periodicities*.

1. With the term *base periodicity*, we will intend the *basic periodicity* and those periodicities obtained by the application of the constructs *Convex_Cal* and *Select_Periods* only.
2. With the term *composite periodicity*, we intend periodicities defined using at least one of the set-operators (i.e., $\cup^C$, $\cap^C$, $-^C$, and $\neg^C$). A composite periodicity can be directly defined by the users, or can also derive from the application of algebraic operators (e.g., $\times^P$ performs intersection $\cap^C$ between periodicities).

This distinction is important from the *operational* point of view. In fact, while in most relevant cases the relation between two base periodicities C1 and C2 (i.e., Λ^C, $=^C$, $\subset^C$, $\supset^C$, $\neq_N{}^C$, or $\neq_C{}^C$) can be obtained by rules operating on the definition of C1 and C2 only (see Section 5.5), the same is not true in case C1 and C2 are composite periodicities. In this section, we define a way of symbolically computing $\cup^C$, $\cap^C$, and $-^C$ between two base periodicities C1 and C2 (assuming $C1 \neq \varnothing$ and $C2 \neq \varnothing$), on the basis of the relation between C1 and C2.

Table 1 represents a set of *conditional transformation rules* of the form:

$$\forall \mathbf{C1, C2\ IF\ C1\ R\ C2\ THEN\ Expr1 \rightarrow Expr2},$$

where C1 and C2 are two user-defined base periodicities, R is one of the six relations between periodicities in Section 5.2, Expr1 is the symbolic expression "$C1 \cap^C C2$", "$C1 \cup^C C2$," or "$C1 -^C C2$," and Expr2 the output symbolic expression. For example, the entry <row: $C1 \subset^C C2$; column: $C1 \cap^C C2$> in Table 1 represents the rule:

$$\forall \mathbf{C1, C2\ IF\ C1 \subset^C C2\ THEN\ C1 \cap^C C2 \rightarrow C1}.$$

In other words, if **C1 $\subset^C$ C2** holds, then the result of the symbolic evaluation of **C1 $\cap^C$ C2** is C1. Thus, for example, with the usual definitions of Mondays and Weeks, we would have Mondays $\subset^C$ Weeks, and thus the symbolic evaluation of "Mondays" $\cap^C$ "Weeks" gives as result "Mondays."

TABLE 2
Relation between $\neg^C$ C1 and C2, Given the Relation between Two Base Periodicities C1 and C2

Relation between C1 and C2	Relation between $\neg^C$C1 and C2
$C1 =^C C2$	$(\neg^C C1) \neq_C{}^C C2$
$C1 \subset^C C2$	$(\neg^C C1)\ \Lambda^C C2$
$C1 \supset^C C2$	$(\neg^C C1)\ \neq_N{}^C C2$
$C1 \neq_C{}^C C2$	$(\neg^C C1)\ =^C C2$
$C1 \neq_N{}^C C2$	$(\neg^C C1) \supset^C C2$
$C1\ \Lambda^C C2$	$(\neg^C C1)\ \Lambda^C C2$

Complexity. The relation (Λ^C, $=^C$, $\subset^C$, $\supset^C$, $\neq_N{}^C$, $\neq_C{}^C$) between any two user-defined periodicities can be computed as soon as the periodicities are inserted (and, thus, offline with respect to the query process), and can be stored into a dedicated table (henceforth, called Relation Table). Thus, in our approach, intersection, union, and differences of base periodicities can be simply performed via a lookup in the Relation Table, plus a lookup in Table 1. In the following analysis, we assume that look-up operations require constant time.

The *correctness* of the rules for symbolic evaluation can be defined in terms of the (extensional) semantics of the operators $\cup^C$, $\cap^C$, $-^C$ (on the basis of the properties of the corresponding operators $\cup^T$, $\cap^T$, and $-^T$ on TE's) by showing that, for each rule

$$\forall \mathbf{C1}, \mathbf{C2}\ \mathbf{IF}\ \mathbf{C1}\ \mathbf{R}\ \mathbf{C2}\ \mathbf{THEN}\ \mathbf{Expr1} \rightarrow \mathbf{Expr2}$$

in Table 1, Ext(Expr1) = Ext(Expr2) (see the proof in [41]).

Property 9. *The rules of symbolic evaluation in Table 1 are correct.*

Also, the *completeness* of our symbolic evaluation can be defined in terms of the (extensional) semantics of periodicities and operators on periodicity. In particular, we should have to prove that, for any two user-defined periodicities C1 and C2, if Ext(C1) = Ext(C2), then there is a way of obtaining C2 from C1 by the applications of our symbolic transformation rules. Our symbolic evaluation method is not complete in this sense. Its goal is less ambitious: It only aims at simplifying symbolic expressions by the elimination of redundancies and the detection of empty periodicity. We have proven that our symbolic evaluation above is complete as regards these forms of simplifications (see [41]). For example, we have proven that, given two base periodicities C1 and C2 and an operation OP^C ($\cup^C$, $\cap^C$, or $-^C$) between them,

1. whenever $\mathrm{Ext}(C1\ OP^C\ C2) \subseteq \mathrm{Ext}(C2)$, there is a rule such that the symbolic expression "$C1\ OP^C\ C2$" is transformed into "C2" (or vice-versa, if $\mathrm{Ext}(C1\ OP^C\ C2) \subseteq \mathrm{Ext}(C1)$).
2. whenever $\mathrm{Ext}(C1\ OP^C\ C2) = \oslash$, there is a rule such that the symbolic expression "$C1\ OP^C\ C2$" is transformed into "Ø."

Property 10. *The rules of symbolic evaluation in Table 1 are complete as regards redundancy elimination and emptyness detection on intersection, union, and difference of base periodicities (but not considering frame times).*

5.4 Symbolic Evaluation of $\cup^C$, $\cap^C$, $-^C$, and $\neg^C$ on Composite User-Defined Periodicities

Complement ($\neg^C$). Table 2 reports the relation between $\neg^C$ C1 and C2, given the relation between two base periodicities C1 and C2. In Table 2, we assume that neither Ext(C1) nor Ext(C2) cover the whole timeline and that $C1 \neq \oslash$ and $C2 \neq \oslash$. These special cases are managed by easy "ad hoc" rules. Given Table 2 (and the relations between base periodicities), one can treat the complement of a base periodicity C1 as a *base periodicity* C1′ in the symbolic evaluation.

In the following, we suppose that composite periodicities are put in normal form. The transformation can be obtained using the standard properties (e.g., distributivity, DeMorgan's laws), which also holds for the operators $\cap^C$, $\cup^C$, $-^C$, and $\neg^C$ on periodicities (see Property 2). Thus, in the following, we consider composite periodicities of the form:

$$(X1 \cap^C \ldots \cap^C Xn) \cup^C (Y1 \cap^C \ldots \cap^C Yk) \cup^C \ldots \cup^C (Z1 \cap^C \ldots \cap^C Zh),$$

where each Xi, Yi, and Zi are base periodicities or complements of base periodicities (called *literal periodicities*—or *literals*, for short—henceforth). Given the transformations in normal form, set-operators can be computed by the operations 1) and 2) below:

1. **Symbolic Intersection** $(\mathbf{X1} \cap^C \ldots \cap^C \mathbf{Xn}) \cap^C (\mathbf{Y1} \cap^C \ldots \cap^C \mathbf{Ym})$. Given two sets S1 and S2 of literal periodicities (representing two intersections $X1 \cap^C \ldots \cap^C Xn$ and $Y1 \cap^C \ldots \cap^C Ym$ to be intersected), the goal of symbolic intersection is to determine the resulting set of literal periodicities (which constitute the resulting intersection). The algorithm considers the list L containing the literal periodicities in S1 and those in S2 (function Append) and remove from it all literal periodicities which are "redundant" since they contain (properly or not) other literal periodicities in the intersection. In case two, disjoint (relation $\neq_N{}^C$ or $\neq_C{}^C$) literal periodicities are found "Ø" is given as output.[6] The function

6. The algorithm assumes that the two input intersections ($X1 \cap^C \ldots \cap^C Xn$) and ($Y1 \cap^C \ldots \cap^C Ym$) have already been simplified independently of each others and performs the simplification on the new intersection. This assumption can be easily removed by modifying the For Each statement in Symbolic_Intersection into "**For Each** pair $< Xj, Yi >$ such that $Xj \in S1 \cup S2$, $Yi \in S1 \cup S2$ **do .....** "

Delete(x,L) deletes an occurrence of x from L. The function build_sym_expr(OP, {S1, .., Sn}), given an operator ($\cap^C$ or $\cup^C$) and a set {S1, .., Sn} of user-defined symbolic periodicities, gives as a result the symbolic expression "S1 OP .... OP Sn" and operates in linear time.

Symbolic_Intersection({X1, ..., Xn}, {Y1, ..., Ym})
Let $S1 = \{X1, \ldots, Xn\}$, $S2 = \{Y1, \ldots, Ym\}$
INTERS ← append (S1, S2) /* list of all literals in S1 and S2, possibly with repetitions */
For Each pair < Xj, Yi > such that $Xj \in S1$, $Yi \in S2$ $(1 \leq j \leq n, 1 \leq i \leq m)$ **do**
If $((Yi \neq_N {}^C Xj) \vee (Yi \neq_C {}^C Xj))$
Then Return ("Ø")
Else If $((Yi =^C Xj) \vee (Yi \supset^C Xj))$
Then Delete(Yi,INTERS)
Else If $(Xj \supset^C Yi)$ **Then** Delete(Xj,INTERS) **od**
If INTERS $\neq$ null **Then Return**
(build_sym_expr($\cap^C$, INTERS)) Else Return("Ø")

For example, the intersection of

$$S1 = \{1^{st}\text{Mondays\_of\_January}\} \text{ and}$$
$$S2 = \{\text{January}, \text{Mondays}\}$$

is $\{1^{st}\text{Mondays\_of\_January}\}$.

Complexity. Let n and m be the number of literals in the two input terms. For each pair of literals, Symbolic_Intersection looks up, in the Relation Table, the relation between them and uses it in order to evaluate the conditions in the "if" statements. Since the pairs are n*m, at most O(n*m) operations (table lookups, condition evaluations, and deletions) are performed.

2. **Symbolic Union** $(\mathbf{X1} \cap^C \ldots \cap^C \mathbf{Xn}) \cup^C (\mathbf{Y1} \cap^C \ldots \cap^C \mathbf{Ym})$. Given two sets of literal periodicities (representing two intersections $X1 \cap^C \ldots \cap^C Xn$ to which union has to be applied) symbolic union builds up their union eliminating "redundant" sets. Union(Si,Sj) removes the redundant set Sj if Sj is such that All_Contain(Si,Sj) holds (or vice-versa). All_Contain(S1,S2) operates on two sets of literals and checks whether, for each $Xj \in S1$, there is a $Yi \in S2$ contained by Xj—properly or not —(i.e., such that $(Xj =^C Yi) \vee (Xj \supset^C Yi)$).

Symbolic_Union({X1, ..., Xn}, {Y1, ..., Ym})
Let $S1 = \{X1, \ldots, Xn\}$, $S2 = \{Y1, \ldots, Ym\}$
If All_Contain(S1,S2)
Then Return (build_sym_expr($\cup^C$,S1))
Else If All_Contain(S2,S1)
Then Return (build_sym_expr($\cup^C$, S2))
Else Return (build_sym_expr($\cup^C$,
{build_sym_expr($\cap^C$, S1),
build_sym_expr($\cap^C$, S2)}));

For example, the union of

$$(1^{st}\text{Mondays\_of\_Months} \cap^C 1^{st}\text{4\_Days\_of\_Months} \cap^C \text{Working\_Days})$$

and

$$(1^{st}\text{5\_Days\_of\_Months} \cap^C \text{Mondays})$$

is $(1^{st}\text{5\_Days\_of\_Months} \cap^C \text{Mondays})$ since

$$1^{st}\text{5Days\_of\_Months} \supset^C 1^{st}\text{4Days\_of\_Months}$$

(e.g., 1^{st}5Days_of_Months can be defined as "[1-5] / Days NSDur Months") and

$$\text{Mondays} \supset^C 1^{st}\text{Mondays\_of\_Months}.$$

Complexity. Let n and m be the number of literals in the two input sets S1 and S2. All_contain considers all n*m pairs of literals and looks up the Relation Table to check the relation between them. Thus, the complexity is O(n*m).

The following properties hold for our symbolic evaluation on *composite* periodicities:

Property 11. *Symbolic evaluation on* composite *periodicities is correct.*

Property 12. *Symbolic evaluation is* not *complete regarding redundancy elimination and emptyness detection on intersection, union, and difference on* composite *periodicities.*

Corollary 1. *The correctness of our symbolic implementation of the* algebraic operators *in Section 4 trivially follows from the correctness of our symbolic evaluation method of set operators on periodicities. In fact, the algebraic operators in Section 4 operate in a standard way on the atemporal attributes, and perform symbolic union, intersection, and difference on the periodicities in the validity time attribute (together with trivial operations on the frame times).*

5.5 Complexity of the Symbolic Management of Validity Times in the Algebraic Operations

Until now, we considered the complexity of our symbolic operations of intersection, union, and difference on user-defined periodicities independently of the context in which they are used, i.e., independently of the algebraic operators in the queries. The algebraic operators of projection (π^P) and union ($\cup^P$) perform the union ($\cup^C$) of the periodicities in the validity time (henceforth, VT), Cartesian product ($\times^P$) perform intersection and difference ($-^P$) performs the difference ($-^C$) of the periodicities in the VT (while the other algebraic operators do not modify the VT). Since more than one of these VT-modifying operators can be applied in the same query, it is interesting to see how our symbolic approach behaves in these cases. For the sake of simplicity and clarity, we consider only union ($\cup^P$) and Cartesian product ($\times^P$) and we take into account tables containing only one tuple each (the generalization of the following analysis to tables containing more than one tuple is easy). We do not pay attention to the atemporal part of the tuples (and to the frame time) and assume that the periodicity in the VT of tuples in base tables has always the same form and dimension:

(per1) $(\mathbf{X_{11}} \cap^C \ldots \cap^C \mathbf{X_{1i}}) \cup^C (\mathbf{X_{21}} \cap^C \ldots \cap^C \mathbf{X_{2i}}) \cup^C \ldots \cup^C (\mathbf{X_{u1}} \cap^C \ldots \cap^C \mathbf{X_{ui}})$,

i.e., it consists of a union of *u intersection-terms*, each one consisting of an intersection of *i* literals. We only consider the *worst case* for our approach, i.e., the case when *no simplification* can ever be performed during the repeated application of symbolic operations on the periodicities in the VT of tuples.

Union ($\cup^{\mathrm{P}}$). Let us suppose that we perform the union between two tables T1 and T2 containing one tuple each with periodicity of the form (and dimension) of (per1) above and with the same data part *dp* (if the data part if different, no manipulation is made on the periodicities in the VT's); then let us suppose to perform the union of the resulting table with a new table T3, containing again a tuple with data part *dp* and periodicity like (per1), and so on. The first union ($\cup^{\mathrm{P}}$) applies Symbolic_Union to the two periodicities. Since each periodicity consists of the union ($\cup^{\mathrm{C}}$) of *u* intersection-terms, and Symbolic_Union operates on pairs of intersection-terms, we need u^2 applications of Symbolic_Union (one for each pair $< \mathrm{it_k}, \mathrm{it_h} >$—where $1 \leq \mathrm{k}, \mathrm{h} \leq \mathrm{u}$—of intersection-terms such that $\mathrm{it_k}$ belongs to the periodicity of the first tuple, and $\mathrm{it_h}$ belongs to the periodicity of the second tuple). Since each intersection-term is an intersection of i literals, the complexity of each application of Sybolic_Union is $\mathrm{O}(\mathrm{i}^2)$. Thus, under the above assumptions, the complexity of performing union ($\cup^{\mathrm{P}}$) on a pair of tuples is $\mathrm{O}(\mathrm{u}^2 * \mathrm{i}^2)$. Furthermore, in the worst case in which no simplification can be performed, the dimension of the periodicity of the resulting tuple becomes *2u*i*, i.e., we have *2u* intersection-terms consisting each one of *i* literals.

After $\mathrm{n}-1$ applications of union ($\cup^{\mathrm{P}}$) as above (and if no simplification can be performed), the periodicity of the tuple in the resulting table consists of $\mathrm{n}^*\mathrm{u}$ intersection-terms each one consisting of (the intersection $\cap^{\mathrm{C}}$ of) *i* literals. At the nth step, we make the union of such a resulting table with a table containing a unique tuple, with data part *dp* and periodicity like (per1). Thus, the time complexity of the nth application of union ($\cup^{\mathrm{P}}$) is $\mathrm{O}(\mathrm{n}^*\mathrm{u}^{2\,*}\mathrm{i}^2)$, and the periodicity in the resulting tuple consists of $(\mathrm{n}+1)^*\mathrm{u}$ intersection-terms of *i* literals each.

Cartesian Product ($\times^{\mathbf{P}}$). As above, let us suppose that we perform incrementally n operations of Cartesian Product, taking at each step the result of the previous operations and a new table. Each application of Cartesian Product applies Symbolic_Intersection to the input periodicities.[7] In particular, in the first application of Cartesian Product, one has to apply Symbolic_Intersection to each pair $< \mathrm{it_k}, \mathrm{it_h} >$ (where $1 \leq \mathrm{k}, \mathrm{h} \leq \mathrm{u}$) of intersection-terms such that $\mathrm{it_k}$ belongs to the periodicity of the first tuple and $\mathrm{it_h}$ belongs to the periodicity of the second tuple. Since each intersection-term consists of *i* literals, the complexity of the first application of Cartesian Product is $\mathrm{O}(\mathrm{u}^{2\,*}\mathrm{i}^2)$. However, in the worst case in which no simplification can be performed, the number of intersection-terms and their dimension increases: the periodicity in the resulting tuple consists of u^2 intersection-terms of $2^*\mathrm{i}$ literals each.

Thus, after $\mathrm{n}-1$ applications of Cartesian Product ($\times^{\mathrm{P}}$) as above (and if no simplification can be performed), the periodicity of the tuple in the resulting table consists of u^n intersection-terms each one consisting of (the intersection $\cap^{\mathrm{C}}$ of) n^*i literals. Thus, the time complexity of the nth application of $\times^{\mathrm{P}}$ is $\mathrm{O}(\mathrm{n}^*\mathrm{u}^{\mathrm{n}+1\,*}\mathrm{i}^2)$, and the periodicity in the resulting tuple consists of u^{n+1} intersection-terms each one consisting of $(n+1)^*i$ literals each. Thus, in the worst case in which no simplification can be performed, the dimension of the periodicity in the validity times of tuples grows exponentially with the number of Cartesian Products in the query and the time complexity grows up accordingly.

This growth is, in our opinion, the necessary price each symbolic approach has to pay in order to maintain a symbolic representation of periodicities in the VT when applying several VT-modifying query operators in the same query. However, it seems to us a quite reasonable price, since we believe that, in many practical applications, the number of such operators is quite limited, and many simplifications can be performed. In particular, simplifications are very frequent in case many "standard" Gregorian periodicities are used, since they tend to be either disjoint (relations $\neq_{\mathrm{N}}{}^{\mathrm{C}}$ and $\neq_{\mathrm{C}}{}^{\mathrm{C}}$, e.g., Mondays vs. Tuesdays) or contained (relations $\subset^{\mathrm{C}}$ and $\supset^{\mathrm{C}}$, e.g., Mondays vs. days, days vs. weeks), and disjointness and containment relations are at the core of all the simplification algorithms we developed (see Sections 5 and 6). Furthermore, it is a low price when compared to the complexity of the approach in [34] (which is, to the best of our knowledge, the only other approach that supports user-defined symbolic periodicities in the data of DB's; see Section 8.2).

5.6 Computing the Temporal Relation between Base User-Defined Periodicities

We defined a set of rules for determining automatically which one of the six relations ($=^{\mathrm{C}}$, $\subset^{\mathrm{C}}$, $\supset^{\mathrm{C}}$, $\neq_{\mathrm{N}}{}^{\mathrm{C}}$, $\neq_{\mathrm{C}}{}^{\mathrm{C}}$, or Λ^{C}) holds between two user-defined periodicities (expressed in the formalism in Section 3) on the basis of their definition only, without having to generate their extensions. We divided such rules into three different groups:

- Rules determining the relation between two periodicities occurring in the same definition. For instance, rule (IT) states that a definition of the form **C1 = n / C2 NSDur C3** implies $\mathrm{C1} \subset^{\mathrm{C}} \mathrm{C3}$ and $\mathrm{C1} \subset^{\mathrm{C}} \mathrm{C2}$ (e.g., from the definition Aprils = 4 / Months NSDur Years, we have that Aprils $\subset^{\mathrm{C}}$ Years and Aprils $\subset^{\mathrm{C}}$ Months).
- Rules determining the relation between two periodicities occurring in two different definitions. For instance, rule (II2) states that, given two definitions of the form $\mathbf{C} = [\mathbf{ni\text{-}nj}]/\mathbf{C_1}\ \mathbf{NSDur}\ \mathbf{C_2}$ and $\mathbf{C'} = [\mathbf{nh\text{-}nk}]/\mathbf{C_1}\ \mathbf{NSDur}\ \mathbf{C_2}$, then, if $[\mathrm{n_i} - \mathrm{n_j}] = [\mathrm{n_h} - \mathrm{n_k}]$, we have $\mathrm{C} =^{\mathrm{C}} \mathrm{C'}$; if $[\mathrm{n_i} - \mathrm{n_j}] \subset [\mathrm{n_h} - \mathrm{n_k}]$, then $\mathrm{C} \subset^{\mathrm{C}} \mathrm{C'}$ (for instance, from Mondays = 1/ Days NSDur Weeks and Working-Days = [1-5]/ Days NSDur Weeks we have Mondays $\subset^{\mathrm{C}}$ Working-Days); if $[\mathrm{n_i} - \mathrm{n_j}] \cap [\mathrm{n_h} - \mathrm{n_k}] = \oslash$, then $\mathrm{C} \neq_{\mathrm{N}}{}^{\mathrm{C}} \mathrm{C'}$ in case the two selections $[\mathrm{n_i} - \mathrm{n_j}]$ and $[\mathrm{n_h} - \mathrm{n_k}]$ do not cover the time line, $\mathrm{C} \neq_{\mathrm{C}}{}^{\mathrm{C}} \mathrm{C'}$, otherwise (e.g., from Mondays above and Wednesdays = 3/ Days NSDur Weeks, we have Mondays $\neq_{\mathrm{N}}{}^{\mathrm{C}}$ Wednesdays, while from Working-Days above and Week-Ends = [6-7] / Days NSDur Weeks, we have Working-Days $\mathrm{C} \neq_{\mathrm{C}}{}^{\mathrm{C}} \mathrm{C'}$ Week-Ends).
- Rules determining the relation between two periodicities on the basis of the relations. For example, from the transitivity of $\subset^{\mathrm{C}}$ and from Mondays $\subset^{\mathrm{C}}$ Working-Days and Working-Days $\subset^{\mathrm{C}}$ Weeks, we have Mondays $\subset^{\mathrm{C}}$ Weeks.

7. Moreover, after the repeated application of Symbolic_Intersection, one obtain a new periodicity $\mathrm{per'}$. Thus, one could also choose to repeatedly apply Symbolic_Union to each pair of intersection-terms in $\mathrm{per'}$, in order to look for further simplifications. We do not consider this possibility in this analysis for the sake of clarity and brevity.

Since the user is completely free in the use of the language in Section 3 for defining new periodicities, our rules do not cover all possible cases. For example, given the formalism in Section 3, the user has many different ways of defining temporally equal periodicities (e.g., s/he could define pairs of days—or triplets, or so on ...—and redefine the other periodicities—e.g., years—in terms of them) and our rules do not cover all of them. However, our rules proved to be powerful enough to cover "nonexceptional" cases. Whenever the relation between a pair of periodicities is needed and it is not obtained by the application of our rules, the relation can be asked to the user (or computed via extensional evaluation—see Section 6). The rules can be applied whenever a new user-defined periodicity is entered by a user, offline with respect to the query process. The relations between periodicities find out by the rules can then be stored into a dedicated table (the Relation Table), which can be initialized with the relations holding with the most commonly used periodicities.

5.7 Temporal Operators and Temporal Functions

The evaluation of the function Extension (Section 4.5) explicitly requires the generation of the extension of input periodicities, on the basis of their definitions. We adopt an attribute grammar and use attributes in order to synthesize the extension of a periodicity, on the basis of the components in its definitions. A brief description of the algorithms we use can be found in [41]. Analogous algorithms concerning Leban's formalism [27] have been described in [9], [14].

6 Semisymbolic Evaluation Method

We now extend the fully symbolic evaluation method in Section 5 to capture also the simplifications due to the frame time. Moreover, the semisymbolic approach can be used in order to determine those relations between periodicities which are not captured by the ad hoc rules (see Section 5.6). The fully symbolic evaluation method above is exploited whenever possible. The semisymbolic approach is based on the four functions below (where C, C1, and C2 stand for user-defined periodicities, SC, SC1, and SC2 are sets $\{C1, \ldots, Ck\}$ of periodicities to be intersected (via $\cap^C$), and FT is a frame time).[8]

8. Another useful function is **period(C1,C)** (defined also in [9]), which, given a periodicity C1 and the basic periodicity C, gives as output the number of intervals of Ext(C) after which the pattern of intervals in Ext(C1) is repeating (e.g., Period(Weeks,Days)=7; Period(Years,Days)=1,461). ContainedExt (ContainedExtInt) could generate the extensions of the two input periodicities (sets of periodicities) "in parallel," and stop the generation as soon as the first extension for which there is no containment is found. In the worst case (i.e., if containment holds), ContainedExt(C1,C2,FT) has to generate all the extensions in the shortest interval between the frame time FT and LCMP (an acronym for Least Common Multiple Period), where LCMP=lcm(period(C1,C),period(C2,C)), and C is the basic periodicity (e.g., if C1=Mondays, $C2 = 1^{st}$Day-of-Years, C=Days—and considering leap years—LCMP=1,461*7=10,227). In other words, since periodicities repeats regularly over a given period, no more than a whole repetition (covering one period) need to be generated. The case on ContainedExtInt is analogous. Similarly, the evaluation of EmptyExt and EmptyExtInt does not usually imply generating all the extensions in the frame time FT since these functions may return as soon as the first extension intersecting the frame time is generated. However, in the worst case, EmptyExt(C1,FT) has to generate all the extensions in the shortest interval between FT and period(C1,C).

EmptyExt(C,FT): returns false if there is at least an interval in the extension of C that intersects the frame time FT (i.e., if BExt(FT,C) is not empty), true otherwise.

ContainedExt(C1,C2,FT): returns

1. C2 if all the intervals in BExt(FT,C2) are contained into intervals in BExt(FT,C1) (i.e., if

$$(\forall i \;\; i \in \mathrm{BExt}(\mathrm{FT}, \mathrm{C2}) \Rightarrow (\exists j \;\; j \in \mathrm{BExt}(\mathrm{FT}, \mathrm{C1}) \wedge \mathrm{NSDur}(i, j))$$

   holds; or, in other words, if

$$\mathrm{BExt}(\mathrm{FT}, \mathrm{C1}) \supset \mathrm{BExt}(\mathrm{FT}, \mathrm{C2})),$$

2. C1 if all the intervals in BExt(FT,C1) are contained into intervals in BExt(FT,C2),
3. the pair $< \mathrm{C1}, \mathrm{C2} >$ if both containments above hold (i.e., $\mathrm{BExt}(\mathrm{FT}, \mathrm{C1}) = \mathrm{BExt}(\mathrm{FT}, \mathrm{C2})$), and
4. null otherwise.

EmptyExtInt(SC,FT): returns false if there is at least an interval in the extension of the intersections periodicities in SC which intersects the frame time FT (i.e., if $\mathrm{BExt}(\mathrm{FT}, \mathrm{C1} \cap^C \ldots \cap^C \mathrm{Ck})$ is not empty), true otherwise.

ContainedExtInt(SC1,SC2,FT): let $\mathrm{SC1} = \{\mathrm{C'1}, \ldots, \mathrm{C'k}\}$ and $\mathrm{SC2} = \{\mathrm{C''1}, \ldots, \mathrm{C''h}\}$; it returns SC2 if all the intervals in $\mathrm{BExt}(\mathrm{FT}, \mathrm{C''1} \cap^C \ldots \cap^C \mathrm{C''h})$ are contained into intervals in $\mathrm{BExt}(\mathrm{FT}, \mathrm{C'1} \cap^C \ldots \cap^C \mathrm{C'k})$, SC1 if the opposite containment holds, $< \mathrm{SC1}, \mathrm{SC2} >$ if both containments hold and null otherwise (i.e., if no containment holds).

On the basis of these functions, we can extend the evaluation of set operators on *base periodicities* proposed in Section 5. The *fully symbolic* method in Section 5 is correct and simplification-complete on *base* periodicities (see Properties 9 and 10), apart for the treatment of simplifications due to frame times. Thus, we only need to consider the extensions required in order to capture simplifications due to frame times. Thus, in all cases in Table 1 where the output of the symbolic evaluation was "Ø," we retain the fully symbolic evaluation, and no extensional evaluation is needed. Whenever the symbolic evaluation leads to a redundancy elimination in Table 1, the extensional evaluation just need to check whether the extension of the symbolic output is not empty in the given frame time. For example, if $\mathrm{C1} =^C \mathrm{C2}$, the rule in Table 1 is "if $\mathrm{C1} =^C \mathrm{C2}$, then $\mathrm{C1} \cap^C \mathrm{C2} \rightarrow \mathrm{C1''}$, and the output of the semisymbolic evaluation must be C1 if EmptyExt(C1,FT) is false, "Ø" otherwise. Whenever no simplification is obtained by the rules in Table 1 (e.g., "if $\mathrm{C1} \; \Lambda^C \; \mathrm{C2}$, then $\mathrm{C1} \cap^C \mathrm{C2} \rightarrow \mathrm{C1} \cap^C \mathrm{C2''}$), the extensional evaluation has to check simplifications due to frame times, as shown by the procedure Ext_Base_Intersection below. The algorithms for union and difference are analogous (see [41]; comments are between "/*" and "*/").

Ext_Base_Intersection(C1,C2,FT) /* applied only in case $\mathrm{C1} \; \Lambda^C \; \mathrm{C2}$ */
/* in case at least one of the extensions is empty in FT, then return "Ø" */

If EmptyExt(C1,FT) OR EmptyExt(C2,FT)
Then Return("Ø") **Else**
begin X ← ContainedExt(C1,C2,FT);
/* in case both containments hold, or in case the extensions of C1 are contained into those of C2, return "C1" */
If (X=<"C1","C2">) OR (X="C1")
Then Return("C1")
Else If (X="C2") **Then Return**("C2")
Else If Null(X) **Then**
/* if no containment holds, and if the intersection of the extensions in FT (i.e., $\mathrm{BExt}(\mathrm{FT}, \mathrm{C1} \cap^{\mathrm{C}} \mathrm{C2})$) is empty, return "Ø" */
If EmptyExtInt({C1,C2},FT) **Then Return**("Ø")
Else Return("$\mathrm{C1} \cap^{\mathrm{C}} \mathrm{C2}$")
end;

Extensional evaluation is used also to extend the symbolic intersection and union operations on *composite periodicities*. In both cases, the symbolic evaluation is performed first in order to obtain as many simplifications as possible. Then, the results of symbolic evaluation are further refined as follows. In case of intersection, the input to the extensional evaluation is a set $\mathrm{S} = \{\mathrm{C1}, .., \mathrm{Ck}\}$ of periodicities and a frame time FT. If any of C1,..,Ck has an empty extension in FT, then Ext_Comp_Intersection returns "Ø." Otherwise, ContainedExt(Ci,Cj,FT) is invoked on each pair $< \mathrm{Ci}, \mathrm{Cj} >$ of periodicities in S to delete redundancies. Finally, EmptyExtInt is invoked on the resulting set of periodicities, to check whether the extension of the resulting set is empty within the frame time FT. Analogously, the semisymbolic union also exploits the symbolic evaluation and the extensional functions above to perform all possible simplifications. A detailed description of these algorithms is outside the goals of this paper, and is given in [41].

To conclude, it is important to notice that our semisymbolic evaluation method: 1) exploits the results obtained by the fully-symbolic evaluation, 2) limits as much as possible the generation of extensions (see also footnote 8 above), and 3) *always provides symbolic outputs* (extensions are given in output only if they are explicitly requested by users via the use of the Extension function) using the generation of extensions just in order to perform simplifications in the symbolic expressions. Finally, and more important, we have proven Property 13 (see the proof in [41]):

Property 13. *Our Semisymbolic evaluation is correct and complete as regards redundancy elimination and emptyness detection.*

Corollary 2. *As in the case of the fully symbolic evaluation method, the* correctness *and* simplification completeness *of our symbolic implementation of the algebraic operators in Section 4 directly follows from Property 13.*

7 Examples

Let us suppose that we have a database containing the periodic tables in Fig. 2 and, in addition, 1) a set of dedicated tables to store the user definitions of periodicities and 2) the Relation Table and let us consider the following query:

When (intensional interpretation) **was room R01 busy between 1/12/98 and 31/1/99 on Mondays?**

which can be expressed in our extended temporal algebra as follows:

(Q1)
$\mathrm{SelectInPer}_{[1/12/98,31/1/99],\mathrm{Mondays}}((\pi^{\mathrm{P}} \oslash (\sigma^{\mathrm{P}}_{\mathrm{Room-Cod=R01}}(\mathrm{MEET})) \cup^{\mathrm{P}} (\pi^{\mathrm{P}} \oslash (\sigma^{\mathrm{P}}_{\mathrm{Room-Cod=R01}}(\mathrm{COURSES})))))$.

Notice that any query considering a user-defined periodicity C (e.g., Mondays in the example above) involves the fact that a definition of C is available in the dedicated tables.[9] Atemporal selections do not involve any special operation on the temporal attributes and, in this specific example, give as output the tables MEET and COURSES unchanged. Projection makes the union of the periodicities of tuples with equal data part (i.e., of all tuples in our example since we used projection on the empty set of data attributes; notice that $\pi^{\mathrm{P}} \oslash$ projects over the validity time attribute only). This involves, among other things, the symbolic evaluation of the periodicities in the validity times T of the resulting tuples. The results of atemporal selection and projection are the tables $\mathrm{R1} = \pi^{\mathrm{P}} \oslash (\sigma^{\mathrm{P}}_{\mathrm{Room-Cod=R01}}(\mathrm{COURSES}))$ and $\mathrm{R2} = \pi^{\mathrm{P}} \oslash (\sigma^{\mathrm{P}}_{\mathrm{Room-Cod=R01}}(\mathrm{MEET}))$ in Fig. 6. Then, $\mathrm{R1} \cup^{\mathrm{P}} \mathrm{R2}$ must be performed. The final result is shown by Table R3 in Fig. 6. Finally, $\mathrm{SelectInPer}_{[1/12/98-31/1/99],\mathrm{Mondays}}(\mathrm{R3})$ involves the evaluation of the intersection $\cap^{\mathrm{I}}$ of the frame times of tuples with [1/12/98 - 31/1/99] and the intersection of the periodicity with "Mondays," resulting in Table R4 in Fig. 6.

In case one is interested in the list of time intervals which satisfy the condition (i.e., in an extensional evaluation of the periodicity) in an interval of time (say, [1/12/98, 31/12/98]), one could apply the Extension operator, asking the query:

(Q2) $\mathrm{F}_{\mathrm{Extension}}([1/12/98, 31/12/98], \mathrm{SelectInPer}_{[1/12/98,31/1/99],\mathrm{Mondays}}((\pi^{\mathrm{P}} \oslash; (\sigma^{\mathrm{P}}_{\mathrm{Room-Cod=R01}}(\mathrm{MEET})) \cup^{\mathrm{P}} (\pi^{\mathrm{P}} \oslash (\sigma^{\mathrm{P}}_{\mathrm{Room-Cod=R01}}(\mathrm{COURSES}))))))$.

In such a case, the output would be a table like R4 with an additional Ext_Attr attribute containing the list of extensions, for each tuple. For example, the extensions of the first tuple in R4 above would be: {[7/12/98 at 9, 7/12/98 at 9], [7/12/98 at 12, 7/12/98 at 12], [7/12/98 at 14, 7/12/98 at 15], [14/12/98 at 9, 14/12/98 at 9], [14/12/98 at 12, 14/12/98 at 12], [21/12/98 at 9, 21/12/98 at 9], [21/12/98 at 12, 21/12/98 at 12], [28/12/98 at 9, 28/12/98 at 9], and [28/12/98 at 12, 28/12/98 at 12]}.

Notice that, in this and many other cases, a *symbolic output* might be more significant for users than a long list of extensional results (as one could obtain, e.g., with an extensional approach; see the discussion in the conclusions).

As further examples, consider the following queries (we use "#" to mark the starts and ends of each tuple).

9. In a practical implementation, the user might choose among a predefined menu of predefined periodicities, or use a user-friendly interface (based on the language in Section 3) to introduce in the dedicated table a periodicity not already present in such a table.

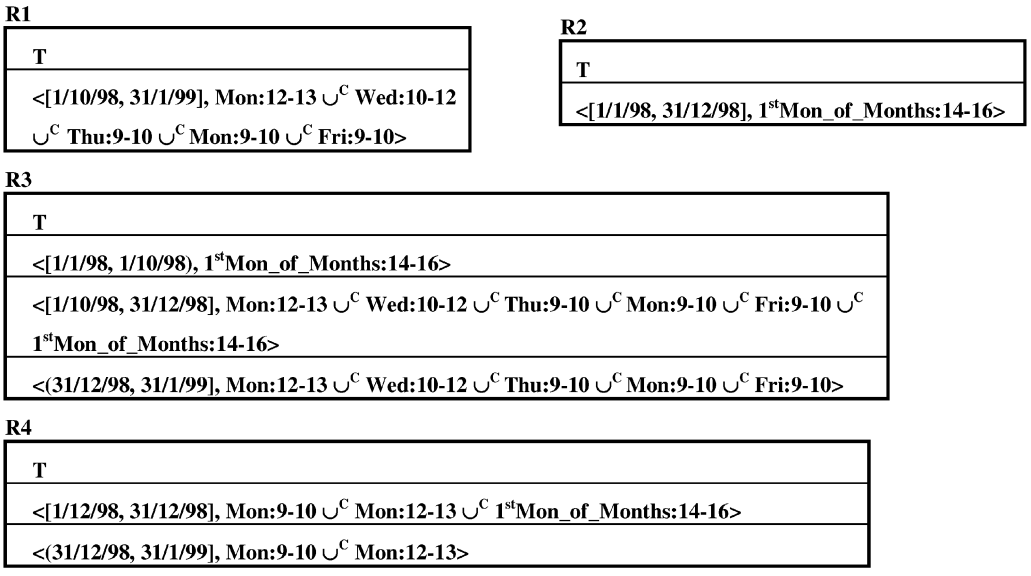

Fig. 6. Stepwise symbolic evaluation of the query Q1.

Which teachers had courses or office hours during meeting M03 in 1998?

(Q3) $\mathrm{SelectIn}_{[1/1/98,31/12/98]}((\pi^{\mathrm{P}}_{\mathrm{Teach-Cod}}(\mathrm{COURSES}) \cup^{\mathrm{P}} \mathrm{OFFICE}) \times^{\mathrm{P}} (\pi^{\mathrm{P}} \oslash; (\sigma^{\mathrm{P}}_{\mathrm{Meet-Cod=M03}}(\mathrm{MEET}))))$.

Answer:
$\{\#\mathrm{A02}| < [1/1/98\text{-}31/12/98], 1^{\mathrm{st}}\mathrm{Mon\_of\_Months}\colon 14\text{-}16 > \#\}$
(the symbolic intersection $\mathrm{Mon}\colon 14\text{-}16 \cap^{\mathrm{C}} 1^{\mathrm{st}}\mathrm{Mon\_of\_Months}\colon 14\text{-}16$ gives as result $1^{\mathrm{st}}\mathrm{Mon\_of\_Months}\colon 14\text{-}16$).

Give me the code of meetings held after 30/4/98 on the first day of each month. (where $1^{\mathrm{st}}\mathrm{Day\_of\_Month}$ has been defined by the user in the dedicated table).

(Q4) $\pi^{\mathrm{P}}_{\mathrm{Meet-Cod}}(\mathrm{SelectPtPer}_{\mathrm{After},30/4/98,1\mathrm{stDay\_of\_Month}}(\mathrm{MEET}))$
Answer: $\{\#\mathrm{M03}| < (30/4/98, 31/12/98], 1^{\mathrm{st}}\mathrm{Mon\_of\_Months}\colon 14\text{-}16 \cap^{\mathrm{C}} 1^{\mathrm{st}}\mathrm{Day\_of\_Month} > \#\}$.

8 Comparisons and Conclusions

In this paper, we proposed a temporal relational model and a temporal algebra dealing with user-defined "symbolic" periodicity in both data and queries, to the overall goal of *getting relational databases closer to a human-oriented way of dealing with temporal information*. Some approaches in the temporal database literature defined a temporal algebra and a temporal model to deal with periodicity, mostly representing periodicity as mathematical formulae; other approaches (also in artificial intelligence) mainly focused on the definition of high-level symbolic languages to model periodicity in a user-friendly way. In this paper, we proposed an original approach which integrates the advantages of both types of approaches.

We proposed a comprehensive and **integrated approach** (see again Fig. 1 in Section 2) to deal with user-defined periodicities (both in the data and in the queries) in temporal databases. In order to achieve such a goal, we introduced 1) a **temporal model and algebra** based on 2) a **symbolic language to express user-defined periodicity** whose semantics and properties are given on the basis of the domain of the temporal elements, as well as a 3) a **fully-symbolic and a semisymbolic evaluation method** of the set-operators on periodicities. It seems to us that such an integrated treatment of all these aspects provides crucial advantages since, among the other things, allowed us to

1. provide a semantics for the operators (e.g., Select_Periods, $\cap^{\mathrm{C}}$) of our high-level language and formally compare the expressiveness of our language with others in the literature,
2. prove the properties of the operators e.g., (Select_Periods, $\cap^{\mathrm{C}}$) of our high-level language
3. prove the properties of the operators of our temporal algebra (e.g., equivalence and reduction), and
4. prove the correctness and simplification-completeness of our symbolic and semisymbolic evaluation.

In our opinion, none of the approaches in the literature considers in detail all of the issues 1-3 above (and the related proofs). Moreover, we strongly believe that our (semi)symbolic evaluation method may have interesting applications also in other fields than TDB, including access control models to DB (see, e.g., [7]), data broadcasting [21], active databases, multimedia scenarios, etc.

8.1 Related Works and Comparisons

Periodicity and its related notions of *granularity* and *calendars* [8] have been widely studied in many different areas of logic and computer science. For instance, within the artificial intelligence community, most attention has been devoted to the treatment of qualitative temporal constraints (e.g., "before") between events which repeat in time (see, e.g., [25], [26], [28], [33], [40], [43]). On the other hand, approaches in classical temporal logic (see, e.g., [1], [30]) deal with truth and validity of logical formulae, specifying how predicates change value over time. However, [50] showed that classical temporal logics do not address the

issue of periodicity since, e.g., the classical logics with the *next* and *until* operators cannot define all periodic sets. In ETL [50], temporal connectives have been introduced in order to overcome this drawback; however, infinitely many connectives should be required. To remedy this problem, Vardi [48] introduced the temporal fixpoint calculus μTL, whose limitations are widely discussed in [46], which also provided an interesting overview of many approaches to periodic events. Regarding (nonmodal) first order logics, Enderton [19] showed that a standard first-order language, enhanced with integer congruence relations, captures periodicity.

In the area of temporal vatabases, periodicity has been widely studied. For example, periodicity and/or different time granularities are supported in [4], [5], [6], [7], [18], [22], [23], [29], [34], [45], [46], [49]. Many approaches only focused on the treatment of periodicity in the *query* (see, e.g., [46] and Oracle8; see www.oracle.com\databases\timeseries.html). Considering the approaches dealing with periodicity also in the *data*, Baudinet [5] distinguished between the approaches using deductive rules (e.g., [16], who dealt with periodicity in the context of $\mathrm{Datalog_{1s}}$) and those using *constraints*. For instance, Kabanza et al. have proposed a very influential and comprehensive approach based on constraints, carefully dealing with all the related aspects involved by the introduction of user-defined periodicity into relational temporal databases [22], [23]. In particular, they extended classical relational models to deal with periodic data by representing infinite temporal information by *generalised tuples* defined by *unions* of *linear repeating points* (points of the form $c_1 + c_2X$; henceforth LRP) and constraints on these points (e.g., $X_1 = X_2 + c_3$). They proved that the *expressive power* of their formalism corresponds to that of Presburger arithmetic. They also defined the temporal extension of the *algebraic operations* and a first order query language to deal also with LRP, and studied their *computational complexity*. However, we believe that a main drawback of Kabanza's approach (which was also recognized by one of the coauthors of Kabanza's approach in the paper [34]) is the fact that "mathematical" languages such as LRP are not "human-oriented" (see Section 8.2).

Regarding **symbolic languages dealing with periodicity**, [7], [9], [27], [34] are the works most closely related to our approach. However, these approaches (except [34]) did ***not*** devise relational temporal models and algebras, and did not introduce symbolic evaluation methods to deal with periodicities (which are the main tasks of our work).

Niezette and Stevenne [34] extended the notion of LRP to *Linear Repeating Intervals* (LRI) and applied it to the definition of calendars. In their approach, calendars are periodic infinite sets of consecutive intervals, and can be expressed using expressions of the form $O_1.C_1 + \ldots + O_n.C_n \diamond D$, where the sum identifies the starting point of the intervals and D their duration. C_i are calendars and O_i are integers (to indicate a selection) or the keyword all (to indicate that all intervals have to be considered). For example, $\mathrm{all.Years} + \{1, 2, 5\}.\mathrm{Months} + 3.\mathrm{Days} \diamond 3\ \mathrm{Hours}$ has as extension the set of all time intervals which represent the first three hours of the third days of January, February, and May of each year (the ordered list of calendars used in a defintion—e.g., <Years,Months,Days> in the example—is called *hierarchy* in [34]). However, their symbolic language is mainly a symbolic interface built upon LRI (and, thus LRP), thus inheriting some of the disadvantages of this formalism, as shown in Section 8.2. Moreover, as proven by Bettini and De Sibi [9], calendars in Niezette's and Stevenne's language do not correspond to *periodic granularities* (see [8], [9]). Niezette's and Stevenne's language has been used in [7] to deal with periodic authorizations in access control models for DBMS. In their approach, an authorization is a pair consisting of a time interval (the bounding of the authorization, corresponding to the *frame time* in our approach) and a periodicity (expressed using Niezette's and Stevenne's language) which are translated into Toman's constraint based language [45] to manipulate them. Thus, basically, manipulation of periodicities in [7] is not symbolic.

Basically, however, our high-level symbolic language can be seen as a revision of Leban et al.'s language in [27] (used also, e.g., in [14]). Leban introduced a way of defining primitive sets (called *collections*) of time intervals which cover the time line (e.g., seconds, days, etc., called *calendars*) that basically corresponds to our Convex-Cal operator (except our use of an anchor time point; thus, our Convex_Cal operator corresponds to *generate* in [7]). *Dicing* and *slicing* are Leban's operators used to build new collections on the basis of other collections. Basically, our Select_Periods operators correspond to a combined application of a dicing and a slicing. In fact, Leban's dicing operator allows one to divide each collection into another collection using a relational operator (e.g., "Days .During. Weeks" divides each one-week-long interval in the extension of Weeks into the collection of Days contained in it) and slicing selects an interval from a collection (e.g., "1/ Days .During. Weeks" selects Mondays). However, Leban also allows independent (and possibly nested) applications of these operators and, in particular, of dicing. Thus, the semantics of Leban's language cannot be given in terms of Gadia's temporal elements [20] but requires *nested sets* of time intervals. Moreover, in [27], the same interval may appear more than once in the resulting collections, and the semantics of dicing expressions such as "Days .<. Months" seems to us ambiguous and unclear. Finally, a major difference regards our *introduction of the set operators of intersection* $\cap^C$, *union* $\cup^C$, *difference* $-^C$, and *complement* $\neg^C$ (set operators are not used by Leban; Niezette and Stevenne [34] only considered union; set operators have been considered, e.g., in [14], [17], [24]).

Only a few approaches extended the **temporal model and algebra** dealing with periodicity in the validity time of tuples. The approaches in [22], [23], [24], [34] are the most closely related to our work. Kurt and Ozsoyoglu [24] defined the temporal type of *periodic elements* to model strictly periodic and also partially periodic events. Periodic elements consist of both an aperiodic part and a periodic part, represented by the repetition pattern and the period of repetition. They also defined the set-operations of union, intersection, complement, and difference, which are closed with respect to periodic elements. While Kurt and Ozsoyoglu [24] devoted most attention to the definition of a

periodic temporal object oriented SQL, they just sketched the language to define the *repetition pattern* and the *period*. Moreover, they do not mention any facility for introducing user-defined *periods*. Thus, in practice, only little support is provided to deal with user-defined periodicity. Moreover, Kurt did not define the semantics set-operators on periodic elements and did not show the implementation of these operators.

Finally, the approach by Niezette and Stevenne [34] is the more closely related to our approach. As discussed above, they proposed an approach using a symbolic layer built upon generalized databases (i.e., databases in which tuples are associated with LRP to express periodicity [22], [23]). They also proposed a semisymbolic evaluation method for operations on symbolic periodicity (actually, in [34], *only the treatment of intersection* between periodicities is discussed). Such a method is only sketched in [34], where they presented *only examples* showing that they exploit the symbolic level to perform efficient symbolic simplifications whenever periodicities share the same *hierarchy*. Otherwise, the computation of intersection over two periodicities $X = O_1.C_1 + \ldots + O_n.C_n$ and $Y = O'_1.C'_1 + \ldots + O'_m.C'_m$ basically transforms X and Y in such a way that they use the very same hierarchy of calendars C_i and then performs simple operations on the selections O_i and O'_i. This transformation involves very complex operations of calendar *completion* and *synchronization*. Moreover, case X and Y contain different calendars which are not in a *subcalendar* relation,[10] this process involves the mapping onto linear repeating intervals, and the output may be very cumbersome and not friendly for the users (see the example in Section 8.2). Moreover, in [34], the *correctness and completeness* of their semisymbolic evaluation method are not discussed.

8.2 Symbolic Periodicity vs. Mathematical-Based Periodicity vs. Extensional Approaches

Finally, it is worth reconsidering our choice of dealing with "symbolic" user-defined periodicity. An "intensional" (called "implicit" in [5]) approach has many advantages with respect to an extensional one, e.g., it allows one to deal also with *infinite periodicities* (e.g., "each Mondays," with frame time $(-\infty, +\infty)$) and is cheaper as regards both memory allocation and computational costs. More important, there is an additional fundamental advantage from the "human-oriented" point of view which inspires our approach: In many cases, extensional results are only scarcely informative to users, who should have to reinterpret them in order to obtain an "intensional" significant result. The comparison between the extensional and the symbolic part of the answer to query Q2 in Section 7 is just one of the many examples of this fact. For these reasons, most TDB approaches chose to adopt an "intensional approach" (see, e.g., [4], [5], [6], [7], [22], [23], [34], [45], [49] and the survey [46]).

However, from the "human-oriented" point of view, also constraint-based approaches using mathematical formulae to express periodicity (see, e.g., [22], [23], [45] and the survey in [46]) seem to us inadequate (see, e.g., the sharp criticism in [27], [34], and notice that Stevenne was one of the coauthors of the LRP paper [22]). Let us consider, e.g., the definition of "the first Mondays of each month" (henceforth 1^{st}-MM for short) in Kabanza's mathematical formalism. Weeks and months are not synchronous (or, in the terms in [34], the *subcalendar* relation does not hold between them), and the minimum common period over which they repeat regularly is 28 years long. Thus, in order to express 1^{st}-MM using Kabanza's unions of LRP [22], [23], one has to explicitly list all the terms of the union. For example, taking days as basic periodicity and 1/1/1998 as reference time, one has to state a list (union) of 336 formulae such as

$$336 \text{ formulae} \begin{cases} 10,227 * n + 4,\ 10,227 * n + 4, X1 = X2 \\ 10,227 * n + 32,\ 10,227 * n + 32, X1 = X2 \\ 10,227 * n + 60,\ 10,227 * n + 60, X1 = X2 \\ ..... \end{cases}$$

where the first and the second formulae in each row define the LRP representing the starting and ending point of each Monday, and the third formula in each row states the constraint that these points are equal. 10,227 is the number of days in 28 years, four represents the start of the first Monday after the reference time (i.e., 5/1/1998). The union contains 336 formulae since there are 336 months in 28 years. It is evident that this formalism is neither user-friendly nor human-oriented since it forces users to introduce very long lists of cumbersome formulae and does not provide a perspicuous and commonsense representation of the data.

A further major drawback is that, as already noticed by Leban, in mathematical approaches all formulae must be built from scratch, instead of building them in a compositional and incremental way (e.g., in Kabanza et al. [22], [23] 1^{st}-MM must be defined from scratch instead that in terms of the definitions of Months and of Mondays).

On the other hand, in the symbolic language by Niezette and Stevenne, 1^{st}-MM can be easily represented by

$$\text{all.months} + 1.\text{weeks} + 1.\text{days}.$$

However, Niezette and Stevenne strongly rely on the mapping onto LRP in order to deal with queries involving set operations on periodicities. Consider, e.g., the intersection between 1^{st}-MM and the fifth days of each month (5^{th}-M for short; defined by all.months+5.days). Since 1^{st}-MM and 5^{th}-M have different calendar hierarchies (weeks is missing in the definition of 5^{th}-M) intersection finds out a common hierarchy. Since Weeks and Months are not in a *subcalendar* relation, this process involves the introduction of their "minimum common multiple periodicity" (called "bigwheels" by Niezette [34]), which is a 28-years-long calendar corresponding to the one implicitly used in the mathematical LRP-based example shown above. Thus, the definition of 1^{st}-MM is transformed into the union below of 336 terms

10. Niezette and Stevenne [34] defined the subcalendar relation as follows: A calendar X is a subcalendar of a calendar Y if each interval of Y is exactly covered by a finite number of intervals of X. For instance, Days is a subcalendar of Months, while Weeks is not.

$$\begin{aligned}&\text{bigwheels} + 1.\text{months} + 5.\text{days}\\ &\cup \text{bigwheels} + 2.\text{months} + 2.\text{days}\\ &\cup \text{bigwheels} + 3.\text{months} + 2.\text{days} \cup .....\end{aligned}$$

and 5^{th}-M into the definition "bigwheels + all.months + 5.days," and the intersection between the two definitions is the union below (of 48 terms):

$$\begin{aligned}&\text{bigwheels} + 1.\text{months} + 5.\text{days}\\ &\cup \text{bigwheels} + 10.\text{months} + 5.\text{days}\\ &\cup \text{bigwheels} + 16.\text{months} + 5.\text{days} \cup .....\end{aligned}$$

Actually, this output seems to us scarcely more informative and human-oriented than an extensional or mathematical one.

In our symbolic approach, we can easily cope with the examples above in a user-friendly, compositional, and incremental way by introducing the Definitions 1, 2, and 3, and given the definitions of Days, Weeks, and Months:

1. $\text{Mondays} = 1/\text{DaysNSDurWeeks}$,
2. $1^{\text{st}}\text{-MM} = 1/\text{MondaysNSDurMonths}$, and
3. $5^{\text{th}}\text{-M} = 5/\text{DaysNSDurMonths}$.

Finally, in our approach, the intersection of 1^{st}-MM and 5^{th}-M is easily computed using the conditional transformation rules in Table 1 (the temporal relation between 1^{st}-MM and 5^{th}-M is Λ^{C}), and gives as result the compact symbolic output "1^{st}-MM $\cap^{\text{C}}$ 5^{th}-M." Even in this case, our semisymbolic evaluation is exploited in order to consider the simplifications due to a given frame-time FT, our output is either "Ø" (if the intersection has empty extension in FT) or "1^{st}-MM $\cap^{\text{C}}$ 5^{th}-M." In fact, we use the extensional evaluation just in order to detect redundant or empty periodicities, so that (differently from [34]), the user-friendliness of our output is not affected by the adoption of extensional evaluations.

In the case of many applications of VT-modifying algebraic operators (e.g., $\times^{\text{P}}$) in the same query, we might have a growth in the dimension of the periodicities (see Section 5.5). However, notice that, in our approach, such a growth is only related to the number of VT-modifying operators in the queries, while in Niezette and Stevenne it also depends on the least common multiple period of input periodicities (which is usually several orders of magnitude greater than the number of operators in a query).[11] Moreover, no simplification (e.g., redundancy elimination) is performed in Niezette's approach.

8.3 Future Work

We are starting to devise a prototypical implementation of our approach by exploiting the abstract data types facilities provided by DBMS. In such an implementation, e.g., validity times are stored simply as a string of characters; obviously, a complete implementation should involve also an extension of SQL and of the operators to insert, delete, and/or modify tuples. We also plan to extend our approach to cover also *nearly periodic events* [46]. In such an extension, we will probably rely on an *interval-based* extensional semantics (see [42], [44]) instead that on Gadia's [20] classical point-based one. Moreover, we want to apply our symbolic language to deal with user-defined periodicity and its symbolic implementation in the area of multimedia scenario authoring [2].

Finally, a long term project involves the treatment of qualitative temporal constraints (e.g., "before") between tuples of periodic tables not only in the queries but also *in the data* (to deal, e.g., with temporal *constraints* such as "each Monday between 1/10/98 and 31/1/99 the lesson of History is Before Mathematics"—see [40]—in the *data* of temporal databases). This would involve the definition of an integrated architecture to couple a temporal reasoner about such constraints (such as, e.g., TeMP [40]) with the data model and algebra in this paper (thus, extending the modular architecture we proposed in [13], that coupled the LaTeR temporal reasoner [11], [12] with a temporal DB [10] in order to deal with qualitative constraints on nonperiodic events only).

11. Moreover, even if in [34] Niezette and Stevenne didn't describe the application of intersection on composite periodicities (e.g., on unions of base periodicities), one could argue that they would exploit the distributivity of intersection over union. In such a case, n repeated nested applications of intersection (as discussed in Section 5.5 for our approach) of periodicities consisting of unions of u intersection terms would lead, in Niezette's approach, to a resulting periodicity of $\text{u}^{\text{n+1}}$ *lcm$(\text{Period}(\text{Per}_1, \text{BP}), \ldots, \text{Period}(\text{Per}_\text{n}, \text{BP}))$ terms. Thus, in the worst case, the dimension would be $(\text{u}^{\text{n+1}} * \text{Period}(\text{Per}_1, \text{BP})^* \ldots^* \text{Period}(\text{Per}_\text{n}, \text{BP}))$, where $\text{Period}(\text{Per}_\text{i}, \text{BP})$ indicates the period of a periodicity with respect to the basic periodicity.

ACKNOWLEDGMENTS

The author is very grateful to R.T. Snodgrass and C.S. Jensen for many useful discussions, comments, and suggestions on early versions of this paper and to L. Egidi for her help in the revision of the latest version. The author is also very indebted to C. Bettini, R. Chandra, D. Cuckierman, C. Jensen, R. Morris, M. Ozsoyoglu, B. Pernici, A. Segev, R. Snodgrass, and P. Wolper, who sent him copies of their papers and other related papers in the literature and/or gave him many relevant references. The author is also very grateful to the anoymous reviewers for their inspiring suggestions, their in-depth comments, and their constructive criticism.

REFERENCES

[1] M. Abadi and Z. Manna, "Temporal Logic Programming," *J. Symbolic Computation,* vol. 8, no. 3, pp. 277-295, Sept. 1989.

[2] S. Adali, L. Console, M.L. Sapino, M. Schenone, and P. Terenziani, "Representing and Reasoning with Temporal Constraints in Multimedia Presentations," *Proc. Seventh Int'l Workshop Temporal Representation and Reasoning (TIME'00),* pp. 3-12, 2000.

[3] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. ACM,* vol. 26, no. 11, pp. 832-843, Nov. 1983.

[4] M. Baudinet, M. Niezette, and P. Wolper, "On the Representation of Infinite Temporal Data and Queries," *Proc. ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems,* pp. 280-290, May 1991.

[5] M. Baudinet, J. Chomicki, and P. Wolper, "Temporal Databases: Beyond Finite Extensions," *Proc. Int'l Workshop Infrastructure for Temporal Databases,* June 1993.

[6] M. Baudinet, J. Chomicki, and P. Wolper, "Temporal Deductive Databases," *Temporal Databases,* A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev and R. Snodgrass, eds., 1993.

[7] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati, "An Access Contro Model Supporting Periodicity Constraints and Temporal Reasoning," *ACM Trans. Database Systems,* vol. 23, no. 3, pp. 231-285, 1998.

[8] C. Bettini, C. Dyreson, W. Evans, R. Snodgrass, and X. Wang, "A Glossary of Time Granularity Concepts," *Temporal Databases: Research and Practice,* 1998.

[9] C. Bettini and R. De Sibi, "Symbolic Representation of User-Defined Time Granularities," *Proc. Sixth Int'l Workshop Temporal Representation and Reasoning (TIME'99),* pp. 17-28, 1999.

[10] V. Brusoni, L. Console, B. Pernici, and P. Terenziani, "Extending Temporal Relational Databases to Deal with Imprecise and Qualitative Temporal Information," J. Clifford and A. Tuzhilin, eds., *Recent Advances of Temporal Databases,* pp. 3-22, 1995.

[11] V. Brusoni, L. Console, and P. Terenziani, "On the Computational Complexity of Querying Bounds on Differences Constraints," *Artificial Intelligence,* vol. 74, no. 2, pp. 367-379, 1995.

[12] V. Brusoni, L. Console, B. Pernici, and P. Terenziani, "LaTeR: An Efficient, General-Purpose Manager of Temporal Information," *IEEE Expert,* vol. 12, no. 4, pp. 56-64, 1997.

[13] V. Brusoni, L. Console, B. Pernici, and P. Terenziani, "Qualitative and Quantitative Temporal Constraints and Relational Databases: Theory, Architecture, and Applications," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, no. 6, pp. 948-968, Nov./Dec. 1999.

[14] R. Chandra, A. Segev, and M. Stonebraker, "Implementing Calendars and Temporal Rules in Next Generation Databases," *Proc. Int'l Conf. Data Eng.,* pp. 264-273, 1994.

[15] J. Chomicki and T. Imielinsky, "Temporal Deductive Databases and Infinite Objects," *Proc. Seventh ACM Symp. Principles of Database Systems,* pp. 61-73, Mar. 1988.

[16] J. Chomicki and T. Imielinsky, "Finite Representation of Infinite Query Answers," *ACM Trans. Database Systems,* vol. 18, no. 2, pp. 181-223, June 1993.

[17] D. Cukierman and J. Delgrande, "Expressing Time Intervals and Repetition within a Formalization of Calendars," *Computational Intelligence,* vol. 14, no. 4, pp. 563-597, 1998.

[18] C.E. Dyreson, W.S. Evans, H. Lin, and R.T. Snodgrass, "Efficiently Supporting Temporal Granularities," *IEEE Trans. Knowledge and Data Eng.,* vol. 12, no. 4, pp. 568-587, 2000.

[19] H. Enderton, *A Mathematical Introduction to Logic.* New York: Academic Press, 1972.

[20] S.K. Gadia, "A Homogeneous Relational Model and Query Languages for Temporal Databases," *ACM Trans. Database Systems,* vol. 13, no. 4, pp. 418-448, 1988.

[21] T. Imielinski, "Data on Air—What's in It for the Database Theorist," *Proc. Conf. Database Theory,* pp. 1-13, 1995.

[22] F. Kabanza, J.-M. Stevenne, and P. Wolper, "Handling Infinite Temporal Data," *Proc. ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems,* pp. 392-403, Apr. 1990.

[23] F. Kabanza, J.-M. Stevenne, and P. Wolper, "Handling Infinite Temporal Data," *J. Computer and System Sciences,* vol. 51, pp. 3-17, 1995.

[24] A. Kurt and M. Ozsoyoglu, "Modelling and Querying Periodic Temporal Databases," *Proc. Workshop Sixth Int'l Conf. Database and Expert Systems Applications (DEXA),* pp. 124-133, 1995.

[25] P. Ladkin, "Primitive and Units for Time Specification," *Proc. Fifth Nat'l Conf. Artificial Intelligence,* pp. 354-359, Aug. 1986.

[26] P. Ladkin, "Time Representation: A Taxonomy of Interval Relations," *Proc. Fifth Nat'l Conf. Artificial Intelligence,* pp. 360-366, Aug. 1986.

[27] B. Leban, D.D. McDonald, and D.R. Forster, "A Representation for Collections of Temporal Intervals," *Proc. Fifth Nat'l Conf. Artificial Intelligence,* pp. 367-371, Aug. 1986.

[28] G. Ligozat, "On Generalized Interval Calculi," *Proc. Ninth Nat'l Conf. Artificial Intelligence,* pp. 234-240, July 1991.

[29] N. Lorentzos and R. Johnson, "Extending Relational Algebra to Manipulate Temporal Data," *Information Systems,* vol. 13, no. 3, pp. 289-296, 1988.

[30] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1992.

[31] E. McKenzie and R. Snodgrass, "Schema Evolution and the Relational Algebra," *Information Systems,* vol. 15, no. 2, pp. 207-232, 1990.

[32] L. McKenzie and R. Snodgrass, "Evaluation of Relational Algebras Incorporating the Time Dimension in Temporal Databases," *ACM Computing Surveys,* vol. 23, no. 4, pp. 501-543, 1991.

[33] R.A. Morris, W.D. Shoaff, and L. Khatib, "Domain Independent Temporal Reasoning with Recurring Events," *Computational Intelligence,* vol. 12, no. 3, pp. 450-477, 1996.

[34] M. Niezette and J.-M. Stevenne, "An Efficient Symbolic Representation of Periodic Time," *Proc. First Int'l Conf. Information and Knowledge Management,* Nov. 1992.

[35] R. Snodgrass, "The TSQL2 Query Language," Kluwer Academic, 1995.

[36] R. Snodgrass and I. Ahn, "Temporal Databases," *Computer,* vol. 19, no. 9, pp. 35-42, Sept. 1986.

[37] R.T. Snodgrass, M.H. Bohlen, C.S. Jensen, and A. Steiner, "Transitioning Temporal Support in TSQL2 to SQL3," *Temporal Databases—Research and Practice,* O. Etzion, S. Jajodia, S. Sripada eds., pp. 150-191, 1998.

[38] M. Soo and R. Snodgrass, "Multiple Calendar Support for Conventional Database Management Systems," *Proc. Int'l Workshop an Infrastructure for Temporal Databases,* June 1993.

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

[40] P. Terenziani, "Integrating Calendar-Dates and Qualitative Temporal Constraints in the Treatment of Periodic Events," *IEEE Trans. Knowledge and Data Eng.,* vol. 9, no. 5, pp. 763-783, 1997.

[41] P. Terenziani, "Symbolic User-defined Periodicity in Temporal Relational DataBases, version II," Technical Report RT58-00, Dipartimento di Informatica, Univ. di Torino, http://www.di.unito.it/~terenz/TECH-REP/, Year?

[42] P. Terenziani, "Is Point-Based Semantics Always Adequate for Temporal Databases?" *Proc. Seventh Int'l Workshop Temporal Representation and Reasoning (TIME'00),* pp. 191-199, 2000.

[43] P. Terenziani and P. Terenziani, "Integrated Temporal Reasoning with Periodic Events," *Computational Intelligence,* vol. 16, no. 2, pp. 210-256, May 2000.

[44] P. Terenziani and R.T. Snodgrass, "Reconciling Point-Based and Interval-Based Semantics in Temporal Relational Databases: A Proper Treatment of the Telic/Atelic Distinction," Technical Report TR-60, TIMECENTER, June 2001.

[45] D. Toman, J. Chomicki, and D. Rogers, "Datalog with Integer Periodicity Constraints," *Proc. Int'l Logic Programming Symp.,* pp. 189-203, 1994.

[46] A. Tuzhilin and J. Clifford, "On Periodicity in Temporal Databases," *Information Systems,* vol. 20, no. 8, pp. 619-639, Dec. 1995.

[47] F. Van Eynde, "Iteration, Habituality and Verb Form Semantics," *Proc. Third Conf. European Chapter of the Assoc. for Computational Linguistics,* pp. 270-277, Apr. 1987.

[48] M.Y. Vardi, "A Temporal Fixpoint Calculus," *Proc. 15th Ann. ACM SIGACT-SIGPLAN Symp. Principles of Programming Languages,* pp. 250-259, 1988.

[49] X. Wang, C. Bettini, A. Brodsky, and S. Jajodia, "Logical Design for Temporal Databases with Multiple Granularities," *ACM Trans. Database Systems,* vol. 22, no. 2, pp. 115-170, 1997.

[50] P. Wolper, "Temporal Logic Can be More Expressive," *Information and Control* vol. 56, pp. 72-99, 1983.

Paolo Terenziani received both the Laurea degree (in 1987) and the PhD degree in computer science (in 1993) from the Universitá di Torino, Italy. He is currently a full professor with the Dipartimento di Scienze e Tecnologie e Avanzate of the Universitá del Piemonte Orientale "Amedeo Avogadro," Alessandria, Italy. His research interests mainly focus on the treatment of time and time-dependent phenomena in different areas, including databases (in the subareas of temporal relational databases, and of semantics) and artificial intelligence (natural language, knowledge representation, temporal reasoning, constraint propagation techniques). He has published more than 50 papers about these topics in refereed journals and conferences.