Symbolic Representation of User-defined Time Granularities

Claudio Bettini Roberto De Sibi
DSI, University of Milan
via Comelico 39, 20135 Milan, Italy
bettini@dsi.unimi.it, rdesibi@it.oracle.com

Abstract

In the recent literature on time representation, an effort has been made to characterize the notion of time granularity and the relationships between granularities, in order to have a common framework for their specification, and to allow the interoperability of systems adopting different time granularities. This paper considers the mathematical characterization of finite and periodical time granularities, and it identifies a user-friendly symbolic formalism which captures exactly that class of granularities. This is achieved by a formal analysis of the expressiveness of well-known symbolic representation formalisms.

1. Introduction

There is a wide agreement in the AI and database community on the requirement for a data/knowledge representation system of supporting standard as well as user-defined time granularities. Examples of standard time granularities are days, weeks, months, while user defined granularities may include business-weeks, trading-days, working-shifts, school-terms, with these granularities having different definitions in different application contexts. The work in [3, 4] represents an effort to formally characterize the notion of time granularity and the relationships between granularities, in order to have a common framework for their specification and to allow the interoperability of systems adopting different time granularities. The formal definition, however, is essentially a mathematical characterization of the granules, and it is not suitable for presentation and manipulation by the common user. The goal of this paper is identifying an intuitive formalism which can capture a significant class of granularities within the formal framework and which is closed for this class with respect to the operations it allows. This class can be intuitively described as containing all finite granularities, as well as all periodical ones. Instead of inventing yet another symbolic formalism for this purpose, in this work we consider some existing proposals, analyzing their expressiveness with respect to our goal.

A symbolic formalism, based on collections of temporal intervals, was proposed in [11] to represent temporal expressions occurring in natural language and used in several application domains like appointment scheduling and time management. This formalism has been adopted with some extensions by many researchers in the AI [9, 15, 6] and Database area [8, 5]. From the deductive database community, a second influential proposal is the *slice* formalism introduced in [14], and adopted, among others, in [2]. None of these formalisms and extensions seems to have the expressive power we are seeking, despite some of the proposals include features that go beyond what is needed in our framework. For example, [6] provides a powerful formalism to represent calendars and time repetition, including existential and universal quantification, which supports the representation of uncertainty, a feature not considered in our framework. Moreover, some *calendar expressions* in [6] go beyond the specification of granularities, as defined in [4, 3] and in this paper, allowing the representation of overlapping granules of time. The formalism can represent recurring events in the form of non-convex intervals, but it does not seem to be able to represent what in the following we call gap-granularities, where gaps may not only occur between one granule and the next, but also within granules. A business-month seen as an indivisible time granule defined as the union of all business-days within a month is an example.

Relevant work on non-convex intervals and repetition includes [10, 13], but the emphasis in these works is more on reasoning with qualitative relations than on calendar expression representation. In addi-

tion to the research cited above, significant work on time granularity includes [16, 12, 7].

The contribution of this paper is twofold: on one side we give results on the expressiveness of the formalisms proposed in [11] and [14] which we identify as the two basic approaches to symbolic representation, while, on the other side, we propose an extension to one of these formalisms that allows to capture exactly the class of finite and infinite periodical granularities we defined in [3].

In the next section we introduce the formal notion of time granularity. In Section 3 we briefly describe the *collection* and *slice* symbolic representation formalisms, and we evaluate their expressiveness and formal properties. In Section 4, we propose an extension to the collection formalism to capture gap-granularities, and we conclude the paper in Section 5. Appendix A summarizes the syntax of the symbolic formalism, and Appendix B contains the proofs of the results in the paper.

2. Characterization of time granularities

In this section we introduce the mathematical characterization of time granularities as proposed in [4] and further refined and summarized in [3]. Granularities are defined with respect to an underlying *time domain*, which can be formally characterized simply as a set whose elements are ordered by a relationship. For example, integers $(\mathbf{Z}, \leq)$, natural numbers $(\mathbf{N}, \leq)$, rational $(\mathbf{Q}, \leq)$, and real numbers $(\mathbf{R}, \leq)$ are all possible choices for the time domain.

Definition 1 *A* granularity *is a mapping G from the integers (the* index set*) to subsets of the time domain such that: (1) if $i < j$ and $G(i)$ and $G(j)$ are non-empty, then each element of $G(i)$ is less than all elements of $G(j)$, and (2) if $i < k < j$ and $G(i)$ and $G(j)$ are non-empty, then $G(k)$ is non-empty.*

The first condition states that granules in a granularity do not overlap and that their index order is the same as their time domain order. The second condition states that the subset of the index set that maps to non-empty subsets of the time domain is contiguous. While the time domain can be discrete, dense, or continuous, a granularity defines a countable set of granules, each one identified by an integer. The index set can thereby provide an "encoding" of the granularity in a computer.

The definition covers standard granularities like `Days`, `Months`, `Weeks` and `Years`, bounded granularities like `Years-since-2000`, granularities with non-contiguous granules like `Business-Days`, and gap-granularities, i.e., granularities with non-convex intervals as granules like `Business-Months`. As an example of the encoding, `Years-since-2000` can be defined as a mapping G, with $G(1)$ mapped to the subset of the time domain corresponding to the year 2000, $G(i+1)$ to the one corresponding to the year 2001, and so on, with $G(i) = \emptyset$ for $i < 1$.

Independently from the integer encoding, there may be a "textual representation" of each non-empty granule, termed its *label*, that is used for input and output. This representation is generally a string that is more descriptive than the granule's index (e.g.,"August 1997", "1/2/2000", etc.) .

Among the many relationships between time granularities (see [4]), the following defines an essential concept for this paper.

Definition 2 *A granularity H is periodical with respect to a granularity G if*

1. *For each $i \in \mathbf{Z}$ there exists a (possibly infinite) subset S of the integers such that $H(i) = \bigcup_{j \in S} G(j)$;*

2. *There exist $R, P \in \mathbf{Z}^+$, where R is less than the number of non-empty granules of H, such that for all $i \in \mathbf{Z}$, if $H(i) = \bigcup_{j \in S} G(j)$ and $H(i+R) \neq \emptyset$ then $H(i+R) = \bigcup_{j \in S} G(j+P)$.*

The first condition states that any non-empty granule $H(i)$ is the union of some granules of G; for instance, assume $H(i)$ is the union of the granules $G(a_1), G(a_2), \ldots, G(a_k)$. The periodicity property (condition 2 in the definition) ensures that the R^{th} granule *after* $H(i)$, i.e., $H(i+R)$, if non-empty, is the union of $G(a_1+P), G(a_2+P), \ldots, G(a_k+P)$. This results in a periodic "pattern" of the composition of R granules of H in terms of granules of G. The pattern repeats along the time domain by "shifting" each granule of H by P granules of G. P is also called the "period" of H. The condition on R enforces that at least one granule of H is a periodic repetition of another granule.

A granularity H which is periodical with respect to G is specified by: (i) the R sets of indexes of G $S_0, \ldots, S_{R-1}$ describing the non-empty granules of H within one period; (ii) the value of P; (iii) the indexes of first and last non-empty granules in H, if their value is not infinite. Then, if $S_0, \ldots, S_{R-1}$ are the sets of indexes of G describing $H(0), \ldots, H(R-1)$, respectively, then the description of an arbitrary granule

$H(j)$ is given by[1] $\bigcup_{i \in S_{j \bmod R}} G(P * \lfloor j/R \rfloor + i)$.

Many common granularities are in this kind of relationship, for example, `Years` is periodical with respect to both `Days` and `Months`. `Business-Months` is periodical with respect to `Business-Days`, which in turn is periodical with respect to `Days`. Most practical problems seem to require only a granularity system containing a set of time granularities which are all periodical with respect to a *basic* granularity. Usually `Days`, `Hours`, `Seconds` or `Microseconds` take this role, depending on the accuracy required in each application context. In this paper, for simplicity, we assume there is a fixed basic granularity covering the whole time domain.

Definition 3 *We say that a granularity G is periodical if it is periodical with respect to the basic granularity.*

In Figure 1 we represent the whole set of granularities, according to Definition 1, partitioned in two main subsets: those having all granules with contiguous values (NO-GAP) and those admitting granules with non-contiguous values (GAP). The inner circle identifies finite and periodical granularities: finite granularities are divided (dash line) into finite irregular and finite periodical[2] while infinite periodical granularities are divided into those having a first non-empty granule and no last granule (INFINITE-R), those having a last non-empty granule and no first granule (INFINITE-L), and those infinite on both sides (INFINITE). This classification will be useful when considering the expressive power of symbolic formalisms.

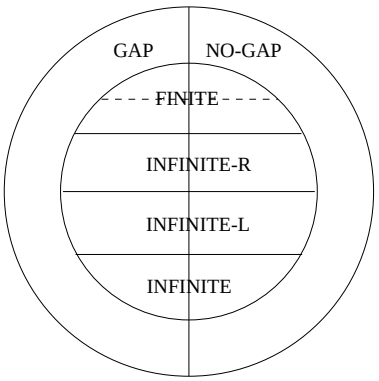

Figure 1. A classification of time granularities.

3. Two approaches to symbolic representation

In this section we first remind the syntax and semantics of *collection* and *slice* formalisms, and then analyze their expressiveness with respect to the class of periodical granularities.

3.1. Collections and slices

The temporal intervals collection formalism was proposed in [11]. A *collection* is a structured set of intervals where the order of the collection gives a measure of the structure depth: an order 1 collection is an ordered list of intervals, and an order n ($n > 1$) collection is an ordered list of collections having order $n - 1$. Each interval denotes a set of contiguous moments of time. For example, the collection of `Months`, where each month is represented as the collection of days in that month, is a collection of order 2. In order to provide a user-friendly representation of collections, the authors introduce two classes of operators on collections and the notion of *calendar*, as a primitive collection. A *calendar* is defined as an order 1 collection formed by an infinite number of meeting[3] intervals which may start from a specific one. The two classes of operators are called *dicing* and *slicing*. A dicing operator allows to further divide each interval within a collection into another collection. For example, `Weeks:during:January1998` divides the interval corresponding to `January1998` into the intervals corresponding to the weeks that are fully contained in that month. Other dicing operators are allowed, adopting a subset of Allen's interval relations [1]. Slicing operators provide means of selecting intervals from collections. For example, `[1,-1]/Weeks:during:January1998` selects the first and last week from those identified by the dicing operator above. In general, slicing can be done using a list of integers, as well as with the keyword `the`, which identifies the single interval of the collection (if it is single), and the keyword `any`, which gives nondeterministically one of the intervals. Collection expressions can be arbitrarily composed using these two classes of operators starting from calendars, which are explicitly specified either by a periodic set of intervals, or as a grouping of intervals from previously defined

[1] This formula is correct provided that no granule of H is empty, but it can be easily adapted to the case with finite index for first and last non-empty granules.

[2] Despite this formal distinction, finite granularities will be treated uniformly in the results.

[3] Interval I_1 meets interval I_2 if I_2 starts when I_1 finishes.

calendars.

The slice formalism was introduced in [14] as an alternative to the collection formalism in order to have an underlying evaluation procedure for the symbolic expressions. It is based on the notions of *calendar* and *slice*. Similarly to the collection formalism, calendars are periodic infinite sets of consecutive intervals, but there is no first nor last interval. Intervals in a calendar are indexed by consecutive integers. Once a basic calendar is given in terms of the time domain, other calendars can be defined dynamically from existing ones by the construct $Generate(sp, C, l_1, \ldots, l_m)$ which generates a new calendar with m intervals in each period, the first one obtained grouping l_1 granules of calendar C, starting from $C(sp)$, the second grouping the successive l_2 granules, and so on, with $l_1, \ldots, l_m$ treated as a circular list. A calendar C_1 is a subcalendar of C_2 ($C_1 \sqsubseteq C_2$) if each interval of C_2 is exactly covered by finite number of intervals of C_1. `Weeks`, `Days`, `Months` are calendars with `Days`$\sqsubseteq$`Months`, `Days`$\sqsubseteq$`Weeks`, `Weeks`$\not\sqsubseteq$`Months`. A slice is a symbolic expression built from calendars and denoting a (finite or infinite) set of not necessarily consecutive intervals. It has the form $\sum_{i=1}^{n} O_i.C_i \triangleright D$ where the sum identifies the starting points of the intervals and D their duration. Each C_i is a symbol denoting a calendar and O_i is either a set of natural numbers or the keyword *all*. If the sum is simply $O_1.C_1$, it denotes the starting points of the intervals of C_1 whose index belongs to O_1, or the starting points of all intervals if $O_1 = all$. If the sum is $\sum_{i=1}^{n-1} O_i.C_i + O_n.C_n$ with $O_n = \{o_n\}$ it denotes the starting points of the o_n-th interval of C_n following each point in $\sum_{i=1}^{n-1} O_i.C_i$. For example, the sum `all.Years + {2,4}.Months + {1}.Days` denotes the set of points corresponding to the beginning of the first day of February and April of each year. The duration D has the form $h.C_d$ where C_d is a symbol denoting a calendar such that $C_d \sqsubseteq C_n$, and h is the number of successive intervals of C_d specifying the duration. Hence, the slice `all.Years + {2,4}.Months + {1}.Days` $\triangleright$ `2.Days` denotes a set of intervals corresponding to the first 2 days of February and April of each year.

3.2. Expressiveness and relationships

Both collections and slices essentially characterize periodic sets. Similarly to granularities, even in these formalisms there is the notion of a basic calendar, which defines the finest time units in the domain. Without loss of generality, in the following of the paper we assume that this basic calendar (denoted by C) is the basic granularity we mentioned in Section 2. A *period*, in terms of C can be associated with each slice expression S as well as with any collection expression E. Intuitively, the period indicates the number of instants of C after which the same pattern of intervals denoted by the expression is repeating; each interval in a period can be obtained by a constant shift of the corresponding interval in another period. If $C_1, \ldots, C_n$ are the calendars appearing in the expression, then the period is the least common multiple of $Period(C_i/C)$. Technically, $Period(C_i/C)$, is defined as $\sum_{j=1}^{\xi} elem(j, Duration(C_i/C))$, where $Duration(C_i/C)$ is a list of integers, each one denoting the duration of an interval of C_i in terms of C, $\xi = length(Duration(C_i/C))$, $elem(j, list)$ returns the j^{th} element of the $list$, and $length(list)$ returns the number of elements in the $list$. For example, $Duration($`Years/Days`$) = \langle 365, 365, 365, 366 \rangle$,[4] and, hence, $Period($`Years/Days`$) = 1461$.

We now consider the expressiveness of slice expressions with respect to the formal notion of granularity introduced in Section 2. If all the intervals denoted by a slice S are disjoint, we call S a *disjoint slice*. We also say that a granularity G is *equivalent* to a slice S, if each granule of G is formed by the union of a set of granules of the basic granularity (C) and this set is represented by one of the intervals denoted by the slice; moreover, each of the intervals must describe one of these sets.

Theorem 1 *Given a disjoint slice S, there exists a no-gap finite granularity, or a no-gap infinite periodical granularity G equivalent to S.*

Technically, if $S = \sum_{i=1}^{n} O_i.C_i \triangleright h.C_d$ is an infinite slice ($O_1 = all$), we have an algorithm to derive the intervals $\{[r_1, r_1 + H_1 - 1], \ldots, [r_n, r_n + H_n - 1]\}$, where H_i is the length in terms of the basic calendar C corresponding to h granules of C_d, starting at r_i. These intervals are the ones denoted by S within a slice period. Then, a periodical granularity G can be defined by taking $R = n$, $P = Period$, where $Period$ is the slice period in terms of C, and $G(i) = \bigcup_{x=r_i}^{r_i + H_i - 1} C(x)$ for each $i = 1, \ldots, n$. It is shown that G is equivalent to S. When S is finite, the same algorithm can be easily adapted to derive all the intervals S denotes. Then, the equivalent granularity is simply defined explicitly mapping each granule to one of these intervals. Disjointness ensures that the result of this mapping is indeed a granularity.

[4] Ignoring exceptions to leap years.

Example 1 Let S=`all.Weeks + {2,3}.Days ▷ 12.Hours` be an infinite slice and $Hours$ be the basic calendar. The slice $Period$ is 168 hours (the number of hours in a week) and in the period containing `Hours(1)` the slice denotes the set of intervals $\{[25, 36], [49, 60]\}$. The periodical granularity G, equivalent to S, is defined by taking $R = 2$ (the number of intervals in a period), $P = Period$, $G(1) = \bigcup_{x=25}^{x=36} \texttt{Hours}(x)$ and $G(2) = \bigcup_{x=49}^{x=60} \texttt{Hours}(x)$. □

The following example shows that if a slice is non-disjoint, then there is no equivalent granularity.

Example 2 Let S=`all.Weeks + {2,3}.Days ▷ 3.Days`. According to the slice semantics, this expression denotes all intervals spanning from Tuesday to Thursday and all intervals from Wednesday through Friday. By Definition 1, no pair of granules of the same granularity can overlap. Hence, no granularity can be found which is equivalent to S. □

To understand the expressiveness of the slice formalism with respect to granularities, we still need to check if any granularity in the identified classes is representable by a disjoint slice.

Theorem 2 *Given a no-gap finite granularity or a no-gap infinite periodical granularity, there exists an equivalent slice.*

The theorem states that any finite (periodical or not) granularity can be represented by a slice, and that the same holds for periodical granularities which are unbounded on both sides. INFINITE-R and INFINITE-L granularities cannot be represented by a slice, since the only way to denote an infinite set of intervals with a slice is to have $O_1 = all$, and there is no way within the slice formalism to impose a minimum or a maximum on that set.[5]

From the above results we can conclude that disjoint slices can represent exactly the set of granularities identified in Figure 2, while non-disjoint ones do not represent granularities at all. Unfortunately, it seems that there is no way to enforce disjointness by simple syntax restrictions.

We now consider the collection formalism.

Proposition 1 *Any collection E resulting from the application of a dicing or slicing operator is such that any two intervals t and u contained in E are either equal or disjoint.*

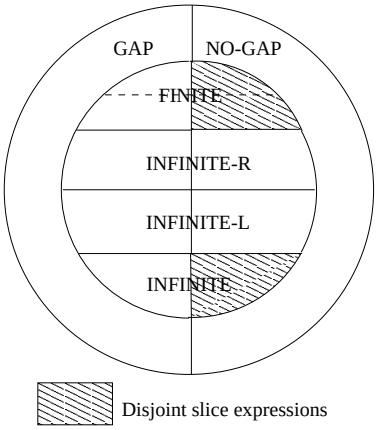

Figure 2. The subset of the granularities captured by the slice formalism

Proposition 1 follows from the semantics of the operators, and from the fact that each calendar contains only disjoint intervals. Similarly to slices, we say that a granularity G is *equivalent* to a collection E, if each granule of G is formed by the union of the granules of C represented by one of the intervals in the collection; moreover, each interval in the collection describes the composition of one of the granules of G.

Theorem 3 *Given a collection expression, there exists an equivalent no-gap periodical or finite non periodical granularity.*

Similarly to Theorem 1, we developed an algorithm to parse the expression, to derive its period, the intervals it denotes within the period[6], and lower/upper bounds if present. Once the intervals are derived, we have all the data that is needed to define the granularity G, since it will have the same period, the intervals within the period define the corresponding granules, and the lower/upper bounds are used to impose a starting/ending non-empty granule.

Example 3 Consider E=`{1/Mondays:during:Years.>.2000}`. This collection expression identifies an order 1

[5] Note however, that the addition of a *reference interval* (bound) to each slice, as used in [2], provides an easy extension to capture all no-gap periodical granularities.

[6] The intervals may be structured in a collection of order higher than 1, but this is irrelevant with respect to the time granules that the expression denotes.

collection that contains all first Mondays of each year starting since Monday, January 1st 2001. We assume `Days` is the basic calendar with `Days`(1) = Saturday, Jan 1st 2000. We first have to compute the expression period. Since `Mondays` is defined as `1/Days:during:Weeks` with the periods of `Days` and `Weeks` equal to 1 and 7 respectively, $lcm(1,7) = 7$ is the period computed for `Mondays`. Similarly, since the period for `Years` with respect to the basic calendar is 1461 (4 years in `Days`), the whole expression period is computed as $lcm(7,1461) = 10227$ (28 years in `Days`). Then, G is defined as having period $P = 10227$, $R = 28$ (the number of granules in each period), $G(1) =$ `Days`(367) (1/1/2001), $G(2) =$ `Days`(738) (7/1/2002), ..., $G(28) =$ `Days`(10228) (3/1/2028), and $G(i) = \emptyset$ for $i < 1$. To obtain these intervals the algorithm first restricts years to those after 2000, then it represents all Mondays within those years, and in the end it extracts the intervals corresponding to the first Monday. □

We also have the counterpart of Theorem 3.

Theorem 4 *Given a no-gap periodical or finite non-periodical granularity, there exists an equivalent collection expression.*

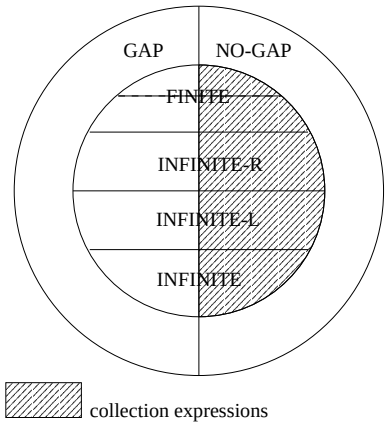

Figure 3. The subset of the granularities captured by the collection formalism

Note that in this case, all granularities in the right side of the inner circle of Figure 3 are captured. We can conclude that slices and collections have incomparable expressiveness, since slices can represent sets of overlapping intervals, and collections can represent INFINITE-R and INFINITE-L periodical granularities. From the above results, it is clearly possible to translate from one formalism to the other, when considering expressions denoting FINITE or INFINITE granularities, but it seems to be difficult to devise general rules to translate at the symbolic level, preserving the intuitiveness of the expression. Indeed, despite the `+` operator in slices may be intuitively interpreted as equivalent to `:during:` in collections, they actually have a different semantics.

The collection formalism has been extended with some additional operators in [8]. In particular, control statements `if-then-else` and `while` are introduced to facilitate the representation of certain sets of intervals, as for example, *the fourth Saturday of April if not an holiday, and the previous business-day otherwise*. Unfortunately, the syntax allows the user to define collections which contain overlapping intervals[7]. This implies that there are collection expressions in the extended formalism for which there does not exist an equivalent granularity.

4. An extension proposal

Both the collection and slice formalisms as well as their known extensions cannot represent gap granularities. Indeed, this requires a non-convex interval representation for each granule which is formed by non-contiguous instants. For example, they cannot represent `Business-Months`, where each granule is defined as the set of `Business-Days` within a month, and it is perceived as an indivisible unit. We propose an extension to the collection formalism in order to capture the whole set of periodical granularities.

We introduce the notion of *primitive collection*, which includes calendars as defined in the collection formalism as well as order 1 collections of non-convex intervals, where each of the intervals represents a granule. A primitive collection PC can be specified by $PC = Generate(sp, C_0, P, X)$, where sp is a synchronization point with respect to an existing calendar C_0, P is the period expressed in terms of C_0, and X is the set of non-convex intervals[8] identifying the position of granules of PC within a period. The synchronization point sp says that $PC(1)$ will start at the same instant as $C_0(sp)$.

[7] For example, consider an expression representing *a semester following the last day of the month, if it is a Sunday, otherwise the week following that day*. Considering 31/5/1998 and 30/6/1998, both the semester starting 1/6/1998 and the week starting 1/7/1998 will be denoted, with the first properly containing the second.

[8] Each $x_i \in X$ is the non-convex interval representing the i-th granule.

Example 4 Suppose a company has 2 weekly working shifts for its employees: `shift1`={Monday, Wednesday, Saturday} and `shift2`={Tuesday, Thursday, Friday}. It may be useful to consider these as two periodic granularities, where each shift is treated as a single time granule within a week. If Thursday 1/1/1998 is taken as `Days`(1), `shift1` = $Generate(5, \texttt{Days}, 7, \{\langle[1,1],[3,3],[6,6]\rangle\})$. Indeed, the synchronization point is 5, since the first granule of `shift1` following `Days`(1) starts on Monday January 5th 1998 which is 5 days later. C_0 is `Days`, the period P is 7 days and X is composed by $x_1 = \langle[1,1],[3,3],[6,6]\rangle$ which identifies the single granule within the period, formed by the first, third, and sixth day, starting from 5/1/1998, and repeating every 7 days. Similarly, `shift2` = $Generate(6, \texttt{Days}, 7, \{\langle[1,1],[3,3],[4,4]\rangle\})$ denotes the first, third and fourth day, starting from 6/1/1998, and repeating every 7 days. □

The user can specify collection expressions by arbitrarily applying dicing and slicing operators starting from primitive collections. Since operators now apply to non-convex intervals, we need to revise their definition. Let t and u be non convex intervals, with $t = \langle[a_1, b_1], \ldots, [a_n, b_n]\rangle$, and $u = \langle[c_1, d_1], \ldots, [c_m, d_m]\rangle$; moreover, let $S_t = \{x : a_1 \leq x \leq b_1 \vee \ldots \vee a_n \leq x \leq b_n\}$ and $S_u = \{y : c_1 \leq y \leq d_1 \vee \ldots \vee c_m \leq y \leq d_m\}$ be the sets of values represented by t and u respectively. Dicing operators are based on the following binary relations on non-convex intervals:[9]

$t\ during\ u$ iff $S \subseteq S'$
$t\ intersects\ u$ iff $S \cap S' \neq \emptyset$
$t\ starts\ u$ iff $(a_1 = c_1)$ and $(b_n \leq d_m)$
$t\ meets\ u$ iff $c_1 = b_n + 1$
$t > u$ iff $a_1 > d_m + 1$
$t < u$ iff $b_n < c_1$

A dicing operator $.rel.$ takes an order 1 collection $\{t_1, \ldots, t_k\}$ as its left operand and an interval u as its right operand, and it returns an order 1 collection $E = \{t \mid t = t_i \text{ for some } i = 1 \ldots k \text{ and } t\ rel\ u\}$. If the strict form $:rel:$ is used, then $E = \{t \cap u \mid t = t_i \text{ for some } i = 1 \ldots k \text{ and } t\ rel\ u\}$, i.e., only the portion of t which is contained in u is part of the resulting collection. When the right operand is a collection, instead of a single interval, the same procedure is applied for each of its intervals, resulting in a collection of one order higher. A slicing operator k/E replaces each order 1 collection contained in E with the k-th non-convex interval in that collection, while $\{k_1 \ldots k_n\}/E$ replaces it with the collection made of the subset of intervals whose position in the collection is specified by $\{k_1 \ldots k_n\}$.

Example 5 Consider the collection expression `Weeks:>:2/shift1:during:1998/Years` where `shift1` was defined in Example 4. This expression denotes all weeks following the end of the second work-shift of 1998. `Years` is the order 1 collection $\{\ldots\langle[1..365]\rangle\ldots\}$, and, for simplicity, we assume the interval $[1..365]$ corresponds to year 1998. Then, the slicing `1998/Years` returns the interval $\langle[1..365]\rangle$, and the dicing `shift1:during:1998/Years` returns the finite collection of order 1 composed by all the work-shifts during 1998: $\{\langle[5,5],[7,7],[10,10]\rangle, \ldots, \langle[355,355],[357,357],[360,360]\rangle\}$. The selection of the second of those work-shifts returns the non-convex interval $\langle[12,12],[14,14],[17,17]\rangle$. Finally, the dicing `Weeks:>:`$\langle[12,12],[14,14],[17,17]\rangle$ generates the order 1 collection of all the weeks that start after January 17-th, i.e., $\{\langle[19,23]\rangle, \langle[26,30]\rangle, \ldots\}$. □

We state a formal property of the proposed extension.

Theorem 5 *The extended collection formalism can represent all and only the granularities which are either periodical or finite non-periodical.*

To support this result, the algorithm used in the proof of Theorem 3 has been extended to consider non-convex intervals. The granularities captured by the proposed extension are shown in Figure 4.

5. Conclusions

In this paper we have considered a recently proposed theoretical framework for time granularities and we have analyzed two of the most influential proposals for calendar symbolic representation. On one side, we have shown that the theoretical framework is general enough to capture all the sets of disjoint intervals representable by those formalisms. On the other side we have shown exactly which subclass of granularities can be represented by each formalism. From this

[9] This set of relations is similar to the one chosen in [11] for convex intervals. We consider it only as a good basic set which allows the representation of most common granularities while having a simple implementation. It can be extended to a richer set considering, for example, the taxonomy of relations given in [10].

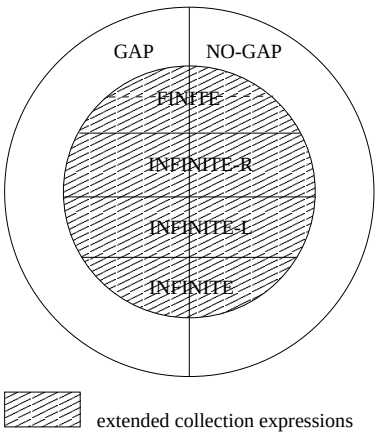

Figure 4. The subset of granularities captured by the proposed extension

analysis, we have proposed an extension of the *collection formalism* which captures a well-defined and large class of granularities, providing a good coverage of granularities that may be found in database and temporal reasoning applications.

We are currently working at the definition and implementation of set operations, performed at the symbolic level, among extended collection expressions. This problem has interesting applications (see e.g., [2]) but it is not addressed in [11] and derivative work for collections, and only briefly investigated in [14] for slices.

References

[1] James F. Allen, Maintaining Knowledge about Temporal Intervals, *Communications of the ACM*, 26(11):832–843, 1983.

[2] Elisa Bertino, Claudio Bettini, Elena Ferrari and Pierangela Samarati, An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning, *ACM Transactions on Database Systems*, 23(3), 1998.

[3] C. Bettini, C.E. Dyreson, W.S. Evans, R.T. Snodgrass, X.S. Wang, A glossary of time granularity concepts, in book: Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (Eds.), LNCS State-of-the-art Survey 1399, pp. 406-413, Springer, 1998.

[4] C. Bettini, X. Wang, and S. Jajodia, A General Framework for Time Granularity and its Application to Temporal Reasoning, *Annals of Mathematics and Artificial Intelligence*, 22(1,2):29–58, Baltzer Science Publ., 1998. A preliminary version appeared in Proc. of TIME96.

[5] F. Casati, B. Pernici, G. Pozzi, G. Sanchez, J. Vonk, Conceptual workflow model, in book *Database support for workflow management: the WIDE project*, chap.3, Kluwer, in press.

[6] D. Cukierman, J. Delgrande, Expressing Time Intervals and Repetition within a Formalization of Calendars, *Computational Intelligence*, 14(4):563–597, 1998.

[7] I.A.Goralwalla, Y.Leontiev, M.T.Ozsu, D.Szafron, C.Combi, Temporal Granularity for Unanchored Temporal Data, in *Proc. of CIKM*, pp. 414-423, ACM Press, 1998.

[8] R. Chandra, A. Segev, and M. Stonebraker, Implementing calendars and temporal rules in next generation databases, in *Proc. of ICDE*, pp. 264–273, 1994.

[9] J. Koomen, Reasoning about recurrence, *Int. Journal of Intelligent Systems*, 6:461–496, 1991.

[10] P. Ladkin, Time representation: a taxonomy of interval relations, in *Proc. of AAAI Int. Conf.*, pp. 354–359, 1986.

[11] B. Leban, D. Mcdonald, and D. Foster, A representation for collections of temporal intervals, in *Proc. of AAAI*, pp. 367–371, 1986.

[12] A. Montanari. Metric and Layered Temporal Logic for Time Granularity, ILLC Dissertation Series 1996-02, Institute for Logic, Language and Computation, University of Amsterdam, 1996.

[13] R. Morris, W. Shoaff, and L. Khatib, Domain-independent temporal reasoning with recurring events, *Computational Intelligence*, 12(3):450–477, 1996.

[14] M. Niezette and J. Stevenne, An efficient symbolic representation of periodic time, in *Proc. of CIKM*, Baltimore, pp. 161–168, 1992.

[15] P. Terenziani, Reasoning About Periodic Events, in *Proc. of 2nd International Workshop on Temporal Representation and Reasoning (TIME)*, Melbourne Beach, Florida, 1995.

[16] *The TSQL2 Temporal Query Language*, R. T. Snodgrass ed., Kluwer Academic Pub., 1995.

A Syntax

In the following, we summarize the syntax of the symbolic formalism. Note that C is the basic calendar in the considered system, and text variables (e.g., `day`, `month`, `...`) are often used in symbolic expressions to denote the calendars (PC) generated from C.

E	:==	⟨interval-list⟩ | PC | E_1 :rel: E_2 | E_1 .rel. E_2 | sel/E
PC	:==	$Generate(sp, C, P, X)$
rel	:==	$starts$ | $during$ | $intersects$ | $meets$ | $<$ | $>$
interval-list	:==	interval | interval,interval-list
interval	:==	[integer,integer]
sel	:==	integer | [integer ... integer] | {integer-list}
integer-list	:==	integer | integer,integer-list

B Proofs

Proof of Theorem 1.
Let S be $\sum_{i=1}^{n} O_i.C_i \triangleright h.C_d$ and C be the basic calendar. We first consider an infinite slice ($O_1 = all$), and we illustrate an algorithm to derive the points $\{r_1, \ldots, r_n\}$, where each r_i represents the starting point of the i-th interval denoted by S in terms of C.

The algorithm works as follows: As the first step we add "$+1.C$" as the last element in the sum describing the slice S. (This is a technical trick to obtain the starting points in terms of C.) The main loop considers each granule of C_1 within a slice period, starting from $C_1(1)$. This avoids checking an infinite number of granules. Then, we consider each one of the possible combinations of the offsets, and we compute the index in terms of C of the single granule denoted by the corresponding slice.

```
INPUT: ∑_{i=1}^{n} O_i.C_i, Period;
OUTPUT: {r_1, ..., r_n};
METHOD:
j := 1;
for x := 1 to ⌈Period⌉^{C_1} do
begin
 k := x;
 for every element in O_2 do
  ⋮
  for every element in O_n do
   begin
    indef := 0;
    for each C_i (i = 1 to n − 1) do
     begin
      k := find_index(C_i, k, o^{i+1}, C_{i+1});
      if (k = indefinite) then
       begin
        indef := 1;
        break;
       end;
     end;
    if (indef = 0) then
     begin
      r[j] := k;
      j := j + 1;
     end;
   end;
end;
```

The function $find\_index()$ receives as input two calendars (C_i and C_{i+1}) and two integer numbers (k and o^{i+1}) and it computes the index of the o^{i+1}-th interval of C_{i+1} which comes after the starting point of the k-th interval of C_i.
The function $\lceil z \rceil_{C_1}^{C_2}$ returns the index z' of the granule of C_2 which contains the z-th granule of C_1 if one exists, and `indefinite` otherwise.
Each point r_i identified by the algorithm is the beginning of an interval with duration $h.C_d$. However, in order to derive intervals in terms of C we need to translate this duration in terms of C. If r_i is contained in the m-th granule of C_d, then the new duration H_i is equal to $r_i' - r_i$ where r_i' is the maximum index of a granule of C contained in the $(m + h)$-th granule of C_d. This allows to derive the intervals $\{[r_1, r_1 + H_1 - 1], \ldots, [r_n, r_n + H_n - 1]\}$ denoted by the slice S in one period. The intervals in other periods are simply given by $[Period * s + r_i, Period * s + r_i + H_i - 1]$ where $1 \leq i \leq n$, $Period$ is the slice period in terms of C, and s is an integer. Then, a periodical granularity G can be defined by taking $R = n$, $P = Period$, and $G(i) = \bigcup_{x=r_i}^{r_i+H_i-1} C(x)$ for each $i = 1, \ldots, n$. It is easily seen that G is indeed a granularity, since the disjointness of the slice guarantees that granules of G do not overlap. By definition of periodical granularity, if $G(i) = \bigcup_{x=r_i}^{r_i+H_i-1} C(j_x)$ for some $j_{r_i}, \ldots, j_{r_i+H_i-1}$ and $G(i + R) \neq \emptyset$, then $G(i + R) = \bigcup_{x=r_i}^{r_i+H_i-1} C(j_x + P)$.
Suppose now, by contradiction, that there exists t within an interval denoted by S but t is not the index of a granule of C contained in any granule of G, i.e., $t \in [P * s + r_{\bar{z}}, P * s + r_{\bar{z}} + H_{\bar{z}} - 1]$, for some integer s, and $\bar{z} \in \{1, \ldots, n\}$ but $C(t)$ is not included in $G(i)$ for any i. Then, since S is periodical, $(t - P * s) \in [r_{\bar{z}}, r_{\bar{z}} + H_{\bar{z}} - 1]$, and by construction of granularity G, $G(\bar{z}) = C(r_{\bar{z}}) \cup \ldots \cup C(r_{\bar{z}} + H_{\bar{z}} - 1)$. Then, by the periodicity of G, $G(s * n + \bar{z}) = C(P *$

$s + r_{\bar{z}}) \cup \ldots \cup C(P * s + r_{\bar{z}} + H_{\bar{z}} - 1)$. Hence $C(t)$ is contained in $G(s * n + \bar{z})$ contradicting the hypothesis. We can conclude that $\forall z \in \{1, \ldots, n\} \forall s \exists i$ s.t. $\forall t (t \in [P * s + r_z, P * s + r_z + H_z - 1] \Rightarrow C(t) \subseteq G(i))$. On the other side, suppose there exists t s.t. $C(t)$ is contained in $G(\bar{z})$, for some integer $\bar{z}$, but $t \notin [P * s + r_i, P * s + r_i + H_i - 1]$ for any integer s and $1 \leq i \leq R$. Since G is periodical, there must exist an integer $\bar{s}$ such that $C(t + P * \bar{s})$ is contained in $G(z)$ where $1 \leq z \leq R$. By construction of granularity G, if $G(z) = C(r_z) \cup \ldots \cup C(r_z + H_z - 1)$, then the interval $[r_z, r_z + H_z - 1]$ is one of those denoted by S. Hence, $t + P * \bar{s}$ belongs to an interval denoted by S. However, since S is periodical with period P, also t must be in one of these intervals. this contradicts the hypothesis. Hence, $\forall i \exists z \in \{1, \ldots, n\} \exists s$ s.t. $\forall t\ (C(t) \subseteq G(i) \Rightarrow t \in [P * s + r_z, P * s + r_z + H_z - 1])$.

When S is finite, the same algorithm can be easily adapted to derive all the intervals S denotes. Then, the equivalent granularity is simply defined explicitly mapping each granule to one of these intervals. Disjointness of the slice S ensures that the result of this mapping is indeed a granularity. In the finite case the equivalence between G and S is trivial.

Proof of Theorem 2.
Let G be a no-gap infinite periodical granularity. By Definition 3, G is periodical wrt the basic granularity which we identify with the basic calendar C. Then, G has a period P in terms of C, R granules within each period, and an explicit description of these granules within a period by sets $S_0, \ldots, S_{R-1}$ of indexes of C. Since the granularity has no gaps, each set can be represented by an interval $[a_z, b_z]$ with $z = 0 \ldots R - 1$. Let Cal be the calendar generated from C including these intervals as well as the largest intervals that can be added to these ones in order to cover the whole time line. Formally, $Cal = Generate(a_0, C, b_0 - a_0 + 1, a_1 - b_0 - 1, \ldots, b_k - a_k + 1, a_{k+1} - b_k - 1, \ldots, b_{R-1} - a_{R-1} + 1, a_0 + P - b_{R-1} - 1)$, where the calendar constructor $Generate()$ was defined in Section 3.1. In this case a_0 is used as a synchronization point, and P is the period of G. (If one of the values in the $Generate()$ expression evaluates to 0 it is ignored.) Let Cal' be a calendar such that each interval starts where a granule of G starts: $Cal' = Generate(a_0, C, a_1 - a_0, \ldots, a_{k+1} - a_k, \ldots, a_0 + P - a_{R-1})$.
We now show that G is equivalent to the slice expression $all.Cal' \triangleright 1.Cal$. First, if $G(i) = C(a) \cup \ldots \cup C(b)$, then $[a, b]$ must be an interval in $all.Cal' \triangleright 1.Cal$. By construction, Cal' contains an interval starting on a. Indeed, this is trivial if $1 \leq i \leq R$, otherwise, by the periodicity of G we know that there exists $G(z)$ containing as first granule $C(a + P * \lfloor i/R \rfloor)$ with $z = (i \text{ MOD } R) + 1$. Then, by construction, $a + P * \lfloor i/R \rfloor$ is the starting point of an interval in Cal' and, since Cal' and G have the same period P, a must also be a starting point of an interval of Cal'. Moreover, by construction of Cal, $[a, b]$ is also one of the intervals in Cal, since Cal contains all intervals denoted by G plus the newly inserted ones. Then, according to the the slice semantics, Cal' gives the starting points and Cal the duration, implying that $[a, b]$ is among the intervals denoted by the slice.

It is easily seen, by the construction of Cal and Cal', that the vice-versa also holds, i.e., for each interval [r,s] denoted by the slice, there exists i such that $G(i) = C(r) \cup \ldots \cup C(s)$.

If the granularity is finite (periodical or not), Cal and Cal' are generated explicitly listing all the intervals. The equivalence proof is analogous.

Proof of Proposition 1.
The proposition follows from the semantics of the operators. Indeed, the expression $E.rel.E'$, apart from possibly structuring the intervals into a collection of higher order, only selects some of the intervals in E. Hence, for each interval in E' the collection we derive is a subset of that in E. Since E has only disjoint intervals, its subsets are also disjoint. However, there may be some interval u in E satisfying relation $.rel.$ with more than one interval in E'. This leads to the multiple presence of u in the resulting collection. Strict operators (: rel :) select common subintervals of intervals in E and E', and there is no way of generating overlapping intervals if there are none in E and E'. The other type of operators (slicing) simply select some intervals from a collection E based on their index. Since we assumed that collection expression are built from calendars recursively applying only slicing and dicing operators, and no calendar has overlapping intervals, we can conclude that the derived collections only contain equal or disjoint intervals.

Proof of Theorem 3.
The proof is based on an algorithm to compute the intervals denoted by a collection expression within its period if it is infinite, or all the intervals if it is finite. Clearly, the algorithm is trivial when the collection is explicitly given as a set of intervals; Otherwise, for each *dicing* and *slicing* operator allowed in the expression, the algorithm has to generate the intervals resulting from its application. The algorithm represents each infinite collection of order 1 by a finite set of intervals

within one of the periods, by the period value, and possibly by lower and upper bounds identified by values min and max. The choice of the beginning of the period is induced either by the calendar definition (if the order-1 collection is a defined calendar) or by the specific operations that have been applied. The algorithm evaluates the collection expression from right to left. When the algorithm applies *dicing* to a collection, its order is increased by 1, while if it applies *dicing* to an interval it becomes a collection of order 1. On the other side, if the algorithm applies *slicing* (with a single slicing value) to a collection, its order is decreased by 1, resulting in a single interval if the collection had order 1. If the algorithm applies "$<$" or "$>$" dicing to a finite collection expression it returns a lower or upper bounded collection expression, while for other operators a finite collection is returned. If the algorithm generates a bounded collection expression, it stores the minimum value min or the maximum max bounding the collection. At the end of the algorithm, if we have $P = undefined$, then the collection expression is finite, otherwise the collection is periodic. If min is defined, then it is lower bounded and if max is defined, it is upper bounded.

The algorithm to calculate the intervals identified by a collection expression uses a set of procedures implementing each slicing and dicing operator. Here, for the sake of brevity, we only illustrate the case of the procedure that implements the $.overlaps.$ operator. Let $E = X.overlaps.Y$ be the collection expression identifying the intervals of X overlapping those of Y, where X and Y are collection expressions themselves. Assume that the intervals denoted by Y are $[c_1, d_1], \ldots, [c_n, d_n]$. Then, for each interval $[c_i, d_i]$ with $i = 1 \ldots n$, we execute the following code, where G_X denotes the granularity identifying the same granules (intervals) as the collection X, C is the basic calendar, and $\lfloor a \rfloor_C^{G_X}$ is a function returning the indexes of granules of C (an interval) forming the a-th granule of G_X.

```
INPUT: [c_i, d_i];
OUTPUT: {[l_1, r_1], ..., [l_m, r_m]}; # all the
intervals of X which overlap [c_i, d_i]
METHOD:
j := 1;
a_j := ⌈c_i⌉_C^{G_X} ;
[l_j, r_j] := ⌊a_j⌋_C^{G_X} ;
while (r_j + 1 ≤ d_i) do
begin
j := j + 1;
a_j := ⌈r_{j-1} + 1⌉_C^{G_X};
[l_j, r_j] := ⌊a_j⌋_C^{G_X};
end;
```

Details of the algorithm are reported in the extended version of this paper.

Once we have derived the intervals denoted by a collection expression, we still need to find an equivalent granularity. We first consider an arbitrary collection expression E denoting an infinite number of intervals. Suppose that, by the global algorithm described above, we obtain the set of intervals $\{[b_1, e_1], \ldots, [b_n, e_n]\}$ identified by E within one of the collection's periods, the period P and, if they exist, min and max. Then, we can define the periodical granularity G as having period P equal to that of the collection, R equal to n (the number of intervals in the period), and $G(i) = C(b_i) \cup \ldots \cup C(e_i)$ for $i = 1 \ldots n$. If the collection is bounded by a minimum value min and/or a maximum one max in terms of C, then we impose $G(i) = \emptyset$ for each $G(i)$ that would contain granules of C with indexes lower than min or greater than max. This can be checked by the formula given in Section 2 to derive granules of periodical granularities. If E is finite not periodical, we can similarly apply the algorithm, this time deriving all the intervals denoted by E. Then, each interval defines explicitly a granule of G. It is easily seen that, if the algorithm computing the intervals denoted by E is correct, the above construction leads to a granularity equivalent to E. While this is trivial in the finite non-periodical case, the periodical case must be considered in detail. Each granule is the union of a set of granules of C specified by one of the intervals denoted by E, since they have been explicitly defined this way within one period, and the period length has been taken equal for E and G. We show that for each interval there is a corresponding granule. Suppose $[b, e]$ is an interval denoted by E, and there is no j such that $G(j) = C(b) \cup \ldots \cup C(e)$. E must denote more intervals than those contained in a period, and $[b, e]$ must not have been in the period considered by the algorithm, otherwise $G(j)$ would have been explicitly defined that way. Then, there must be an interval $[b_k, e_k]$ among the R intervals returned by the algorithm, and an integer s such that $b = s*P+b_k$ and $e = s*P+e_k$, where P is the period. Then, by the property of G being periodical wrt C, we know that for each j, $G(j) = \bigcup_{i \in S_{j \bmod R}} C(P * \lfloor j/R \rfloor + i)$ unless $G(j)$ has been explicitly defined as empty. Consider $j = s*R+k$. Either $G(j) = \bigcup_{i \in \{b_k, \ldots, e_k\}} C(P * s + i)$, or $G(j) = \emptyset$. The first would contradict the assumption that there is no j such that $G(j) = C(b) \cup \ldots \cup C(e)$. The second, by the construction of G, can only be true if $G(j)$, as given by the formula above, would contain

$C(i)$ with $i < min$ or $i > max$. Both cases lead to a contradiction, since $b \leq i \leq e$, and we assumed $[b, e]$ to be one of the intervals denoted by E, that is $min < b \leq e < max$. Hence, we can conclude that G is equivalent to E.

Proof of Theorem 4.
Let G be the granularity considered in the theorem. Following the first part of the proof of Theorem 2, we construct the calendars Cal and Cal'. If G is finite non-periodical or INFINITE periodical, it is easily seen that the collection expression $Cal.starts.Cal'$ is equivalent to G, where $.starts.$ is defined on generic intervals as follows: $[x, y].starts.[v.w]$ if $x = v$ and $y \leq w$. A slightly more elaborate construction makes use of the $. > meets.$ relation defined in [11] instead of $.starts.$. FINITE, INFINITE-R, and INFINITE-L periodical granularities require an additional term in their equivalent collection expression. If the definition of G specifies a first non-empty granule $G(i) = [m, n]$, then we define E as $(Cal.starts.Cal'). > .(m-1)/C$. Similarly, for a last non-empty granule $G(i) = [M, N]$, using $. < .(N+1)/C$. If the definition of G specifies a first and a last non-empty granule $G(min) = [m, n]$ and $G(max) = [M, N]$, then we define E as $(PC.starts.PC') : during : [m..N]/C$.

Proof of Theorem 5.
Introducing non-convex intervals as granules, Theorems 3 and 4 are extended with analogous proof techniques to show that the gap counterpart of all the periodical granularities is captured by the extended formalism. Regarding the extension of Theorem 3, the only relevant part that needs to be modified is the algorithm. In particular, the procedures implementing dicing operators must follow the new semantics when applied to non-convex intervals. Details of the algorithm are reported in the extended version of this paper. Depending on the expression being periodical or not, all the non-convex intervals or only those in a period are derived, each one defining a granule of the gap granularity G. Then G is easily shown to be equivalent to the extended collection expression. Let us consider now the extension of Theorem 4. The finite non-periodical case is trivial, hence we consider here the periodcal case. Let G be a gap granularity with a period P, R granules (with gaps) within each period described by sets $S_0, \ldots, S_{R-1}$ of indexes of the basic granularity C. Each set can be represented by a non-convex interval y_z with $z = 0 \ldots R-1$. If G has no first nor last granule, then we can define a primitive collection PC equivalent to G. Let y_z above be $\langle [a_0^z, b_0^z], \ldots, [a_{m_z}^z, b_{m_z}^z] \rangle$. Then, PC is given by $Generate(a_0^0, C, P, X)$, where X contains R non-convex intervals $x_0, \ldots, x_{R-1}$, where each interval in x_z is given by $[a_j^z - a_0^0 + 1, b_0^z - a_0^0 + 1]$ for each $j = 0 \ldots m_z$ and $z = 0 \ldots R-1$. Essentially, we just transformed the granularity description in the equivalent notation for primitive collections. Similarly to Theorem 4 FINITE, INFINITE-R, and INFINITE-L periodical gap-granularities are accommodated by using the operators ". > .", ". < .", and ": $during$:".