Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Spalto Marengo 33, 15100 Alessandria
http://www.di.unipmn.it

Orthogonal operators for user-defined symbolic periodicities

Author: Lavinia Egidi (lavinia.egidi@unipmn.it),
Paolo Terenziani (paolo.terenziani@unipmn.it)

TECHNICAL REPORT TR-INF-2004-04-06-UNIPMN
(April 2004)

The University of Piemonte Orientale Department of Computer Science Research Technical Reports are available via WWW at URL `http://www.di.mfn.unipmn.it/`.
Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

2004-05 RHENE: A Case Retrieval System for Hemodialysis Cases with Dynamically Monitored Parameters, Montani, S., Portinale, L., Bellazzi, R., Leonardi, G., March 2004.

2004-04 Dynamic Bayesian Networks for Modeling Advanced Fault Tree Features in Dependability Analysis, Montani, S., Portinale, L., Bobbio, A., March 2004.

2004-03 Two space saving tricks for linear time LCP computation, Manzini, G., February 2004.

2004-01 Grid Scheduling and Economic Models, Canonico, M., January 2004.

2003-08 Multi-modal Diagnosis Combining Case-Based and Model Based Reasoning: a Formal and Experimental Analisys, Portinale, L., Torasso, P., Magro, D., December 2003.

2003-07 Fault Tolerance in Grid Environment, Canonico, M., December 2003.

2003-06 Development of a Dynamic Fault Tree Solver based on Coloured Petri Nets and graphically interfaced with DrawNET, Codetta Raiteri, D., October 2003.

2003-05 Interactive Video Streaming Applications over IP Networks: An Adaptive Approach, Furini, M., Roccetti, M., July 2003.

2003-04 Audio-Text Synchronization inside mp3 file: A new approach and its implementation, Furini, M., Alboresi, L., July 2003.

2003-03 A simple and fast DNA compressor, Manzini, G., Rastero, M., April 2003.

2003-02 Engineering a Lightweight Suffix Array Construction Algorithm, Manzini, G., Ferragina, P., February 2003.

2003-01 Ad Hoc Networks: A Protocol for Supporting QoS Applications, Donatiello, L., Furini, M., January 2003.

2002-06 Stochastic modeling, analysis techniques and tools for dependable reactive systems, Codetta Raiteri, D., Bobbio, A., October 2002.

2002-05 Stochastic modeling, analysis techniques and tool for dependable reactive systems, Bernardi, S., Gribaudo, M., Bobbio, A., October 2002.

2002-04 Interactive MPEG video streaming over IP-Networks: a performance report, Furini, M., Roccetti, M., September 2002.

2002-03 A fuzzy approach to case-based reasoning through fuzzy extension of SQL, Portinale, L., Montani, S., Bellazzi, R., July 2002.

2002-02 From FPN to NuSMV: The temperature control system of the ICARO cogenerative plant, Horvath, A., Gribaudo, M., Bobbio, A., February 2002.

Orthogonal operators for user-defined symbolic periodicities

Lavinia Egidi
Dipartimento di Informatica
Università del Piemonte Orientale
Spalto Marengo, 33
15100 Alessandria Italy

Paolo Terenziani
Dipartimento di Informatica
Università del Piemonte Orientale
Spalto Marengo, 33
15100 Alessandria Italy

Abstract

The treatment of user-defined calendars and periodicities is attracting an increasing attention within the AI and the DB fields. In this paper, we identify a set of orthogonal properties characterizing periodicities; based on these we define a lattice of classes (of periodicities). For each property, we introduce a language operator and, this way, we propose a family of symbolic languages (each one corresponding to a *subset of operators*), one for each point in the lattice. Therefore, the expressiveness and meaning of each operator, and thus of each language in the family, are clearly defined, and a user can select the language that *exactly* covers the properties of her domain. To the best of our knowledge, our language covering the top of the lattice (i.e., all of the properties) is more expressive than any other symbolic language in the AI and DB literature.

1 Introduction

Calendars and periodicity play a fundamental role in human modeling of the world, and are extremely relevant in many applications, spanning from financial trading to scheduling, from manufacturing and process control to office automation and data broadcasting. In most practical application, supporting a "standard" calendar (i.e., the Gregorian calendric system) does not suffice, since *user-defined* periodicities need to be used (see, e.g., [13]). Thus, many different AI and DB approaches have been devised to deal with periodicity and its related notions of *granularity* and *calendric systems* [2, 13] (see, e.g., the survey in [15]). Besides *logical* approaches ([4] and approaches in classical temporal logics) and *mathematical* ones (e.g., Kabanza's *linear repeating points* [9]), *symbolic* approaches to user-defined periodicities are gaining an increasing relevance (consider, e.g., [11, 12, 3, 14]). The latter provide a set of *operators* which can be combined in order to define complex periodicities in an *incremental* and *compositional* way. For instance, in [11], "the first day of each month" can be modeled by first applying a basic operator in order to define "day" and "month", then applying a *dicing* operators, to split months into sets of days, and finally a *slicing* operator, to select the first day in each set.

In this paper, we propose a *family* of new symbolic languages, that we designed according to the following intuitions:

1. the expressiveness of (symbolic) languages for periodicities can be charaterized on the basis of a set of *orthogonal properties*;
2. the less the operators are interdependent, the clearer is the underlying semantics of (symbolic) language operators: operators should correspond as much as possible to specific properties, without affecting each other;
3. there is no "best language" per se: the goodness of a language depends on the domain of application; the most suitable language is the one that meets more precisely the expressiveness requirements.

With this in mind,

- we identified (Section 2) five orthogonal properties that characterize user-defined periodicity. The definitional process was driven by considerations on the significance and usefulness of the structures that must be allowed, on homogeneity arguments and on orthogonality requirements (cf. Item 1 and 2 above);
- we defined, for any subset of these properties, a class of periodicities. The classes, ordered by set inclusion, form a lattice (Section 2). In [6] this lattice was used in order to classify different (symbolic) languages in the literature;
- we defined two basic (symbolic language) operators to express periodicities for which none of the five properties hold; and five additional operators, one for each property (Section 3). Any subset of the five operators, together with the basic ones, defines a language. Languages, ordered by set inclusion, define a lattice;
- we proved that the lattice of languages matches the lattice of periodicities, in that each node of the former has enough expressive power in order to define *exactly* the omologous node in the latter (Section 4).

From the point of view of end-users, our symbolic approach makes, on the one hand, the task of choosing a priori a language easier, since the choice only depends on the properties of interest in her domain/application (see Issue 3 above). On the other hand, at a later stage, when the user needs to define a new periodicity, the choice of the operators required is easier because it depends on the properties of that periodicity.

To the best of our knowledge, our top language (i.e., the one that can deal with all of the properties) is more expressive than any other symbolic language in the AI and DB literature.

Moreover, the modularity of our approach can be exploited by language designers and implementers, since in order to operate on the whole family of languages, one can simply focus on the seven operators. For instance:

- the semantics of all the languages in the family can be provided on the basis of the semantics of the seven operators. In [7], a semantics in terms of Presburger Arithmetic is given;
- interpreters/compilers for each one of the languages can be built in a modular way.

In Section 5 we briefly compare our approach to others in the literature.

2 Properties

Intuitively, *periodicities* are events that repeat in a regular way in time. As in many approaches in the literature, we choose to focus only on the *temporal extent* of periodic events, disregarding other properties (e.g., agent, location). We adopt *discrete*, *linearly ordered* and *unbounded* time; *time points* are the basic temporal primitives. The basic structures on time points are *time intervals* i.e., non empty sets of points. A *convex* time interval is the set of all points between two endpoints.

We take *calendars* as the basis of our construction, as in most symbolic approaches in the literature (e.g., [11, 12, 3, 14]). Calendars model the discrete time axis, partitioning it into a regularly repeating pattern of adjacent intervals. In particular, a *basic calendar* (also called *basic granularity*, or *chronon*) defines the tick of the given (temporal) system. Examples of calendars are "minutes", "days", "months". We call *Cal* the set of all calendars.

Approaches in the literature witness that the need for more complex structures has been felt. An analysis of the various proposals led us to identify five crucial properties. We propose here definitions filtered through our intuition and our demand for orthogonality.

Operators to *select* some portions of the time line, dropping others, have been provided in, e.g., [11, 12, 3, 14]. The resulting periodicities have the **non-adjacency** property. We give a preliminary definition of the property, that captures the above:

NA - (prelim) *A periodicity P has the* non-adjacency *property, if it has no interval I, that is not P's rightmost interval, such that for no $J \in P$, $Meet(I, J)$, where $Meet([s_1, e_1], [s_2, e_2]) \leftrightarrow s_2 = e_1 + 1$ (see [1]).*

For instance, "Mondays" has the non-adjacency property, as well as (ex.1) "Mondays from 8 to 18".

In some cases, one may want to capture the fact that a periodic event occurs over time intervals with gaps (i.e., holes) in them (see, e.g. [3]). For instance, (ex.2) "Working hours on Mondays, from 8 to 12 and from 14 to 18", may be represented using a gap interval for each Monday.

G - *A periodicity P has* Gaps *if it contains gap intervals.*

Notice that the property of having gaps is orthogonal with respect to non-adjacency, since the former is about inner holes in intervals, and the latter about outer gaps between intervals. For instance, (ex.2) has both the NA and the G property, while (ex.1) has only NA, and "each day, with the lunch-break gap (from 12 to 14)" has only G.

Intervals may overlap [5] (exact overlaps, i.e., multiple occurrences of the same interval, are special cases of overlaps). For instance, (ex.3) "Tom's and Mary's working hours on Mondays" may be a case of (exact) overlap—consider e.g., the same periodicity (ex.2) for both Tom and Mary.

In order to disambiguate the relationship between gaps and overlaps, we introduce the notion of time span. We define the *time span* of an interval (or sets thereof) as the interval whose endpoints are the minimum and maximum points belonging to the interval (or sets thereof).

O - *A periodicity P has* Overlaps *if the time-spans of some of the intervals have non-empty intersection.*

So, O is a property of interval time spans, and therefore totally independent from G.

In order to preserve our intuition of non-adjacency, and to keep O and NA orthogonal, we settle on a more articulate version of NA-(prelim):

NA - *A periodicity has the* non-adjacency *property if it can't be split in a finite number of periodicities that don't satisfy NA-(prelim).*

Other approaches pointed out the importance of dealing with *bounded* periodicities (consider, e.g., [2]) or with periodicities consisting of a finite aperiodic set of intervals plus a periodic part [10]. We introduce the property of being *eventually periodic* (terminology taken from [8]) to model both (notice that a bounded periodicity can be seen as a degenerate eventually periodic one with an empty periodic part).

EP - *A periodicity P is* Eventually Periodic *if it can't be expressed giving a repeating pattern and a positive period.*

Consider, e.g., (ex.4) "Tuesday January 13, 2004 plus Tom's and Mary's working hours on Mondays (defined as in ex.3), starting from January 14, 2004".

EP does not interact with G and NA, since EP is a property of the sequence of intervals as a whole, whereas the other two are local properties. Also, EP and O are orthogonal, since the periodic and/or aperiodic parts may have overlaps or not.

Finally, Leban et al. [11] pointed out the importance of grouping intervals into structured sets. For instance, the periodicity in (ex.4) may be structured by grouping intervals into sets, one for each month. Nested groupings (e.g., into months and years) are also allowed.

To cope with structure, *order−n* collections are introduced by Leban et al:

Definition 1 (Order-n collections) *For each $n > 0$, an* order-n collection *is a multiset of order-$(n-1)$ collections. A time interval is an* order-0 collection.

It is customary to use the set notation (with braces) for order-n collections with $n > 0$, whereas intervals are represented using square brackets even though the time axis is discrete.

S - *A periodicity P has* Structure *if it is an* order-n collection, *with $n > 1$.*

Thus, the extent of non-structured periodicities is an *order-1 collection*, while *order-n collections* are needed for structured periodicities. S is trivially orthogonal to NA, G, O and EP, since it operates at a different level.

So far, we defined five orthogonal properties of periodicities. Even though the definitions are tailored in order to meet user needs, and obtain only meaningful structures, we must place further constraints on the combinations of the five properties that we wish to allow, based on homogeneity observations.

Remark (Homogeneity) In order-n collections, braces are not annotated, i.e. no semantics can be attached to them.

Therefore, to preserve a clean semantics, we only admit *homogeneous* structures, i.e. structures *in which all data has been grouped according to the same calendars.*

A consequence of this choice is that *overlaps between order-n collections ($n > 0$) are not possible.*

3 Orthogonal operators

We define two basic operators, that can be applied only initially and never after the other operators:

- **Basic_Granularity:**
  No input;
  Output: an order-1 collection right and left infinite, containing adjacent one-element no-gap intervals.
- **Cal_Def:**
  Input: a calendar C in *Cal*, an 'anchor' time point p which is the start point of an interval in C, and a list of positive natural numbers $n_1, \ldots n_k$;
  Output: a calendar, built by grouping cyclically $n_1, \ldots n_k$ intervals of C starting from p (both directions), i.e. the union over $\mathbb{N}$ of the intervals I_h defined as:

$$\bigcup_{\substack{j=1,\ldots,k \\ s \in \mathbb{N}}} \left\{ I_s | l(h) + \sum_{i=1}^{j-1} n_i < s \leq l(h) + \sum_{i=1}^{j} n_i \right\},$$

  where $\{I_s\}$ is the set of intervals in C, ordered according to the natural ordering, $l(h) = ap + h \cdot \sum_{i=1}^{k} n_i$ and ap is the index of the interval starting at the anchoring point.

In the following we need two relational operators:

Definition 2 (Strictly During, Non-Strictly During) *Let I (resp. I') be the time span of interval J (resp. J'). I is* strictly during *I' (I* **SDur** *I') if I' contains I and the endpoints of I are both different from the endpoints of I'; I is* non strictly during *I' (I* **NSDur** *I') if I' contains I.*

Then we can introduce the five operators, paired with the five orthogonal properties. We place constraints on the inputs that they can take, based on the following remarks: (a) in the cases of *Select* and *Drop*, the structure of the second argument is explicitly disregarded, therefore we prescribe an unstructured input; (b) for homogeneity reasons (see Sect. 2) unions make sense only over order-1 collections, which reflects on the definitions of *Union* and *Replace*; (c) operators must either modify intervals or treat them as black boxes, for orthogonality (this reflect on the definitions of *Replace* and *Group_by*); (d) intuition suggests that a user first collects raw data and then organizes it logically, therefore *Group_by* can only be applied last (all other operators have unstructured inputs).

- **Select (for non adjacency)**
  Input: $N = \{n_1, ..., n_k\}$, $n_i \in \mathbb{N}$, order-1 collections C_1 and C_2;
  Output: An order-1 collection (or the empty set):

$$\bigcup_{n_i \in N, j \in C_2} \left\{ n^{th}(\{i \in C_1 | i \text{ NSDur } j\}, n_i) \right\}$$

  where $n^{th}(C, i)$ selects the i-th interval in the collection C, if it exists, it is the empty set otherwise. (See Remarks (a) and (b), above.)
- **Drop (for gaps)**
  Input: $N = \{n_1, ..., n_k\}$, $n_i \in \mathbb{N}$, order-1 collections C_1 and C_2;
  Output: An order-1 collection:

$$\bigcup_{i \in C_1} \left\{ i \setminus \bigcup_{n_i \in N, j \in C_2} n^{th}(\{j \in C_2 | j \text{ SDur } i\}, n_i) \right\}$$

  where $\setminus$ is set minus, and $n^{th}(C, i)$ selects the i-th interval in the collection C, if it exists, it is the empty set otherwise. (See Remarks (a) and (b) above.)

- **Union (for overlaps)**
  Input: two order-1 collections C_1 and C_2;
  Output: the order-1 collection

$$\bigcup_{i \in C_1} \{i\} \cup \bigcup_{j \in C_2} \{j\}.$$

  ($\cup$ and $\bigcup$ denote *multiset union*. For the constraints on the arguments, see Remark (b) above.)

- **Replace (for eventual periodicity)**
  Input: Order-1 collections C_1, C_2 and C_3, two of which can be the empty set, and time points p_1, p_2 such that $p_1 < p_2$ and if p_1 belongs to some intervals of C_1 or C_2, it is their right endpoint, and similarly for p_2 relative to C_2 and C_3;
  Output: An order-1 collection:

$$\{i \in C_1 | i \text{ NSDur}(-\infty, p_1]\} \bigcup \bigcup \{i \in C_2 | i \text{ NSDur}[p_1 + 1, p_2]\} \bigcup \bigcup \{i \in C_3 | i \text{ NSDur}[p_2 + 1, \infty)\}.$$

  (This operator corresponds to the intuition that a complex periodicity consists of bounded portions of basic periodicities. It is defined on order-1 collections—see Remark (b). For the constraints on p_i, see Remark (c).)

- **Group-by (for structure)**
  Input: An order-n collection C and a calendar C', such that for each interval $i \in C$ there is an interval $j \in C'$ such that i NSDur$j$;
  Output: An order-$(n+1)$ collection:

$$\bigcup_{j \in C'} \left\{ \bigcup_{i \in C} \{i \in C | i \text{ NSDur} TS(j)\}_C \right\},$$

  where $TS(j)$ is the time span of interval j and the subscript C indicates that the structure of C is preserved.

  (The constraint on the second input stem from orthogonality—Remark (c) above. See also Remark (d).)

We show how to define with our operators the periodicities in the examples from Section 2 (we choose *hours* as the basic granularity, and the first hour of Jan first, 2004, as the origin):

- (ex.1) $\mathit{Mon8\text{-}18} = \mathit{Select}(2, C, \mathit{days})$
  where $C = \mathit{Cal\_Def}(\mathit{Basic\_Granularity}(), 0, \langle 8, 10, 6 \rangle)$
  and $\mathit{days} = \mathit{Cal\_Def}(\mathit{Basic\_Granularity}(), 0, \langle 24 \rangle)$
- (ex.2) $WH = \mathit{Drop}(\langle 4, 5 \rangle, \mathit{Mon8\text{-}18}, \mathit{Basic\_Granularity}())$
- (ex.3) $M{+}T = \mathit{Union}(WH, WH)$
- (ex.4) $P = \mathit{Replace}(\emptyset, 288, \mathit{days}, 312, M{+}T)$,
- (ex.5) $\mathit{Group\_by}(P, \mathit{months})$, where (ignoring leap years)
  $\mathit{months} = \mathit{Cal\_Def}(\mathit{days}, 0, \langle 31, 28, 31, 30, \ldots, 31 \rangle)$.

We prove that each operator affects a single property:

Theorem 1 (Orthogonality) {Select, Drop, Union, Replace, Group-by} *is an orthogonal set of operators.*

Proof (sketch) We defined an operator introducing each property. We now show, for each property, that no other operator can introduce it.
NA. The definition of NA itself is tailored so that Union can't introduce it. The use of SDur ensures that *Drop* doesn't introduce it. The same holds for Replace, because of the constraints on p_i. The constraints

on the definition of Group-by guarantee that the latter doesn't affect the underlying sequence of intervals.
G. The only operators that affect intervals are *Cal_Def* and *Drop*. Only *Drop* can introduce gaps, since *Cal_Def* compounds adjacent intervals.
O. Select and Drop can't introduce overlaps by their nature. Replace can't because disjoint portions of periodicities are used to obtain the new one. Group-by, as noticed, doesn't affect the underlying sequence of intervals.
EP. Select, Drop and Group-by work periodically, and the period of their output is the least common multiple (lcm) of the periods of the inputs, if those were unbounded and with no aperiodic part. Similarly, Union doesn't add EP, since if periodicities C_1 and C_2 are not bounded and have no aperiodic part, $C = \text{Union}(C_1, C_2)$ is a pure unbounded periodicity with period $p = lcm(p_1, p_2)$ and repeating pattern the multiset union of $lcm(p_1, p_2)/p_2$ adjacent occurrences of RP_2 and $lcm(p_1, p_2)/p_1$ repetitions of RP_1 over the same time span—where p_i (resp. RP_i) is the period (resp. repeating pattern) of C_i.
S. By definition, Union and Replace output an order-1 collection, Select and Drop preserve the structure of one of the arguments. Therefore, only Group-by can add structure. □

4 Expressiveness

The operators we defined can be combined to obtain a family of languages. Let $\mathcal{L}$ be the basic language $\{Basic\_Granularity, Cal\_Def\}$. We prove that it defines exactly the class Cal. Adding any combination of the other operators to it, we define a lattice of languages, ordered by set inclusion. We prove that the lattice of languages matches exactly the lattice of classes of periodicities, in the sense that language $\mathcal{L} \cup \{O_{\pi_1}, \ldots, O_{\pi_k}\}$ defines all and only the periodicities in $Cal^{\pi_1 \cdots \pi_k}$ (where O_{π_i} is the operator that introduces property π_i).

The first part of this result is proven by Theorem 1. The converse (that the languages are expressive enough to define the corresponding classes) is proven by induction. In order to simplify the proof, it is convenient to fix an order in which operators are applied. This is not restrictive for our goals: in the lattice, all classes corresponing to all the combinations of properties are present, and therefore it is not relevant in which order the properties are added or removed.

A few technical lemmata prove the induction steps.

In the following, let $\mathcal{L} = \{Basic\_Granularity, Cal\_Def\}$, let $RP(C)$ be the repeating pattern of C and $p(C)$ its period.

Lemma 1 *The language $\mathcal{L}$ defines at least* Cal.

Proof. Let C be a generic calendar, whose repeating pattern consists of intervals of widths $w_1, \ldots, w_n$. Let ap be the starting point of one occurrence of the repeating pattern. Then

$$C = Cal\_Def(ap, Basic\_Granularity(), \langle w_1, \ldots, w_h \rangle).$$ □

Lemma 2 *The language $\mathcal{L} \cup \{Select\}$ defines at least Cal^{NA}.*

Proof. Let $C \in Cal^{NA}$. If $C \in Cal$, then the thesis follows from Lemma 1.

Otherwse, define C_1 so that $RP(C_1)$ is obtained adding to $RP(C)$ all convex intervals necessary to have adjacent intervals, and adjacent occurrences of the repeating pattern; C_1 is a calendar. Let C_2 be a periodicity with $RP(C_2)$ the time span of $RP(C_1)$, and $p(C_2) = p(C_1)$; C_2 is a calendar. Both C_1 and C_2 belong to Cal and are definable in $\mathcal{L}$.

Let $N = \{n_1 \ldots, n_k\}$ be the positions of the intervals of C in $RP(C_1)$. Then, $C = Select(N, C_1, C_2)$, as required. □

Lemma 3 *If the language $\mathcal{L} \cup \{O_1, \ldots, O_k\}$ defines at least Cal (resp. Cal^{NA}), then $\mathcal{L} \cup \{O_1, \ldots, O_k, Drop\}$ defines at least Cal^G (resp. $Cal^{NA,G}$).*

Proof. Let $C \in Cal^G$ (resp. $Cal^{NA,G}$). If it doesn't have property G, then the thesis follows from Lemma 1 (resp. 2).

So, let us assume that it has gap intervals. Then let C_2 be the interval closure of C, i.e. the periodicity consisting of the time spans of all intervals in C. Of course, the interval closure has the same properties as the original periodicity, except for gaps. Thus $C_2 \in Cal$ (resp. Cal^{NA}) and can be defined in $\mathcal{L} \cup \{O_1, \ldots, O_k\}$, by hypothesis.

We add gaps appropriately one interval at a time in the repeating pattern of C_2, using suitable periodicities C_1. C_1 is identical to C_2 except for the interval I that we need to modify. That one is replaced in C_1 by the subintervals that must be removed, and all the other, adjacent ones, to make sure that if C has adjacent intervals, so has C_1. N is set to $\{n_1, n_2 \ldots, n_k\}$ to identify the intervals in C_1 that must be removed from I. Then Drop(N, C_1, C_2) yields a new periodicity with the same gaps as C in all occurrences of I. The other intervals are unchanged.

With a finite number of applications of Drop as above, C is obtained. Notice that after the first application, C_1 will have (legitimately) gaps. □

Lemma 4 *If the language $\mathcal{L} \cup \{O_1, \ldots, O_k\}$ defines at least $Cal^{\pi_1,\ldots\pi_k}$, with $\{\pi_1, \ldots \pi_k\} \subseteq \{NA, G\}$, then $\mathcal{L} \cup \{O_1, \ldots, O_k$, Union$\}$ defines at least $Cal^{\pi_1,\ldots,\pi_k,O}$.*

Proof. Let C be any periodicity in $Cal^{\pi_1,\ldots,\pi_k,O}$. If it has no overlaps, it can be defined in $\mathcal{L} \cup \{O_1, \ldots, O_k\}$ by hypothesis.

Otherwise, it can be split in a finite number of periodicities with no overlaps. This introduces no gaps and no structure. It doesn't need introduce EP since if C is purely periodic, then overlaps must repeat periodically. It doesn't need to introduce NA either, by the definition of NA. Therefore we obtain a finite number of periodicities belonging to $Cal^{\pi_1,\ldots,\pi_k}$, by hypothesis definable in $\mathcal{L} \cup \{O_1, \ldots, O_k\}$. By a finite number of applications of Union we obtain C as required. □

Lemma 5 *If the language $\mathcal{L} \cup \{O_1, \ldots, O_k\}$ defines at least $Cal^{\pi_1,\ldots\pi_k}$, with $\{\pi_1, \ldots \pi_k\} \subseteq \{NA, G, O\}$, then $\mathcal{L} \cup \{O_1, \ldots, O_k$, Replace$\}$ defines at least $Cal^{\pi_1,\ldots,\pi_k,EP}$.*

Proof. Let $C \in Cal^{\pi_1,\ldots,\pi_k,EP}$. If it doesn't have the property EP, it can be defined in $\mathcal{L} \cup \{O_1, \ldots, O_k\}$ by hypothesis.

Otherwise, in the most general case, split it in the obvious two infinite periodicities and the finite one. Define C_1 by removing the right bound to the left infinite periodicity obtained (call that bound p_1), and C_3 similarly (defining p_2 as the left bound that has been removed); define C_2 by extending the aperiodic part A to a periodicity with $RP(C_2) = A$ and $p(C_2)$ the time span of A. (If one of the three parts is empty, so is the resulting C_i.) This extensions don't introduce G, nor NA, nor O, but simply reproduce them periodically if they appear in C.

Therefore $C_i \in Cal^{\pi_1,\ldots,\pi_k}$, and by hypothesis are definable in $\mathcal{L} \cup \{O_1, \ldots, O_k\}$.
And $C =$ Replace$(C_1, n_1, C_2, n_2, C_3)$. □

Lemma 6 *If the language $\mathcal{L} \cup \{O_1, \ldots, O_k\}$ defines at least $Cal^{\pi_1,\ldots\pi_k}$, with $\{\pi_1, \ldots \pi_k\} \subseteq \{NA, G, O, EP\}$, then $\mathcal{L} \cup \{O_1, \ldots, O_k$, Group-by$\}$ defines at least $Cal^{\pi_1,\ldots,\pi_k,S}$.*

Proof. Let $C \in Cal^{\pi_1,\ldots,\pi_k,S}$. If it has no structure, it can be defined in $\mathcal{L} \cup \{O_1, \ldots, O_k\}$ by hypothesis.

So, let C have structure. First assume that it is an order-2 collection. Remove its order-1 braces (i.e. the inner braces), transforming it to an order-1 collection C_1. Define C_2 as a periodicity whose intervals are the time spans of the order-1 subcollections of C. Notice that C_2 has no EP, by the definition of S. It has no gaps, no overlaps, and no structure by definition, and has NA only if C has the NA. Therefore $C_1, C_2 \in Cal^{\pi_1,\ldots,\pi_k}$ and can be defined in $\mathcal{L} \cup \{O_1, \ldots, O_k\}$. Moreover, $C =$ Group-by(C_1, C_2).

If C is an order-n collection, the same construction can be applied $n-1$ times, adding braces from the inside out, to the order-1 collection obtained removing all structure from C. □

Let O_{π_k} the operator introducing property π_k (thus O_{NA}=*Select*, O_G=*Drop*, etc.). The lemmata above, together with Theorem 1 prove that

Theorem 2 *The language $\mathcal{L} \cup \{O_{\pi_1}, \ldots, O_{\pi_k}\}$ defines exactly the class $Cal^{\pi_1 \cdots \pi_k}$.*

Proof. By Theorem 1, $\mathcal{L} \cup \{O_{\pi_1}, \ldots, O_{\pi_k}\}$ defines at most the class $Cal^{\pi_1 \cdots \pi_k}$.

The converse is proven by induction on k. For $k = 0$, the basis of the induction is Lemma 1. For positive k, the inductive step is proven as follows: if $S \in \{\pi_1 \ldots \pi_k\}$, by Lemma 6; otherwise, if $EP \in \{\pi_1 \ldots \pi_k\}$ (but not S), by Lemma 5; otherwise, if S and EP are not in $\{\pi_1 \ldots \pi_k\}$, but O is, by Lemma 4; otherwise, if $G \in \{\pi_1 \ldots \pi_k\}$, by Lemma 3; if only NA holds, by Lemma 2. □

Theorem 2 can be rephrased as follows:

Corollary 1 *$Cal^{\pi_1 \cdots \pi_k}$ is the minimal closure of Cal, with respect to $\{O_{\pi_1}, \ldots, O_{\pi_k}\}$.*

5 Conclusions

We presented a family of symbolic languages for user-defined periodicity, that has significant features and solves some drawbacks of languages in the literature.

We briefly notice again that the modularity of our approach allows for a clear management of properties characterizing periodicities. We explicitly avoid side-effects, that impair the clarity of semantics elsewhere. Consider, for instance, in the language of [11], the definition of "the first Monday of each month": group Mondays into months, using *Dice*, then select the first Monday in each group, with the *Slice* operator "[1]\". Counterintuitively, this defines a collection of collections each containing a Monday.

Moreover, modularity allows to obtain the desired expressiveness for an application without the burden of unnecessary features. This is not obvious. For intance, consider [3], in which it is explicitly stated that all the structure carried over from [11] is of no use, but is there as an unavoidable heritage. The effort to get rid of the structure was then done in [14], and results in rather complex operators.

Our top language is more expressive than any other symbolic language in the literature, to the best of our knowledge, since it allows to deal with all the five properties we defined, including the special case of exact overlaps, not treated by any other symbolic approach.

But the expressiveness is enhanced not only in a quantitative way, but also qualitatively, so to speak. Indeed, our design of the languages is based on an analysis of the structures that it makes sense to define, based on usage, homogeneity constraints and our aim at modularity. Overlaps are a notable example—for instance, in [11] overlaps of a more general kind are possible, resulting in structures of no clear meaning.

References

[1] J.F. Allen, Maintaining Knowledge about Temporal Intervals, *Communications of the ACM* 26(11):832843, 1983.

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

[3] C. Bettini, R. De Sibi, Symbolic Representation of User-defined Time Granularities, *Proc. TIME'99*, IEEE Computer Society, 17-28, 1999.

[4] J. Chomicki, and T. Imielinsky, Finite Representation of Infinite Query Answers, *ACM ToDS* 18(2), 181-223, 1993.

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

[6] L. Egidi and P. Terenziani, A lattice of classes of user-defined symbolic periodicities, *TIME'04*, 2004.

[7] L. Egidi and P. Terenziani, A mathematical framework for the semantics of symbolic languages representing periodic time, *TIME'04*, 2004.

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

[9] F. Kabanza, J.-M. Stevenne, and P. Wolper, Handling Infinite Temporal Data, *Journal of Computer and System Sciences* 51, 3-17, 1995.

[10] A. Kurt, M. Ozsoyoglu, Modelling and Querying Periodic Temporal Databases, *Procs. DEXA'95*, 124-133, 1995.

[11] B. Leban, D.D. McDonald, and D.R. Forster, A representation for collections of temporal intervals, *AAAI'86*, 367-371, 1986.

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

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

[14] P. Terenziani, Symbolic User-defined Periodicity in Temporal Relational Databases, *IEEE TKDE* 15(2), 489-509, 2003.

[15] A. Tuzhilin and J. Clifford, On Periodicity in Temporal Databases, *Information Systems* 20(8), 619-639, 1995.