FORMALIZING THE TEMPORAL DOMAIN WITH HIERARCHICAL STRUCTURES OF TIME UNITS.

by

Diana Cukierman

Computer System Engineer, Universidad de la Republica, Uruguay, 1989

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
August 1994

APPROVAL

Name:	Diana Cukierman
Degree:	Master of Science
Title of thesis:	Formalizing the Temporal Domain with Hierarchical Structures of Time Units.

Examining Committee: Dr. Arvind Gupta
Chair

[illegible]

Dr. James Delgrande
Senior Supervisor

[illegible]

Dr. Veronica Dahl
Supervisor

[illegible]

Dr. Frederick Popowich
S.F.U. Examiner

Date Approved: August 18, 1994

There was nothing so remarkable in that, nor did Alice think it so very much out of the way to hear the Rabbit say to itself, " Oh dear, oh dear, I shall be too late!"; ... But when the Rabbit actually took a watch out of its waist-coat-pocket, and looked at it, and then hurried on, Alice started to her feet, ... and burning with curiosity, she ran across the field after it...

Alice in Wonderland, Lewis Carroll.

Acknowledgements

It is a pleasure to acknowledge the help and support I have received from many people during the efforts that have culminated in this thesis.

I am most grateful to my senior supervisor Jim Delgrande for his guidance and encouragement. I also want to thank him for carefully reading the many drafts of the thesis, and his advice and corrections for both, the research results and writing style. His constructive comments have helped shape this thesis, and his enthusiasm for the topic has made this research a pleasure to undertake.

Thanks to the other members of the committee, Veronica Dahl, for her comments and kind counsel and Fred Popowich, for his gracious acceptance as external supervisor and valuable suggestions.

Thanks to the faculty and the staff of the school, who have helped me in many ways, specially Kersti Jaager, Lou Hafer and Arvind Gupta, and for the support from Bill Havens and the Intelligent Systems laboratory in early stages of my research. I also want to thank the Knowledge Representation group of the Computing Science department at University of Toronto for welcoming me during a short but fruitful stay to work with Jim Delgrande during his Sabbatical year there.

Encouragement from and discussions with friend fellow graduate students helped to make this experience a very pleasant one. Andrew Fall read part of the thesis, both Andrew and Joerg Ueberla got enthusiastically involved in philosophical and thesis related discussions, and provided for useful suggestions and comments, and friendly support. Patrice Belleville helped me patiently with LaTeX details. Also Eli, Micheline, Alicja, Marie-Ange, Luciana, Johanna, Chris, Sheelagh, Pepe, Manuel, Edwin, Brigitte, Dave and other innumerable friends and colleagues have made my stay at SFU enjoyable and unforgettable. I thank them all for sharing very good times and for the moral support they gave me during hard moments. My most special thanks to Sergio.

I want to express my deepest appreciation to my family and friends in Uruguay and other places, for their support throughout all my studies and for being so close to me though physically thousand of kilometers away, specially my grandfather, parents, sister Susi, aunts, and good friends Alicia and Sylvie.

This research was mainly funded by the CIDA (Canadian International Development Agency) Human Uruguay Resource Development Project No. 964/15779, and also by support from Simon Fraser University, through graduate fellowships and private scholarships. Finally, I want to acknowledge the support from In.Co. and the people there, (Instituto de Computación), in Facultad de Ingeniería, Universidad de la República, Uruguay.

Abstract

We investigate a formal representation of calendars and time units as restricted temporal entities for reasoning about activities. Calendars can be considered as repetitive, cyclic temporal objects. We examine the characteristics of *time units* as particular classes of time intervals, and provide a categorization of relations among them. The motivation for this work is to ultimately be able to reason about schedulable, repeated activities, such as going to a specific class every Tuesday and Thursday during a semester.

Calendar Structures are defined as an abstract hierarchical structure of time units. We investigate the structural and mathematical properties of this framework and the specific relations among the units that compose it. We propose this formal apparatus as a system of measures which subsumes calendars and any other system that can be based on discrete units and a repetitive containment relation. One of the abstractions introduced in this thesis is that of relations among classes of intervals. This is an expansion of the interval algebra framework defined by J. F. Allen and used throughout temporal reasoning, particularly in scheduling applications.

Contents

List of Tables

List of Figures

List of Definitions

Chapter 1

Introduction

The White Rabbit put on his spectacles.
"Where shall I begin please your Majesty?" he asked.
"Begin at the beggining," the King said gravely,
"and go till you come to the end; then stop."
Alice in Wonderland, Lewis Carroll.

The motivation for this work is to ultimately be able to reason about schedulable, repeated activities, specified using calendars. Examples of such activities include going to a specific class every Tuesday and Thursday during a semester, attending a seminar every first day of a month, and going to swim every other day. Defining a precise representation and developing or adapting known efficient algorithms to this domain would provide a valuable framework for scheduling systems.

In a representation scheme for such activities, one would ideally be able to determine consistency among several repeated activities or verify whether an activity is conflicting with others or not. Another complex task that would result in a useful automated facility is to find a minimal set of potential ways repeated activities may interact. Work has been done on repeated activities, for example [PB91], where a main concern is the implementation of variants of constraint propagation algorithms to detect overlapping repeated activities. We search for a different, more general and formalized representation of the temporal entities. We explore further the *date* concept, building a structure that formalizes dates in calendars.

Human schedulable activities are based on conventional systems called *calendars.* We use as a departure point "calendar" in the traditional sense of the word, that is: "a system for fixing the beginning, length, and divisions of the civil year, and arranging days and longer divisions of time (as weeks and months) in a definite order" [Web]. Examples of calendars include the traditional Gregorian calendar, university calendars, and business calendars, the last two groups defined in terms of the Gregorian. A goal of this work is to define a generic calendar abstract structure, which subsumes the mentioned calendars, and arguably any system of measures based on discrete units. Therefore we also take into account calendars of historical interest to extrapolate their general characteristics and provide completeness to the developed framework. Examples include the French Revolution calendar, the World (or Universal) calendar and the Hebrew calendar [Col71].

Calendars can be considered as repetitive, cyclic temporal objects. We define an abstract structure that formalizes calendars as being composed of *time units*, which are related by a *decomposition* relation. The decomposition relation is a containment relation involving repetition and other specific characteristics. Time units decompose into contiguous sequences of other time units in various ways. A *calendar structure* is a hierarchical structure based on the decomposition of time units. This structure expresses relationships that hold between time units in several calendars.

At this point it is worth introducing the concept of time unit *instances*, how they relate to time units and the different levels we encounter. Scheduling applications are concerned with specific time unit instances; single or repeated activities occurring in the time line during specific days, weeks, hours, etc. Whereas "month" refers to a time unit, "March" is referred to as a *named time unit.* March is one of the 12 occurrences of the notion of *month* with respect to *year*, and viewed extensionally it represents the set of all possible occurrences of March. Finally, "March 1994" is one specific instance of a month. Hence, calendar structures provide a formal system of units on which date-based temporal objects can be built. Intervals, time points and moments in the time line can be represented in terms of these units.

In this thesis we explore calendar structures and systematically analyze the structural and mathematical properties they hold. We suggest a notation to express specific time unit instances. We envision that this notation should provide a straightforward way of representing the temporal counterpart of repeated activities. We define *chains* in the time unit hierarchy corresponding to time units that decompose in a specific way. It is expected that reasoning with instances of such time units will produce very efficient operations.

We envision this formalism as a basis for reasoning about repeated activities; however, we also intend the structure to be sufficiently general that it may be used for representing any system of measures based on discrete units with a repetitive containment relation. Hence a goal of this work is that it also be applicable to systems of (spatial) measures, such as the Metric or Imperial systems.

To wrap up this introduction, we comment on our work as viewed from a knowledge representation perspective, extracted from [MB83],

> "There are two major requirements for a time specialist: First, it must be *formally adequate* and second it must be *computationally effective.* The first condition is met if the formal system is coherent and consistent, and contains sufficient mechanisms to be able to represent all temporal specifications and perform all the deductions we want. The second requirement is essential in order to have a program produce answers with a reasonable amount of effort."

This thesis tackles the knowledge representation aspect of the temporal domain, based on calendars and units within. We define time unit hierarchies or calendar structures, a formal framework of time units which has been shown to be coherent and consistent. *Calendar expressions* are suggested to represent time unit instances so that the temporal counterpart of repeated and simple activities can be represented. We leave for further research the definition of a formal system where deductions can be done in the defined language. Nonetheless, the necessary basis for the definition of a language is exposed in this thesis. As for the implementation aspect of the time specialist, we analyze computational aspects of dealing with calendar structures per se and we envision that efficient algorithms are possible to develop. Particularly efficient results should be expected when dealing with time unit instances that come from specific sequences of time units, which we define as chains.

1.1 Organization of the thesis

The next section presents background material and related work. Specially we present Allen's *interval algebra* [All83]. Characteristics of our approach are compared to the different research results as they are exposed.

Chapter 2 builds an extension to the interval algebra. In this thesis we define relations between *classes of intervals* and particularly decomposition.

Chapter 3 develops the concept of *time unit class.* The main attributes are described both informally and functionally. The duration of a time unit and the naming of time unit instances are the main issues dealt with. Numerous examples accompany the development of these concepts.

Chapter 4 describes and defines decomposition relations between the particular class of intervals: time units. *Constancy* and *alignment* are proposed as the only aspects needed to characterize this decomposition. The formal definition is based on the decomposition relation defined in Chapter 2 applied to the particularities of time units.

Chapter 5 presents the definition of calendar structures or time unit hierarchies. Examples of several calendar structures are given, such as Gregorian and university calendars. The structure is systematically analyzed, resulting in a parallelism with divisibility concepts. An alternative structural characterization is studied, providing for a starting point for the definition of a formal language of time units. In the final sections, computational aspects of using the calendar structure are addressed and a particular notation is suggested to represent time unit instances.

The last chapter highlights the main contributions and special issues of interest in this thesis. Moreover, it is worth emphasizing that this thesis leaves open an important amount of further research venues. Suggestions and research possibilities that are laid at by this thesis conclude the last chapter. The appendix contains the proofs of the main theorems in Chapter 4.

1.2 Background

The choice of time points or time intervals as primitive temporal objects is an issue that has been long discussed in philosophy (see for example [Van83]). It is also a matter dealt with extensively in the AI literature. Discussions that are related to ontological aspects include for example whether or not there exist in reality instantaneous events, which would occur during a time point, by nature having zero-duration. Another concern is how assertions that occur in the time line should be interpreted. An objection that Allen [All83] poses over interpreting assertions over time points is that this leads to paradoxical situations. The proposed example is that of a time interval during which a light bulb is on, meeting another interval during which the same light bulb is off. Where these two intervals meet, there is a transition of the truth value of the property *on(light)*. The question that arises is, what is the truth value of this property in the meeting point? Is the light on, off, both or neither? Having assertions interpreted only over intervals, the law of excluded middle is maintained, that is, at all times a property is either true or false. Our formalism deals with time units, which are a special kind of time interval, with inherent durations. Therefore we will base our research on *time intervals* as the basic temporal objects.

The next sections present background material of the interval algebra and further related work. Different issues of our work are compared in perspective with the material presented.

1.2.1 Interval algebra

[All83] defined an *interval algebra* of relations between (convex) time intervals, with intervals considered as primitive temporal objects[1]. There are 13 basic relations between convex intervals; these are the combinatorial possibilities of how two intervals can relate, like for example *before, during*, etc. The elements of the algebra are the relations that may exist between intervals of time. Temporal relations can be indefinite, therefore there are 2^{13} possible relations between convex intervals. A transitivity

[1]Convex intervals are those that do not have any gaps within them.

table, expressing the results of composing any of two basic relations appears in [All83]. These relations appear in Table 2.1, Chapter 2. The transitivity table forms part of the algebra definition, as it defines one of the operations between relations, referred to as composition or multiplication. This table is the basis of the implementation of algorithms for propagation of temporal constraints among intervals. [LM94] subsumes results related to the formalization of Allen's work [All83] in terms of relational algebras.[2]

[All83] presents a precise computational model of a temporal inference system. Different axiomatizations of interval relations have appeared in the literature, which subsume those of Allen's with the 13 basic relations among intervals. Allen and Hayes [AH85, AH89] define a first-order formal theory where they express every possible basic relation among intervals with only one relation, *meets*, considered as primitive. Other axiomatizations take into account a basic distinction between interval relations: those relations of strict precedence as opposed to those that share common points. The case of relations with common points has been axiomatized with the *overlap* relation or with the *inclusion* relation, respectively Kamp's and Van Benthem's axiomatizations [BL92].

The problem of determining consistency of assertions in the full interval algebra was proven to be NP-complete, as well as determining all consequences of these assertions [VK86]. Allen's polynomial constraint propagation closure algorithm in [All83] was proven to be incomplete. In [VK86], the *time point algebra* is developed, based on the notion of *time point* in place of interval. The time point algebra has known (complete) polynomial closure algorithms (see also [VKV89]). There are only three basic relations among points: *before, equal* and *after.* Convex intervals can be represented by their two extreme points and consequently, the 13 basic relations among intervals, and disjunctions of them, are expressible in terms of the point algebra. However, there are certain disjunctive combinations of interval-to-interval relations that are not expressible with the point algebra. The paradigmatic example used in the

[2]The term "relational algebra" refers to algebras in Tarski's sense, [JT52]. Briefly, they are structures based on boolean algebras, where the elements are relations. (These are not the algebraic formulation of operations on relational data bases).

related literature is that of an interval being *before or after* another. This situation is common in scheduling applications.

The tradeoff is then between a more expressive NP-complete, full interval algebra vs. a less expressive, tractable point algebra. More specifically, the efficiency that can be obtained depends on which disjunctive combinations of relations are allowed between intervals and/or points. Without disjunction, both interval and point based representations have the same expressive power and have closure algorithms of $O(n^2)$ time, where n is the number of points or intervals. If disjunction of interval relations is allowed then the problem becomes NP-hard [VK86]. Disjunctions of point relations are less expressive, but tractable, with a closure of $O(n^3)$ time. Allowing for only certain specific disjunctions among intervals, interval relations can be expressed with the time point algebra, thereby obtaining a tractable interval subalgebra. This subalgebra has been called *pointisable interval algebra* [VKV89]. If relations are combined with conjunctions and disjunctions arbitrarily, then both point and interval representations are equivalent, and the closure problem is NP-hard in both cases.

Nonetheless, some specific applications do not require the full expressive power of the interval algebra, and can benefit from efficient representations. Research has been done on finding efficient and yet useful subalgebras of the full interval algebra by limiting the allowable relations. The above mentioned pointisable interval algebra is one such example, and others have been studied as well [GS92]. Work has also been done restricting the temporal structure to specific characteristics of the intended application. For example, [MS90] have developed a sequence chain representation taking into account the time structure originating from narrative texts. This representation based on points allows for very efficient operations. Also work has been carried out organizing intervals into a hierarchy so that constraint propagation among intervals is more efficient [Koo89]. Hence it is of interest to study restrictions of the interval algebra. In this respect, the present framework deals with restricted kinds of intervals within a hierarchical structure. These restrictions are based on the particular characteristics of the motivation problem: reasoning with repeated activities within calendars and dates. The intent is that the hierarchy provides a basis for obtaining efficient algorithms for certain operations. This may be contrasted with [Koo89] which considers a hierarchical structure in the general interval algebra.

1.2.2 Further related work

Non-convex intervals are employed when using time units in a repetitive way or when referring to recurring periods [Lad86a, Lad86b]. We envision that only a narrow subset of the set of non-convex relations is required to relate schedulable repeated activities. Again, such a restriction could allow for efficient, yet useful inferences. [Lad86b] also defines a concept of time units as a unique linear hierarchy of sequences of integers corresponding to "[year, month, day, hour, ...]". Thus these sequences have a fixed order, closed at the left and open at the right. *Interval types* are defined from these units, based on an axiomatization that relies on the fixed format of the sequences. Units that do not appear in the sequence are defined through iterating the *meets* operation combined with operators like *convexify*. (Convexify produces the smallest convex time interval containing two subintervals). In contrast, we consider our building block to be a *"time unit class"* (for example *year* or *week*). The time unit hierarchy proposed in our work generalizes [Lad86b], in that temporal objects in the time line can be represented by any sequence of composed time units, as opposed to fixed in Ladkin's approach. Thus valid temporal objects in our framework include sequences which represent, for example, February 1st 1994, the 32nd day of 1994, the 5th day within the 17th week of the year 1994. Moreover, we are able to *combine* systems of measurement, and so talk about the third month of a company's business year as corresponding with June in the Gregorian calendar. *Duration* of time units is axiomatized in [Lad86b]. This approach, however, does not take into account the varying duration of specific time instances, for example, the duration of a generic month in days (28 to 31 days) is different from *February's* duration (28 or 29 days), which in turn is different from the duration of *February 1994* (28 days). This matter is addressed and formalized in our work.

[MSK93] elaborate on the notions of non-convex interval relations defined in [Lad86a, Lad86b]. They define an algebra of relations between what they call *n-intervals*, a subclass of Ladkin's non-convex intervals. The operations defined for the algebra allow for the application of a variation of Allen's constraint propagation algorithm to their relations. In this respect, our work defines operations relating decomposition relations which constitute a basis for defining a relational algebra of such relations. [Lig91] proposes a generalization of non-convex intervals. This work defines a generalized interval as an ordered, finite sequence of points in a linear order. Relations among these generalized intervals are expressed as conjunctions of conditions on the interval defining points. The study of these relations as they constitute an ordered structure and the definition of a calculus of these relations is a main concern in Ligozat's paper. One motivation of this study is the understanding about the topology of relations. This motivation is also present in our thesis. In our case, we study the order structure and algebraic operations among decomposition relations. [LMF86] deals with repetition and time units. This work relies on sequences of consecutive intervals combined into "collections". The collection representation makes use of "primitive collections" (essentially circular lists of integers), and two basic operators, *slicing* and *dicing*, which subdivide an interval and select a subinterval respectively. The problem of months and weeks not fitting exactly is addressed in Leban et al's paper. They distinguish strict from relaxed dicing operators, allowing for example to differentiate the first week of a month (whether it is complete or not) from the first complete week of a month. One of the characterizations of the decomposition relation we define deals with time units fitting exactly or not among others.

[PB91] is mainly concerned with the implementation of algorithms to detect overlapping repeated activities. This work relies on temporal constraint satisfaction results and algorithms [DMP91]. [PB91] also introduces the concept of using dates as reference intervals to make constraint propagation more efficient. From a different perspective, we envision that our proposed formalism also profits from dates and calendars to establish a sound basis that will lead to efficient temporal reasoning operations. [CR88] expose a set theoretic structure for the time domain with a calendar perspective. They formalize the temporal domain with sets of "constructed intervallic partitions" which have a certain parallel to our decomposition into contiguous sequences. However, in their case only time units that decompose in an aligned way are considered; (in their terminology, their model excludes weeks). This is to be contrasted with our formalism which is more complete and general. To mention some of the differences; our work takes alignment and non-alignment of time units as a special concern, abstracts time units and decomposition relations. We also address the issue of naming and durations of classes of time units and time unit instances, study the calendar structures with a mathematical perspective. As well, we set up a basis for developing a formal language for time units, a matter not included at all with their work. Their work builds the formalism with a different perspective than ours, and in fact they were conceived completely independently. It is interesting to observe however that in formalizing the same domain, some of the problems encountered are similar.

Preliminary reports of the research developed for this thesis appear in [CD94a, CD94b].

Chapter 2

Classes of intervals and decomposition

This method is, to define as the number of a class the class of all classes similar to the given class. **Principles of Mathematics, pt.II, ch.11, sect.iii (1903), Bertrand Russell.**

In this chapter we define some specific relations between intervals based on the basic relations of the interval algebra [All83]. Of special interest are the decomposition relations of an interval into a *contiguous sequence* of intervals. We also introduce the concept of *classes of intervals* and relations between them. Finally *decomposition* relations between (generic) classes of intervals are defined. The objects of study in this chapter constitute the formal building blocks for the definition of *decomposition between time units*, a central issue in this work. As well, we envision that the concepts developed here have potential utility for other applications in temporal reasoning, aside from the usage in this thesis.

2.1 Basic interval relations

This work takes time intervals as primitives and bases definitions on interval algebra basic relations [All83]. These relations appear with the notation that will be used in this document in Table 2.1.

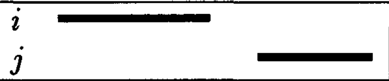
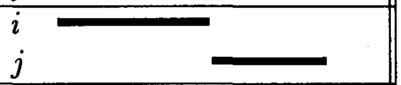
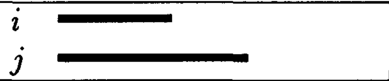
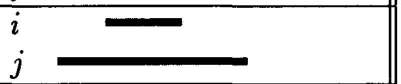
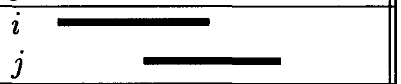
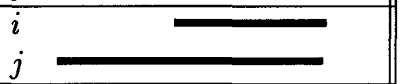
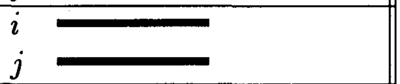
Relation		Graphical Representation[1]
before after	$.before.$ $.before^{-1}.$	
meets met by	$.meets.$ $.meets^{-1}.$	
starts started by	$.starts.$ $.starts^{-1}.$	
during contains	$.during.$ $.during^{-1}.$	
overlaps overlapped by	$.overlaps.$ $.overlaps^{-1}.$	
finishes finished by	$.finishes.$ $.finishes^{-1}.$	
equals	$.equals.$	

Table 2.1: Basic 13 binary interval relations

2.2 New interval relations

Two containment relations, *.strict-in.* and *.in.* are defined as specific disjunctions of basic interval containment relations. The relation *.in.* also appears for example in [AH89].

Definition 1 (Interval relations "strict-in" and "in")
Let i and j be two intervals,
i.strict-in.j $\equiv$ *i.during.j* $\bigvee$ *i.starts.j* $\bigvee$ *i.finishes.j*
i.in.j $\equiv$ *i.strict-in.j* $\bigvee$ *i.equals.j.*

Table 2.2 shows a graphical representation of these relations.

[1]Direct relations are graphically represented between intervals i and j: $i\ .r.\ j$, therefore, the inverse holds between j and i: $j\ .r^{-1}.\ i$.

[2]Direct relations are graphically represented between intervals i and j: $i\ .r.\ j$

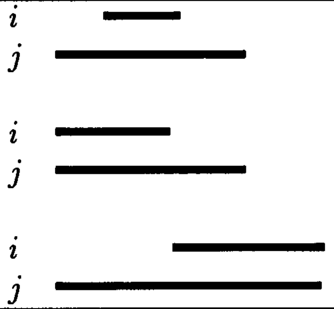
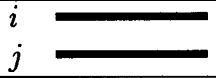
Relation		Graphical Representation[2]
strict in	*.strict-in.*	
equals	*.equals.*	

Table 2.2: *Strict-in* and *Equals* relations. *In* includes both, strict-in and equals.

Theorem 1 Relations *.equals.* and *.strict-in.* are disjoint.

Proof: Allen's 13 basic interval relations are disjoint. By definition, the new relation *.strict-in.* is disjoint with the relation *.equals.*. ■

Definition 2 (Ordered sum of intervals) [3] *Given two time intervals that meet, there exists a unique longer interval that constitutes these two intervals. This longer interval is called the "ordered union" or "sum" of the adjacent intervals.*

The ordered sum (k) of two meeting intervals i and j is denoted as $k = i + j$. The existence and uniqueness of such longer interval covering two adjacent ones is guaranteed for example within the axiomatization of the time interval algebra in terms of meet, by two of the axioms in this theory, namely axioms *[M4]* and *[M5]* in [AH89], page 227.

Definition 3 (Contiguous sequence of n intervals) *A* contiguous sequence of n intervals *is the ordered union of* n *adjacent intervals:*
$< j_1, \ldots, j_n > \equiv (j_1 + \ldots, +j_n)$.

By definition of ordered sum, it holds that:
j_i .meets. j_{i+1} $\forall\, i,\ i = 1, \ldots, n-1$.

[3]This definition is based on [AH89]

2.2.1 Decomposition into contiguous sequences of intervals

Two relations are defined that relate one interval with a contiguous sequence of intervals: aligned decomposition into a sequence, and non-aligned decomposition into a sequence.

Definition 4 (Aligned decomposition into sequence) *An interval decomposes in an* aligned *way into a contiguous sequence of intervals iff it is equal to the ordered sum of the intervals in the sequence. This is denoted by: i .dec-alig-into-seq.* $< j_1, \ldots, j_n >$ *iff* $i = (j_1 + \ldots, +j_n)$.

The first and last interval in the sequence relate with the interval (i) as follows: (j_1 .starts. i) *and* (j_n .finishes. i).

Figure 2.1.a graphically shows this relation.

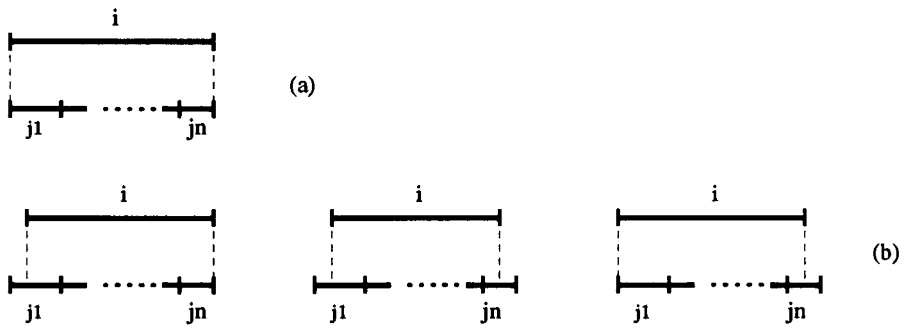


Figure 2.1: Graphical representation of aligned and non aligned decomposition

Definition 5 (Non-aligned decomposition into sequence) *An interval decomposes in a* non-aligned *way into a contiguous sequence of intervals, i.e.*
i .dec-non-alig-into-seq. $< j_1, \ldots, j_n >$ *iff*

1. *The interval i relates with a .strict-in. relation with the ordered sum of the intervals in the sequence.*
   $k = (j_1 + \ldots, j_n)$ *and i .strict-in. k.*

2. *The following relations hold between the interval and the extreme intervals in the sequence:*

   $(j_1$ *.overlaps.* $i) \bigwedge (j_n$ *.finishes.* $i)$ *or*
   $(j_1$ *.overlaps.* $i) \bigwedge (i$ *.overlaps.* $j_n)$ *or*
   $(j_1$ *.starts.* $i) \bigwedge (i$ *.overlaps.* $j_n)$.

3. *There is at least one subinterval completely contained in the sequence:* $\exists\ j_x, 1 \leq\ x \leq\ n$ *such that* i *.contains.* j_x *(or simply varying the notation,* i *.during*$^{-1}$*.* j_x*).*

Figure 2.1.b graphically shows this relation.

The reason why the third additional constraint is imposed on non-aligned decomposition is to enforce the intuitive notion holding in a decomposition relation that a composed interval indeed decomposes into the component and not the other way around. Figure 2.2 shows the problematic situation that would arise if non-aligned decomposition were defined without this restriction. It should be noted that non-aligned decomposition cannot include the limit case of an interval decomposing into itself (which is correct and is *not* eliminated in the case of *aligned* decomposition).

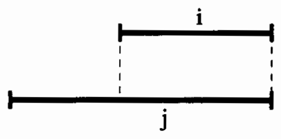


Figure 2.2: Example of problematic situation avoided with definition of non-aligned decomposition

Theorem 2 *An interval decomposes in an aligned way into a sequence of intervals iff it does not decompose in a non-aligned way into that same sequence of intervals.*
i *.dec-alig-into-seq.* $< j_1, \ldots, j_n >$ *iff*
not i *.dec-non-alig-into-seq.* $< j_1, \ldots, j_n >$.

Proof: It follows from the definition of these two relations, and the fact that *.equals.* and *.strict-in.* are disjoint relations. Hence, an interval is either equal or

exclusively strictly in a contiguous sequence of intervals (into which it decomposes).
■

Theorem 3 (Number of subintervals in a non-aligned decomposition) *If an interval i decomposes into a sequence of intervals $< j_1, \ldots, j_n >$ in a non-aligned way, then $n \geq 2$. More precisely,*

1. *if $(j_1$ .overlaps. $i) \bigwedge (j_n$ .finishes. $i)$ or $(j_1$ .starts. $i) \bigwedge (i$ .overlaps. $j_n)$ then $n \geq 2$.*
2. *if $(j_1$ .overlaps. $i) \bigwedge (i$ .overlaps. $j_n)$ then $n \geq 3$.*

Proof: It follows from the definition of non-aligned decomposition. More precisely, this follows from the fact that there is always at least one complete subinterval contained in the composed interval (guaranteed by the third characteristic in the non-aligned decomposition definition). ■

2.3 Classes of intervals

Basic binary interval relations are generalizable to relations among *classes* of intervals. This generalization is clearly also applicable for the new definitions developed at the interval level, such as ordered sum of intervals. In addition, a *binary* decomposition relation among classes of intervals is defined based on decomposition of a class of intervals into sequences of classes.

Definition 6 (Class of intervals) *A class of intervals is a set of intervals with some common properties. The intervals are said to be instances of the class.*

Two predicates are defined relating instances of intervals and classes, *in-class* and *subclass*. These two predicates are to be interpreted as follows: *in-class*(i, I) iff i is an instance of I, and *subclass*(J_x, J) iff $\forall j$ *in-class*$(j, J_x) \supset$ *in-class*(j, J).

2.3.1 Generalization of interval relations to class relations

Definition 7 (Binary relations between two classes of intervals) *A class of intervals relates to another class with a generalization of a basic binary interval relation iff for all instances of the former, there is an instance of the latter such that they relate with that basic relation. More precisely, I and J are classes of intervals and .R. is a relation between the two classes,*
$(I.R.J)$ *iff* $\forall i\, \exists j\, (\textit{in-class}(i, I) \bigwedge \textit{in-class}(j, J) \supset i.r.j)$,
where .r. is one of the 13 basic interval relations in Table 2.1. In terms of notation, .R. is the generalized relation corresponding to r.

Any disjunction of the 13 basic relations is generalizable from the interval level to the class of intervals level in an analogous way, particularly also the relations *.in.* and *.strict-in.* above defined. The notation convention used is that relations between two intervals are written with lower case letters, (for example *.meets.*), and relations between two classes of intervals are denoted with upper case letters, (for example *.MEETS.*). Likewise, intervals are represented by lower case letters, for example *i*, *j*, and classes of intervals with upper case letters, for example *I*, *J*.

According to the definition presented, when generalizing a relation from the interval (instance) level to the class of intervals level it is *not* required that every instance of one class relates to *every* instance of the other, but only that for every instance of the former, *there exists* an instance of the latter such that the relation at the instance level holds. (The generalization has a universal/existential formulation.)

An example with decomposition relations among time units should give an intuition of why generalization is done in this manner. *Month* from the Gregorian calendar is considered to be a class of time intervals: the class representing all possible months, January 1900, February 1994, December 2000, and so on. *January* for example still is a class, subclass of *month*, representing all possible *January's*. We may want to express that the subclass *January .MEETS.* the subclass *February.* Clearly, not all instances of January meet all instances of February, but only those belonging to the same year. The universal/existential generalization of the relation applies to this case: for all instances of January, there exists one instance of February such that January meets February. It should be noted that unless additional constraints are imposed in relations at the class level, there can be more that one possible binary relation between the same two classes.

For example another possible relation between February and January is that *February .BEFORE. January.* This in fact holds for every instance of February, considering any January instance in a subsequent year.

Based on the meets relation between two classes, the notions of ordered sum of intervals and contiguous sequence of intervals above defined are also generalizable from the instance level to the class level. The universal/existential generalization is also applied.

Definition 8 (Ordered sum of classes of intervals) *K equals the ordered sum $(I+J)$ iff $\forall k\, \exists i\, \exists j$ (in-class(k,K) $\bigwedge$ in-class(i,I) $\bigwedge$ in-class(j,J)) $\supset$ $(k=i+j)$.*

Definition 9 (Contiguous sequence of n classes of intervals) *A* contiguous sequence of n classes of intervals *is the ordered union of n classes of intervals:*
$< J_1, \ldots, J_n > \equiv (J_1 + \ldots, J_n)$.

By definition of ordered sum, it holds that:
J_i .MEETS. J_{i+1} $\forall\, i,\ i = 1, \ldots, n-1$.

Furthermore, the relations "aligned decomposition into a sequence" and "non-aligned decomposition into a sequence" are extensible to classes.

Definition 10 (Aligned decomposition into a sequence. Class level) *A class of intervals I decomposes in an aligned way into a contiguous sequence of classes of intervals iff*

1. *I is equal to the ordered sum of the classes of intervals in the sequence. This is denoted by the uppercase version of the corresponding instance level, i.e:*
   I.DEC-ALIG-INTO-SEQ. $< J_1, \ldots, J_n >$ iff $I = (J_1 + \ldots, J_n)$.

2. *Neither of the classes of intervals J_x contain the class I. In terms of instances, there is no instance of any of the classes J_x that contains an instance of the class I.*

$\forall J_x$, $1 \leq x \leq n$, $\neg[J_x.CONTAINS.\ I]$ *which can be expressed also as:*
$\forall j\ \forall i\ (in\text{-}class(j, J_x) \bigwedge in\text{-}class(i, I)) \supset \neg\ (j.contains.\ i)$

The last restriction in the definition of decomposition at the class level is essential to guarantee that decomposition is a containment relation, as is presented in the following theorem. It should be noted that this is essential to include as part of the definition at the class level and not at the interval level, to ensure a unique relation between the classes of intervals involved. Recall that two classes may relate with more that one relation given the universal/existential generalization of relations between classes of intervals. Figure 2.3 shows a scenario that could occur in case of an aligned decomposition of a class I into a sequence of subclasses of J if this restriction was not imposed on the decomposition definition of classes. In this scenario, there is an interval i_1, instance of the class I that decomposes aligned into $< j_1, j_2 >$, both instances of J. If the above mentioned restriction is not imposed, there could be another interval i_2, of a larger extension than i_1, that decomposes into $< j_3, j_4 >$. But in this scenario j_3 has in fact a larger extension than i_1, and therefore it holds that an instance of I decomposes into an instance of J and reciprocally, an instance of J decomposes into an instance of I. This scenario does not seem reasonable, and is avoided with the restriction. This restriction can be exemplified within a university calendar like the one from Simon Fraser University. This calendar has units which we call *periods* of two weeks, three weeks, and thirteen weeks (corresponding to exam periods, break periods and class periods respectively). *Periods* can not be said to decompose into *months* nor reciprocally.

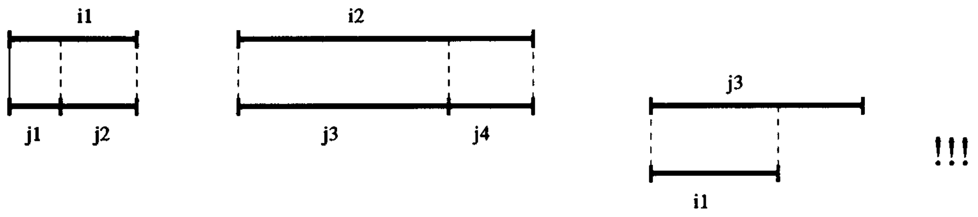


Figure 2.3: Example of problematic situation avoided with the definition of class decomposition

Theorem 4 (Aligned decomposition as containment) *Aligned decomposition into sequences at the class level is a containment relation. If a class of intervals I decomposes into a sequence of intervals $< J_1, \ldots, J_n >$ in an aligned way, then every subinterval $J_x, 1 \leq x \leq n$, is contained in it, and I is not contained in any of the classes J_x.*

I.e. $\forall J_x,\ 1 \leq x \leq n,\ (J_x.\mathrm{IN}.\ I) \bigwedge \neg\ (I.\mathrm{CONTAINS}.\ J_x)$.

Proof: The result follows from the definitions of the decomposition relation. The second restriction in the definition plays an essential role so that this theorem holds.

1. If $n = 1$ then I.equals. J_1, which is a particular case of the $.IN.$ relation.

2. If $n > 1$, then all classes of subintervals $J_x,\ 1 \leq x \leq n$, are contained in I. I is not contained in any of the J_x.

Definition 11 (Non-aligned decomposition into sequence. Class level) *A class of intervals I decomposes in a non-aligned way into a contiguous sequence of intervals, i.e. I .DEC-NON-ALIG-INTO-SEQ. $< J_1, \ldots, J_n >$ iff*

1. The class I relates with a .strict-in. relation with the ordered sum of the intervals in the sequence, i.e.
   I .STRICT-IN. K, where $K = (J_1 + \ldots, J_n)$

2. The following relations hold between the class of intervals and the extreme classes of intervals in the sequence:

   $(J_1$ .OVERLAPS. $I) \bigwedge (J_n$ .FINISHES. $I)$ or
   $(J_1$ .OVERLAPS. $I) \bigwedge (I$ .OVERLAPS. $J_n)$ or
   $(J_1$ .STARTS. $I) \bigwedge (I$ .OVERLAPS. $J_n)$

3. At least one of the classes in the sequence $(J_x,\ 1 \leq x \leq n)$ is completely contained in the class I: $\exists\ J_x$ such that I.CONTAINS. $J_x,\ 1 \leq x \leq n$

4. *Neither of the classes of intervals J_x contain the class I. In terms of instances, there is no instance of any of the classes J_x that contains an instance of the class I.*

   $\forall J_x,\ 1 \leq\ x \leq\ n,\ \neg[J_x.CONTAINS.\ I]$ *which can also be expressed as:*
   $\forall j\ \forall i\ (in\text{-}class(j, J_x) \bigwedge\ in\text{-}class(i, I))\ \supset\ \neg\ (j.contains.\ i)$

Just as in the case of aligned decomposition of classes, it should be noted that the last restriction in the definition is essential at the class level and not at the interval level, because two classes may relate with more that one relation given the universal/existential generalization of relations between classes of intervals.

Theorem 5 (Non-aligned decomposition as containment) *Non-aligned decomposition at the class level is a containment relation. If a class of intervals I decomposes into a sequence of intervals $< J_1, \ldots, J_n >$ in a non-aligned way, then every subinterval $J_x, 1 \leq\ x \leq\ n$ is contained in it, and I is not contained in any of the classes J_x.*

I.e. $\forall\ J_x,\ 1 \leq\ x \leq\ n,\ (J_x.\text{IN.}\ I) \bigwedge\ \neg\ (I.\text{CONTAINS.}\ J_x)$.

Proof: This theorem is analogous to the aligned case. It follows from the definitions of the decomposition relation of I .DEC-NON-ALIG-INTO-SEQ. $< J_1, \ldots, J_n >$. All classes of subintervals $J_x,\ 1 \leq x \leq n$, are contained in I. I is not contained in any of the J_x. ∎

2.3.2 Decomposition: a binary relation between classes of intervals

Based on the decomposition relations (aligned and non-aligned) of a class of intervals into a sequence of classes, a new binary relation between classes of intervals is defined. This *decomposition* relation plays a central role in the formalism developed in this thesis.

Definition 12 (Decomposition relation between two classes) *A class of intervals is said to decompose into another iff the former (or composed class) decomposes*

into a sequence of subclasses of the latter (or component class) either in an aligned or in a non-aligned way.
I.DECOMPOSES-INTO. J iff there exist r subclasses $J_1, \ldots, J_r$ of J such that
I.DEC-ALIG-INTO-SEQ. $< J_1, \ldots, J_r >$ or
I.DEC-NON-ALIG-INTO-SEQ. $< J_1, \ldots, J_r >$.
The number of classes, r is referred to as the repetition factor *of the contiguous sequence and also of the decomposition. Given the two classes of intervals, I and J, a lower and a higher bound for the repetition factor are uniquely determined.*

Theorem 6 (Decomposition is a containment relation) *If one class of intervals decomposes into another (I.DECOMPOSES-INTO. J), the composed class (I) contains or is equal to the component class (J), and the component class does not contain the composed class.*

Proof: The result follows from the definition of the .DECOMPOSES-INTO. relation, i.e. decomposition in an aligned or non-aligned way into a sequence of subclasses of the component class. Therefore this theorem summarizes the theorems above of decomposition of classes as containment relation. ■

More intuitive and concrete examples of decomposition between classes of intervals and other concepts defined in this chapter are presented in Chapter 4, where the classes of intervals are time unit classes, a particular case of general classes of intervals. Table B.1 in Chapter 3 summarizes and compares concepts of time intervals and classes of time intervals with time unit instances and time unit classes.

Chapter 3

Time units and time unit instances

Thirty days hath September,
April, June, and November;
All the rest have thirty-one,
Excepting February alone,
And that has twenty-eight days clear
And twenty-nine in each leap year.
Stevins MS. (c.1555).

The central element of our formalism is that of a *time unit.* Time units represent *classes* of time intervals, each with certain commonalties and which interact in a limited number of ways. For example, *year* and *month* are time units. Something common to every *year* is that it can be decomposed into a constant number of months. A characteristic of *month* is that it decomposes into a non-constant number of days, which vary according to the instance of the month. Properties that are common to time units determine the time unit class attributes. Some of the attributes clearly belong to the time unit class, like for example an identifier. Some others correspond (in essence) to the time unit class, however they strictly belong to the relation between two classes. For example, the duration of a time unit in terms of another one is an attribute of the relation between this time unit and the one acting as the unit of measure.

In the following sections, the *identifier* of a time unit class and the *(general) duration* are presented. Relative and general durations are compared. Related to durations, the concept of an *atomic time unit* is introduced. Time unit instances and their numbering and naming is described as well. The reference of a time unit, a concept closely related to instance values (which are numbers or names), is defined. All attributes are formally defined in a functional way. Numerous examples are intercalated among the different definitions, and the final section presents a summary with examples from the Gregorian and Simon Fraser university calendars which show how all the definitions apply.

3.1 Identifier of a time unit

The first attribute of the time unit class we consider is a unique *identifier*, which is composed of a time unit name, for example *year* or *month*. If different calendars have different time units that happen to be named the same (for example a week in the Gregorian calendar is 7 days, while in the French Revolution calendar a week is 10 days), some convention will distinguish them, for example the mentioned weeks would be: $week_{GC}$ and $week_{FRC}$. This is to say, they really are different time units.

Also we will distinguish time unit names when the time units have the same duration but differ in their origin. For example, *years* in the Gregorian calendar start in January, but *school years*, in university calendars in the northern hemisphere, start in September. However these two *years* differ from each other in a more subtle way than the previous weeks example. Year and school year have several properties in common that we want to contemplate. They have the same duration, and they decompose into several common time units. We consider year and school year to belong to two different, but yet very closely related calendars, the Gregorian calendar and a university calendar. A university calendar is called a *variant* of the Gregorian, and as such, it is based on it, and shares some time units, for example: week, month, day.

Functional definition of a time unit identifier

A function *id* is defined on the sets of time units;

id : Set of time units $\longrightarrow$ Set of time unit names
Let T be a time unit, $id(T) = Name$

However, we will usually overload notation. The identifier of the time unit class will be used to refer to the class itself, usually in italics, unless otherwise specified.

3.2 Duration of a time unit

A common characteristic of time units is that they inherently involve durations; a time unit expressly represents a standard adopted to measure periods of time. In passing, this confirms the notion that time units are special kinds of *time intervals* (as opposed to time points).

Different time unit instances of the same time unit class can have different durations. For example different months have different number of days, from 28 to 31. Accordingly, the attribute representing a duration of *a class* is a *range* of possible values. We will represent this range by a pair of integers, the extremes of the range of possible lengths any time unit instance can have. Thus, *month* as a class has a duration of $(28, 31)$. It should be noticed that we are using the term "duration" to express a *range* of "possible lengths" of instances of a class. Sometimes the range associated to a time unit class contains one single value, and we will also call it duration. This is the case for example of week, having a duration of $(7, 7)$ days. Finally, a specific time unit *instance* as opposed to a class, has one specific length, and therefore not a range of possible lengths. When there is no problem of ambiguity we will overload notation, and also refer to a unique length as "duration". An example in this situation is February 1994 having 28 days. Formally, one function is defined to associate the pair indicating the range of possible lengths to the time unit class, (dur). Another function is defined to associate the unique length to a time unit instance, ($dur@$).

3.2.1 General vs. relative duration

A duration referred to as *general* will be expressed in a *basic* unit, common to all the time units in one calendar. For example, using *day* as a common or basic unit, *month* has a (general) duration of $(28, 31)$ days, *week* has a (general) duration of $(7, 7)$ days. A specific month, for example, *February 1994*, has a duration of 28 days.

We also define another kind of duration; a *duration relative to another time unit.* Relative durations are, strictly speaking, attributes of the relation between the two time units involved. A time unit can have several relative durations expressed with respect to several others. For example, *year* can have its duration expressed in terms of *months*, or *seconds*; respectively $(12, 12)$ or $(365 * 24 * 60 * 60, 366 * 24 * 60 * 60)$.

A time unit can be measured in terms of another one if the former decomposes into the latter.

It follows from the description of general and relative durations that the general duration is in fact a relative one, expressed in units of a special *basic* unit in the calendar. One property we impose on a time unit to be suitable as a basic unit is that it be *atomic* for the intended application, as explained below. Other properties of basic time units are analyzed when describing calendar structures or time unit hierarchies, in Chapter 5. An important reason why (general) durations of all time units in a calendar are defined based on the same basic measure unit is to be able to compare them. This notion plays a fundamental role in the partial order among time units in the same calendar or variants of a calendar.

Equal and non-equal durations time unit

A time unit class can have the property of being of *equal duration* or of *non-equal duration.* Strictly, this property depends on the basic measure time unit selected, and so, again, it is not an absolute characteristic of a time unit. However, the intuition is quite strong in considering certain time units as of equal durations and certain others as of different durations. For example, days are usually considered as always having the same length. In the examples we consider *days* as of equal durations. Any measure of time is completely conventional however, and it is of equal duration if it is defined so. It is interesting to point out some astronomical facts in this respect. In fact, the Earth does not rotate at a uniform rate, and thus day is not an exactly constant measure. From January 1972 there is the "international atomic time", with a second defined as "9192631770 periods of the radiation corresponding to the transition between two hyperfine levels of the ground state of the caesium atom 133." [Moe82]. Interestingly, current days are occasionally inserted a "leap second", to ensure that noon is at 12 o'clock.

Definition 13 (Equal duration and non-equal duration time units) *General durations of all time units in a calendar are expressed in terms of a predetermined basic time unit. Any time unit class in the calendar is of* equal duration *if all its instances have the same (general) duration. A time unit is of* non-equal duration *if there exist instances with different (general) durations.*

For example, if *day* or *hour* are taken as the basic time unit for the Gregorian calendar, *week* is of equal duration. Any week lasts 7 days or $7 * 24$ hours; a week (general) duration is expressed by the pair $(7, 7)$ in days, $(7 * 24, 7 * 24)$ in hours, etc. Year is of non-equal duration; the range of possible year-instance lengths includes more than one value. A year general duration is $(365, 366)$ in days, $(365 * 24, 366 * 24)$ in hours, etc.

Atomic time units: Moments of time

In this thesis, the time intervals under study are very special ones: they are time units, they represent a class of temporal objects with pre-defined names, inherent durations, etc. It seems reasonable that time units be considered to decompose into smaller units up to a finite level after which a time unit is *atomic* — a non-divisible interval. When the time interval is non-decomposable it is called a *time moment*, following related ideas and terminology of [AH89].

Time units therefore model time in a *discrete fashion.* Time units may be considered atomic in one application and decomposable into smaller time units in other applications. This depends on the intended granularity for the application [Hob85].

We impose the convention that a basic time unit in a hierarchy used to measure (general) durations be atomic. In doing so, all general durations of any time unit in a hierarchy are expressible in terms of a time unit into which they decompose. Also, as a consequence, all durations are representable with integer numbers. For example, if *day* was considered as a basic time unit but decomposable into *hours*, the general duration of the class *hour* would have to be expressed with non-integer numbers, in terms of a time unit higher in the hierarchy. (Recall that day decomposes into hour, so day is higher in the decomposition hierarchy than hour.)

The option of modeling time units as intervals that decompose down to a certain atomic level excludes the possibility of intervals having zero duration, i.e. we exclude the existence of instantaneous events. The issue whether or not intervals can be considered to have zero duration has appeared in Philosophy and Artificial Intelligence (AI) temporal reasoning literature frequently. See [AH89] and references therein.

3.2.2 Functional definition of duration

Initial definitions

We first define the concept of *min-max* pair, which will be later used.

Definition 14 (Min-max pair) *A min-max pair is an ordered pair of natural numbers (a, b) such that $a \leq b$. a is called the minimum component of the pair, b is called the maximum component of the pair.*

Definition 15 (Order relation between min-max pairs) *A strict order relation between min-max pairs is denoted by $\overset{P}{<}$. Equality between min-max pairs is denoted by $\overset{P}{=}$. $\overset{P}{\leq}$ abbreviates $\overset{P}{<}$ or $\overset{P}{=}$.*
Let (a_1, b_1) and (a_2, b_2) be two min-max pairs.

1. $(a_1, b_1) \overset{P}{=} (a_2, b_2)$ *iff* $a_1 = a_2$ *and* $b_1 = b_2$.

2. $(a_1, b_1) \overset{P}{<} (a_2, b_2)$ *iff*
   $b_1 < a_2$ *or*
   $(b_1 = a_2$ *and* $(a_1, b_1) \overset{P}{\neq} (a_2, b_2)$

3. $(a_1, b_1) \overset{P}{\leq} (a_2, b_2)$ *iff* $(a_1, b_1) \overset{P}{<} (a_2, b_2)$ *or* $(a_1, b_1) \overset{P}{=} (a_2, b_2)$.

Examples of relation between min-max pairs

$(1,2) < (10,20)$
$(1,2) < (2,2)$
$(1,2) < (2,3)$
$(2,2) = (2,2)$
$(1,2) = (1,2)$

Theorem 7 $\overset{P}{\leq}$ *is a partial order.*

Proof: See Appendix A.1. ■

A function *dur* associates a time unit class to the range of possible lengths any time unit instance can have. A function *dur@* associates a time unit instance to its unique duration value. The sets of time units and of time unit instances are associated to one calendar and its variant calendars.

Definition 16 (Duration functions) *Two functions are defined, dur and dur@.*

1. *dur is a function that associates a min-max pair to a time unit class in terms of a measure time unit. This min-max pair represents the minimum and maximum value of the length that any instance of the time unit can have, in units of the* measure *time unit.*

2. *dur@ is a function that associates a unique length to a specific time unit instance measured in terms of a* measure *time unit.*

More precisely, let TUS be a set of time units in a certain calendar and its variants and let TUINS be the set of all associated instances.

$$dur : \text{TUS} \times \text{TUS} \longrightarrow N \times N$$
$$dur@ : \text{TUINS} \times \text{TUS} \longrightarrow N$$

Let T be a time unit and $t@$ be a time unit instance of T, let M be a time unit in the calendar that can be a measure of T;

$dur(T, M) = (a, b) \iff_{\text{def}} dur@(t@, M) \in \{a, \ldots, b\}$.

where a and b are measured in units of M.

Examples

The following are examples of the previous functional definition of durations of time units:

$dur(week, day) = (7, 7)$

$dur(month, day) = (28, 31)$

$dur(year, day) = (365, 366)$

$dur(year, month) = (12, 12)$

$dur(century, month) = (1200, 1200)$

$dur@(year\ 1994, day) = 365$

$dur@(month\ February\ 1994, day) = 28$

$\forall$ year-instances y, $dur@(month\ March\ y, day) = 31$

$\forall$ week-instances w, $dur@(w, day) = 7$.

When the same time unit is used as the measure unit, durations can be compared using the order defined for min-max pairs.

$dur(week, day) \overset{P}{<} dur(month, day)$, *since* $(7, 7) \overset{P}{<} (28, 31)$

$dur(month, day) \overset{P}{<} dur(year, day)$, *since* $(28, 31) \overset{P}{<} (365, 366)$.

3.3 Instance names, Reference time unit

A time unit is a class of objects; a time unit instance is a specific temporal object in the time line. There is, however, more than one level of generalization for time units. Consider for example the notion of *month.* First, there is the time unit class *month*, the class of all months. A particular or *named* month (such as March) is a subclass of the time unit *month.* "March" then (extensionally) corresponds to the *set* of all specific instances of the month March. Finally "March 1994" is a specific instance of a month. Notice that via this distinction we obtain appropriate refinements of the domains of attributes of the time unit class. For example, *month* has associated with it possible lengths of 28-31 days. February has a possible length of 28-29 days. Specific *instances* of February will have a duration 28 or 29 days.

The *name or number* a time unit instance has can be expressed in several ways. Similarly to durations, the name is relative to another time unit, a *reference time unit.* For example, a *day* instance can be named from *Sunday* to *Saturday* or numbered 1 to 7 if *week* is the reference time unit of *day.* (Names can be thought of as abbreviations of the numbers). More specifically:

Definition 17 (Reference time unit) *Let R and T be two time unit classes such that instances of T can be given names or numbers within R, then R is called a* reference time unit *for T. The characteristic a time unit must have to be a reference of another one is that the reference decomposes into the latter.*

Instances of time units which do not have any reference time unit are numbered relative to a conventional *reference point* or *zero point* of the calendar. A zero point marks the beginning of an *Era.* For example, March 1994 is a specific instance in the Gregorian calendar, where 1994 accounts for 1994 years from the beginning of the Christian Era.

The concept of *reference* (as opposed to *reference time unit*), adds the possibility of reference time points and reference time unit instances:

Definition 18 (Reference of a time unit) *Let T be a time unit. There are three possible naming or numbering* references *for T. A reference can be another time unit class. A reference can also be a specific time unit instance, (which would be an instance of a reference time unit class). Finally a reference of a time unit can be a conventional reference point. If a time unit T is top-most in the time unit hierarchy (and therefore has no reference time unit), its reference is a conventional reference point.*

For example, a possible reference time unit for *days* is *week*, and within this reference, a possible instance is *Monday*. A possible reference time unit instance for *days* is a *specific year*, for example *year 1994*, and within this year a possible instance is *day 147*. Finally the *Christian Era* acts as a reference point for *year*.

One time unit can have more than one reference time unit. Referring again to the previous examples, days can be numbered within a *week* or within a *year*. Therefore, the *domain of possible names or numbers* within a reference (i.e. *all the possible* names or numbers a time unit can have) is in fact an attribute of the relation between the reference and the time unit class. A *main* reference is defined for a class, which is clearly an attribute of a time unit class (and not of the relation).

Definition 19 (Main reference of a time unit) *There are several references possible for the same time unit. One is selected as a* main reference.

For example, *day* has *year* as the main reference (time unit), *year* has the *Christian Era*, (or zero-point of the Gregorian calendar) as the main reference (time point).

Sequences of names and cardinality

All the possible names or numbers the instances of a time unit can take within a reference can be expressed extensionally by a sequence of names (if there are names associated to the time unit class). Names are indeed abbreviations of instance numbers. For example *days* can be named with a name from the sequence {*Sunday, Monday, ..., Saturday*} within a (reference) *week*. Other time units don't have names associated to them, but only numbers. For example *days* within *year* are numbered from 1 to 366 but not named.

A pair of numbers can represent the range of maximum possible instance numbers of a class. For example, the range of maximum possible *days* within a *month* is $(28,31)$. In case of time unit instances referable by names, this range represents the maximum cardinalities of the sequence of all possible names. For example, the sequence of *day* names within *week* is {*Sunday, Monday, ..., Saturday*} , with cardinality 7. (In this example the range of possible maximum numbers $(7,7)$ contains only one value, since *every* week lasts 7 days.)

A function associates a time unit class and a reference with the sequence of all possible instance names (*name*). This is a partial function, since not all time units have names associated. Another function associates a time unit class and a reference with the range of possible numbers (*number*). *name@* and *number@* are respectively the functions that associate name and number to a specific time unit instance and a reference.

Instances values and origin of counting

A particular instance name or number of a time unit within a reference reflects the relative position of the instance within the reference, given a certain *origin* where counting of instance values starts. For example, months are counted from 1 to 12 in a year in the Gregorian calendar, starting from 1, (or equivalently they are named {*January, February,..., December*} starting from *January*). But, in case of months counted within a *school year*, the origin of instance numbering is not the month number 1, but rather the 9th (or *September*), so the 3rd month of a *school year* is the 11th month of all the possible months name sequence, (or *November*). *Circular counting* is assumed. Following the example, *Month* 1 comes after *Month* 12, (or *January* after *December*). It should be noted that *Month* 1 (or *January*) is a possible number (name) for a month within a *school-year*, it is just that it will not be the first month within the school-year. In fact *January* is positioned 5th with respect to school-year.

We define one function that associates the origin of instance name (or number) counting to a time unit class and a reference (*origin*). For example, *origin* maps *month* and the reference *year* to the origin position 1, whereas *origin* maps *month* and the reference *school-year* to the origin position 9. Another function maps a time unit class, a reference and a relative position to the instance name (or number) in that relative position (*value-wrt-rel-position*). Thus for example, *value-wrt-rel-position* maps *month*, the reference *year* and the 5th relative position to *May*. On the other hand, *value-wrt-rel-position* maps *month*, the reference *school-year* and the 5th relative position to *January*. Precise definition of these functions and more examples appear below.

3.3.1 Connection between durations and instance values

Possible lengths of a time unit A are measured in units of another time unit B, which acts as a *measure* time unit. Possible instance names or numbers of a time unit B are those relative to a *reference* A. In fact the two notions express somehow the same concept: in the first case the concept is taken from the point of view the "decomposition" relation, in the latter one from the point of view of a "composition" relation between two time units. In the first case, B is seen as the measure time unit for A, in the second case, A is seen as the reference for B.

For example, *day* has as possible instance numbers (within a *week*) the range from 1 to 7, and *week* has a duration of $(7,7)$ days. Days have as possible numbers 1 to 365 or 366 within a *year*, the class *year* has a duration of $(365,366)$ days, and a specific year has a duration of 365 or 366.

Possible implementations can profit from this inter-relation; however conceptually they have their own characteristics. Furthermore, there is a difference which also justifies the existence of two series of definitions and concepts. The correspondence of definitions does not apply when time unit instances of B are numbered with respect to a conventional reference point A. There is no such thing as the relative duration of a reference point with respect to a measure time unit.

3.3.2 Functional definition of instance values

The following functions are related to the numbering or naming of time units and time unit instances: *number*, *number@*, *name*, *name@*, *origin* and *value-wrt-rel-position.*

Definition 20 (Instance numbers functions) *Functions associating instance numbers to time units and time unit instances.*

1. number *is a function that associates a min-max pair to a time unit and a reference. The reference can be either another time unit, a time unit instance or a conventional reference point. In fact this pair is exactly the same as the duration*

of the reference in terms of the time unit, when the reference is a time unit or a time unit instance(i.e., not a reference point). In general, the pair reflects the range of possible maximum values of instances. That is to say, it indicates the minimum and maximum cardinality of the instance values domain within the reference.

2. number@ *is a function that associates a number to a specific time unit instance within a reference.*

More precisely, let
TUS be a set of time units in a certain calendar and its variants,
TUINS be the associated set of time unit instances,
REF-POINT be a set of conventional reference points, and
REFS = TUS ∪ TUINS ∪ REF-POINT,

$$\mathit{number}: \text{TUS} \times \text{REFS} \longrightarrow N \times N$$
$$\mathit{number@}: \text{TUINS} \times \text{REFS} \longrightarrow N$$

Let T be a time unit and $t@$ be a time unit instance of T,
let R be a reference of T;

$$\mathit{number}(T, R) = (\text{min-card}, \text{max-card}) \Longleftrightarrow_{\text{def}}$$
$$\mathit{number@}(t@, R) \in \{1, \ldots, \text{max-card}\}$$

Examples

$month \in$ TUS,
$February \in$ TUS, (subclass of $month$),
$February$ 1994 $\in$ TUINS,
$Christian\text{-}era \in$ REF-POINT

$number(day, month) = (28, 31)$
$number(day, February) = (28, 29)$
$number(day, February\ 1994) = (28, 28)$
$number(day, year) = (365, 366)$
$number(day, year\ 1994) = (365, 365)$
$number(month, year) = (12, 12)$
$number(month, school\text{-}year) = (12, 12)$
$number(month, century) = (1200, 1200)$
$number(year, Christian\text{-}era) = (3000, 3000)$ (3000 is an application-dependent bound)

if m is a month and $number@(m, year) = 3$, then m is an instance of March.

Definition 21 (Instance names functions) *Functions associating instance names to time units and time unit instances. These are partial functions, since not every time unit instance has a name associated to it.*

1. name *associates all possible instance names to a time unit and a reference. The possible instance names are conceived as a sequence and not as a set, since their ordering is relevant, and indicates the relative position of the instance within the reference.*

2. name@ *associates the name to a specific time unit instance within a reference.*

More precisely, let
TUS be a set of time units in a certain calendar and its variants,
TUINS be the associated set of time unit instances,

REF-POINT be a set of conventional reference points, and
REFS = TUS ∪ TUINS ∪ REF-POINT,

$$name : \text{TUS} \times \text{REFS} \longrightarrow \text{Set of sequences of instance names}$$
$$name@ : \text{TUINS} \times \text{REFS} \longrightarrow \text{Set of instance names}$$

Let T be a time unit and $t@$ be a time unit instance of T,
let R be a reference of T;

$name(T, R) =< name_1, \ldots, name_{\text{max-card}} > \Longleftrightarrow_{\text{def}}$
$name@(t@, R) = name_i,\ with\ 1 \leq i \leq$ max-card $\Longleftrightarrow_{\text{def}}$
$number(T, R) =$ (min-card, max-card)

Examples

week, *year*, *school-year* $\in$ TUS,

$name(day, week) =< Sunday, \ldots, Saturday >$
$number(day, week) = (7, 7)$

$name(month, year) =< January, \ldots, December >$
$name(month, school\text{-}year) =< January, \ldots, December >$
$number(month, year) = (12, 12)$
$number(month, school\text{-}year) = (12, 12)$

$name@(a\ Sunday\ day, week) = Sunday$
$number@(a\ March\ month\ of\ any\ year, year) = 3$
$name@(a\ March\ month\ of\ any\ year, year) = March$

Definition 22 (Instance values counting) *Function associating origin of instance number counting to a time unit class and function associating a specific instance given a relative position.*

1. origin *associates a relative position from where counting of instance values start to a time unit and a reference.*

2. value-wrt-rel-position *is a function that associates a specific instance of a time unit given a reference and a relative position. This relative position is evaluated from the origin where counting starts.*

More precisely, let TUS be a set of time units and TUINS be the associated set of time unit instances.

$origin : \text{TUS} \times (\text{TUS} \cup \text{TUINS}) \longrightarrow N$

$value\text{-}wrt\text{-}rel\text{-}position : \text{TUS} \times (\text{TUS} \cup \text{TUINS}) \times N \longrightarrow N$

Let T be a time unit and R be a reference of T;

$value\text{-}wrt\text{-}rel\text{-}position(T, R, i) = (i + origin(T, R) - 1) \bmod \text{card}$,

where $\text{min-card} \leq \text{card} \leq \text{max-card}$

Examples

$origin(month, year) = 1$

$value\text{-}wrt\text{-}rel\text{-}position(month, year, 3) = (1 + 3 - 1) \bmod 12 = 3$

$origin(month, school\text{-}year) = 9$

$value\text{-}wrt\text{-}rel\text{-}position(month, school\text{-}year, 3) = (9 + 3 - 1) \bmod 12 = 11$

$value\text{-}wrt\text{-}rel\text{-}position(month, school\text{-}year, 6) = (9 + 6 - 1) \bmod 12 = 2$

Connecting instance numbers and durations

The connection between the notions of relative duration and numbering within a reference can be expressed with the functions defined for the case when a time unit is named or numbered within a time unit or time unit instance (and not a reference point):

Property 1 (Connecting duration and instance value functions) *Let R be a time unit reference of T and $R@$ an instance of R. Then the number of an instance of T within a reference R (or $R@$) is contained in the same range as the duration of R (or $R@$) in terms of T.*

More precisely,

$number(T, R) = (a, b) \quad iff \; dur(R, T) = (a, b).$

$number(T, R@) = (d, d) \quad iff \; dur@(R@, T) = d.$

Examples

$number(day, year) = (365, 366) \quad iff\ dur(year, day) = (365, 366).$

$number(day, year1994) = (365, 365) \quad iff\ dur@(year1994, day) = 365.$

3.4 Summary - attributes of a time unit

As a summary of the description about attributes of time units, several time units from the Gregorian calendar and from a university calendar are presented in Table 3.1 and Table 3.2. The correspondence between relative duration and possible instance numbers above discussed can be observed in Table 3.2. The general duration is the duration relative to the same time unit (*day*) for all the time units in the table.

Appendix B summarizes main concepts related to time units.

Time Unit	(general) Duration	Main reference
year	(365,366)	zero-point
month	(28,31)	year
week	(7,7)	year
day	(1,1)	week
school-year	(364,371) [1]	zero-point
semester	(119,133)	school-year

Table 3.1: Examples of attributes of some time units

[1]The general duration of the school year is calculated in terms of number of weeks, and is therefore a loose bound of it.

Reference	Measure TU	Relative duration	Possible instance values	Origin
year	*month*	(12,12)	{Jan, Feb,..., Dec}	1
year	*week*	(53,53)	1 to 53	1
year	*day*	(365,366)	1 to 366	1
month	*day*	(28,31)	1 to 31	1
Christian-era	*year*	—	1 to 3000 (application)	(appl.)
year 1994	*day*	(365,365)	1 to 365	1
February	*day*	(28,29)	1 to 29	1
February 1994	*day*	(28,28)	1 to 28	1
school-year	*semester*	(3,3)	{Fall, Spring, Summer}	1
school-year	*month*	(12,12)	{Jan, Feb,..., Dec}	9
semester	*week*	(17,19)	1 to 19	1

Table 3.2: Examples of attributes related to durations and instance names or numbers of time units.

Chapter 4

Decomposition of time units

- Y qué necesidad de cambiar de año? Acaso este no sirve? Total, son todos lo mismo, no?
- No, lo mismo no, este fue bisiesto, así que el 65 *tendrá un día menos.*
- Ah! Encima nos encajan un año de inferior calidad!

- And what's the need of changing the year? Isn't this one useful? Aren't they all the same thing?
- No, not the same. This one was a leap year, so year 65 *will have one day less.*
- Ah! So on top of that, they impose on us a lower quality year!

Mafalda questioning her mother, in Mafalda inédita, by Quino.

The primary relation among time units is that of *decomposition.* When A decomposes into B, A will be referred to as the *composed* time unit, and B will be referred to as the *component* unit. For example, a *year* decomposes into *months* and a *month* into *days.* Also a *month* decomposes into *weeks*, a *week* into *days*, etc. Clearly there are different kinds of decompositions: a year decomposes exactly into 12 months, whereas a month decomposes in a non-exact way into weeks – between 4 and 6, where the extreme weeks of the month may be complete or incomplete weeks. Also, the number of components can vary: months decompose into different numbers of days depending on the specific month. In the case of February, the number of days even depends on specific *month-instances.* We propose that all these variations in the decomposition relation can be captured with two different aspects of the relation: *Alignment* and *Constancy.*

First we present intuitions of these two aspects and later we present them in a formal way, as part of the definition of the decomposition relation. Decomposition between time units is defined as a particular case of decomposition between classes of intervals. (We claim and justify that decomposition is the main relation of interest between time units.) Definitions therefore will combine those concepts of abstract decomposition introduced in Chapter 2 as well as concepts associated to time units and time unit instances, developed in Chapter 3. The last part of this chapter combines ideas related to time units, particularly their durations, with the decomposition relation. The main result of this formal development is a group of theorems where decomposition is shown to be a partial order on the set of time units.

4.1 Introduction: Aspects of the decomposition relation

4.1.1 Alignment

A time unit may decompose into another in an *aligned* or *non-aligned* fashion. The decomposition is *aligned* just when the composed time unit starts exactly with the first component and finishes exactly with the last component. Consequently, a certain number of *complete* components fit exactly into the composed time unit. Examples of aligned decompositions include *year* into *months*, *month* into *days*, and *week* into *day.* A time unit decomposes into another in a *non-aligned* way when the composed time unit does not start exactly with the first component time unit and/or does not finish exactly with the last component time unit. Examples of non-aligned decompositions include *year* into *weeks* and *month* into *weeks.* A graphical picture of an aligned and non-aligned decomposition can be seen in Figure 4.1. This figure in fact looks exactly like Figure 2.1 in Chapter 2. The one in the previous chapter was introduced for time unit instances, this one associated to time unit classes.

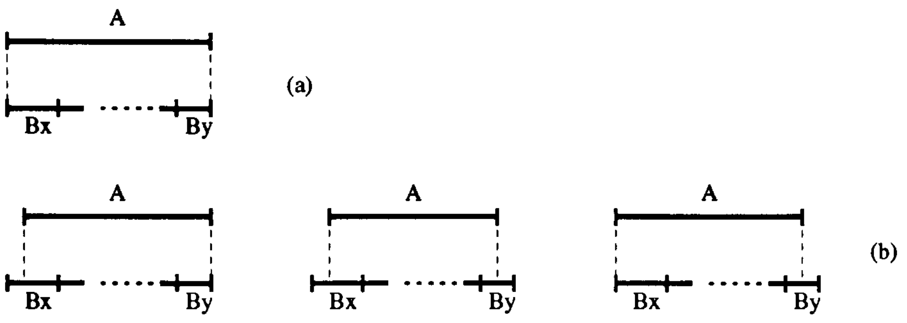


Figure 4.1: Graphical representation of Alignment

4.1.2 Constancy in the number of components

A time unit may decompose into another in a *constant* or *non-constant* fashion. The decomposition is *constant* when the component time unit is repeated a constant number of times for *every* time unit instance. Examples of constant decompositions include *year* into *months* and *week* into *days*, since any year decomposes into 12 months and any week decomposes into 7 days. A time unit decomposes into another in a *non-constant* way it decomposes into different numbers of component time units. Examples of non-constant decompositions include *month* into *days* and *year* into *days*.

A *month* decomposes into 31 or 30 days depending on which named-month it is, for example respectively January and April. It can also depend on the specific instance of the *month*, it can decompose into 28 or 29 days if it is February, depending on whether the year is a normal year or a leap year respectively.

4.1.3 Combining the two aspects of the decomposition

There are four possible combinations resulting from these two aspects of alignment and constancy. These combinations cover examples from the various calendars analyzed. Some calendar structures are graphically represented in Chapter 5, Figures 5.1 and 5.2. These graphs show how the combination of these two aspects provides enough expressive power to describe decomposition alternatives between time units.

The constancy aspect of decomposition also covers the very important attribute of a time unit, its (general) duration. Recall that the general duration of a time unit has been defined as a relative duration where the measure unit is a basic time unit in the calendar. Furthermore, a time unit has been classified as of *equal durations* or *non-equal durations* based on its general duration. The relation of these definitions and the constancy aspect of decomposition is straightforward; a time unit is of equal duration if it decomposes in a constant way into the basic time unit. Non-equal duration is the parallel concept to non-constant decomposition into the basic time unit.

One goal of our work is to represent relationships encountered in any system based on discrete units and holding a repetitive containment relation among them. (Arguably) all the relationships of interest in such systems are covered by the variants resulting from combining alignment and constancy of decomposition.

One may wish to consider the decomposition of time units into different *types* of time units – for example the astronomical year decomposing into a special *universal-year* of 365 days and a *universal-day*. Again, constancy and alignment cover this variation, adding no further complication to the formal apparatus, since it is effectively an elaboration of a decomposition into a non-equal durations time unit. However, a detailed analysis is beyond the scope of this thesis.

4.2 Decomposition between time units

Decomposition relations were defined in Chapter 2 in an abstract fashion, based on the interval algebra basic binary relations. In general terms, two main levels of abstraction were distinguished: *intervals on the time line* and *classes-of-intervals*, which constitutes a higher level of abstraction introduced in this thesis. Correspondingly, decomposition relations have been defined at both levels, at the instance level and at the class-of-intervals level. In Chapter 2 no special characteristic is imposed on the classes-of-intervals. Relations between them result from a universal/existential generalization from the instance level. The following sections in this chapter define decomposition between special classes-of-intervals; time units.

4.2.1 Contiguous sequences definitions

Contiguous sequences of time unit instances and time unit classes will be defined based on concepts and definitions related to numbering (or naming) of time unit instances. These definitions respectively extend those of contiguous sequence of intervals and contiguous sequence of classes of intervals.

Definition 23 (Contiguous sequence of time unit instances) *A contiguous sequence of time unit instances of the same time unit class is a* contiguous sequence of intervals, *where the intervals are instances of the same time unit class, named or numbered with respect to a reference. The number of time unit instances in the sequence is referred as the* repetition factor *of the sequence.*

Since it is a sequence of instances, (and not classes) the reference for numbering them is a reference time unit *instance* or a conventional reference point (and not a time unit class).

Formally, a contiguous sequence of time unit instances can be denoted as follows: Let T be a time unit class, let $R@$ be a reference time unit *instance* of T (or conventional reference point)
$T@_{R@}^{(s,r)} =< t@_s, \ldots, t@_{s+r-1} >$ is a contiguous sequence of r intervals, where each interval $t@_i$, with $s \leq \ i \leq \ s+r-1$, is a time unit instance of T numbered or named with respect to $R@$.

It should be noted that in a contiguous sequence of time unit instances, names or numbers of the instances relative to the reference will be in increasing order. When the reference $R@$ is a time unit instance (as opposed to reference point), this is a special restricted case of a contiguous sequence of intervals. In this case the intervals in the sequence are time unit instances of T that span over at most *one* time unit instance, the reference $R@$. The repetition factor is then bounded by the cardinality of the domain of T numbered within the reference time unit instance $R@$. For example, there will not be more than 12 specific months within one specific year.

This can be expressed using the function *number* which associates the minimum and maximum cardinality of the domain of T numbered within a reference, defined

in Section 3.3.2:
If *number*$(T, R@)$ = (min-cardinality, max-cardinality), then $1 \leq r \leq$ max-cardinality. (As a detail, since $R@$ is an instance, min-cardinality = max-cardinality).

Examples

The following are examples of contiguous sequences of time unit instances within one reference time unit instance.

1. $month@^{(4,3)}_{Year1994}$ = <*April 94, May 94, June 94*>
2. $day@^{(1,20)}_{May1994}$ = <*May 1 1994, ..., May 20 1994*>
3. $year@^{(1990,4)}_{Christian\ Era}$ = <*year 1990, ..., year 1993*>

Specific time intervals which span more than one specific time unit instance, as for example days numbered with respect to two different, specific months, (for example from May 15 1994 until June 15 1994) are also contiguous sequences of day instances. The representation of this is suggested in a later chapter. The definition of contiguous sequences of time unit instances as presented so far acts as a preamble for the definition of contiguous sequences of time unit classes. Classes in these contiguous sequences are *named* or *numbered* time units.

Definition 24 (Named (or numbered) time unit) *A named (or numbered) time unit is a subclass of a time unit. All the instances in this subclass have a unique instance name (or number) with respect to some reference time unit. A named time unit is not yet a specific instance, rather it represents a set of specific instances (all with the same name).*

Since a named (or numbered) time unit class has a unique name (or number) by definition, the function that associates a unique number or name to a time unit instance can be applied to it. (even though a named time unit is a subclass of the time unit and not yet an instance). The property defining a named (or numbered) time unit can then be expressed as:

1. if names are associated to a time unit,
   $\forall$ time unit instance $t@$ of a named time unit Named-T,
   there exists a name m common to all the instances in the subclass:
   $\exists$ m $name@(t@, R)$ = m = $name@(Named\text{-}T, R)$

2. $\forall$ time unit instance $t@$ of a numbered time unit Numbered-T
   $\exists$ n $number@(t@, year)$ = n = $number@(Named\text{-}T, R)$

Examples of named time units

- Named-*month* relative to *year*: *February*
  (extensionally it represents the set of all the specific instances of the months February)
  $number@(subclass\ of\ February, year) = 2$
  $name@(subclass\ of\ February, R)$ = "February"

- Named-*day* relative to *week*: *Monday*

- Named-*day* relative to *month*: *First day of a Month*

Definition 25 (Contiguous sequence of named time units) *A contiguous sequence of named time units is a contiguous sequence of subclasses of the same time unit class. More precisely, it is a contiguous sequence of named time units of the same class. Names or numbers are relative to a reference (which is a time unit class). Numbering (or naming) is cyclic. The number of classes in the sequence is referred as the repetition factor.*

Formally this can be expressed as follows:
Let T be a time unit class, let R be a reference of T.
$T_R^{(s,r)} =< T_s, \ldots, T_{s+r-1} >$ is a contiguous sequence of r subclasses of T, where each subclass T_i, with $s \leq i \leq s+r-1$, is a named time unit of T numbered or named (in cyclic, increasing order) within R.

The fact that numbering (or naming) of the components of the sequence is cyclic can be expressed using the functions defined as follows:
Let (min-cardinality, max-cardinality)= $number(T, R)$
$number@(T_i, R) = number@(T_{i+card}, R)$,
$name@(T_i, R) = name@(T_{i+card}, R)$,
where min-cardinality $\leq$ *card* $\leq$ max-cardinality.

Examples

The following are examples of contiguous sequences of named time units.

1. $month^{(4,3)}_{Year}$ = *<April, May, June>* = *<month4, month5, month6>*
2. $day^{(1,20)}_{Month}$ = *<day1, ..., day20>*
3. $month^{(4,15)}_{Year}$ = *<April, May, ..., December, January, ..., April, ..., June>*
4. $month^{(12,3)}_{Year}$ = *<December, January, February>* = *<month12, month1, month2>*

As the examples above show, the repetition factor is not restricted by cardinality bounds in case of contiguous sequence of classes (as opposed to contiguous sequences of instances). The 3rd example is a sequence of 15 months named relative to *year*. The cyclic nature of numbering can be seen in examples 3 and 4. Furthermore, cyclic counting is important in the definition of time unit classes having the origin of instance number counting different than 1, like for example *months* numbered within a *school-year*.

4.2.2 Decomposition, relation of interest between time units

Having defined contiguous sequence of *time unit* classes as special classes of intervals other relations between classes of intervals can be defined for this particular case.

The main relation we will define between time units is that of decomposition. We argue that when classes of time intervals are time *units*, decomposition is sufficient to cover all relations of interest between them. That is to say, time units are related by a special inclusion relation and not by any precedence relation. We state this fact in the following claim:

Claim 1 *Time units do not relate with precedence relations. As a consequence, the possible binary interval relations that will hold between time units are those characterizable with inclusion.*

These ideas can be intuitively explained as follows: time units are classes of time intervals. Hence a time unit does not precede other time units; rather it is specific instances that precede other instances. For example, the year 1993 is before 1994, but *year* is not before nor after *month*; rather *year* can be decomposed into a sequence of *months*. Precedence among subclasses of time units as opposed to specific time unit instances (for example, *Monday* as opposed to *Monday, May 16 1994*) is not straightforward. For example, it could be said that *Mondays* precede *Tuesdays*, but this is true if there is an implicit assumption that both belong to the same week. In this case however, the ones compared are in fact specific *time unit instances*. Therefore, this issue does not alter the assumption taken in this research: time units do not relate with precedence relations. Precisely, one of the axiomatizations of the interval algebra is that of considering two main kind of relations: precedence and inclusion. Given that precedence relations do not correspond, decomposition, an inclusion relation should be enough to cover the interesting relations among time units.

The situation of time units with the same duration but a "different origin", like for example *year* in the Gregorian calendar and *school-year* in a university calendar is not of decomposition, but rather overlapping. (In the generalized class level sense defined *year .OVERLAPS. school-year.*) It should be noted that time units in the same calendar don't present this situation. All time units in the same calendar have different durations; decomposition accounts for the main interesting relation between them.

But even in case of variant time units, decomposition plays an important role, determining their relation. Variant time units are different ones, but closely related. Both *year* and *school-year* decompose into *month*, and they differ in the origin of instance number counting, as was analyzed in Chapter 3.

4.2.3 Definitions, alignment and constancy

Two main characteristics are the ones distinguished in decomposition of time units: alignment and constancy.

The aligned and non-aligned distinction of decomposition concerns the specific instance level; we can conceive a specific year decomposing in an aligned way or non-aligned way into specific months or specific weeks. Also, alignment is a characteristic to consider in the class level. For example year as a class can decompose in an aligned or non-aligned way into other time unit classes. Constancy, however, is a matter that concerns only the class level, since it is related to *all instances of the class* decomposing into the same number of subintervals or not.

Aligned and non-aligned decompositions of a time unit class into a sequence of time units are definable based on the general case (aligned and non-aligned decomposition into a sequence of classes of intervals, in Chapter 2). Furthermore, also based on the general definition of decomposition between two classes in that chapter, the decomposition relation between *two* time unit classes can be defined.

Binary decomposition between two time unit classes will be denoted with a special symbol,($\trianglerighteq$), resembling a symbol associated to an order relation. Indeed, a main theorem about the decomposition between time units is that this relation is a partial order on this set. Alignment and Constancy will be expressed as binary predicates holding between time units. Alignment is definable directly based on the abstract decomposition definitions in Chapter 2. Constancy is formally defined as a property that depends on the repetition factor.

Definition 26 (Decomposition relation between time units) *Let A and B be two time units in the same calendar or variant of the calendar. ($A, B \in TUS_{CAL}$). A decomposes into B, written $A \trianglerighteq B$, iff A .DECOMPOSES-INTO. B.*

A is called the *composed time unit* and B is called the *component time unit.* Decomposition of time units can be aligned or non-aligned into a contiguous sequence of subclasses of the component time unit. The repetition factor of the contiguous sequence is also considered the repetition factor of the decomposition.

Alignment of decomposition

Definition 27 (Aligned and non-aligned decomposition of two time units) *Let A and B be two time units in the same calendar or variant of the calendar. ($A, B \in TUS_{CAL}$)*

- *A decomposes into B in an aligned way (i.e. Alig(A,B)) iff A decomposes aligned into a contiguous sequence of named B's, i.e. A.DEC-ALIG-INTO-SEQ. $< B_1, \ldots, B_r >$. The number r is the repetition factor of the sequence and of the decomposition.*
- *A decomposes into B in a non-aligned way (i.e. Non-Alig(A,B)) iff A decomposes non-aligned into a contiguous sequence of named B's, i.e. A.DEC-NON-ALIG-INTO-SEQ. $< B_1, \ldots, B_r >$. The number r is the repetition factor of the sequence and of the decomposition.*

Examples

year $\trianglerighteq$ month,	*Alig(year,month)*
	(*year*.DEC-ALIG-INTO-SEQ. $< January, \ldots, December >$).
month $\trianglerighteq$ day,	*Alig(month,day)*
week $\trianglerighteq$ days,	*Alig(week,days)*
year $\trianglerighteq$ week,	*Non-Alig(year,week)*
month $\trianglerighteq$ week,	*Non-Alig(month,week)*

Theorem 8 *Aligned and non-aligned are two mutually exclusive relations. Let $A, B \in TUS_{CAL}$, such that $A \trianglerighteq B$. Alig(A,B) iff not Non-Alig(A,B).*

Proof: This follows from the analogous theorem about aligned decomposition between general classes of intervals (Chapter 2). ■

Constancy of decomposition

Definition 28 (Constant and non-constant decomposition of two time units)

Let A and B be two time units in the same calendar or variant of the calendar. $(A, B \in TUS_{CAL})$

1. *A decomposes into B in a constant way with a* unique *repetition factor r (i.e.Cons(A,B,r)) iff every A-instance decomposes into a contiguous sequence of r named B's.*

2. *A decomposes into B in a non-constant way (i.e.Non-Cons(A,B,(lower-r,higher-r))) iff*

   - *Every A-instance decomposes into a contiguous sequence of r named B's, where the number r is the repetition factor of the contiguous sequence (and extending notation, the repetition factor of the decomposition). The repetition factor is bound such that lower-r $\leq$ r $\leq$ higher-r and*
   - *There exist at least two A-instance decompositions into B with a different repetition factor.*

Examples

year $\trianglerighteq$ month, *Cons(year,month,12)*
week $\trianglerighteq$ days, *Cons(week,days,7)*

month $\trianglerighteq$ day, *Non-Cons(month,day,(28,31))*
year $\trianglerighteq$ day, *Non-Cons(year,day,(365,366))*

The repetition factor can be considered as a min-max pair (lower-r,higher-r) in both cases, constant and non-constant decomposition. In the constant situation, *lower-r* = *higher-r*. This pair corresponds exactly to the duration of the composed time unit in terms of the component one, as the following property expresses:

Property 2 (Equivalence of repetition factor and duration) *Let A and B be two time units in the same calendar or variant $(A, B \in TUS_{CAL})$. A decomposes into B with a repetition factor (lower-r,higher-r) iff $dur(A, B)$ = (lower-r,higher-r)*

The following property relates the equal or non-equal duration of a time unit with the constancy aspect, as was already discussed intuitively. This property outlines the fact that a time unit is of equal durations if it decomposes into the basic time unit (measure for general durations), in a constant way. A time unit is of non-equal durations if it decomposes in a non-constant way into the basic time unit.

Property 3 *If a time unit A decomposes into the basic time unit of a calendar in a constant way, then A will be of equal durations. I.e., all A-instances will have the same general duration.*

If a time unit A decomposes into the basic time unit in a calendar in a non-constant way, then A will be of non-equal durations. I.e., there exist A-instances with different general durations.

Proof: This property follows from the definition of general duration and constancy of the decomposition relation. ■

4.3 Connecting decomposition and durations

Definitions of time units, time unit instances, their durations and decomposition relation were done to formalize the following idea: when a time unit decomposes into another, the composed time unit instances are all of a larger duration span than any of the component time unit instances. Based on the fact that this holds, the formalization presented in this thesis is consistent with the fact that decomposition is a *partial order* on the set of time units. Because decomposition on time units is a partial order, a hierarchy is definable, based on it. This is the structure this thesis proposes as the basic structure time units relate. Moreover, the time unit hierarchy and its properties will lead to other alternative structures which show how time units relate.

The rest of the chapter is organized as follows. First, some properties that derive from definitions and assumptions taken so far are presented. A central theorem in this thesis is presented next, relating duration and decomposition of time units. The precise proof of the theorem appears in Appendix A.2. Corollaries from this theorem lead to the theorem proving that decomposition is a partial order.

4.3.1 Decomposition: a partial order

Property 4 *Let A and B be two time units in the same calendar or variant ($A, B \in TUS_{CAL}$). Then they have the same duration only if they are the same time unit or if they are variant time units. I.e., If $dur(A, Basic) = dur(B, Basic)$ then $A = B$ or A and B are variant time units.*

The assumption taken in this work is that different time units in the same calendar have different (general) durations. Recall that a general duration is the one measured relative to a "Basic" time unit in the calendar, i.e. in the hierarchy associated to TUS_{CAL}.

Time units with the same duration but "different origin" are not considered to be different time units, but rather *variant time units*, as for example *year* in the Gregorian calendar and *school year* in a university calendar.

Property 5 *Variant time units are not related by decomposition but rather by the .OVERLAPS. relation. The relation is determined by decomposition.*

This issue has been discussed previously (in section 4.2.2). It appears here as a property to stress that the next theorem 9, deals with two time units that decompose into each other. This includes time units within the same calendar (i.e. all time units in the Gregorian calendar, for example *year* into *day*). It also includes time units from variant calendars. For example *school-year* from a university calendar decomposes into *month*, in the Gregorian calendar. The case that is not included in the theorem, because they do not decompose into each other, is that of variant time units. For example, *year* and *school-year*. However, they both decompose into *month*, and the relation between them is precisely based on how *month-instances* are numbered.

Property 6 *A time unit decomposes into itself only in a constant aligned way, with repetition factor* 1.

The only situation that a time unit can decompose into itself is if every instance of the composed time unit is exactly the same as the (single) instance of the component time unit. There is only one possibility in this situation, a constant/aligned decomposition with repetition factor one.

Theorem 9 (Decomposition and duration) *Let A and B be two time units in the same calendar or variant ($A, B \in TUS_{CAL}$) such that A decomposes into B ($A \trianglerighteq B$). Then A has a greater or equal general duration than B.*

Considering that durations are min-max pairs, this theorem can be reformulated using the order relation defined between min-max pairs:
Let A and B be two time units in the same calendar or variant ($A, B \in \mathrm{TUS}_{CAL}$) such that A decomposes into B (A $\trianglerighteq$ B), then $dur(A, Basic) \overset{\mathrm{P}}{\geq} dur(B, Basic)$. "Basic" is the time unit used to measure general durations in the hierarchy associated to TUS_{CAL}.

Proof: We mainly give here an idea of the the proof of this theorem and specially the implications of it in the corollaries. The precise proof of this theorem appears in Appendix A.2.

Rephrasing the theorem statement, we restrict the parameters of the decomposition relation so that if A decomposes into (a sequence of) B's, then any A instance has a larger time span than any single B instance. The idea we follow is to restrict the time units and decomposition parameters so that the theorem holds. Specifically, we relate the duration of the component time unit and the repetition factor of the decomposition. The restrictions are not arbitrary, they reflect the essence of decomposition of a unit into a contiguous sequence of another unit. An example of the kind of restrictions is that no time unit can decompose into a single occurrence of an incomplete time unit. That is, in case of a non-aligned decomposition, at least one complete occurrence of the component time unit must be part of the component sequence. ■

Corollary 10 *Component time units do not contain composed time units.*

Proof: In this case of decomposition of time units and not just any class of intervals, the constraint imposed in the decomposition relation about component classes not containing the composed class (f. ex. a day not containing a month) is not necessary. (This detail is taken into account in theorems 4 and 5, chapter 2 in case of decomposition between generic classes). The reason relies precisely on this previous theorem, which associates decomposition with duration, and the fact that duration min-max pairs are related with a partial order. ■

Corollary 11 *Decomposition is a particular case of an inclusion relation. Let $A, B \in TUS_{CAL}$. If A decomposes into B, ($A \trianglerighteq B$) then*

1. *B is contained or is equal to A, I.e. B .IN. A, where .IN. is the extension of the relation .in. defined between intervals to the class level.*
2. *A .IN. $B_A^{(x,y)}$ I.e the composed time unit is equal or is contained into the contiguous sequence of (complete) components.*

Proof:

1. B is contained or is equal to A (B .IN. A) - This is a direct consequence of the previous theorem. Refer to Figure 4.2 for a graphical view.
2. A .IN. $B_A^{(x,y)}$ - This is a consequence of decomposition being either aligned or non-aligned. The relation *.IN.* is defined as the union of *.EQUALS.* and *.STRICT-IN.* The relations *.EQUALS.* and *.STRICT-IN.* are precisely the relations that hold between A and $B_A^{(x,y)}$ in the aligned and non-aligned case respectively. ■

Theorem 12 (Decomposition - a partial order on time units) *The decomposition relation is a partial order on the set of time units in a calendar or a variant calendar.*

Proof: When a composed time unit A decomposes into a component time unit B, the composed time unit A has a bigger (general) duration than the component time

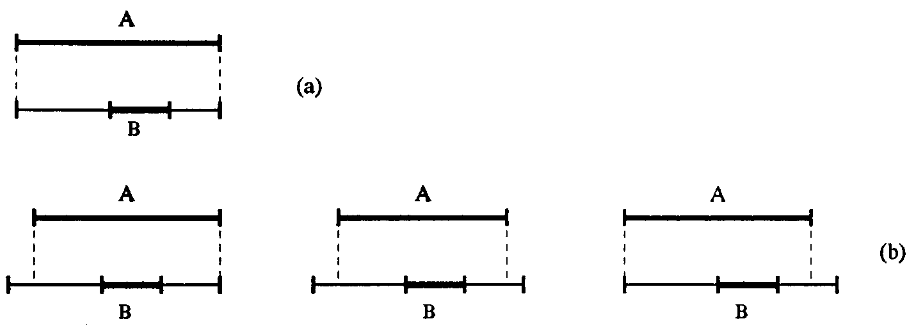


Figure 4.2: If A decomposes into a sequence of B's, then A contains B

unit B. This the main result just presented. I.e., for any $A, B \in$ TUS$_{CAL}$, A $\trianglerighteq$ B iff $\mathit{dur}(A, \mathit{Basic}) \overset{p}{\geq} \mathit{dur}(B, \mathit{Basic})$ This duration has been defined in terms of min-max pairs. These pairs relate with a partial order ($\overset{p}{\geq}$). Based then on the partial order between min-max pairs, decomposition is a partial order. ■

Or, another proof:

Proof: Decomposition is a containment relation. I.e, for any $A, B \in$ TUS$_{CAL}$, A $\trianglerighteq$ B iff B .IN. A, from theorem 6. Since containment is a partial order, decomposition is a partial order. ■

Chapter 5

Calendar Structures: time unit hierarchies

"And how many hours a day do you do lessons?" - said Alice in a hurry to change the subject.

"Ten hours the first day," - said the Mock Turtle: "nine the next and so on." "What a curious plan!" - exclaimed Alice.

"That's the reason they are called lessons, " the Gryphon remarked, "because they lessen from day to day."

This was quite a new idea to Alice, and she thought it over a little bit before she made her next remark. "Then the eleventh day must have been a holiday?" "Of course it was" - said the Mock Turtle. "And how did you manage on the twelfth?" Alice went on eagerly.

"That's enough about lessons" - the Gryphon interrupted.
Alice in Wonderland, Lewis Carroll

A calendar structure is basically a pair: a set of time units and a decomposition relation. A result proved in the previous chapter is that decomposition is a particular case of containment, and constitutes a partial order on the set of time units. Therefore, a calendar structure is defined as a containment *time unit hierarchy.*

The structure is defined so that *variants* of a specific calendar can be defined in terms of a basic one. Such would be the case of a university or business calendar, based on the Gregorian calendar. Such calendars have the same time units as the basic one, with possible new time units or a different conventional beginning point.

For example, many university calendars would have a *semester* time unit, where the school year begins in September.

The systematic analysis of the transitivity properties of the decomposition relation between the time units in the hierarchy gives rise to properties that hold in the structure. For example, we analyze the composition (or product) of two decomposition relations. Thus, if two aligned decomposition relations are multiplied, the resulting decomposition relation is also aligned. An interesting result arises when composing two decomposition relations which are non-constant. The result is a third relation which can be either constant or non-constant. But, the result of the composition operation is not arbitrarily constant or non-constant; rather, the resulting relation follows a pattern in the time unit hierarchy. *Periodic relations*, *cycles* and *chains* become apparent as a direct consequence of these characteristics of transitivity of alignment and constancy.

The sections in this chapter present the definition of calendar structures and examples of different calendars: the Gregorian, the SFU, the World or Universal and the Hebrew calendars. Graphs and tables with the attribute values are used to show these structures. Subsequently we study operations on decomposition. Transitivity of constancy and alignment is a result of this study. The hierarchy is then characterized structurally in terms of the constancy and alignment of decomposition, with cycles, higher and lower time unit bounds, etc. This study makes apparent a parallelism of decomposition in the hierarchy with divisibility concepts. Definitions and properties highlighting this parallel are developed. We then define the concept of chains of time units as an alternative structural characterization of the hierarchical structure. This characterization provides for a different point of view, and we envision that it constitutes a basis for a formal language of time units. In the final section computational aspects of using the calendar structure are addressed. In addition a particular notation is suggested to represent time unit instances which exposes the expressive power of the developed formalism.

5.1 Definitions and examples of calendar structures

Definition 29 (Calendar structure or time unit hierarchy) *A* calendar structure or time unit hierarchy *is essentially a poset, a pair composed of a set (of time units) and a (decomposition) relation, which is a partial order on the set. The pair will be written as:* $< TUS_{CAL}, \trianglerighteq >$. *As it was previously expressed,* TUS_{CAL} *denotes the set of time units in a calendar* CAL *and its variants.*

Since calendar structures are partial orders, they can be represented by a directed acyclic graph. The set of nodes in a calendar structure represents the set of time units. Edges represent the decomposition relation.

Definition 30 (Readings of an edge in the time unit hierarchy) *An edge in the time unit hierarchy is directed from a time unit* A *to a time unit* B *iff* A *decomposes into* B, *(written* $A \trianglerighteq B$*). This can also be read in other ways:* A *is greater than* B, *or* A *is higher in the time unit hierarchy than* B, *or* A *is a predecessor of* B.

Depending on the application, a certain level in the time unit hierarchy is defined as the lowest level in the hierarchy. Time units in that level were defined as *atomic* in Chapter 3. A non-atomic time unit decomposes into one or more time units. An atomic time unit does not decompose.

In terms of the time unit hierarchy, the level of granularity desired prunes the structure so that lower level time units than the atomic ones are not taken into consideration. In the same way, high level time units can be pruned for a particular application, so that only the units involved in the application are included in the time unit hierarchy.

Definition 31 (Height of the time unit hierarchy) *The* height *of the structure is the longest path from an atomic level time unit up to the top-most time unit.*

For example, the Gregorian calendar structure, as represented in Figure 5.1, has an height equal to 5.

Examples in this thesis are mainly based on the Gregorian calendar. As an example for a variant calendar we use a university calendar structure, particularly, as organized at Simon Fraser University (SFU). Figure 5.1 represents the Gregorian calendar structure. Figure 5.2 represents the SFU calendar structure. Edges are labeled with the two decomposition characteristics, constancy and alignment. C stands for constant, *nc* stands for non-constant and similarly, *a* and *na* stand for the alignment possibilities; not all edges are drawn for clarity purposes. The complete description of these graphs in terms of constancy and alignment of decomposition appear in Figure 5.8 and Figure 5.9 with the adjacency matrix of both graphs. It can be noticed that *month* and *period* in the SFU calendar structure do not relate with decomposition. *Period* stands for class period, exam period and break period, lasting respectively 13 weeks, 2 weeks and 2 or 3 weeks. That is to say, *period* is a non-equal durations time unit, as is *month*. They are not comparable exactly because their respective durations are not comparable. The duration of *month*, $(28, 31)$, is not comparable to the duration of *period*, $(2*7, 13*7) = (14, 91)$, an expected result which confirms the soundness of the whole apparatus defined. (Some instances of *month* are larger than some instances of *period* and viceversa.)

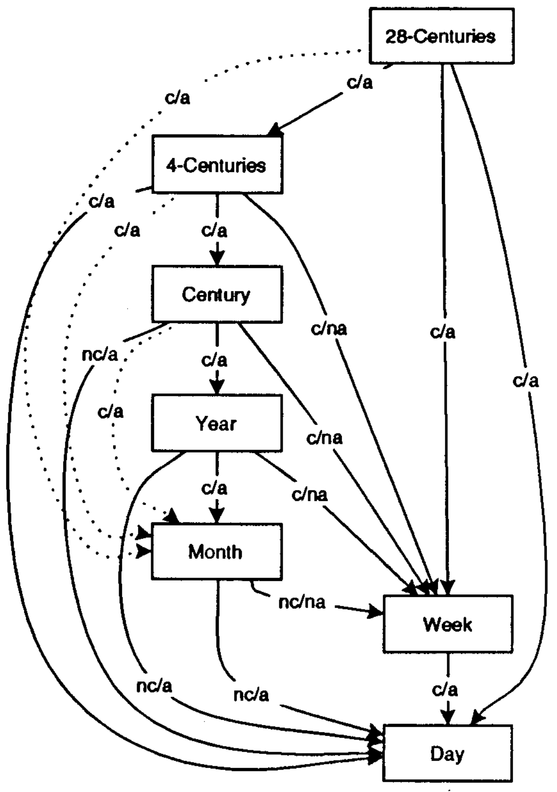
Time Unit	(general) Duration[1]	Main reference time unit
4-centuries	(146097,146097)	zero-point
century	(36524,36525)	zero-point
year	(365,366)	zero-point
month	(28,31)	year
week	(7,7)	year
day	(1,1)	whole-week

Table 5.1: Attributes of time units in the Gregorian calendar

To completely define these two calendars the attributes of the time units and of the decomposition relations have to be specified. These have appeared in examples in Chapter 3 and are summarized in Tables 5.1, 5.2 and 5.3.

[1]In this case, durations are measured in *days*, the atomic time unit in the structure

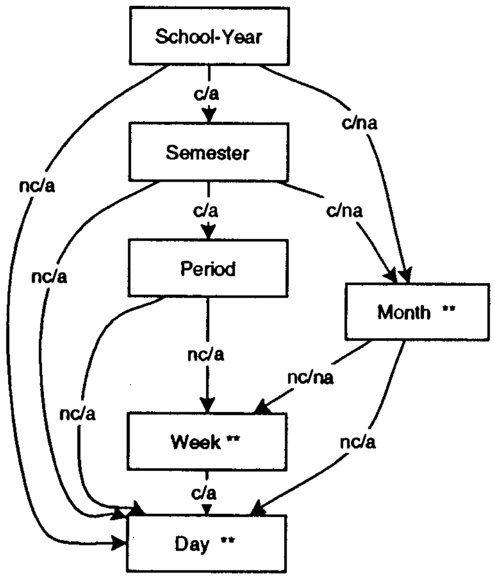
[2]These time units are related to time units in the Gregorian calendar



References:

c/a	constant, aligned
nc/a	non-constant, aligned
c/na	constant, non-aligned
nc/na	non-constant, non-aligned

Figure 5.1: Gregorian calendar structure



References:

c/a	constant, aligned
nc/a	non-constant, aligned
c/na	constant, non-aligned
nc/na	non-constant, non-aligned
X **	time unit from Gregorian calendar

Figure 5.2: Simon Fraser University calendar structure

Time Unit[2]	(general) Duration	Main reference time unit
school-year	(52*7,53*7)	zero-point
semester	(17*7,19*7)	school-year
period	(2*7,13*7)	semester

Table 5.2: Attributes of special time units in the Simon Fraser university calendar

The graphs of two other calendar structures, the World calendar and the Hebrew calendar appear in figures. The *World calendar* is a proposal that came as a result of inter-governmental initiatives in cooperation with some non-governmental parties, such as the International Astronomical Union, and religious authorities, from the 1920's till 1940's. This reform has been proposed for example at the United Nations (1947), and treated at Vatican Councils (in 1963). However this reform has not finally been approved. (see [Col71, Rus82]). This calendar attempts to address two main "problems" of the Gregorian calendar; inequalities of the divisions of the year (months having different number of days), and the changes of the day of the week (thus if the first day of one year is a Monday, the first day of the next year is a Tuesday or Wednesday). In terms of the terminology introduced in this thesis, this reform attempts to deal with the non-constancy and non-alignment of the decomposition between time units. We present this calendar here as a note about related material to calendars. A key point in studying this calendar however has been to have a more complete set of study cases. The apparatus developed in this thesis takes into consideration the structure of the World calendar. Briefly, the astronomical year is divided exactly into four three-months periods, of ninety-one days each, with more (fixed) internal structure. This makes a total of 364 days. The odd remaining day is out of the scheme of days and months. It is a spare day, a holiday called "world day". Leap years have an extra world day. January 1st is always a Sunday, Christmas Day comes on a Monday. More details can be found in the referenced literature. In our formalism we would treat the special world day as a unit of another type into which the astronomical year decomposes. However, this adds no further complication to the formal apparatus, since it is effectively an elaboration of a decomposition into non-equal duration time units.

Composed TU	Component TU	Constancy	Repetition factor	Alignment	Begin	Instance values
year	*month*	cons	(12,12)	alig	1	Jan, Feb,...
year	*week*	cons	(53,53)	non-alig	1	1..53
year	*day*	non-cons	(365,366)	alig	1	1..366
month	*week*	non-cons	(4,6)	non-alig	1	1..6
month	*day*	non-cons	(28,31)	alig	1	1..31
week	*day*	cons	(7,7)	alig	1	1..7
school-year	*semester*	cons	(3,3)	alig	1	Fall,...
school-year	*month*	cons	(12,12)	non-alig	September	Jan, Feb,...
school-year	*week*	non-cons	(52,53)	alig	1	1..53
school-year	*day*	non-cons	(365,366)	alig	Labor Day	1..366
semester	*month*	cons	(4,4)	non-alig	1	1..4
semester	*period*	cons	(3,3)	alig	1	classes, ...
semester	*week*	non-cons	(17,19)	alig	1	1..19
period	*week*	non-cons	(2,13)	alig	1	1..13

Table 5.3: Attributes of decomposition between time units

The *Hebrew calendar* is an example of a lunar-based calendar, where other decomposition variations appear, all of them contemplated by our formalism. This calendar has 12 months of 29 and 30 days alternating, having a total of 354 days per year. The calendar is corrected adding an extra month to seven years of the nineteen-years "Metonic cycle". These years are the so called "embolismic" years, the 3rd, 6th, 8th, 11th, 14th, 17th and 19th of the cycle. The zero point of this calendar is the "Creation of the World", established as October 3761 B.C. So for example 1969 of the Gregorian calendar was 5730 in the Hebrew calendar. Another example not represented in this chapter, but also contemplated by the developed formalism (and of historical curious interest), is that of the *French Revolution calendar.* This calendar was introduced in 1793 by the French revolutionaries and abolished in 1806. It is based on a more "rational", decimal system. They proposed a year composed of 12 months of 30 days, composed in turn by three weeks of 10 days each. Five or six revolution days end the year. Hours, minutes and seconds are also based on the decimal system.

5.2 Operations on decomposition relations

Three operations will be considered between decomposition relations; *composition, inverse* and *intersection.* The composition operation between relations shouldn't be confused with the name of the relation, which happens to be "decomposition". Composition will be referred also as *product* of decomposition relations and intersection as *sum*, following the terminology used in the definition of relational algebras (for example [VK86]).

The operation that we specially want to study is composition. With composition, transitivity properties about constancy and alignment are studied, and these in turn are the basis for the analysis of structural properties in the time unit hierarchy.

The other two operations are presented here for completeness purposes. It should be noticed that these three operations (composition, intersection and inverse) are here defined among decomposition between time unit *classes.* This is to be contrasted to the relational algebras defined in the literature, and constraint propagation algorithms, which deal with relations between time intervals in the time line, i.e., at

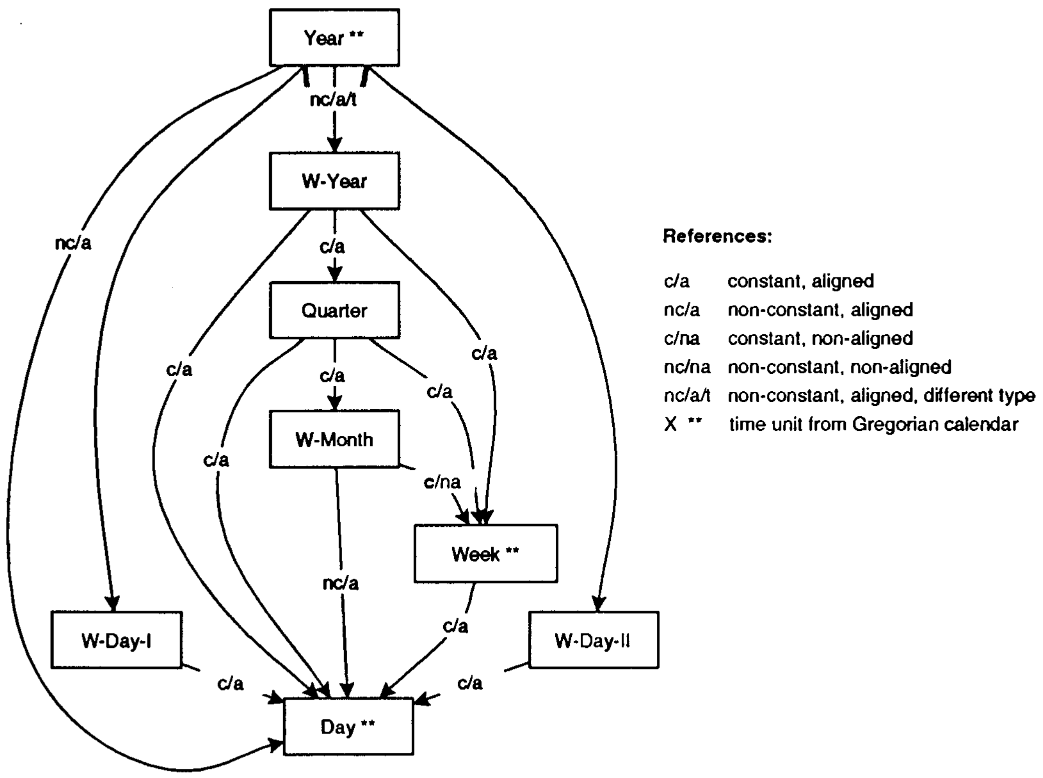

Year **
nc/a/t
W-Year
c/a
Quarter
c/a
W-Month
c/na
Week **
nc/a
W-Day-I
W-Day-II
Day **
References:
c/a constant, aligned
nc/a non-constant, aligned
c/na constant, non-aligned
nc/na non-constant, non-aligned
nc/a/t non-constant, aligned, different type
X ** time unit from Gregorian calendar

Figure 5.3: World calendar structure

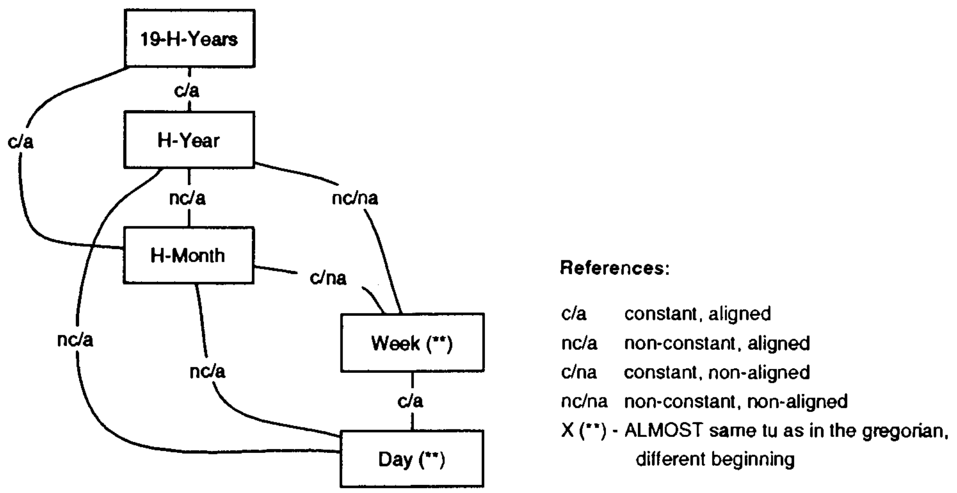

19-H-Years
c/a
H-Year
nc/a
nc/na
H-Month
c/na
Week (**)
Day (**)
References:
c/a constant, aligned
nc/a non-constant, aligned
c/na constant, non-aligned
nc/na non-constant, non-aligned
X (**) - ALMOST same tu as in the gregorian, different beginning

Figure 5.4: Hebrew calendar structure

the instance level (for example [All83, MSK93, GS92, LM94].)

5.2.1 Composition or product

The composition or product of two decomposition relations is defined when the component time unit of one of the relations is the same time unit as the composed time unit of the other relation. For example *year* decomposes into *month*, *month* decomposes into *day*. These two decomposition relations can be composed (or multiplied) to obtain the relation of *year* decomposing into *day*.

Decomposition characteristics like constancy, alignment and repetition factor can be analyzed when multiplying these relations.

Definition 32 (Composition or product of decomposition) *Let A, B, C be time units $\in TUS_{CAL}$, such that $A \trianglerighteq_1 B$ and $B \trianglerighteq_2 C$. The composition or product of the two decomposition relations $\trianglerighteq_1$ and $\trianglerighteq_2$ is a third decomposition relation $\trianglerighteq_3$ such that $A \trianglerighteq_3 C$. This will be written as $\trianglerighteq_3 = \trianglerighteq_1 \otimes \trianglerighteq_2$.*

Constancy and alignment

A systematic study of each possible product of decomposition relations with respect to constancy and alignment is schematically presented in Figure 5.5 and Figure 5.6. Examples are from the Gregorian calendar unless otherwise specified. Combinations that are not possible are crossed out. The results are summarized in Table 5.4 and Table 5.5.

Some situations are determined. The product of a *constant* decomposition relation with another *constant* decomposition relation is a third *constant* relation. For example, (*week* $\trianglerighteq_1$ day) and (*day* $\trianglerighteq_2$ hour), with both decompositions being constant *Cons(week,day)* and *Cons(day,hour)*. Multiplying these two relations produces $\trianglerighteq_3$ such that (*week* $\trianglerighteq_3$ hour) and *Cons(week,hour)*.

Some other situations are undetermined; for example, the product of a constant decomposition with a non-constant decomposition relation can be either constant or non-constant. (For example *4-centuries, year,day* and *year,month,day* respectively.)

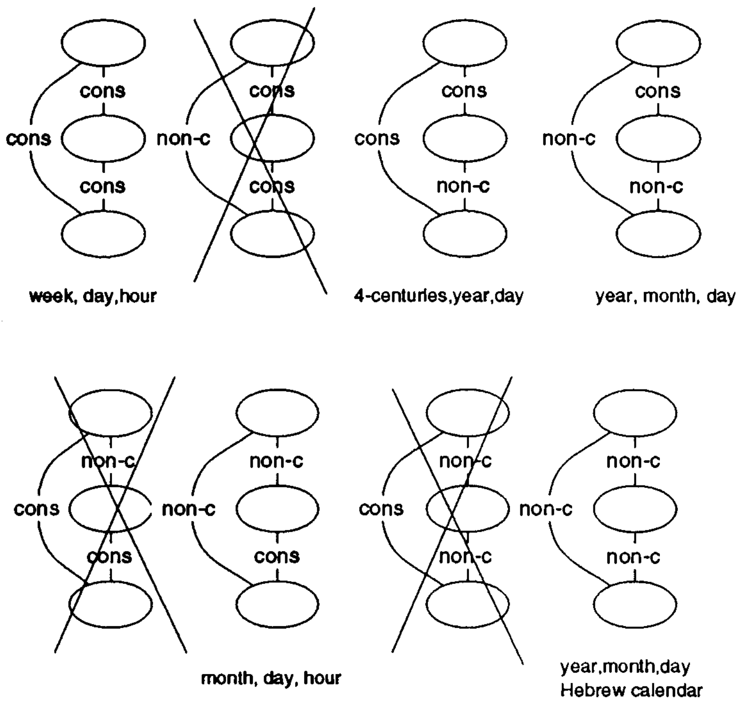


Figure 5.5: Transitivity of Constancy

However, this non determinism is not random. As it will be analyzed in detail, (in section 5.3) the resulting relation follows a precise pattern in the structure; *periodic relations* and *cycles* become apparent as a direct consequence of these characteristics of the transitivity.

2 1 ⊗	*cons*	*non-cons*
cons	cons	{cons, non-cons}
non-cons	non-cons	non-cons

Table 5.4: Transitivity of constancy of decomposition relations

The results above shown analyze constancy and alignment separately. There are $2^3 = 8$ possible combinations in each separate case. A complete analysis combining constancy and alignment involves the study of $4^3 = 64$ scenarios. Several of those cases are a direct consequence of the behavior of each aspect separately. Some cases however present situations that show the inherent interconnection between these two aspects.

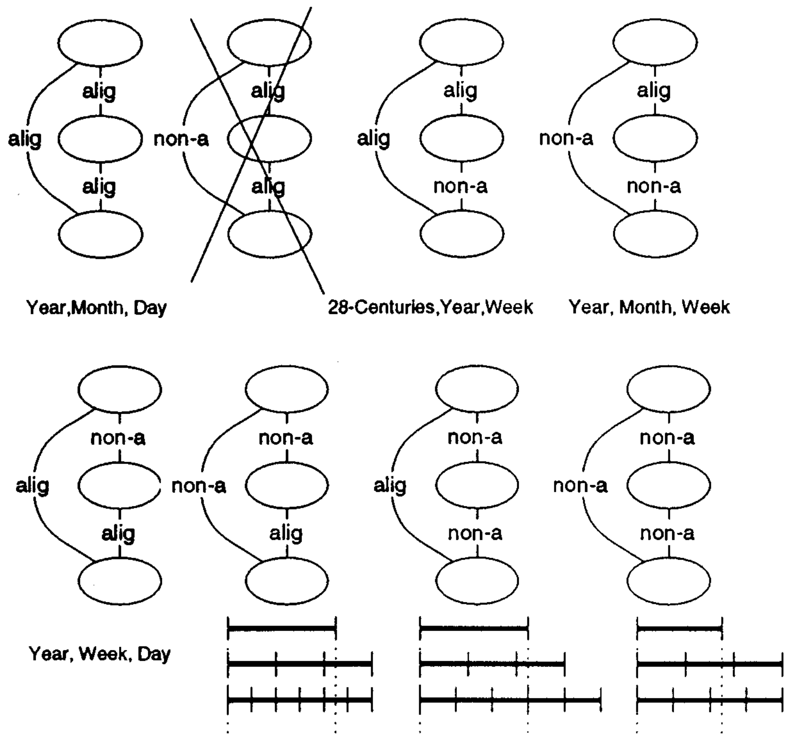


Figure 5.6: Transitivity of Alignment

2 1 ⊗	*alig*	*non-alig*
alig	alig	{alig, non-alig}
non-alig	{alig, non-alig}	{alig, non-alig}

Table 5.5: Transitivity of alignment of decomposition relations

We present one pattern of such an interconnection between alignment and constancy. These examples suggest that if the product of two decomposition relations is non-exact (either non-constant or non-aligned) and one of the relations is exact (either constant or aligned), then the other decomposition relation must be non-exact. Figure 5.7 graphically shows this situation with associated examples. A complete study of all the possible scenarios would prove this pattern and others relating transitivity with constancy and alignment. This is left for future investigation.[3]

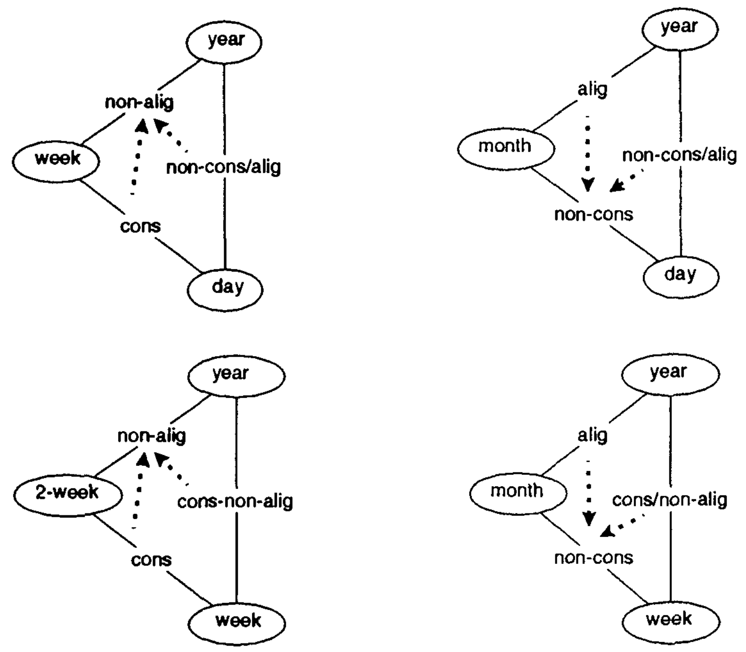


Figure 5.7: Interconnection between constancy and alignment

Repetition factor

The repetition factor of the product of two decomposition relations can be obtained by multiplying the respective repetition factors. Sometimes only a bound of the repetition factor can be deduced. Repetition factors have been defined as min-max pairs, and we define multiplication as multiplying respectively the minimums and the maximums.

[3]I would like to thank Andrew Fall for his enthusiasm in discussing these and different interesting patterns.

If the relations that are composed (or multiplied) are *constant and aligned*, the repetition factor obtained as the multiplication of the two pairs is exactly the repetition factor of the product relation. For example, *century* decomposes into *year* in a constant and aligned way, with a repetition factor of $(100, 100)$. *Year* decomposes into *month* also in a constant and aligned way, with a repetition factor of $(12, 12)$. *Century-month* is the resulting composition (or product) of these decomposition relations. It has a repetition factor of $(12 * 100, 12 * 100)$.

However, when some of the relations composed (or multiplied) are non-constant or non-aligned, the multiplication of the repetition factors does not produce exactly the repetition factor of the product relation, but a looser bound. For example, *year-month* has a repetition factor of $(12, 12)$, is constant and aligned; *month-day* has a repetition factor of $(28, 31)$, is non-constant and aligned. Multiplying the repetition factors, produces the pair $(12 * 28, 12 * 31) = (336, 372)$ which is in fact a looser bound of the *year-day* repetition factor: $(365, 366)$. Multiplication of repetition factors can be used when creating or modifying calendar structures, see Section 5.5.

5.2.2 Intersection

The *intersection* or *sum* of two decomposition relations is defined if both decomposition relations have the same composed and component time unit.

Definition 33 (Intersection or sum of decomposition) *Let A, B be time units $\in$ TUS_{CAL}, such that $A \trianglerighteq_1 B$ and $A \trianglerighteq_2 B$. The intersection or sum of the two decomposition relations $\trianglerighteq_1$ and $\trianglerighteq_2$ is a third decomposition relation $\trianglerighteq_3$ such that $A \trianglerighteq_3 B$, and the attributes of $\trianglerighteq_3$ are common to the other two. This will be written as $\trianglerighteq_3 = \trianglerighteq_1 \oplus \trianglerighteq_2$.*

Intersection in this context is a limit case operation. Time units are in fact *classes* of intervals, and likewise, decomposition relations are classes of decompositions. Therefore two time units only have *one* possible decomposition relation between them. For example *year* always decomposes in a constant and non-aligned way into *day*, with a repetition factor of $(365, 366)$. Therefore, there is only one single decomposition relation between two classes, and thus intersection is not of so much interest from a computational point of view. Contrast this to intersection of relations between intervals at the instance level, an issue dealt extensively in the literature to deal with scheduling of activities occurring during specific time intervals. Time unit instances can relate in several possible ways. For example, *specific weeks* can occur before, after, during, or mostly any of the basic interval relations with respect to a *specific year.* That is to say, there are several possible consistent scenarios of relations between specific time unit instances. At this point in our discussion, we are not dealing with relations among specific time intervals, but rather with relations about the time units classes, which constitute the basis of a formal system of measures.

Intersection of two decomposition relations between time unit classes is still an interesting operation if relations are not yet defined completely, i.e., it is not yet determined whether a decomposition relation is aligned or non-aligned. For example, this could be the case while the time unit hierarchy is created, or if a new time unit is added to the structure.

The characteristics of the sum of two decomposition relations results from intersecting their characteristics in the usual set-theoretic way. For example, if $\trianglerighteq_1$ is $\{$*constant, non-constant*$\}$ and $\trianglerighteq_2$ is *constant* then $\trianglerighteq_3 = \trianglerighteq_1 \oplus \trianglerighteq_2$ is *constant.*

5.2.3 Inverse

The *inverse* of a decomposes relation is the *composes* relation. (Not to be confused with composition or product!). The characteristics of decomposition and its inverse are the same, constancy, alignment, etc. It is only a matter of how things are expressed, from the composed time unit point of view or the component time unit point of view.

5.3 Structural properties in a time unit hierarchy

This section develops a series of definitions intercalated with examples and properties about the calendar structures related to propagation of constancy and alignment. Cycles, higher and lower bounds are defined. A parallelism is presented with divisibility concepts. At the end of this section all the properties are summarized.

5.3.1 Cycles in the time unit hierarchy

In a time unit hierarchy we find *cycles.* These are related to the transitivity of constancy and alignment of the decomposition relations in the hierarchy as they are composed (or multiplied).

Up cycles

A constancy up cycle appears when a time unit decomposes in a constant way into some "extreme" time unit even though there may be "intermediate" time units in the hierarchy that decompose in a non-constant way into that extreme one.

For example, sequences of four centuries decompose into $(365 * 400 + 4 * 25 - 3)$ *days*. This formula comes from considering all years as non-leap years (i.e. with 365 days) and then adding the additional leap days. So 400 years contain 400 "years of 365 days", plus 4 times the leap days in a century. (4 times comes from 4 centuries). Since there is a leap year every 4 years, each century has (in general terms) $100/4 = 25$ leap years. However, years that are multiple of 100 are not leap years, except those that are multiple of 400. (Year 2000 is a leap year, year 1900 is not). That is why, every 400 years there are **3** less leap days than according to the general rule of "every four years", and that is reflected by the "-3" in the formula. For example, any sequence of 100 year could have different number of days, depending whether the year which is multiple of 100 is also multiple of 400 or not. It is only every 400 years that there is a fixed number of days. This explains why *4-Centuries* decompose into a constant number of days, but not *100-years*, nor *year*, etc. *4-Century* sequences and *day* are the extremes of the cycle. We can also say that *4-Centuries* closes the cycle.

Intermediate time units in this cycle are *century*, *year* and *month* which decompose in a non-constant way into *day*.

Definition 34 (Constancy up-cycle) *There is a* constancy up-cycle *in the time unit hierarchy involving time units A, C and B_i, with $0 \leq i \leq n \in N$, when A decomposes in a constant way into C, and considering the time units B_i and A, A is the least time unit that so decomposes. That is, if there exists any time unit B_i lying in the hierarchy between A and C, such a time unit decomposes in a non-constant way into C. A is referred to as a* Higher bound *of C (in a constancy sense). The cycle is formed by the extreme time units (A and C) and the intermediate time units (B_i).*

In the previous example,*4-centuries* stands for A, *day* stands for C, and there are 3 intermediate B_i's: *century*, *year* and *month.* Figure 5.8 shows a graphical representation of this cycle.

Another example showing a limit case is *week* decomposing into 7 *days. Week* and *day* are the extremes of the cycle, there are no "intermediate" time units.

These two previous examples also illustrate that the same time unit (in this case *day*) can participate in more than one cycle.

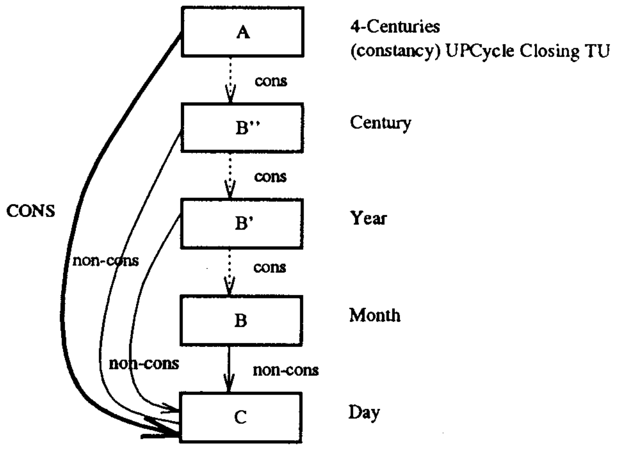


Figure 5.8: Constancy up-cycle: *4-Centuries* and *day* are the extremes of the cycle

Analogously, the concept of an *alignment* up-cycle is defined.

Definition 35 (Alignment up-cycle) *There is an* alignment up-cycle *in the time unit hierarchy involving time units A, C and B_i, with $0 \leq i \leq n \in N$ when A decomposes in an aligned way into C, and considering the time units B_i and A, A is the least time unit that so decomposes. That is, if there exists any time unit B_i lying between the two extremes of the cycle (A and C), they decompose in a non-aligned way into C. A is referred to as a* Higher bound *of C (in an alignment sense). The cycle is formed by the extreme time units (A and C) and the intermediate time units (B_i).*

For example, sequences of 28 centuries decompose in an aligned way into *week* (*28-Centuries* is the least time unit in the hierarchy that so decomposes.) *28-Centuries* and *week* are the extremes of the cycle; *4-Centuries*, *century*, *year* and *month* are intermediate time units in the cycle, they decompose in a non-aligned way into *week.* Figure 5.1 of the Gregorian calendar includes this cycle.

Property 7 (Existence of a higher bound) *Any time unit hierarchy can be extended so that any given time unit has a higher bound, both in the constancy and alignment sense.*

Proof: This stems from the fact that it is always possible to add to a time unit hierarchy a time unit which is equal to a contiguous sequence of the given time unit. This added time unit decomposes in an aligned and constant way into the given one. ■

For example *centuries* has as a higher bound *2-centuries*, both in the constancy and alignment sense.

Down cycles

We also define *down-cycles.* Alignment and constancy differ in their down-cycles. In case of alignment, there exists a *lower bound* into which some time unit decomposes in an aligned way, even though this time unit decomposes in a non-aligned way into intermediate time units. In case of constancy, if there are intermediate time units decomposing in a non-constant way, the cycle is not closed. More precise definitions and examples follow.

Definition 36 (Alignment down-cycle) *There is an* alignment down-cycle *in the time unit hierarchy involving time units A, C and B_i, with $0 \leq i \leq n \in N$ when A decomposes in an aligned way into C, and considering the time units B_i and C, C is the highest time unit into which A so decomposes. That is, if there exists any time unit B_i lying between the two extremes of the cycle (A and C), A decomposes into them in a non-aligned way. C is referred to as a* lower bound *of A (in an alignment sense). The cycle is formed by the extreme time units (A and C) and the intermediate time units (B_i).*

For example, *year, week, day* is a cycle. *Year* decomposes into *week* in a non-aligned way, whereas *year* decomposes into *day* in an aligned way. Hence, *day* closes the non-aligned down-cycle.

Non-constancy behaves differently than alignment with respect to down cycles.

Property 8 (Propagation of non-constancy down) *Let A be a time unit that decomposes in an non-constant way into another time unit B. A will also decompose in a non-constant way into any time unit into which B decomposes.*

Proof: This property can be seen as a result of the transitivity properties seen in the composition (or product) of decomposition relations. A non-constant decomposition composed with either a constant or a non-constant one results in a non-constant decomposition relation. (See Table 5.4) ■

For example, *month* decomposes into day in a non-constant way. *Month* also decomposes into *hour* in a non-constant way. Still going further down in the hierarchy, *month* also decomposes into *second* in a non-constant way. And this could continue further down. Intuitively, the idea is that since different *month*-instances can decompose into a different number of days, different *month*-instances will decompose into a different number of hours and different number of whatever time unit composes a day, hence there is no lower time unit closing such a non-constant cycle.

From this property, it follows that there exist constancy down-cycles, but only in a limit case:

Property 9 (Constancy down-cycle: no intermediate time units) *There are constancy down-cycles only with* no *intermediate time units. I.e. there exist only limit constancy down-cycles, a cycle involving only two time units, the extremes of the cycle, so that they decompose in a constant way. In this case the lower extreme is the* lower bound *of the higher extreme (in the constancy sense).*

Proof: The result is a direct consequence of the previous property, of propagation of non-constancy down. ■

The same example of *month* decomposing into *day* and *hour* applies as in the previous property.

Property 10 (Existence of lower bound) *Any time unit hierarchy can be extended so that any given time unit has a lower bound, in both, the alignment and constancy sense.*

Proof: This stems from the fact that it is always possible to add to a time unit hierarchy a time unit so that the given time unit is equal to a contiguous sequence of the added one. That is, the given time unit decomposes in an aligned way into new added time unit. The fact that there are no constancy down-cycles with intermediate time units does not alter this result. This result simply comes from adding a (possibly artificial) new time unit to the hierarchy. (And thus creating a cycle with no intermediate time units between the given time unit and the created lower bound.) ■

For example, if the given time unit is *second*, a lower bound (in an alignment and constancy sense) of it could be *mili-second* . In passing, it should be noticed that adding a new time unit in a lower level in the hierarchy may change the atomicity level, and therefore the granularity considered.

Cycles viewed as propagation

To further clarify intuitively the defined cycles, we illustrate them as a "propagation movement" of a "non-aspect" (non-constancy or non-alignment) in a hierarchy structure. In an *up* cycle, the non-aspect propagates upwards, until a time unit "stops" the propagation, or "closes the cycle". The closing unit is a higher bound of the time unit from where the propagation movement started. Figure 5.9.a abstractly shows propagation going upwards, where non_i denotes a decomposition relation that is non-constant going upwards in an ith "iteration", up to a time unit closing the cycle. The example in Figure 5.8 can be paraphrased following this intuition of propagation movement. *Month* decomposes into *day* in a non-constant way. Going up in the hierarchy, *year* also decomposes into *day* in a non-constant way. Still going further up in the hierarchy, *century* also decomposes into *day* in a non-constant way. So far, non-constancy propagated upward in the hierarchy. *4-Centuries* decomposes in a *constant* way into *day*, thus closing the cycle.

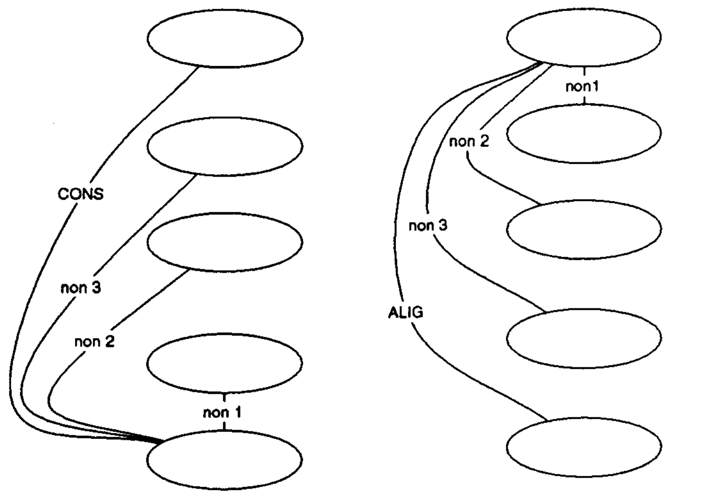

CONS
non 3
non 2
non 1
(a) Propagation of non-constancy "up"
non1
non 2
non 3
ALIG
(b) Propagation of non-alignment "down"

Figure 5.9: Propagation

Analogously, Figure 5.9.b abstractly shows propagation going downwards, where non_i denotes a decomposition relation that is non-aligned going downwards in an ith "iteration" down to a time unit closing the cycle, (i.e. a lower bound.) The alignment down-cycle *year, week, day* can be paraphrased with this view point. *Year* decomposes into *week* in a non-aligned way. Going down in the hierarchy, *day* stops the propagation of non-alignment, since *year* decomposes in an aligned way into *day*.

The non-existence of constancy down-cycles with intermediate time units can be paraphrased as follows: non-constancy propagates downwards and there is no time unit stopping that propagation.

5.3.2 Relation with divisibility concepts

Higher bounds - Multiples

The results obtained reflect the fact that given *two* time units, there is a "least *common* higher bound" that decomposes constantly (aligned) into both. The two given time units may decompose in a constant or non-constant (aligned or non-aligned) way between them.

Two series of definitions are possible, either in the constancy or alignment sense.

Definition 37 (Least common higher bound - constancy) *Given two time units B and C so that B decomposes into C either in a constant or non-constant way, A is the least common higher bound (in the constancy sense) if it simultaneously is a higher bound (in the constancy sense) of both B and C, and it is the least time unit with that characteristic. That is to say, A decomposes into both B and C in a constant way, and it is the least time unit in the hierarchy that so decomposes.*

From the previous example, *4-Centuries* is the least common higher bound that decomposes constantly into the two time units *year* and *day* (with *year* decomposing in a non-constant way into *day*).

The limit case occurs when the two given time units already decompose in a constant way. In this case, the least common higher time unit that decomposes in a constant way into both is the given composed time unit. For example, *year* decomposes into *month* in a constant way. The common time unit that decomposes in a constant way into both, (*year* and *month*) is *year* itself.

Analogously, a least common higher time unit to two time units also exists related to alignment.

Definition 38 (Least common higher bound - alignment) *Given two time units B and C so that B decomposes into C either in an aligned or non-aligned way, the time unit A is the least common higher bound (in the alignment sense) if it simultaneously is a higher bound (in the alignment sense) of both B and C, and it is the least time unit with that characteristic. That is to say, A decomposes into both B and C in an aligned way, and it is the least time unit in the hierarchy that so decomposes.*

The characteristics of higher bounds and least common higher bounds parallel divisibility among integers. Higher bounds parallel to "multiples" and least common higher bounds with least common multiples. Again, two definitions apply, in the constancy and alignment sense.

Definition 39 (Multiple of a time unit) *A time unit A is a* multiple *of B (or B* divides *A) if A decomposes in an constant (aligned) way into B.*

For example, *year* is a multiple (in both constancy and alignment sense) of *month. Year* is a multiple of *day* in the alignment sense, but not in the constancy sense.

Definition 40 (Relative prime time units) Relatively prime time units *are those time units decomposing in a non-constant (non-aligned) way.*

E.g. *Year* is relatively prime to *day* in the constancy sense.

Definition 41 (Least common multiple) *The least time unit in the hierarchy decomposing in a constant (aligned) way into two time units corresponds to a* least common multiple *of those time units.*

E.g. *4-centuries* is the least common multiple of *year* and *day* in the constancy sense.

Property 11 (Existence of a least common multiple) *Any time unit hierarchy can be extended so that any two given time units have a a least common higher bound, and therefore, a least common multiple, both in the constancy and alignment sense.*

Proof: This follows from the existence of higher bounds of any time unit. ■

Lower bounds - Divisors

In analyzing down-cycles, we define the concept of a greatest common lower bound of two given time units.

Definition 42 (Greatest common lower bound - alignment) *Given two time units A and B so that A decomposes into B either in an aligned or non-aligned way, the time unit C is the greatest common lower bound if it simultaneously is a lower bound of both A and B, and it is the greatest time unit with that characteristic. That is to say, both A and B decompose into C in an aligned way, and C is the greatest time unit in the hierarchy into which they so decompose.*

The following example illustrates the situation: *year* decomposes into *week* in a non-aligned way. There exists a common lower time unit into which both decompose in an aligned way, *day*. (And it is the greatest time unit into which they so decompose).

As expected, the existence of a greatest common lower bound in the constancy sense is a limit case. Only time units that relate with a constant decomposition have a greatest common lower bound, and it is the component time unit itself, as the following definition expresses:

Definition 43 (Restricted greatest common lower bound - constancy) *Given two time units A and B so that A decomposes into B (only* in a constant way*), the time unit C is the greatest common lower bound if it simultaneously is a lower bound of both A and B, and it is the greatest time unit with that characteristic. That is to say, both A and B decompose into C in a constant way, and C is the greatest time unit in the hierarchy into which they so decompose. In fact $B \equiv C$, given that A decomposes in a constant way into B, thus A, B constitutes a limit case cycle, with no intermediate time units.*

E.g. the greatest common lower bound in the constancy sense of *year* and *month* is *month.*

Analogously to the up-cycles, a parallel can be established with divisibility concepts for down-cycles.

Definition 44 (Greatest common divisor) *The greatest time unit in the hierarchy into which two time units decompose in an aligned (constant) way corresponds to a* greatest common divisor *of those time units in the alignment (constancy) sense.*

For example, *day* acts as a *greatest common divisor* of *year* and *week*. In case of constancy, only limit cases apply, for example *month* is the greatest common divisor of *year* and *month* itself.

Property 12 (Existence of a greatest common divisor - alignment) *Any time unit hierarchy can be extended so that any two given time units have a a greatest common lower bound, and therefore, a greatest common divisor, in the alignment sense.*

Proof: This follows from the existence of alignment lower bounds of any time unit. ■

Property 13 (Restricted existence of greatest common divisor - constancy) *Any time unit hierarchy can be extended so that two given time units* that decompose in a constant way *have a a greatest common lower bound, and therefore, a greatest common divisor, in the constancy sense.*

Proof: This follows from the restricted existence of constancy lower bounds of any time unit. ■

5.3.3 Summary of results cycles-divisibility

Table 5.6 and Table 5.7 summarize the results exposed in the previous sections.

Alignment sense
• Up-cycle with intermediate time units
• $\exists$ higher bound for any time unit
• $\exists$ least common higher bound for any pair of time units $\equiv$
$\equiv$ $\exists$ least common multiple for any pair of time units
• Down-cycle with intermediate time units
• $\exists$ lower bound for any time unit
• $\exists$ greatest common lower bound for any pair of time units $\equiv$
$\equiv$ $\exists$ greatest common divisor for any pair of time units

Table 5.6: Summary of structural/mathematical results in the alignment sense

Constancy sense
• Up-cycle with intermediate time units
• $\exists$ higher bound for any time unit
• $\exists$ least common higher bound for any pair of time units $\equiv$
$\equiv$ $\exists$ least common multiple for any pair of time units
• Down-cycle with *no* intermediate time units
• $\exists$ lower bound for any time unit
• $\exists$ *restricted* greatest common lower bound for any pair of time units $\equiv$
$\equiv$ $\exists$ *restricted* greatest common divisor for any pair of time units

Table 5.7: Summary of structural/mathematical results in the constancy sense

Intuitions about the parallel decomposition-divisibility

If a time unit decomposes into another in an aligned way, intuitively it can be seen that the composed time unit can be divided into an integer number of component time units. If the decomposition is non-aligned the composed time unit is divided in a certain integer number of component time units "plus" a certain "fraction" of component unit. This intuition strongly suggests that there is a parallel between aligned decomposition and divisibility in the positive integer numbers. Hence a composed aligned time unit can be considered a *multiple* of the component, and so on. For example, one (expected) property that holds in a time unit hierarchy is that there always exists a common multiple time unit of any pair of given time units.

The constancy aspect does not have such a straightforward parallel with divisibility, although a parallel can be established. Time units decomposing in a constant way are multiples, those decomposing in a non-constant way are relatively prime, etc. Thus the constancy-divisibility parallel results in a natural counterpart of the aligned-divisibility and analogous results to alignment hold as well. For example, it holds that for any pair of time units there exists a (constancy) common multiple, i.e. a time unit that decomposes in a constant way into any pair of time units. Intuitively, this result is guaranteed by the very nature of the conventional cycles that have been imposed in calendars to adapt to astronomical facts. (see for example [Moe82, Ped82]). "Period relations" are an intrinsic part of calendars, they state a relation between two time periods, such that an integer number of the first period equals another integer number of the second period. For example, the "Metonic cycle" assumes that 19 solar years equal 235 lunar months. The example used in the previous section is that of 4 centuries having an integer (constant) number of days.

Hence, we parallel a time unit that decomposes in a constant way as a "multiple" of the component unit. We thus obtain the expected result that any pair of time units in the hierarchy has a common multiple in the constancy sense. We can restate this in terms of the introduced terminology; propagation of non-constancy upwards from any pair of given time units is always stopped by some (closing cycle) time unit.

Basic measure time unit in a hierarchy

A unit that divides all units in an *alignment sense* in the hierarchy is in fact the one preferred to be used as the basic unit to measure general durations. Defining a basic unit as the greatest common divisor in the alignment sense of all the time units produces that general durations be defined with integers and with no need of special numbering conventions. That would not be the case if the unit chosen as basic is not a common divisor. For example, a *year* contains 51 complete weeks (from *Sunday* to *Saturday*), and two incomplete pieces of week. (Or in some particular years 52 complete weeks and an incomplete one). Should general durations be expressed with integers, then the incomplete weeks have to be conventionally included or not as part of the year.

A basic unit should also be atomic for the intended application, as explained in Chapter 3.

Property 14 (Characterization of a basic measure time unit) *A basic time unit in a time unit hierarchy used to measure general durations of the units in the hierarchy should divide all units, i.e. all time units should decompose into it in an aligned way. That is, a basic time unit in a hierarchy is the greatest common divisor of all time units in the hierarchy.*

For example, in the Gregorian calendar structure of Figure 5.1, day is the basic time unit. If the hierarchy included *hour* as a time unit, *hour* would be the basic measure time unit. Notice that in defining the measure time unit as atomic all general durations are expressible with integer numbers.

5.4 Chains

We have built a formal apparatus of a hierarchical decomposition of time units. This gave birth to the precise definition of a calendar structure or time unit hierarchy. We have studied structural and mathematical properties that this hierarchy holds, and paralleled concepts to those of divisibility.

In this section we propose an alternative categorization based on the subrelations of decomposition, i.e., considering only constant/aligned decompositions, or aligned only, or constant only. *Calendar substructures* result from these subrelations.

New entities become apparent in these substructures, *chains* and *chain extreme time units*. Chains and chain extreme time units are envisioned as possible building blocks to characterize a language of time units in a constructive and recursive way.

The term "chains" has been inspired in work by [MS90], however in their case chains are defined on time intervals on the time line. In our case we are referring to chains of time unit classes. Nonetheless, chains of time unit classes are directly related to expressions that represent time unit instances, and influence greatly in efficiency matters. To give an immediate example, the operation of converting time unit instances from one time unit to another will be more efficient when the time units decompose in a constant/aligned way; divisions and multiplications can be done, whereas it is necessary to have some iterative process of additions or subtractions when the decomposition is not exact. We will introduce the concept of "calendar substructures" in the following sections as a basis to define chains, as well as to provide an alternative structural categorization of time unit hierarchies.

5.4.1 Aspects of decomposition as different relations

The original structure defined as "calendar structure" is the one that relates time units by the decomposition relation including any variation of constancy and alignment. Other alternative structures rise from considering subsets of the decomposition relation. There can be several possible subsets of the decomposition relation. The following are all the possibilities: constant/aligned, non-constant/aligned, constant/non-aligned, non-constant/non-aligned, constant and aligned. The last two relations are defined as unions of the previous; constant is the union of constant/aligned and constant/non-aligned, and analogously, aligned is the union of constant/aligned and non-constant/aligned.

Adjacency matrix of the calendar structure

Table 5.8 shows the adjacency matrix of the Gregorian calendar structure as one way of visualizing all the elements in the structure (time units) and how they relate. Table 5.9 shows the analogous matrix of the Simon Fraser University calendar structure. The matrix representation provides for another display to represent a calendar structure. *Ca* stands for constant/aligned, nc, na for non-constant and non-aligned respectively. We find this display useful in visualizing the different subsets of decomposition relations and paths in the graph as explained in the following paragraphs.

2 1 $\trianglerighteq$	*day*	*week*	*month*	*year*	*century*	*4-centuries*	*28-centuries*
day	ca						
week	ca	ca					
month	nc-a	nc-na	ca				
year	nc-a	c-na	ca	ca			
century	nc-a	c-na	ca	ca	ca		
4-centuries	ca	c-na	ca	ca	ca	ca	
28-centuries	ca	ca	ca	ca	ca	ca	ca

Table 5.8: Adjacency matrix of the Gregorian calendar structure

2 1 $\trianglerighteq$	*day*	*week*	*month*	*period*	*semester*	*school-year*
day	ca					
week	ca	ca				
month	nc-a	nc-na	ca			
period	nc-a	nc-a	X	ca		
semester	nc-a	nc-a	c-na	ca	ca	
school-year	nc-a	nc-a	c-na	ca	ca	ca

Table 5.9: Adjacency matrix of the Simon Fraser University calendar structure

Adjacency matrices represent the decomposition relation by its two aspects; constancy and alignment. We use the convention of referring first to the composed time unit (in the "matrix row") and then the component unit (in the "matrix column"). Thus for example the entry associated to *(month,week)* is non-constant/non-aligned, which is the way *month* decomposes into *week*. The adjacency matrix is, as expected, a triangular matrix, since decomposition is an antisymmetric relation (i.e., if a time unit A decomposes into a time unit B then B can not decompose into A, unless they are equal). Hence, if there is an entry in position (A, B), there is no entry in position (B, A) unless $A = B$. For example, there is no entry for *(week,month)*. The main diagonal corresponds to the limit case of a time unit decomposing into itself. The entries there are all constant/aligned, since a time unit decomposes into itself in a constant/aligned way, with repetition factor equal to 1, (a result was already exposed in Chapter 4).

One characteristic that can be observed in the matrix is the presence of paths with the same decomposition aspect in the structure. For example *28-centuries, 4-centuries, century, year* all relate with a constant/aligned decomposition in the Gregorian structure. This can be seen in the matrix in the entries *(28-centuries, 4-centuries); (4-centuries, century)*, etc.

5.4.2 Calendar substructures

Substructures of the complete calendar structure are defined as a subgraph considering one particular subrelation. We refer to them as *calendar substructures* of a particular aspect (constant, aligned or constant/aligned).

Definition 45 (Aspect calendar substructure) *Given a calendar structure and an "aspect" (either constant, aligned or constant/aligned), a* calendar aspect substructure *is defined by the graph composed by the same set of time units and the "aspect" decomposition subrelation.*

The graphical representation of the Gregorian constant/aligned substructure appears in Figure 5.10.

Substructures induced by aligned only decomposition (i.e. without taking constancy into account) of the Gregorian calendar appears in Figure 5.11.a. Figure 5.11.b shows the substructure induced by constant only decomposition.

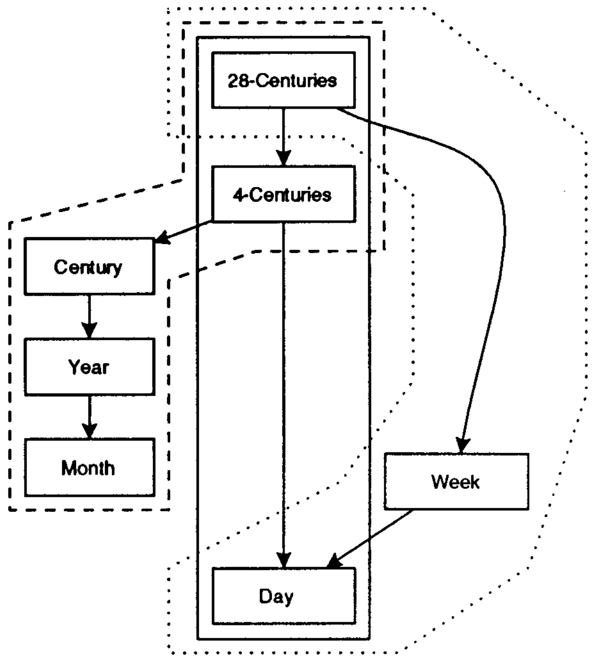


Figure 5.10: Constant/aligned Gregorian calendar substructure

Given a specific aspect the calendar substructure is unique, since the aspect defines a subrelation, (subset of the general decomposition) and thus a (unique) graph relating all time units with that particular sub-relation is determined.

5.4.3 Chains

Chains can be *aligned*, *constant*, or both (i.e. *constant/aligned.*) They are composed by time units in a *path* in the corresponding "aspect" calendar substructure.

Definition 46 (Chain of time units) *An "aspect" (constant, aligned or constant/-aligned) chain of time units is composed by all those time units in the hierarchy that form a consecutive linear sequence such that each time unit decomposes into the next one in the aspect way. Or, equivalently, an aspect chain is a path or sequence of connected time units in the corresponding aspect calendar substructure.*

For example, Figure 5.10 shows three constant/aligned chains in the Gregorian calendar structure: *<28-centuries, 4-centuries, century, year, month>*, *<28-centuries,*

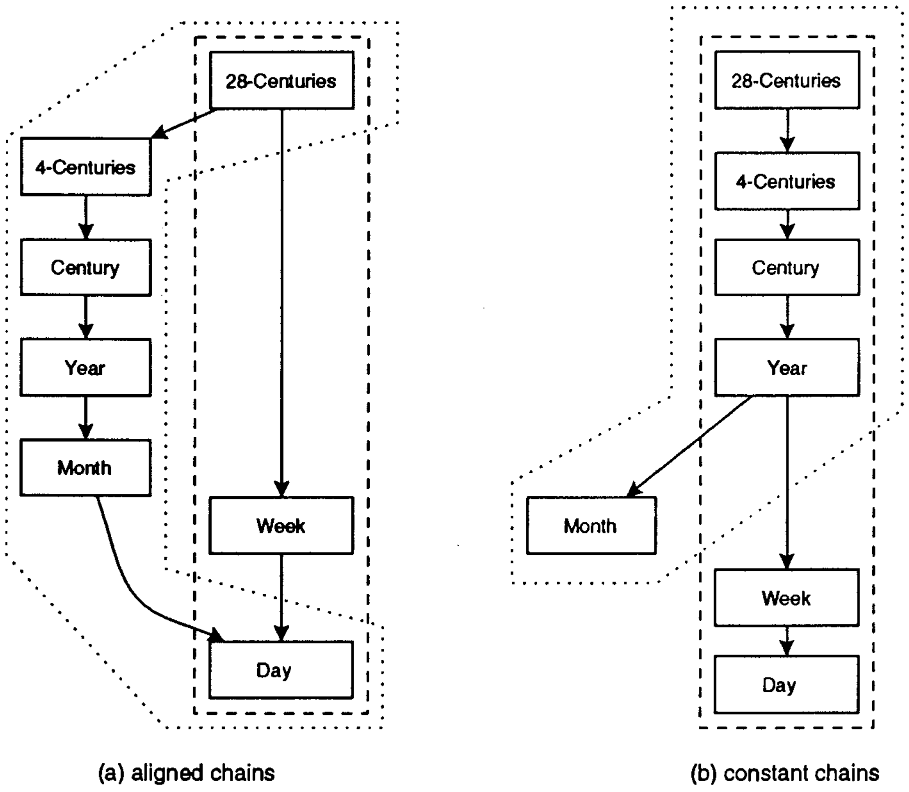


Figure 5.11: Aligned and constant Gregorian calendar substructures

4-centuries, day> and *<28-centuries, week, day>.*

Figure 5.12 shows the constant/aligned chains in the Simon Fraser university calendar structure: *<school year, semester>*, *<month>* and *<period, week, day>.*

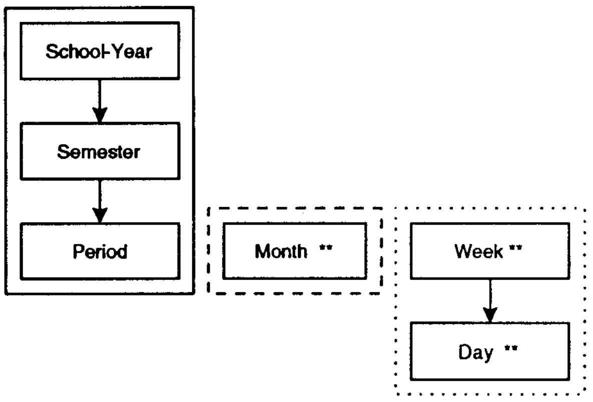


Figure 5.12: Constant/aligned chains in the SFU calendar structure

Property 15 *All pairs of time units within one chain decompose in the same way.*

Proof: This property follows from the definition of chains as belonging to paths in a calendar substructure. Also, the transitive characteristics of constancy and alignment ensure this result, since the product of two constant decomposition relations results in a constant relation and analogously this happens with alignment. ■

Property 16 *All time units in the hierarchy form part of at least one chain.*

Proof: In general, chains of a single time unit it can be conceived, thus proving this property. ■

For example < *month* > in the SFU calendar structure is a one time unit chain, (Figure 5.12).

Property 17 *A pair of time units belonging to two different chains either decompose differently than the aspect characterizing the substructure or do not relate by decomposition.*

Proof: These property follows from the definition; all time units decomposing in the characterizing aspect of the substructure belong to the same chain. ■

For example, *Century* and *week* are two time units that belong to two different constant/aligned chains and decompose in a constant/*non-aligned* way. *Period* and *month* in the SFU calendar structure, belong to different aligned chains and they do not relate with decomposition (in any aspect).

5.4.4 Chain extremes

Definition 47 (Chain extremes) *A time unit is a* chain extreme *if it is a chain* top-extreme *or a chain* bottom-extreme.

A chain top-extreme is the highest time unit in the hierarchy that belongs to that chain. Analogously, a chain bottom-extreme is the lowest time unit in the hierarchy that belongs to that chain.

For example, in the Gregorian constant/aligned substructure, *28-centuries* is a top-extreme of three chains. *Day* is a bottom-extreme of one of the chains.

Property 18 *A top-most time unit in the hierarchy is a particular case of a chain top-extreme. Analogously a bottom-most unit in the hierarchy is a particular case of a chain bottom-extreme.*

Proof: Both, the top-most and bottom-most time units belong to some chain, since every time unit in the hierarchy belongs to some chain. And since they are the respectively highest and lowest in the hierarchy they must be extremes of the chain(s) they belong to. ■

Property 19 (Existence of unique common extreme of all chains) *In any time unit hierarchy, it is possible to add a time unit acting as a common extreme to all the chains in the structure.*

Proof: The existence of a top-extreme is guaranteed by the existence of a common higher bound to any pair of given time units in the time unit hierarchy in both, the constancy and alignment sense. The existence of a lower bound of all time units in the hierarchy guarantees the existence of a common bottom-extreme of every chain, although only in the alignment sense. ■

For example, the Gregorian calendar structure as depicted in Figure 5.10 shows that *28-centuries* is a common top-extreme of the three chains in the structure.

All the development of calendar substructures and chains just presented provides for a better understanding of calendar structures and also establishes a foundation to define a formal language of time units, for example, extreme time units are envisioned as possible candidates for *primitive time units* to define a formal language.

5.5 Creating and modifying calendar structures

In this section we analyze the characteristics of the calendar structures defined that concern algorithms to create and update them. Updating a time unit hierarchy would include for example adding new time units to existing calendars, and derive new decomposition relations relating the added time unit to existing ones in the hierarchy. This is to be contrasted with algorithms dealing with time unit instances which *are based* on the units of this formalism, a topic addressed later.

A few comments are worth mentioning with respect to this (definitional) usage of the calendar structures.

The calendars analyzed, i.e. the Gregorian calendar, a university calendar, and others, typically have less than 10 time units. In fact, any measurement system will have a small, finite number of units. Many reasonable algorithms will do a good job obtaining the characteristics of new decomposition relations. Variations of temporal constraint propagation algorithms, (i.e. path consistency algorithms) could be used for propagating the decomposition characteristics, using the operations defined in Section 5.2 above, i.e. composition, intersection and inverse of decomposition. [VK86, PB91, MSK93] are examples of work within particular sub-cases of [All83], where such algorithms are used on time intervals (on the time line).

In Section 5.2 we observed that time units relate in a unique way. They are *classes* of time intervals. For example, *year* always decomposes into *day* in an aligned, non-constant way. That is to say, there are *not* several possible scenarios for two time unit classes to relate, but only one. In fact, the way one time unit decomposes into another can be seen as defining the time units involved. For example, it could be said that an ***hour*** is defined as a time unit so that a *day* decomposes into it in a constant and aligned way, with repetition factor 24. This restricted situation occurring with the decomposition relation results in intersection of decomposition relations being a trivial operation to perform in propagating constraints between time units.

Another issue to take into account is related to the results that can be obtained through composition, hence using the alignment and constancy transitivity tables, as well as calculating repetition factors via multiplication. In some cases composition results are deterministic, for example, a *non-constant* decomposition relation composed with any other decomposition relation will result in a *non-constant* one. These cases provide for useful constraint propagation. Other cases provide disjunctive information; for example the result of composing an *aligned* decomposition relation with a *non-aligned* one can result in either an aligned or a non-aligned decomposition.Non determined cases correspond to the existence of cycles in the calendar structure, as it was already addressed in previous sections. Thus, in those cases it must be known whether the time unit closes a cycle or does not. But in that case, we know (a priori) how this time unit relates to any other time unit, so there wouldn't be a need of using the transitivity tables.

Therefore, because of the (small) size of calendar structures and the characteristics just exposed, we can conclude that efficiency is not an issue in updating the calendar structures themselves. Furthermore, obtaining information from these structures shouldn't pose any problem either. The characteristics of the decomposition relation between every two time units can be recorded in an initial definition stage of a calendar in an auxiliary structure accessible by any pair of time units. A possible structure could be the adjacency matrix previously introduced. This information would occupy very little space. If m is the number of time units in the calendar (most likely less that 10), a structure with $\frac{C_2^m}{2} = \frac{m!}{2*2!(m-2)!}$ entries would record the information of all decomposition relations. Accessing to this auxiliary structure is an O(1) operation.

5.6 A suggested notation to represent specific time unit instances

We here address the issue of how to represent a specific time unit instance, for example *August 18, 1994*, in terms of the developed formalism. As it was already discussed, a simple instance of a time unit subclass is not yet a specific time unit instance. For example, an instance of *month* in the Gregorian calendar could be the fourth month (abbreviated as April) - but this instance does not represent a specific month yet (unless of course the application is dealing with a specific year only). Viewed extensionally, it represents a set of all specific instances of the month "April".

A specific time unit instance representation will require a sequence of expressions, which we call *calendar expressions.*

A calendar expression is defined by a conventional beginning reference point or *zero point* associated to the calendar and a finite sequence of pairs: $\text{zero}_{CAL} \oplus [(t_1, x_1), \ldots, (t_n, x_n)]$. Each pair (t_i, x_i) contains a time unit t_i from the calendar structure and a numeric expression x_i. Expressions include numbers and variables ranging in the set of integers. The pairs in the sequence are ordered so that any time unit in a pair *decomposes into* the time unit in the *following* pair in the sequence.

The numeric expression x in a pair (t, x) (other than the first pair in the sequence) represents the xth time unit t relative to the beginning of the *preceding* time unit in the sequence. The first pair in the sequence is numbered with respect to a beginning reference point. The last time unit in the sequence provides the precision of the time unit instance. The length of the sequence is limited by the number of levels in the calendar structure.

The following examples express simple calendar expressions in the Gregorian calendar. The zero point is the beginning of the Christian Era (*CE*).

Examples of calendar expressions

1. $CE \oplus [(year, 1994)]$
2. $CE \oplus [(year, X), (month, 4)]$

Fourth month within any year. (Viewed extensionally, it represents the set of all specific instances of the month "April"). X is a free variable.

3. $CE \oplus [(year, X), (day, 175)]$
   The 175th day from the beginning of any year.

4. $CE \oplus [(year, 1994), (week, 17), (day, 5)]$
   5th day within the 17th week of the year 1994.

5. $Curr \oplus [(month, 4), (day, 3)]$
   (In this example, "Curr" stands for the current year. The current year is numbered with respect to the zero point).

It can be observed that a calendar expression with no variables is a time unit instance of the last time unit in the calendar expression. Consequently, $\text{zero}_{CAL} \oplus [(t_1, x_1), \ldots, (t_n, x_n)]$ is a t_n-instance. Examples 3, 4 and 5 above are *day-instances.* Also, a calendar expression with variables is a set of time unit instances; it represents the temporal counterpart of a date-based repeated activity, a specific case of a non convex interval. Example 2 above represents the non convex interval containing the occurrences of the month *April.*

Simple calendar expressions similar to the ones exemplified above could be combined with logical operations like *disjunction* for example. Thus the temporal counterpart of more complex expressions could be represented. The following example corresponds to a repeated activity, within the Simon Fraser university calendar.

Example

The course CMPT 823 is offered every Tuesday at nine and every Thursday at eight every year, every semester:

$$\forall\, Y, S, W\ (CE \oplus [(year, Y), (semester, S), (period, Classes), (week, W), (day, Tuesday), (hour, 9)]\ ) \vee (CE \oplus [(year, Y), (semester, S), (period, Classes), (week, W), (day, Thursday), (hour, 8)]\ )$$

Three free variables are associated to year, semester and week, respectively Y, S, W. Notice that "classes" is a *period-instance name* which abbreviates the number 1, it is the first type of *period* within the reference time unit *semester* in the SFU calendar, i.e. *classes, exams* and *break* periods.

The computation of different useful operations using calendar expressions will be influenced greatly by the way the time units in the calendar expression decompose into each other. Corresponding to aligned (constant/aligned) chains in the time unit hierarchy, we define *aligned (constant/aligned) calendar expressions.* These particular cases are expected to behave very efficiently. An example of an operation that will be more efficient with a constant/aligned calendar expression is the transformation from one format in a variant calendar to a format in the basic calendar associated.

For example a calendar expression expressed in the SFU calendar can be transformed to a certain format in the Gregorian calendar.
$CE \oplus [(year, 1993), (semester, 1), (month, 1)] = CE \oplus [(year, 1993), (month, 9)]$

The development of a formal language to represent time unit instances with calendar expressions and algorithms to reason with them appears as a very promising future research venue.

Chapter 6

Conclusion

Suppose time is a circle, bending back on itself. The world repeats itself, precisely, endlessly...

In this world, there are two times. There is the mechanical time and there is the body time. The first is as rigid and metallic as a massive pendulum of iron that swings back and forth, back and forth. The second squirms and wriggles like a bluefish in a bay...

Suppose that time is not a quantity but a quality, like the luminescence of the night above the trees just when the rising moon has touched the treeline. Time exists, but it cannot be measured...

Imagine a world ... `Einstein's Dreams, Alan Lightman`

This research was inspired by current efforts to develop a system for reasoning about activities that occur within calendars or using dates. The ultimate goal is to be able to represent and reason efficiently about repeated activities using the formalism.

Much of our work is dedicated to defining a hierarchical structure of time units, and elaborating on decomposition among the time units, systematically analyzing their mathematical properties. We take into consideration the distinction between time unit classes, named time units and specific time unit instances. (For example "*month*", "August", and "August 1994".) Durations of time units are precisely defined taking this distinction into consideration. We have developed a hierarchical structure based on these time units, allowing for the definition of different calendars, such as business calendars or university calendars, based on the Gregorian calendar. The detailed study of this domain provides a useful basis for further abstraction and specification of a formal theory dealing with time units and time unit instances. The temporal counterpart of repeated activities is envisioned to be represented in a straightforward way with time unit instances.

The framework developed, including all the structural properties studied, constitutes an extensive study of the temporal domain with time measured in a discrete way, based on calendars and dates. Some of the related research we have referenced (in chapter 1) propose some kind of formalization of time units. Such is the case in, for example, [Lad86b, LMF86, PB91, MSK93]. However, these works mostly treat the representation of temporal entities of a calendar as a simplified side issue; it is not their main concern, whereas it is a main component of our work. Still, [Lad86b] proposes a way of defining time units, but as we commented in chapter 1, it is essentially different from our work and it is based on a more rigid schema than ours. In any case, the extensive study we do of this domain is not present in any of the literature reviewed. We have developed a different, general and formalized representation of temporal entities and time units, building a coherent and consistent structure that formalizes dates in calendars.

In the following section we highlight some issues that are of special interest in this thesis. Finally, future research venues are proposed.

6.1 Highlights

Classes of intervals As part of the process of defining the elements of the domain under study, we propose an initial definition of relations between generic classes of intervals. Classes of intervals and the relations among them extend those of the basic interval algebra of [All83]. The generalization is done in a universal/existential formulation. For example, a generic class of intervals *MEETS* another when every instance of the former *meets* at least one instance of the latter, where *meets* is one of the thirteen basic interval relations in the above mentioned algebra. We envision that this abstraction has a potential utility for other applications in temporal reasoning independent of calendar structures, such as reasoning about recurrence and non convex intervals, and relations between them, and leaves an interesting open venue for future research.

Interconnection of time units and decomposition The main objects in the temporal domain as formalized in this thesis are very interconnected. For example, time units are entities related by the decomposition relation, but also decomposition can be seen as defining the time units. This interconnection is clearly seen when considering, for example, the time units *day* and *week*. In fact, *week* is inherently a series of seven days: "a series of 7-day cycles used in various calendars" [Web]. Having in mind this interconnection, we have taken special care to develop the formal apparatus and definitions in a bottom up way, avoiding any possible circularity.

Global idea on how definitions have been organized In the following we present a global idea on how we based the definition of the main elements in the domain in a bottom-up fashion. The decomposition relation is defined based on Allen's interval relational algebra (Chapter 2). More precisely, we define relations between classes of intervals and further define decomposition on top of this abstraction. Independently, time units have been defined as classes of objects with certain attributes (Chapter 3). These attributes are presented in a functional manner. Such is the case of the "range of possible durations" of time unit instances (referred as "duration" of the class). The two independently developed concepts of decomposition relations and time units and their durations are compared in Chapter 4. Decomposition relations and durations are proven to have strong commonalties, as expected. Of key importance, decomposition is proven to define a partial order among time units, thus giving rise to the calendar structures or time unit hierarchies (Chapters 4 and 5). The set of time units can be formally characterized in a logical language. We envision a language to define the set of time units in a manner such that there are some primitive time units and all the others are defined recursively based a particular case of the decomposition relation. This idea is extended further in the section addressing possible future extensions of this research.

Mathematical foundation One problem that had to be faced is that durations of time units (as a class) are not necessarily fixed given the time unit and the measure. For example, a year lasts 365 or 366 days. We defined the concept of *min-max pairs* to represent range of possible values the instances durations can have, and hence a basis to define durations of classes. These pairs are defined as a mathematical entity, and a partial order is defined on that set. The partial order of decomposition on the set of time units is based on the partial order of min-max pairs. At another level of definition, the calendar structure is proposed as a system of measures, and a parallel of the relations between the units and divisibility concepts is developed. This parallel reflects expected properties of systems of measures, regarding multiple and divisor units.

Concise characterization of decomposition It is worth mentioning that the decomposition relation between time units is captured with only two aspects: *constancy* and *alignment*. Each aspect only has two possible values: positive and negative, (constant and non-constant and analogously aligned and non-aligned), thus generating four possible cases time units can relate with decomposition. We claim that this characterization covers any system of measures based on discrete units and a repetitive containment relation. Operations on decomposition relations have also been analyzed taking into account these two aspects. These operations are composition or product, intersection or sum, and inverse. As part of the study of these operations, transitivity of constancy and alignment is systematically analyzed. As well, these operations are a basis for defining a relational algebra, in an analogous way as a relational algebra has been defined on the basic interval relations. The elements of the algebra in this case would be the four possible decomposition variations.

Structural properties - cycles, chains and other beasts A detailed study has been done regarding the structural properties in the time unit hierarchies. Cycles and cycle-closing time units are related to how constancy and alignment behave "propagating" up and down in the hierarchy. We corroborate that certain properties hold in present-day and historical calendars and propose such properties as a characteristic that holds in any calendar structure. An example of such properties in any hierarchy is that all time units should have a common time unit into which they decompose in an aligned way. An alternative characterization of the hierarchical structure considers subrelations of decomposition according to the different aspects of decomposition: aligned, non-aligned, constant, non-constant and the possible combinations. Calendar substructures are defined then associated to one particular aspect. A path in such a substructure is referred to as a *chain.* Chains therefore are a sequence of time units that decompose in the same way with respect to constancy and/or alignment. We find chains useful to provide more insight into calendar structures, and also as a basis for future work defining a formal language to characterize time units. As well, we envision that reasoning with instances of time units in one chain will produce very efficient operations. Constant/aligned chains are expected to be particularly efficient.

6.2 Future work

There are many possible venues to continue exploring based on this research. We envision that the structural properties studied of calendar structures provide for a direct basis to define a formal language characterizing temporal units, or eventually any unit in a discrete system of measures that relate with containment repetitive relations. Calendar structures can be organized according to different kinds of chains, i.e. aligned, constant, constant/aligned. Based on chains, a characterization of the set of time units can be conceived on a set of primitive time units and the decomposition relation. Extremes of chains could be potential primitive time units of the language.

Also, research should be done formalizing a language to represent time unit *instances.* We have suggested a starting point for a possible notation which we called *calendar expressions* to represent specific time unit instances (in Chapter 5). The examples there show the expressive power of this suggested notation. We envision that these expressions should provide for a straightforward way of representing the temporal counterpart of repeated activities.

Another important research direction is that of studying algorithms that would best fit with this formalism, so that we may obtain efficient inferences when reasoning about repeated activities. Algorithms developed for qualitative and/or quantitative temporal constraint satisfaction problems, or variations, could be considered in this matter [DMP91, KL91, COS93]

Other possible continuations of this work include exploring the results exposed in this thesis in other domains, for example the spatial domain. As well, the extension of the interval relational algebra to relations among classes of intervals constitutes a first step in a certain abstraction direction that should be further explored.

We believe that this thesis contributes to clarify the inner structure of conventionally measured time, providing a deeper understanding of the temporal domain as based on calendars and dates. It provides a coherent and consistent foundation to develop practical and efficient time specialists for dealing with scheduling problems and sets the stage for continuing research in numerous venues.

Appendix A

Proof of theorems

"Manners are not taught in lessons" - said Alice. "Lessons teach you to do sums, and things of that sort."

"Can you do Addition? " - the White Queen asked."What's one and one and one and one and one and one and one and one and one? "

"I don't know."- said Alice - "I lost count."
"She can't do Addition" - the Red Queen interrupted. "Can you do Subtraction? Take nine from eight."

"Nine from eight I can't, you know "- Alice replied very readily: "but –"

"She can't do Subtraction" - said the White Queen. "Can you do a Division? Divide a loaf by a knife" ...
`Through the Looking glass, and what Alice found there, Lewis Carroll`

A.1 Partial order between min-max pairs

In Chapter 3 min-max pairs were defined as: " an ordered pair of natural numbers (a, b) such that $a \leq b$". As well, an order relation was defined. We prove here Theorem 7, i.e. that this relation is indeed a *partial order*.

First, we recall the definition of this relation between min-max pairs: (Definition 15 in chapter 3)

An order relation between min-max pairs is denoted by $\overset{p}{<}$.
Let (a_1, b_1) and (a_2, b_2) be two min-max pairs.

1. $(a_1, b_1) \overset{p}{=} (a_2, b_2)$ *iff* $a_1 = a_2$ *and* $b_1 = b_2$.

2. $(a_1, b_1) \overset{\text{P}}{<} (a_2, b_2)$ *iff*
   $b_1 < a_2$ *or*
   $(b_1 = a_2$ *and* $(a_1, b_1) \;\not\overset{\text{P}}{=}\; (a_2, b_2)$
3. $(a_1, b_1) \overset{\text{P}}{\leq} (a_2, b_2)$ *iff*$(a_1, b_1) \overset{\text{P}}{<} (a_2, b_2)$ *or* $(a_1, b_1) \overset{\text{P}}{=} (a_2, b_2)$.

Theorem 7 $\overset{\text{P}}{\leq}$ *is a partial order.*

Proof: The relation has to be proven to be reflexive, antisymmetric and transitive.

Reflexive
$\forall$ min-max pair (a, b), $(a, b) \overset{\text{P}}{\leq} (a, b)$ - This holds as a direct consequence of the definition. ■

Antysimmetric
For any two min-max pairs it must be proved that:
if $(a_1, b_1) \overset{\text{P}}{\leq} (a_2, b_2)$ and $(a_2, b_2) \overset{\text{P}}{\leq} (a_1, b_1)$, then $(a_1, b_1) \overset{\text{P}}{=} (a_2, b_2)$.

By definition, $(a_1, b_1) \overset{\text{P}}{\leq} (a_2, b_2)$ iff $(a_1, b_1) \overset{\text{P}}{<} (a_2, b_2)$ or $(a_1, b_1) \overset{\text{P}}{=} (a_2, b_2)$,
iff $b_1 < a_2$ *or* $(a_1, b_1) \overset{\text{P}}{=} (a_2, b_2)$.
Similarly, $(a_2, b_2) \overset{\text{P}}{\leq} (a_1, b_1)$ iff $b_2 < a_1$ *or* $(a_2, b_2) \overset{\text{P}}{=} (a_1, b_1)$.
But, by definition of a min-max pair, it holds that: $a_1 \leq b_1$ and $a_2 \leq b_2$,
So it is not possible that simultaneously $b_1 < a_2$ and $b_2 < a_1$ (or, rephrasing it, $a_1 \leq b_1 < a_2 \leq b_2$ is inconsistent with $b_2 < a_1$.)
Therefore, $\overset{\text{P}}{=}$ is the only relation holding between the two pairs. ■

Transitive
For any two min-max pairs it must be proved that:
if $(a_1, b_1) \overset{\text{P}}{\leq} (a_2, b_2)$ and $(a_2, b_2) \overset{\text{P}}{\leq} (a_3, b_3)$, then $(a_1, b_1) \overset{\text{P}}{\leq} (a_3, b_3)$.

On one hand, $(a_1, b_1) \overset{\text{P}}{\leq} (a_2, b_2)$ iff $b_1 < a_2$ *or* $(a_1, b_1) \overset{\text{P}}{=} (a_2, b_2)$.

Similarly, $(a_2,b_2) \overset{P}{\leq} (a_3,b_3)$ iff $b_2 < a_3$ *or* $(a_2,b_2) \overset{P}{=} (a_3,b_3)$.
If either pair is equal to another, then transitivity holds trivially. Else, from the definition of min-max pair, $a_2 \leq b_2$, therefore, $b_1 < a_2 \leq b_2 < a_3$ and it follows that $b_1 < a_3$, which holds when $(a_1,b_1) \overset{P}{<} (a_3,b_3)$. ■

Not a total order
To prove that the order defined is not total it suffices to observe that there exist pairs that are not comparable. For example, $(1,3)$ and $(2,4)$. ■

A.2 Decomposition and duration theorem

We prove here Theorem 9 from Chapter 4.

Theorem 9 (Decomposition and duration) *Let T and S be two time units in the same calendar or variant ($T, S \in TUS_{CAL}$) such that T decomposes into S ($T \trianglerighteq S$). Then T has a greater or equal general duration than S.*

Proof:

Considering that durations are min-max pairs, this theorem can be reformulated using the order relation defined between min-max pairs:
Let T and S be two time units in the same calendar or variant ($T, S \in \mathrm{TUS}_{CAL}$) such that T decomposes into S (T $\trianglerighteq$ S). then $dur(T, Basic) \overset{P}{\geq} dur(S, Basic)$. "Basic" is the time unit used to measure general durations in the hierarchy associated to TUS_{CAL}. To simplify the notation in the proof, we define the general duration of the class T as $d(T) \equiv_{notation} dur(T, Basic)$ and the general duration of instances t of a class T as $d@(t) \equiv_{notation} dur@(t, Basic)$.

The idea we follow is to restrict the time units and decomposition parameters so that the theorem holds. Rephrasing the theorem statement, we restrict parameters so that if T decomposes into (a sequence of) S's, then any T instance has a larger time span than any single S instance. Specifically, we relate the duration of the component time unit (S) and the repetition factor of the decomposition of T into S.

Four cases are analyzed, those stemming from the combination of the two aspects of decomposition; alignment and constancy. In the four cases, the methodology followed is analogous and can be summarized as follows: The duration of the component time unit (S) is assumed known (parametrically). Then, the duration of the composed time unit (T) is expressed as a function of the former. (This is based on the decomposition case; aligned or not, constant or not). I.e. it is assumed that $d(S) \overset{P}{=} (a, b)$ and $d(T)$ is expressed as a function of (a, b), and the repetition factor of the decomposition. Then, the duration of S is imposed to be less than the duration of T: $d(S) \overset{P}{\leq} d(T)$. Restrictions among the parameters involved are obtained and thus, with decomposition restricted in this way, the theorem holds.

Case (I): constant/aligned decomposition

Let T and $S \in \text{TUS}_{CAL}$, $\text{T} \trianglerighteq \text{S}$.
Cons(T,S,n)
Alig(T,S)
(T decomposes into S in a constant/aligned way with a fixed repetition factor n)

Let $d(S) \overset{P}{=} (a, b)$ $(iff\ \forall s\ (\textit{in-class}(s, S))$ it holds that $a \leq\ d@(s) \leq\ b))$
Since it is an aligned and constant decomposition, with repetition factor n, the duration of T is bounded by: $d(T) \overset{P}{=} (na, nb)$ $(iff\ \forall t\ (\textit{in-class}(t, T))$ it holds that $na \leq d@(t) \leq\ nb)$. This is so because T is composed of exactly n S's.

Next we impose $d(S) \overset{P}{\leq} d(T)$ and analyze the two possibilities: $d(S) \overset{P}{=} d(T)$ or $d(S) \overset{P}{<} d(T)$.

1. $d(S) \overset{P}{=} d(T)$ iff $(a, b) \overset{P}{=} (na, nb)$ iff $a = na$ and $b = nb$, which only holds if $n = 1$. By property 6, this holds iff $T = S$, i.e., it is the case of constant aligned decomposition of a time unit into itself.

2. $d(S) \overset{P}{<} d(T)$ iff $(a, b) \overset{P}{<} (na, nb)$, iff by definition,
   i) $b < na$ or

ii) $b = na$ and $(a,b) \overset{P}{\neq}(na,nb)$.

The second case *(ii)* only happens if $\exists s$ $(\textit{in-class}(s,S))/d@(s) = b = na$ and if $\exists t$ $(\textit{in-class}(t,T))/d@(t) = na$, with $n \neq 1$ (since $(a,b) \overset{P}{\neq}(na,nb)$)

That is to say, there exists an instance of S with the same duration as an instance of T and T and S are different time units. This is counterintuitive with the class T decomposing into the class S. So only case *(i)* applies, and the constraint $b < na$ must be imposed.

This constraint reflects the intuition that the largest possible instance of S (with duration b) must have a shorter duration than the shortest instance of T, with duration na. ■

Case (II): non-constant/aligned decomposition

Let T and $S \in$ TUS$_{CAL}$, T $\trianglerighteq$ S.
Non-Cons(T,S,(l,u))
Alig(T,S)
(T decomposes into S in a non-constant/aligned way with a repetition factor bounded by a lower bound l and an upper bound u)

Let $d(S)\overset{P}{=}(a,b)$ $(iff\ \forall s\ (\textit{in-class}(s,S))$ it holds that $a \leq d@(s) \leq b))$
The duration of T is bounded by:
$d(T)\overset{P}{=}(la,ub)$ $(iff\ \forall t\ (\textit{in-class}(t,T))$ it holds that $la \leq d@(t) \leq ub)$.
This is so because there are instances of T composed of r S's, with $l \leq r \leq u$.

Next we impose $d(S)\overset{P}{\leq} d(T)$ and analyze the two possibilities: $d(S)\overset{P}{=} d(T)$ or $d(S)\overset{P}{<} d(T)$.

1. A time unit can not decompose into another in a non-constant way and have the same duration. This option is not possible in this decomposition case.

2. $d(S) \overset{p}{<} d(T)$ iff $(a,b) \overset{p}{<} (la, ub)$, iff by definition,
   i) $b < la$ or
   ii) $b = la$ and $(a,b) \overset{p}{\not=} (la, ub)$.

   The second case *(ii)* only happens if $\exists s$ (*in-class*(s, S))$/d@(s) = b = la$ and if $\exists t$ (*in-class*(t, T))$/d@(t) = la$, with $l \neq 1$ (since $(a,b) \overset{p}{\not=} (la, ub)$)

   That is to say, there exists an instance of S with the same duration as an instance of T and T and S are different time units. This is counterintuitive with the class T decomposing into the class S. So only case *(i)* applies, and the constraint $b < la$ must be imposed.

   This constraint reflects (just as in the first case of constant/aligned decomposition) the intuition that the largest possible instance of S (with duration b) must have a shorter duration than the shortest instance of T, with duration la. The results obtained, resemble (as expected) the results obtained when the decomposition is constant. In this case, the constraint applies to the lower bound of the decomposition repetition factor instead of the (unique) repetition factor. ■

Case (III): constant/non-aligned decomposition

Let T and $S \in \mathrm{TUS}_{CAL}$, T $\trianglerighteq$ S.
Cons(T,S,n)
Non-Alig(T,S)
(T decomposes into S in a constant/non-aligned way with a repetition factor n)

Let $d(S) \overset{p}{=} (a,b)$ (*iff* $\forall s$ (*in-class*(s,S)) it holds that $a \leq d@(s) \leq b$))
By theorem 3, $n \geq 2$. (This result follows from the fact that there exists at least one complete s interval in the decomposition of t, $\forall t$ *in-class*(t, T). (See Figure A.1).

The duration of all instances t of T is (not strictly) bounded by:

$\alpha)$ $(n-2)a < d@(t) < nb$

This situation occurs when there exist instances where both extreme intervals of the

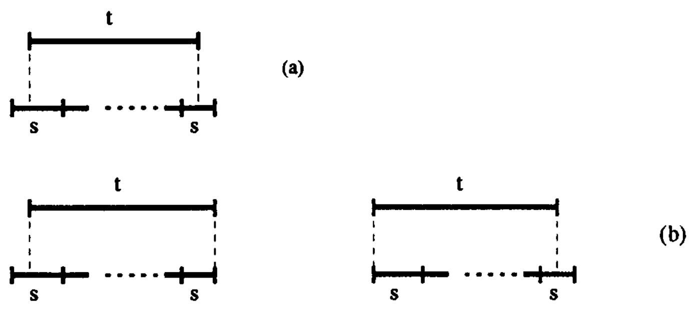


Figure A.1: Two situations of non-aligned decomposition

s sequence overlap with t (as in Figure A.1.a). $n \geq 3$ in this case.

β) $(n-1)a < d@(t) < nb$
This situation occurs when for all instances only one extreme interval of the s sequence overlaps with t and the other either starts or finishes t. (as in Figure A.1.b). $n \geq 2$ in this case.

Considering the two situations (α) and (β) together, the duration of all t instances is bounded by:

$(n-k)a < d@(t) < nb)$, where $k = 1$ or $k = 2$ and $n - k > 1$.
I.e in a non-strict sense, $d(T) \overset{\text{P}}{=} ((n-k)a, nb)$.
(intuitively, $n-k$ stands for the number of complete s intervals contained in t. $n-k > 1$, since the definition (and intuition) of non-aligned decomposition requires at least one complete subinterval)

Next we impose $d(S) \overset{\text{P}}{\leq} d(T)$ and analyze the two possibilities: $d(S) \overset{\text{P}}{=} d(T)$ or $d(S) \overset{\text{P}}{<} d(T)$.

1. $d(S) \overset{\text{P}}{=} d(T)$ can not hold, since $(a, b) \overset{\text{P}}{\neq} ((n-k)a, nb)$, since for any a, b it can not simultaneously hold that $a = (n-k)a$ and $b = nb$, with $k = 1$ or $k = 2$.

2. $d(S) \overset{\text{P}}{<} d(T)$ iff $(a, b) \overset{\text{P}}{<} ((n-k)a, nb)$, iff by definition,
   i) $b < (n-k)a$ or

ii) $b = (n-k)a$ and $(a,b) \not\overset{\text{p}}{=} ((n-k)a, nb)$.

The second case *(ii)* is applicable; as it was just explained $(a,b) \not\overset{\text{p}}{=} ((n-k)a, nb)$ with $k=1$ or $k=2$.

So both cases *(i)* and *(ii)* apply, and the constraint $b \leq (n-k)a$ must be imposed. k depends on the style of non-aligned decomposition as discussed, $k=1$ or $k=2$.

This constraint reflects the intuition that the largest possible instance of S (with duration b) must have a shorter or equal duration than the (strict) bound of the duration of the shortest instance of T, $(n-k)a$. ■

Case (IV): non-constant/non-aligned decomposition

Let T and $S \in \text{TUS}_{CAL}$, T $\trianglerighteq$ S.
Non-Cons(T,S,(l,u))
Non-Alig(T,S)
(T decomposes into S in a non-constant/non-aligned way with a repetition factor bounded by a lower bound l and an upper bound u)

This case combines the characteristics of non-alignment and non-constancy addressed separately in the previous two cases *II* and *III*. We therefore directly write down the result in this case:

Let $d(S) \overset{\text{p}}{=} (a,b)$ $(iff\ \forall s\ (\textit{in-class}(s,S))$ it holds that $a \leq\ d@(s) \leq\ b))$
then $d(T) \overset{\text{p}}{=} ((l-k)a, ub)$ in a non-strict sense, where $k=1$ or $k=2$ and $l-k>1$.
I.e. $\forall t\ (\textit{in-class}(t,T))$ it holds that $(l-k)a < d@(t) < ub)$.

Reasoning analogously to the previous cases, the constraint that must be imposed is obtained: $b \leq (l-k)a$. k depends on the style of non-aligned decomposition. ■

The results obtained with this theorem are summarized in Table A.1. They correspond to the intuition that certain restrictions must be imposed on the repetition factor in case that the component time unit S is of non-equal durations, that is, when there exist in fact instances of S with different durations ($d(S) \overset{\text{p}}{=} (a,b)$ with $a \neq b$.)

The constraints avoid the situation when S instances vary so much in their duration that some instance of S could decompose into some instance of T. In case that S is of equal duration, that is, all instances of S have the same duration ($d(S)\overset{\mathrm{P}}{=}(a,a)$), the constraints are trivial in the four cases: $a < na$ with $n > 1$, $a < la$ with $l > 1$, $a < (n-k)a$ with $(n-k) > 1$ and $a < (l-k)a$ with $(l-k) > 1$.

Case	Parameters Involved	Constraints
aligned, constant	repetition factor $= n$ a,b / $d(S)\overset{\mathrm{P}}{=}(a,b)$	$b < na$ $n > 1$
aligned, non-constant	bounds of rep. fact. l,u a,b / $d(S)\overset{\mathrm{P}}{=}(a,b)$	$b < la$ $l > 1$
non-aligned, constant	repetition factor $= n$ a,b / $d(S)\overset{\mathrm{P}}{=}(a,b)$	$b < (n-k)a$ $k = 1$ or $k = 2$, $n-k > 1$
non-aligned, non-constant	bounds of rep. fact. l,u a,b / $d(S)\overset{\mathrm{P}}{=}(a,b)$	$b < (l-k)a$ $k = 1$ or $k = 2$, $l-k > 1$

Table A.1: Summary theorem: constraints of the decomposition of T into S.

A.2.1 Example with results of the theorem

We present here an example of the situation that would arise in case that the restrictions were not taken into consideration. This shows the reasonable nature of the restrictions obtained.

Let T and $S \in \mathrm{TUS}_{CAL}$, $\mathrm{T} \trianglerighteq \mathrm{S}$.

Cons(T,S,n)

Alig(T,S)

$d(S)\overset{\mathrm{P}}{=}(1,2)$.

The constraint imposed by the theorem in this case requires $2 < n$.

These parameters indicate that $\forall s$ (*in-class*(s,S)) it holds that $1 \leq d@(s) \leq 2)$), and given these durations the repetition factor of T decomposing into S must be greater than 2.

To illustrate, we analyze what would happen if $n = 2$.

Since $d(S) \overset{\mathrm{P}}{=} (1,2)$, $\exists s_1, s2$ (*in-class*(s_1, S) $\wedge$ *in-class*(s_1, S)) with $d@(s_1) = 1$ and $d@(s_2) = 2$. If the repetition factor of the decomposition of T into S is 2, there exist 3 different possible lengths for instances of T:
$t_1 =< s_1, s_1 >$ so that $d@(t_1) = 2$
$t_2 =< s_1, s_2 >$ (or $t_2 =< s_2, s_1 >$) so that $d@(t_2) = 3$
$t_1 =< s_2, s_2 >$ so that $d@(t_3) = 4$

But then, there would exist one instance of T, t_1, with the same length as s_2, an instance of S, which is not reasonable.

Appendix B

Summary of concepts related to time units

Table B.1 summarizes main concepts related to time units.

Concept	Examples	Presented in	Comment
time unit	"month"	Chapter 3	
time unit class	"month"	Chapter 3	Same as time unit, stressing its class nature, particular case of class of time intervals.
named time unit	February	Chapter 3 and Section 4.2.1	Subclass of a time unit class, all instances have the same name.
time unit instance	February 1994	Chapter 3	A specific instance, particular case of time interval.
time interval	Months from February 1994 to April 1994	Chapter 2	Any interval on the time line, not necessarily instance of one single time unit.
class of time intervals	All September's and October's of years 1994, 1995 and 1996.	Chapter 2	Set of time intervals with some characteristic in common.
time unit hierarchy	Gregorian calendar structure	Chapter 5	Structure relating time units with decomposition
chain of time units	< 28-centuries, week, day>	Section 5.4	Sequence of time units decomposing into each other in the same way.

Table B.1: Main concepts related to time units

Bibliography

[AH85] J. F. Allen and P. J. Hayes. A common-sense theory of time. In *Proc. of the Ninth IJCAI-85*, pages 528–531, Los Angeles, CA, 1985.

[AH89] J. F. Allen and P. J. Hayes. Moments and points in an interval-based temporal logic. *Computational Intelligence*, 5:225–238, 1989.

[All83] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[BL92] H. Bestougeff and G. Ligozat. *Logical Tools for Temporal Knowledge Representation.* Ellis Horwood Lmtd, England, 1992.

[CD94a] D. Cukierman and J. Delgrande. Abstract: Time units and calendars. In *Proc. of AAAI'94*, page 1436, Seattle, USA, 1994.

[CD94b] D. Cukierman and J. Delgrande. Hierarchical decomposition of time units. In *Proc. of the Workshop of Temporal and Spatial reasoning, in conjunction with AAAI-94*, pages 11–17, Seattle, USA, 1994.

[Col71] L. Coleman. *A Book of Time.* Lowe and Brydone, London, 1971.

[COS93] D. Cukierman, R. Ovans, and S. Sloseris. Interactive activity scheduling with object-oriented constraint logic programming. In *Proc. of the Eigth International Conference on Applications of Artificial Intelligence in Engineering, AIENG-93*, pages 583–598, Toulouse, France, 1993.

[CR88] J. Clifford and A. Rao. A simple, general structure for temporal domains. In C. Rolland, F. Bodart, and M. Leonard, editors, *Temporal aspects of information systems*, pages 17–28. Elsevier Science Publishers B.V. (North-Holland), 1988.

[DMP91] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[GS92] M. C. Golumbic and R. Shamir. Algorithms and complexity for reasoning about time. In *Proc. of AAAI-92*, pages 741–747, 1992.

[Hob85] J. Hobbs. Granularity. In *Proc. of the Ninth IJCAI-85*, pages 1–2, 1985.

[JT52] B. Jonsson and A. Tarski. Boolean algebras with operators ii. *American Journal of Mathematics*, 74:127–162, 1952.

[KL91] H. A. Kautz and P. B. Ladkin. Integrating metric and qualitative temporal reasoning. In *Proc. of AAAI-91*, pages 241–246, 1991.

[Koo89] J. A. G. M. Koomen. Localizing temporal constraint propagation. In *Proc. of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 198–202, Toronto, 1989.

[Lad86a] P. Ladkin. Time representation: A taxonomy of interval relations. In *Proc. of the AAAI-86*, pages 360–366, 1986.

[Lad86b] P. B. Ladkin. Primitives and units for time specification. In *Proc. of the AAAI-86*, pages 354–359, 1986.

[Lig91] G. Ligozat. On generalized interval calculi. In *Proc. of the AAAI-91*, pages 234–240, 1991.

[LM94] P. B. Ladkin and R. D. Maddux. On binary constraint problems. *Journal of the ACM*, 41(3):435–469, 1994.

[LMF86] B. Leban, D. D. McDonald, and D. R. Forster. A representation for collections of temporal intervals. In *Proc. of the AAAI-86*, pages 367–371, 1986.

[MB83] J. Malik and T. O. Binford. Reasoning in time and space. In *Proc. of the eight IJCAI-83*, pages 343–345, 1983.

[Moe82] K. P. Moesgaard. Basic units in chronology and chronometry. In *Gregorian Reform of the Calendar, Proceedings of the Vatican Conference to Commemorate its 400th. anniversary*, pages 3–14, 1982.

[MS90] S. A. Miller and L. K. Schubert. Time revisited. *Computational Intelligence*, 6(2):108–118, 1990.

[MSK93] R. A. Morris, W. D. Shoaff, and L. Khatib. Path consistency in a network of non-convex intervals. In *Proc. of the IJCAI-93*, pages 655–660, 1993.

[PB91] M. Poesio and R. J. Brachman. Metric constraints for maintaining appointments: Dates and repeated activities. In *Proc. of the AAAI-91*, pages 253–259, 1991.

[Ped82] O. Pedersen. A glossary of technical terms. In *Gregorian Reform of the Calendar, Proceedings of the Vatican Conference to Commemorate its 400th. anniversary*, pages 299–321, 1982.

[Rus82] F. Russo. Contemporary discussions on the reform of the calendar. In *Gregorian Reform of the Calendar, Proceedings of the Vatican Conference to Commemorate its 400th. anniversary*, pages 287–297, 1982.

[Van83] J.F.A.K. VanBenthem. *The Logic of Time.* Synthese Library, Volume 156. D. Reidel Pub. Co., 1983.

[VK86] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proc. of the AAAI-86*, pages 377–382, 1986.

[VKV89] M. Vilain, H. Kautz, and P. VanBeek. Constraint propagation algorithms for temporal reasoning: a revised report. In Weld and de Kleer, editors, *Readings in Qualitative reasoning about physical systems.* Morgan Kaufmann, 1989.

[Web] *Webster's Ninth New Collegiate Dictionary.*