Temporal Constructs for a Web Language

François Bry and Stephanie Spranger

University of Munich, Germany, http://www.pms.informatik.uni-muenchen.de

Abstract

This paper reports on enriching a (logic-based) query and transformation language for the Web, called Xcerpt, with temporal constructs and temporal reasoning capabilities. Advanced Web applications often refer to temporal data, especially to *time points*, complex *time intervals*, and *durations*. Salient to many advanced Web applications is that they often refer to various *calendars*. However, current Web languages and formalisms have rather primitive temporal constructs and temporal data processing capabilities – if any. The focus of the project this paper reports on is not to specify yet another temporal logic and/or calculus. Instead, it is to select from existing temporal logics and calculi a coherent set of concepts and processing primitives to be added to a Web query and transformation language.

1 Introduction

Many traditional Web sites and pages refer explicitly or implicitly to temporal data. Many advanced Web applications such as web-based information systems, adaptive Web systems, and Semantic Web applications also refer to more complex temporal notions, especially to *time points*, *time intervals*, and *durations*. Examples of such advanced Web systems are news servers, scheduling systems, e-commerce systems, and various 'Web services' like flight booking, online banking, or train schedules. Salient to advanced Web applications is that they often explicitely or implicitly use various *calendars*. However, an essential building block of web-based systems, i.e. the query and transformation languages for the Web have rather primitive temporal constructs and temporal data processing capabilities – if any. The W3C standard XML Schema [1, 2, 3] does have temporal data types but very limited temporal reasoning and no multi-calendar reasoning. This paper reports on a project aiming at enriching a logic-based query and transformation language for the Web, called Xcerpt [9, 10], with temporal constructs and temporal reasoning. To this aim, a temporal model is needed that (1) is simple enough for being understood by programmers with limited knowledge in logic and/or temporal reasoning, (2) is structured in conceptual parts (e.g. layers) of various expressive power so as to provide programmers with temporal notions and constructs corresponding to their application's needs, and (3) provides temporal notions and constructs seamlessly integrated in the host language.

Thus, the focus of the project is not to specify yet another temporal logic and/or calculus. Instead, it is to select from existing temporal logics and/or calculi a coherent set of concepts and primitives fulfilling the above-mentioned objectives.

Recently, approaches to temporal reasoning on the Semantic Web have been developed from a functionality centered perspective, e.g. ontology reasoning such as with DAML-Time[1]. Temporal knowledge representation and temporal reasoning have been investigated for a long time, e.g. in (temporal) databases [14, 13, 7, 8] and in Artificial Intelligence [17, 5, 16, 18, 15, 19]. Arguably, temporal reasoning on the Web requires forms of reasoning inherently different from traditional reasoning forms used in databases and Artificial Intelligence because the Web is heterogeneous: The Web can be seen as a (very large) distributed database consisting of interoperable autonomous

[1] http://www.cs.rochester.edu/∼ferguson/daml/

sites. Note that temporal knowledge on the Web is usually represented using different time granularities (e.g. hour, day, week) and different calendars (e.g. the Gregorian, Islamic, Iranian, and Julian calendars) as well as the various ways of numbering years used in East Asia with the Gregorian calendar.
This article is structured as follows. Section 1 is this introduction. Section 2 motivates the need of temporal constructs in Web query and transformation languages in terms of emerging applications. Section 3 proposes a time and calendar model for such languages. Finally, Section 4 gives perspectives of the proposed temporal constructs.

2 Motivation: Three Emerging Applications

The so-called *'Web adaptation'* is receiving increasing attention in Web circles [4]. Adaptation basically means delivering and/or rendering data in a context dependent manner. One distinguishes between 'semantical adaptation' adapting the data themselves and 'representational adaptation' adapting how data are rendered. For example a web-based e-commerce catalog might adapt offers to former purchases of a user and/or render Web pages using font sizes specified by the user and/or the rendering device (desktop, cellular phone, or handheld computer) used. Temporal information on the Web mostly refers to semantical adaptation, and in modeling 'contexts', temporal data often play an essential role. In the following, three scenarii of advanced Web applications that make use of temporal data are described: appointment scheduling, event planning, and budgeting. See [20] for more details.

Web-based Appointment Scheduling. Appointment scheduling is a problem faced daily by many people at work. In general, scheduling an appointment among a number of people requires to consider several temporal constraints (such as 'John does not accept a meeting on a Monday before 9:30 am') as well as the already scheduled appointments registered in the calendars of the persons concerned. An appointment scheduler tries to find a match (or a best match) between the given constraints. Advanced systems might determine priorities on appointments. Appointment scheduling often requires advanced temporal reasoning capabilites (for processing planning requests such as 'Mary plays tennis for one hour in the morning every second week during working time' or for scheduling activities spread over two consecutive days).
It is desirable that a web-based appointment scheduler provides some form of calendar-based semantical adaptation for appointments expressed in terms of the calendar in use in the country (or countries) where one works and lives in. These calendars present more differences than one might think at first. For example Christmas Day means January 7 in Russia and some (but not all) Slavic countries but December 25 in Greece, while in both Russia and Greece Eastern 2003 means April 27. Also, years are numbered differently in Japan and China than in western countries (and not in the same way in Japan, continental China, and Taiwan). Many more such examples could be given. Thus, for being usable world-wide, an appointment scheduler must refer to various calendar systems. Moreover, it should provide with adaptation to the calendar system preferred to each user making it possible to communicate with each other without having to take care of the different calendars in use. In fact, multi-calendar temporal reasoning is an essential, still rather neglected aspect of the so-called 'internationalization'[2] the developers of the Web strive for. Beside various calendars and for obvious reasons, time zones and the various 'daylight saving times' also have to be supported by modern, web-based appointment schedulers.
The following scenario illustrates the temporal notions and temporal reasoning an appointment scheduler calls for.

Example 2.1 *Three business men plan a trip to London. This trip should be arranged within the next three months. They estimate the time to spend on the trip to two and a half days and specify a time interval during which the trip should take place. After having defined these temporal constraints on the agenda of each business man, the appointment scheduler queries the electronic*

[2]http://www.w3.org/International

calendars of the participants for their personal time constraints within the considered time interval. The appointment scheduler reasons over the temporal constraints and returns the (consistent!) answers to the problem, if any. In doing this, various calendars, time zones, as well as general temporal notions such as 'bank holiday' and 'working day' might be involved.

An appointment as considered in Example 2.1 is a convex time interval represented by its ending points, and it has a duration. Other examples similar to 2.1 demonstrate the need for non-convex time intervals. For example the answer to the question 'when are Anna, Bob, and Cindy all staying in New York' might be a non-convex interval. Thus, basic temporal notions needed by an appointment scheduler are *time points*, *convex and non-convex time intervals*, and (time interval) *durations*. Furthermore, an appointment scheduler might refer to conjunctions and disjunctions of temporal constraints expressed in terms of the above-mentioned basic temporal notions. Each of these basic temporal notions can in turn be expressed with different temporal *granularities*, i.e. time point and duration units such as hour, day, week, month, trimester, semester. Note that some of these notions might have different interpretations depending on the used calendar system. For example months are differently defined in the Islamic, Gregorian, Iranian, and Coptic calendars, for citing a few still widely used in some countries and/or communities.

Web-based Event Planning. The events considered in this section are social events (like concerts or theater performance), or professional events (like venues of conventions and conferences). An event planning system is a software aiming at assisting people planning and/or managing a large number of events subject to possibly complex constraints. For example planning the concerts of a large city might have to ensure a regular distribution over the time if those concerts aimed at similar audiences. Event planning is concerned with inter-related time-dependent events. The events to consider might be already finalized, i.e. certain, or instead potential, i.e. subject to cancellation. In contrast to appointment scheduling, event planning is in general a continuous, or at least a long lasting process: While scheduling an appointment can be seen and realized as a punctual task, scheduling events often requires long lasting, intertwined tasks. Thus, the temporal reasoning system subjacent to an event planning system must be able to manage an ongoing planning process. The following scenario illustrates the temporal notions and temporal reasoning aspects an event planning system in general needs.

Example 2.2 *Mary is responsible for planning, managing, and surveying the cultural events of a large city. For some event, the following might have to be planned: Renting a service (e.g. a catering service) could involve calling a catering service in due time, scheduling appointments with a responsible person, conclude a contract, provide with access to premises and facilities at some point of time, oversee the service provided in due time, etc. Indeed, consistency with planned events and their sub-tasks must be checked. For example, two subcontractors cannot necessarily use the same resource, e.g. rooms.*

Thus, an event planning system might recall to workflow management systems [6]. An essential difference is that, in contrast to a standard workflow management system, an event planning system will have to support 'common-sense' or 'real life' temporal notions like hour, day, week or month. An event planning system perfectly fitting the needs of Mary from Example 2.2 will also have to support various calendars and time zones (for many artists come from abroad and the city is likely to have different cultural communities the celebrations of which have to be taken into account in planning cultural events).
In general, events will have to be modeled using non-convex time intervals collected from the convex components corresponding to the several sub-tasks of the events. Furthermore, one probably will have to distinguish between 'fully specified events' having for example a specific time and date occurrence with sub-tasks having fully specified temporal constraints and 'incompletely specified events' referring to sub-tasks which are already committed but not yet fully scheduled. Temporal notions needed in formalizing such events are among others *convex* and *non-convex time intervals*, *beginning time* and *ending time*, *earliest time* and *latest time*, and *maximal duration* and *minimal durations*. Most likely, an event planning system will have to support partially ordered activities

FUZZY	COMMON-S.	Calendar Layer
		Granularity Layer
		Quantitative Layer
		Qualitative Layer

Table 1: The 'Temporal Primitive Tower'.

(or tasks, or sub-tasks), incomplete information, and to verify the consistency of temporal constraints between (inter-related) events.

Web-based Budgeting. A budgeting system might be seen as a temporal planning system tuned to financial control. Budgeting systems take into account both, when and in which order budgeting tasks occur. They also take into account a task's evolution for this is often critical for a correct determination of future budgets and their related budgeting plans. Let us consider a scenario.

Example 2.3 *A budgeting system for public schools guides the school's financial analyst through the process of creating a budget that can be easily managed, consulted, balanced, and compared with previous budgets of the school, with the budgeting plan, and with the current year's budget. The budgeting system computes (and stores) the budgets of all budget sections together with some constraints. It also computes monthly reports including absolute and relative deviations from the running year's budget and extrapolation for the future based on previous year's balance-sheets. Further reports give the budget for each term of references separately as well as the currently available resources.*

Thus, a budgeting system refers to several temporal notions and constraints. Budgeting refers to different time domains because it uses both histories (past data) and extrapolations (future data).

Importance of the Web for the Applications Considered Above. It is worth stressing that the Web provides an infrastructure making it possible for an appointment scheduler, event planner, and budgeting system to refer to the calendars of several persons at different places and possibly moving from place to place. Web applications like those mentioned above suggest that what one might call *'multi-calendar and multi-location temporal reasoning'* is likely to become much more important in the future than it has been in the past. The project reported on in this paper is focused on this emerging issue.

3 The 'Temporal Primitive Tower'

Adding temporal notions and temporal reasoning, in the following referred to as 'temporal primitives', to a programming or query language calls for a simple model describing these primitives at different levels of complexity. The 'Temporal Primitives Tower', short TemPTo (cf. Table 1) is such a model. TemPTo is a layered model, i.e. the temporal primitives it offers are grouped in layers of increasing expressive power and complexity, the lower layer being the less expressive and simpler. Each layer of TemPTo is accessible to a programmer depending on his/her needs.
The bottom layer of TemPTo, called *qualitative layer*, offers a core set of temporal primitives: time points, convex time intervals, and non-convex time intervals, and so-called qualitative operations on these basic time primitives. These qualitative operations are listed in Tables 2, 3 and 4.
The second layer of TemPTo, called *quantitative layer,* provides with (1) arithmetical operations on the basic time primitives (cf. Table 7) of the bottom layer and (2) temporal duration.

The third layer of TemPTo, called *granularity layer*, offers so-called time granularity notions such as hour, day, week.
The third layer of TemPTo might be the only one referred to in some programs, for some application are appropriately specified in terms of this layer's primitives. Other programs might refer to both, primitives of this layer and primitives of lower layers. Again, some more complex programs might combine modules referring either only to the qualitative or quantitative layers, or to the granularity layer. TemPTo might be seen as a programming language's *type system* providing the 'glue' between such modules, thus making it possible for the programmer not to have to explicitely specify conversions between primitives of the qualitative/quantitative layers and of the granularity layer.
The uppermost layer of TemPTo, called the *calendar layer*, provides different *calendar systems.* A calendar system might be defined as composed of a finite set of time granularities, origins of time of calendars, and specific properties (such as leap years).
In addition, a first vertical component, called the *common-sense column* will be added so as to complement TemPTo's layers with common-sense notions such as 'Easter Monday' or the meaning of 'Friday evening' in different cultures (in Western countries, 'Friday evening' denotes the eve of Friday, in some Islamic countries, the eve of Thursday). A second vertical component, called the *fuzzy column*, will be added in the future. It is intended to provide with fuzzy reasoning primitives so as to accommodate widespread imprecise temporal notions such as 'weekend' or 'quinze jours' (French for 'fifteen days' but meaning 'two weeks'!). These two components are 'vertical' because they provide access to all layers of the TemPTo tower.
TemPTo supports time points, convex time intervals, and non-convex time intervals expressed at different levels of abstraction corresponding to different time granularities (like minute, hour, day, week, etc.) referring to different calendar systems (like the Gregorian and the Islamic calendars). In TemPTo, a 'time point' has an associated 'precision', i.e. a time granularity. A TemPTo 'convex time interval' is defined in terms of two time points, called its ending points (having a precision). It has a 'duration' which in turn has a precision, i.e. a granularity. A TemPTo 'non-convex time interval' is a finite set of pairwise disjunct TemPTo convex time intervals.
Selected temporal operations on the above-mentioned TemPTo data types are part of TemPTo. These operations give rise to compare and compute temporal data. It is a working hypothesis of the project that not all possible temporal operations are desirable in the project's context, i.e. that of Web query and transformation languages to be used in implementing web-based systems such as those described in Section 2. The reason for this working hypothesis is that a query and transformation language, even more than a general purpose programming language, must remain simple enough for being understood by programmers with limited knowledge in logic and/or temporal reasoning. In the following, the different layers of TemPTo are described in more detail.

Definition 3.1 *A* **basic time domain** *is a pair $(\mathcal{T}, <_{\mathcal{T}})$ where $\mathcal{T}$ is an infinite set and $<_{\mathcal{T}}$ is a total order on $\mathcal{T}$ such that $\mathcal{T}$ is not bounded for $<_{\mathcal{T}}$.*
If $(\mathcal{T}, <_{\mathcal{T}})$ is a basic time domain, then the elements of $\mathcal{T}$ are called time points.

Examples of time domains are $(\mathbb{Z}, <_{\mathbb{Z}})$, a *discrete* time domain, $(\mathbb{Q}, <_{\mathbb{Q}})$, a *dense* time domain, and $(\mathbb{R}, <_{\mathbb{R}})$, a *dense and continuous* time domain.
In the following, $(\mathcal{T}, <_{\mathcal{T}})$ denotes a fixed time domain.

Qualitative Layer. The qualitative time language constructs are the time points, the convex time intervals, and the non-convex time intervals defined over $(\mathcal{T}, <_{\mathcal{T}})$. The qualitative time also offers basic (qualitative) temporal operations. The qualitative layer is needed, e.g. for appointment scheduling where different people try to find an appointment time for a meeting not overlapping with their already scheduled appointments.

Definition 3.2 *A* **closed convex time interval** *$I = [a, b]$ is a subset of $\mathcal{T}$ where $a \in \mathcal{T}$, $b \in \mathcal{T}$, $a <_{\mathcal{T}} b$ such that $p \in [a, b]$ iff $p \in \mathcal{T}$ and $a \leq p \leq b$.*

time point	EQUALS	time point
time interval	EQUALS	time interval
time point/interval	PRECEDES	time point/interval
time point/interval	DURING	time interval
time point/interval	STARTS	time interval
time point/interval	FINISHES	time interval
time interval	MEETS	time interval
time interval	OVERLAPS	time interval

Table 2: TemPTo Relations on time points / convex time intervals

COMPLEMENTS
EXTRACT
ZIP_ALL(REL)
ZIP_SOME(REL)

Table 3: TemPTo relations on non-convex time intervals

Right-closed, left-closed, and open convex time intervals (denoted, $]a, b]$, $[a, b[$, $]a, b[$, respectively) are defined analogously. Since time intervals are sets, the usual set-theoretic operations union, intersection, and difference can be applied to them. The qualitative layers offers these set-theoretic operations. Note that the set of all time intervals as defined in Definition 3.2 is not closed with respect to these set-theoretic operations.
The following definition is motivated by common applications that refer to non-convex time intervals. Non-convex time intervals result from (i) unions of convex time intervals (e.g. to describe semantically related events such as the activities for planning an event), (ii) recurring time intervals (e.g. a professor's consultation-hour), or (iii) gapped time intervals (e.g. the phone rings during a conference).

Definition 3.3 *A* **non-convex time interval** *J is a finite collection of pairwise disjunct convex time intervals called* **component***s of J.*
Two relations 'before' and 'after' are defined as follows on the components of a non-convex time interval J: (a_1, b_1) before (a_2, b_2) iff $b_1 <_{\mathcal{T}} a_2$ or $b_1 = a_2$, and (a_1, b_1) after (a_2, b_2) iff $b_2 <_{\mathcal{T}} a_1$ or $b_2 = a_1$, where (a, b) denotes a closed, right-closed, left-closed, or open component of J.

Note that a convex time interval can be seen as a non-convex time interval (i.e. a collection with one component).
Table 2, Table 3, and Table 4 give the relation and functions offered by the qualitative layer. The definitions of these relations and functions are as usual (cf. [17, 5, 18]), and supplemented with a core set of functions and relations on non-convex time intervals: An operation (*complements*) for determining the complement of a non-convex time interval according to its convex hull, the *extract* operation for extracting a particular set of components from a non-convex time interval, and pairwise binary comparison operations (*zip_all* and *zip_some*) on the components of non-convex time intervals. In this table, 'REL' denotes any of the relations given in Table 2.

Quantitative Layer. The second layer of TemPTo, the quantitative Layer, offers *origins of time,* time durations, and additional quantitative and/or metric operations on the basic TemPTo primitives. Using the primitives of this layers, one can express *metric constraints* between time points, convex, and non-convex time intervals, absolute numeric values, and more or less complex durations. Metric constraints are handled by systems of linear inequalities. Such linear inequalities can be used together with the qualitative operations. The quantitative layer is needed, e.g. for appointment scheduling when a person specifies the amount of time between two meetings for scheduling a train journey between the two meetings in time.

Function Symbol	Operands	Result
EMPTY	set of time points set of convex time intervals set of non-convex time intervals	Boolean
UNION INTERSECTION DIFFERENCE	set of time points set of convex time intervals set of non-convex time intervals	set of time points set of convex time intervals set of non-convex time intervals
BEGIN	convex time interval non-convex time interval	time point
END	convex time interval non-convex time interval	time point
FIRST	set of time points set of convex time intervals set of non-convex time intervals	time point convex time interval non-convex time interval
LAST	set of time points set of convex time intervals set of non-convex time intervals	time point convex time intervals non-convex time intervals
CONV	non-convex time interval	convex time interval

Table 4: Functions provided on Time Primitives.

Operand 1	arithm. Op.	Operand 2	Result
	$-$	duration	duration
duration	$+$	duration	duration
duration	$-$	duration	duration
duration	$*$	numeric	duration
duration	$/$	numeric	duration

Table 5: Arithmetic Operations provided on Durations ('+' and '-" are commutative).

Operand 1	Relation Symbol	Operand 2
duration	$=$	duration
duration	$<$	duration
duration	$>$	duration

Table 6: Relations provided on Durations.

Function Symbol	Operands	Result
ABSOLUTE	(duration)	duration
SHIFT	(time point,duration)	time point
SHIFT	(convex time interval,duration)	convex time interval
SHIFT	(non-convex time interval,duration)	non-convex time interval
DISTANCE	(time point1,time point2)	duration
LENGTH	(convex time interval)	duration
LENGTH	(non-convex time interval)	duration
CARD	(non-convex time interval)	numeric

Table 7: Arithmetic Operations provided on Basic Time Primitives.

Definition 3.4 *A* **duration** *D is a (positive or negative) amount of time.*

An amount of time can be expressed using a time unit (i.e. time granularity) like minute or day. The quantitative Layer offers standard arithmetic operations and comparison relations on durations, cf. Table 5 and Table 6. Furthermore, operations on variables ranging over basic time primitives are also provided by the quantitative layer, cf. Table 7. These operations provide reasoning across TemPTo's qualitative and quantitative layers.

Granularity Layer. The third layer of TemPTo is the granularity Layer. It introduces *hierarchies of time domains* based on the time partition conventions of the various calendar and clock systems by specifying the semantics of *time granularities* (e.g. minute, hour, day, month). An element of a time granularity , a so-called *granule,* is defined as a set of time points of a given basic time domain. Granules might be seen as duration units (e.g. 31^{st} March 2003 as the last day of working on this paper) or as time point precisions (e.g. 31^{st} March 2003 as the submission deadline for this paper). Granularities are subsequently used in defining calendar and clock systems.

Definition 3.5 *Assume $G = \{g_i \mid i \in \mathbb{N} \setminus \{0\}\}$ is a set of granules. A* **time granularity** *is a (possibly partial) function Γ from G in the power set $\mathcal{P}(\mathcal{T})$ such that for all $i, j \in \mathbb{N} \setminus \{0\}$ with $g_i < g_j$ the following holds:*

1. *If $\Gamma(g_i) \neq \emptyset$ and $\Gamma(g_j) \neq \emptyset$, then for all $\mathcal{T}_i$ in $\Gamma(g_i)$ and for all $\mathcal{T}_j$ in $\Gamma(g_j)$ $\mathcal{T}_i <_{\mathcal{T}} \mathcal{T}_j$.*
2. *If $\Gamma(g_i) = \emptyset$, then $\Gamma(g_j) = \emptyset$.*

The first condition of Definition 3.5 states that granules in a granularity do not overlap, and their index corresponds to their order in their associated time domain. The second condition of Definition 3.5 states that the first granule (if any) in a granularity must start with index 1. Examples of granularities are 'day' and 'month'. Corresponding granules are '13th of May 2003' and 'May'. Most temporal reasoning systems rely on the assumption that temporal constraints referring to different granularities can be translated into constraints referring to a single (usually a 'smallest') granularity, thus giving rise to reasoning with a single granularity. We claim (and postulate in the project reported about here) that temporal reasoning for advanced Web applications requires multiple granularity temporal reasoning. The reason for this claim and postulate is the need to efficiently accommodate heterogeneous calendar (and clock) systems. Choosing an arbitrary reference granularity, for appealing it might seem to the 'temporal technician', would (1) contradict the inherent 'calendar heterogeneity' of the Web and (2) fix a 'smallest' granularity that most likely would at some point of time in the future turn out to be highly inappropriate. Consequently, TemPTo must support operations on time granularities. These operations are offered at TemPTo's calendar layer.

Calendar Layer. The uppermost layer of TemPTo is the calendar layer. It introduces different calendar systems (e.g. Gregorian and Islamic calendars). The calendar layer is needed, e.g. for appointment scheduling between people in various cultural areas. Formally, a *calendar system* is defined as follows.

Definition 3.6 *A* **calendar system** *is a tuple $(\{0_1, ..., 0_n\}, \mathcal{G})$ where $0_1, ..., 0_n \in \mathcal{T}$ are called the calendar's* **origins of time** *and $\mathcal{G}$ is a finite set $\{\Gamma_1, ..., \Gamma_n\}$ of granularities with Γ_1 is the 'finest' and and Γ_n is the 'coarsest' granularity of the calendar system.*

Most calendars have only one origin of time, others, like the calendar currently used in Japan, have several. E.g., December 24, 1926 is referred to in Japan[3] as December 24, Taishō 15 while December 25, 1926 is December 25, Shōwa 1 and January 7, 1989 is January 7, Shōwa 64 while January 8, 1989 is January 8, Heisei 1. The rationale is that, following the old Chinese practice, years are numbered in the Japanese calendar after the emperors' reigns.
'Finer' and 'coarser' granularities are defined in a common manner which is not recalled here.

[3]When the Japanese language is used.

Definition 3.7 *Assume $\mathcal{G}$ is a finite set of granularities $\{\Gamma_1, ..., \Gamma_n\}$. A granularity Γ_i is finer than a granularity Γ_j, denoted $\Gamma_i \prec \Gamma_j$, if every granule in Γ_i is a subset of some granule in Γ_j. If $\Gamma_i \prec \Gamma_j$, then Γ_j is said to be* **coarser than** *Γ_i, and Γ_i is said to be* **finer than** *Γ_j.*

For example with common-sense definitions, day $\prec$ day, day $\prec$ week, but week $\prec$ month does not hold. Note that $\prec$ is a *partial order* on a set of granularities.
In TemPTo, a granule g of some granularity Γ can also be considered as a time point. Indeed, in a calendar system $(\{0_1, ..., 0_n\}, \mathcal{G})$, if $\Gamma_i \in \mathcal{G}$, $\Gamma_j \in \mathcal{G}$ and $\Gamma_i \prec \Gamma_j$, then each $g_j \in \Gamma_j$ is a time interval of 'kind' Γ_i. Note that if $\mathcal{G}$ is a set of granularities, then $(\mathcal{G}, \prec)$ is a *lattice*.

Definition 3.8 *Assume $\mathcal{G} = \{\Gamma_1, ..., \Gamma_n\}$ is a finite set of granularities.
If $\Gamma_i \prec \Gamma_j$ where Γ_i and Γ_j are immediate neighbors, i.e. there is no $\Gamma_k \in \mathcal{T}$ such that $\Gamma_i \prec \Gamma_k \prec \Gamma_j$, then each $g_j \in \Gamma_j$ is a (possibly non-convex) time interval in the time domain of Γ_i.*

Definition 3.8 explicitly introduces a context between immediate neighbors in a calendar system due to temporal reasoning with different granularities. For example the granule '13th of May 2003' of the granularity 'day' corresponds to the interval of 24 hours of the granularity 'hour'. In most calendar systems these time intervals are convex. However, for some applications such as business and management non-convex granularities are desirable (e.g. working month). Definition 3.8 allows for viewing granules from two points of view: First, g_j in the coarser granularity Γ_j can be a (possibly non-convex) time interval anchored on the time domain related to the finer granularity Γ_i. Second, g_j in the coarser granularity Γ_j can be a (time interval) duration w.r.t. the time domain of the finer granularity Γ_i. Corresponding to Definition 3.8, type casting operations for temporal reasoning with different granularities are provided with TemPTo which enables temporal reasoning across TemPTo's qualitative and/or quantitative layers and granularity/calendar layers. Note that practical calendar systems in addition have particular properties such as rules specifying leap years. The calendar layer offers various calendar systems and conversion between their concepts.

4 Perspectives

The TemPTo model has been designed as a temporal type system of temporal types for the Web query and transformation language Xcerpt. This type system as well as its integration to Xcerpt is a subject of ongoing research. Xcerpt [12, 10, 9, 11] is a logic-based language currently developed and tested on web-based systems at the University of Munich. Because of its 'logic bias' Xcerpt appears to be a very convenient 'host' for temporal data types and temporal reasoning. Further investigations will show whether the temporal model TemPTo sketched in this article is a convenient temporal model for implementing the kind of web-based applications described in Section 2. This model goes far beyond the temporal notions proposed in the W3C recommendation 'XML Schema' [1, 2, 3] for Web applications.

References

[1] *W3C* `http://www.w3.org/TR/xmlschema-0/`*: XML Schema Part 0: Primer*. 2001.

[2] *W3C* `http://www.w3.org/TR/xmlschema-1/`*: XML Schema Part 1: Structures*. 2001.

[3] *W3C* `http://www.w3.org/TR/xmlschema-2/`*: XML Schema Part 2: Datatypes*. 2001.

[4] The Adaptive Web. *Special Issues of Communications of the ACM*, 45(5):30–75, 2002.

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

[6] Gustavo Alonso and C. Mohan. Workflow Management Systems: The next Generation of distributed Processing Tools. In *Sushil Jajodia and Larry Kerschberg (Eds.): Advanced Transaction Models and Architectures Ch. 1, Kluwer Academic Publishers*, pages 35–62, 1997.

[7] Marianne Baudinet, Jan Chomicki, and Pierre Wolper. Temporal Deductive Databases. In *Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings*, pages 294–320, 1993.

[8] Claudio Bettini, Sushil Jajodia, and Sean X. Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning.* Springer-Verlag, Berlin, 2000.

[9] François Bry and Sebastian Schaffert. A Gentle Introduction into Xcerpt, a Rule-based Query and Transformation Language for XML. In *International Workshop on Rule Markup Languages for Business Rules on the Semantic Web (invited article), http://www.soi.city.ac.uk/~msch/conf/ruleml, Italy, June*, 2002.

[10] François Bry and Sebastian Schaffert. The XML Query Language Xcerpt: Design Principles, Examples, and Semantics. In *Akmal B. Chaudhri, et al. (Eds.), Lecture Notes in Computer Science (LNCS) on Web, Web-Services, and Database Systems*, pages 295–310, 2002.

[11] François Bry and Sebastian Schaffert. Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification. In *Peter J. Stuckey (Ed.): Logic Programming, 18th International Conference on Logic Programming (ICLP), Denmark, July, Lecture Notes in Computer Science 2401, Springer-Verlag*, 2002.

[12] François Bry, Sebastian Schaffert, and Sacha Berger. Pattern Queries for XML and Semistructured Data (revised version). In 17th *Workshop on Logic Programming, Germany, December, http://www.pms.informatik.uni-muenchen.de/publikationen/#ResearchPapers #PMS-FB-2003-1*, 2003.

[13] Jan Chomicki. Temporal Query Languages: A Survey. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *Temporal Logic: International Conference on Temporal Logics (ICTL'94)*, volume 827, pages 506–534. Springer-Verlag, 1994.

[14] Jan Chomicki and David Toman. Temporal Logic in Information Systems. In *Jan Chomicki, Gunter Saake (Eds.): Logics for Databases and Information Systems, Dagstuhl Seminar: Role of Logics in Information Systems, 1995, Kluwer*, pages 31–70, 1998.

[15] Henry A. Kautz and Peter B. Ladkin. Integrating Metric and Qualitative Temporal Reasoning. In *Proceedings of the 9th National Conference on Artificial Intelligence, July, AAAI Press / The MIT Press*, 1991.

[16] Peter B. Ladkin. Time Representation: A Taxonomy of Intervals. In *Proceedings of the Conference on Artificial Intelligence (AAAI), Morgan Kaufmann*, pages 360–366, 1986.

[17] Drew V. McDermott. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science*, 6:101–155, 1982.

[18] Itay Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. In *Proceedings of the 9th National Conference on Artificial Intelligence, July, AAAI Press / The MIT Press*, 1991.

[19] Hans Jürgen Ohlbach and Dov Gabbay. Calendar Logic. *Journal of Applied Non-classical Logics*, 8(4):291–324, 1998.

[20] Stephanie Spranger. *Representation of Temporal Knowledge for Web-Based Applications.* Diploma Thesis, Institute for Computer Science, University of Munich, http://www.pms.informatik.uni-muenchen.de/publikationen/diplomarbeiten/Stephanie.Spranger/da.pdf, 2002.