Temporal Data Modeling and Reasoning for Information Systems

Stephanie Spranger and François Bry

Abstract

Temporal knowledge representation and reasoning is a major research field in Artificial Intelligence, in Database Systems, and in Web and Semantic Web research. The ability to model and process time and calendar data is essential for many applications like appointment scheduling, planning, Web services, temporal and active database systems, adaptive Web applications, and mobile computing applications. This article aims at three complementary goals. First, to provide with a general background in temporal data modeling and reasoning approaches. Second, to serve as an orientation guide for further specific reading. Third, to point to new application fields and research perspectives on temporal knowledge representation and reasoning in the Web and Semantic Web.

1 Introduction

Temporal knowledge representation and reasoning is a major research field in Artificial Intelligence, in Database Systems, and in Web and Semantic Web research. The ability to model and process time and calendar data is essential for many applications like appointment scheduling, planning, Web services, temporal and active database systems, adaptive Web applications, and mobile computing applications.

An exhaustive overview of all the approaches, results, and applications on temporal and calendric knowledge representation and reasoning would probably require an entire book by itself. This article has three less ambitious goals. First, it aims at providing with a general background in temporal data modeling and reasoning approaches. Second, it aims at serving as an orientation guide for further specific reading. Third, it points to new application fields and research perspectives on temporal knowledge representation and reasoning in the Web and Semantic Web.

This article emphasizes the practical impact of time and calendar modeling and reasoning for various applications, addressing important application scenarios.

This article is structured as follows. First, this article addresses important application fields on the Web and the Semantic Web for temporal and calendric data modeling and reasoning. Second, it surveys well-known and (at least to some extend) established time models. Third, this article reviews some application fields of such time models, in particular, constraint-based temporal and calendric reasoning.

2 Motivating Applications: Temporal Reasoning on the Web

So-called *"Web adaptation"* is receiving increasing attention in Web circles. Adaptation basically means delivering and/or rendering data in a context-dependent manner. One distin-

guishes 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 fact, time is an essential aspect of semantical adaptation on the Web: for example, phone conferences are nowadays supported by some Web applications. Scheduling an appointment for a further phone conference involving persons from France, Tunisia, and Japan would be much easier to handle, if the underlying Web system provides with semantical adaption to time and calendars: everyone's calendar data should be better expressed in the calendar (e.g. Gregorian months in France and Japan, but Islamic months in Tunisia) and time zone (e.g. Central European Time in France but Japan Standard Time in Japan) the person is used to. In the following, three scenarios of advanced Web applications that make use of calendric and temporal data are described: appointment scheduling, event planning, and budgeting.

2.1 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 considering several time 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 and calendric reasoning capabilites for processing planning requests such as "Mary plays tennis for one hour in the morning every second week during working time", for scheduling activities spread over two consecutive days, or for scheduling activities of different calendric types like what is the latest possible working day in February for a Valentine Day present delivered in due time (i.e. before 14^{th} February).

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 7^{th} January in Russia and some (but not all) Slavic countries but 25^{th} December in other European Countries. 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 the developers of the Web strive for (http://www.w3.org/International). 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 concerns 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 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 calendric expressions such as "legal 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. 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 calendric types, i.e. units such as hour, day, week, month, trimester, and semester. Note that some of these notions might have different interpretations depending on the used calendar. For example, months are differently defined in the Islamic, Gregorian, Iranian, and Coptic calendars, for citing a few still widely used.

2.2 Web-based Event Planning

The events considered in this section are social events (like concerts or theater performances), 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 aim at similar audiences. Event planning is concerned with inter-related time-dependent events. The events to be considered 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 that 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 resources, for example rooms.*

Thus, an event planning system might recall to work-flow management systems. An essential difference is that, in contrast to a standard work-flow management system, an event planning system will have to support common-sense or real life calendar expressions 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 (e.g. for many artists that come from abroad). Furthermore, the city is likely to have different cultural communities the celebrations of which have to be taken into account in planning cultural events.
One probably will have to distinguish between "fully specified events" and "incompletely specified events". The former refer to specific time and date occurrence and some event's sub-tasks having fully specified temporal constraints. The latter refers to sub-tasks which are already committed but not yet fully scheduled. Most likely, an event planning system will have to support partially ordered activities (or tasks, or sub-tasks), incomplete information, and it will have to verify the consistency of temporal constraints between (inter-related) events.

2.3 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).

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 temporal data modeling and reasoning"* is likely to become much more important in the future than it has been in the past.

3 Time Models

Knowledge is the *symbolic representation* of aspects of some *discourse universe* such as time and calendars. An example for temporal knowledge represented in natural language is "John plays tennis every Saturday between 10 a.m. and 11 a.m.". Modeling formalisms and reasoning mechanisms to manipulate the knowledge of some discourse universe, to create solutions, and to formulate new problems have been developed. Such formalisms have been proposed as tools to define knowledge in terms of symbols that may be manipulated by computer programs.

In principle, symbolic representation of temporal knowledge allows for describing changes of the reality modeled in some system, for example a database. A computer program that allows for manipulating such knowledge usually provides with inference strategies to reason about time-dependent objects such as schedules, plans, actions, and changes. These and further considerations have given birth to a large field of research that can be summarized as the development of *time models*. Essentially two different approaches to represent time exist: *implicit time models* and *explicit time models*. The computation mechanisms to manipulate temporal knowledge that is defined in such time models are often called *temporal reasoning methods*.
Usually, knowledge about time (and calendars) is obtained by extending a language with means to represent temporal data and to define a temporal reasoning system, i.e. a technique for reasoning about assertions formulated in the extended language. Such assertions usually depend on some intended application (e.g. scheduling, planning, changes and actions), determining which reasoning technique(s) might be applied. A temporal reasoner is usually part of a more general reasoning system.
Following approaches to temporal data modeling and reasoning are surveyed in this section. First, implicit time models which focus on time-dependent entities indicating that a change in time has occurred. Second, explicit time models which consider time itself independent of anything that could happen in it. Third, explicit time models including approaches to model different calendar units in terms of so-called *time granularities*.

3.1 Implicit Time Models

In principle, implicit time models consider time dependent entities like *events* and *actions* indicating that a *change* has occurred in the modeled world, for example a database of facts. With such systems, time is not made explicit. Instead, time is implicitly specified in terms of the changes that have occurred in the past or that might occur in the future.
Implicit time models are particularly used to model events and actions that result in changes in the context of some application. Implicit time models are used to specify the effects of actions and to acknowledge and to react on changes in such systems.
These implicit time formalisms provide a simplistic notion of time, however useful for simple problem-solving tasks: a state describes the world at an instantaneous time, i.e. at a time point. Actions are modeled as functions over states. Well-known implicit time models are the Situation Calculus [MH87] and the Event Calculus [KS86, Kow92]. In the field of program verification[1], Dynamic Logic [Pra76, Har79, Har84, Moo85] is a well-known implicit time formalism which is also applied to problems in temporal knowledge representation and reasoning. Implicit time models are widespreadly applied to active database systems, and recently, to model and process changes and actions in active Web and Semantic Web systems.

3.1.1 Situation Calculus

The Situation Calculus has been introduced by McCarthy and Hayes in 1969 [MH87]. This calculus has long been a foundation for temporal representation and reasoning in Artificial Intelligence. The Situation Calculus has been modified and/or extended with several different means [SG88, Web90, LS95, Ram00, McC02, MPP02]. In addition to research in knowledge

[1]Note that research on program verification is not considered in this survey. The reason is that it is inherently different.

representation and reasoning on actions and changes [GLR91, Lif91], the Situation Calculus is particularly used for practical work in planning [FN71, FHN72, Ló6, GLL97], active database systems [Rei92, Rei95, BAF98], and in agent programming and robotics [BCF+98, LLR99, BRST00, MSZ01, ABH+02].
In the Situation Calculus, temporal knowledge is represented as a sequence of so-called *situations,* each being a description of the world at a time point. The Situation Calculus models a *changing world* in terms of a totally ordered and discrete sequence of situations isomorphic to the integers. The changing world is formalized by a many-sorted predicate calculus with some reserved predicate and function symbols. The sorts of this calculus are *situations, actions,* and *objects. Fluents* are specific relations and functions defined over the sort of situations. Fluents are used to describe the world in each situation. Action performances, action preconditions, and histories are non-decomposable functions from one situation to another.
The Situation Calculus is feasible only in domains where merely one event can occur at a time; in particular, no concept of an event taking time is supported. The Situation Calculus does not provide a notion of persistence: an event that is true at one situation needs to be explicitly verified at any succeeding situation.
Various axiomatizations of the Situation Calculus in some first-order language [Bak91, PR93, Rei93, LPR98, Rei01] as well as some formalizations of the Situation Calculus in terms of logic programs [KS94, BDD95] are proposed.

3.1.2 Event Calculus

The Event Calculus has been introduced by Kowalski and Sergot in 1986 [KS86, Kow92]. Various modifications and/or extensions of this calculus exist [Sha90, CMP93, CCM95, Sha95, SK95]. The most notable variation is the so-called Simplified Event Calculus [Kow92] that has addressed some of the problems appearing with the original calculus. Those problems are discussed in [Mis91, DMB92]. This simplified calculus has been applied to problems in planning [DMB92, YS02] and in active database systems [Kow92, FWP97].
The Event Calculus is a formalism to reason about *events.* It is based on a many-sorted first-order predicate calculus or the Horn clause subset of such a calculus. In most formalizations of the Event Calculus, a linear, point-based time line is assumed. Events initiate and terminate time intervals over which *fluents* hold. Fluents are properties that can have different values which are manipulated by the occurrences of events at different time points.
Various axiomatizations of the Event Calculus and of the Simplified Event Calculus are proposed [Esh88, Sha89, Sha90, Mis91, Kow92, SK95, MS99].

3.1.3 Dynamic Logic

Dynamic Logic is based on classical logics, in particular, propositional and predicate logics and modal logics which are combined with an algebra of regular events. Dynamic Logic enables verification of imperative programs and program specifications. It has been proposed by Pratt [Pra76] (with an emphasis on the modal nature of program interactions), Harel [Har79, Har84], and Moore [Moo85].
The basic idea of Dynamic Logic is to model *programs* by modal operations. Such programs change the values of variables causing changes of the truth values of some formula. Among the numerous formalisms for (formal) program reasoning, Dynamic Logic enjoys the singular advantage being strongly related to classical logics. Therefore, Dynamic Logics gains from

the advantages of classical logics compared to (often more complex) logics frequently applied to reason about programs.
Although Dynamic Logic has been initially introduced as a formalism for program verification, it has been turned out to be a formalism for reasoning about some actions of (natural or artifical) systems. In Artificial Intelligence, Dynamic Logic has been adapted for tasks such as the description of actions in some common-sense world and for specifying particular reasoning systems [SWM95, SGdMM96, Mey99].

3.2 Explicit Time Models

Explicit time models consider *time* itself, independently of any event or action that might occur. The flow of time is explicitly represented, defining a *time model* that specifies one (or more) time lines of time primitives, in particular, *points* and/or *intervals*. Some time models specify more than one time line which are related either in parallel or vertically. The former leads to *hierarchical time models*, in particular used to define calendar units like "day", "week", or "working year" and cultural calendars like the Gregorian and Hebrew calendars or professional calendars, for example of some university. The latter leads to *multi-dimensional time models*, in particular used to define temporal histories, for example of transactions of objects and/or validities of some properties of those objects stored in a system like a database.
Explicit time models are either point-based [Bru72, KG77, McD82, Sho87] or interval-based [Ham72, Dow79, NS80, All83, HA89]. Some proposals have been made to combine point-based and interval-based time models [Vil82, Boc90, Gal90, VS96]. Those models combining points and intervals, however, yield in rather artifical time models, raising several problems [Spr06]. Furthermore, time models for generalized intervals (i.e. intervals which are not necessarily convex) have been proposed [LMF86, Lad87, DC91, NS92, Lig98].
Explicit time models have been widely applied in temporal reasoning [Vil82, All83, vBC90, VKvB90, KL91, DMP91, Kou92, Mei96], temporal database systems [ÖS95, Sno95, Gan99], planning and scheduling [RS90, All91, BRS05], and ontology design for Web and Semantic Web applications and Web services [DAR02, PH04].

3.2.1 Point-based Models

Points are modeled as duration-less portions of time with identical beginning and ending times. Point-based time models have been influenced by physics [New36] where it is common to model time as an unbounded, ordered continuum of (partially) ordered points isomorphic to the real numbers. In Artifical Intelligence, research in this tradition can be found in [Bru72, KG77, McD82, Sho87].
A point-based time model is defined on a structure $(P, <_P)$, where P denotes a set of points and $<_p$ an ordering of P.
A point-based model defines at least a partial (i.e. irreflexive, antisymmetric, and transitive) order on points. Let $p, q, r, s, t \in P$ points.
An **ordering** over $(P, <_P)$ can be defined as follows:

(irreflexivity)	$\neg(p <_P p)$
(antisymmetry)	$((p <_P q) \wedge (q <_P p)) \Rightarrow (p = q)$
(transitivity)	$((p <_P q) \wedge (q <_P r)) \Rightarrow (p <_P r)$

Additionally, it may have (some of) the following properties:

- **left-linearity or linearity:**

$$\begin{array}{ll} \textit{(left-linearity)} & ((q <_P p) \wedge (r <_P p)) \Rightarrow ((q <_P r) \vee (q = r) \vee (r <_P q)) \\ \textit{(linearity)} & (p <_P q) \vee (p = q) \vee (q <_P p) \end{array}$$

- **boundness or unboundness:**

$$\begin{array}{ll} \textit{(right-boundness)} & \forall p \exists q.(q <_P p) \\ \textit{(left-boundness)} & \forall p \exists q.(p <_P q) \end{array}$$

- **discreteness or density:**

$$\begin{array}{ll} \textit{(discreteness)} & \forall p, q.(p <_P q) \Rightarrow \exists r.((p <_P r) \wedge \neg \exists s(p <_P s <_P r)) \\ & \forall p, q.(p <_P q) \Rightarrow \exists r.((r <_P q) \wedge \neg \exists s(r <_P s <_P q)) \\ \textit{(density)} & \forall p, q.(p <_P q) \Rightarrow \exists r.(p <_P r <_P q) \end{array}$$

The properties introduced above are sufficient to achieve a certain level of completeness. Two theories are known to be syntactically complete [vB91]: the unbounded dense linear theory and the unbounded discrete linear theory.
Dense and continuous point-based models provide a means to model continuous changes such that any extend of time can always be partitioned into subintervals; necessary, e.g. to model planning problems where tasks are frequently decomposed into subtasks. In dense and continuous models it is not possible to refer to some next (resp. previous) time. This is however possible in discrete models. Any finite strict partial order (i.e. if p and q are points, then $p < q$ or $q < p$) is automatically discrete. Including a (metric) *duration*, i.e. specific amounts of time with a given length but without begin and end points into a point-based time model, a distinction between continuous and discrete time becomes important: if the time line is continuous, a point has no duration, but in discrete time, it always has some duration. Point-based models provide with a simplified concept useful in simple problem solving tasks such as maintaining different states of a system like a database. Points describe the modeled world at an instantaneous time where events are represented by dates. Actions are modeled in such systems as functions mapping between points. For example, if dates are represented by integers, then the temporal order between two dates could be simply computed using numerical applications. Point-based time models are particularly applied in temporal information systems and temporal databases [ÖS95, Sno95, Gan99].

3.2.2 Intervals-Based Models

In several applications, events simply cannot be assigned to a precise date, e.g. to model the fact that two events cannot happen at the same time. Recall that points cannot be decomposed. Although some events appear to be instantaneous, e.g. one might argue that "step inside the house" is instantaneous, it also appears that such events might be either decomposed or have a duration when varying the "precision" (e.g. from minute to second) of this event. Thus, point-based time models are not expressive enough for many applications.
Starting from a point-based time model, intervals might be implemented into this approach. In a time model of totally ordered points, an interval can be represented by an ordered pair of points with the first point less than the second. To ensure that such intervals can meet

(e.g. "the light was on" and the "the light is off"), i.e. only having one endpoint in common, the endpoints of the intervals must be *less*, i.e. for an interval i with endpoints i^- and i^+ $i^- < i^+$. This can be achieved when the intervals are closed in their starting points and open on their ending points (or vice versa). However, this requirement points out that a time model based on points does not correspond with our intuitive notion of time. The modeling of temporal knowledge is significantly complicated, because both, points and intervals have to be considered. Furthermore, having e.g. an event e holding sometime during an interval i which in turn is during an interval i', then e holds during i', as well. Thus, relations between intervals and events may be "carried forward" such that reasoning can be kept local. It is not clear to maintain those properties when considering intervals modeled by endpoints in a point-based time model.
To overcome the problems of point-based time models, purely *intervals-based* time models have been introduced [Ham72, Dow79, NS80, All83, HA89]. An interval-based time modeled well-known and widely applied in Artificial Intelligence, Knowledge Representation, and Information Systems is Allen's interval calculus [All83], exclusively based on intervals and the 13 basic relations between pairs of them. The 13 basic interval relations are illustrated in Figure 1.
Allen [All83] initially takes a structure $(I, \mathcal{R})$ where I denotes a set of intervals and and $\mathcal{R}$ the set of the 13 relations between pairs of elements of I. $\mathcal{R}$ is informally specified by the following axiom schemas:

1. For all $i \in I$, there exists an interval $j \in I$, $i \neq j$ with $R(i, j)$ for $R \in \mathcal{R}$.
2. The relationships in $\mathcal{R}$ are mutually exclusive:
   $\forall i, j \in I, R \in \mathcal{R}. \forall R' \in \mathcal{R}. R(i, j) \Rightarrow \neg R'(i, j)$
3. The relationships have a transitivity behavior. E.g. if $i\ before\ j$ and $j\ meets\ k$, then $i\ before\ k$. The transitivities between all pairs of the 13 basic relations are given in a transitivity table in [All83].

The relationships between intervals have also been studied by other researchers: van Benthem [vB91] has introduced two basic relations, *precedes* and *contained-in* which are consistent with the structure of intervals over the rational numbers. Ladkin [Lad87] has defined each of the 13 basic interval relations in terms of the relation *precedes*. Allen and Hayes [HA89] have redefined the interval calculus in terms of the relation *meet*. Characteristics, axiomatizations, comparisons, and expressiveness of those three interval calculi have been investigated by Ladkin in [Lad87].

3.2.3 Combined and Generalized Models

In addition to purely point-based and purely interval-based time models, models integrating points and intervals [Vil82, Boc90, Gal90, VS96] as well as models to represent intervals which are not necessarily convex, referred to as *generalized intervals* [LMF86, Lad87, DC91, NS92, Lig98], have been proposed for different purposes.

3.2.3.1 Points and Intervals Combined. Differentiating between events which are instantaneous, i.e. duration-less events and events which have a temporal extend, i.e. continuous events has been motivated by accomplishing events like "to close the door" and continuous

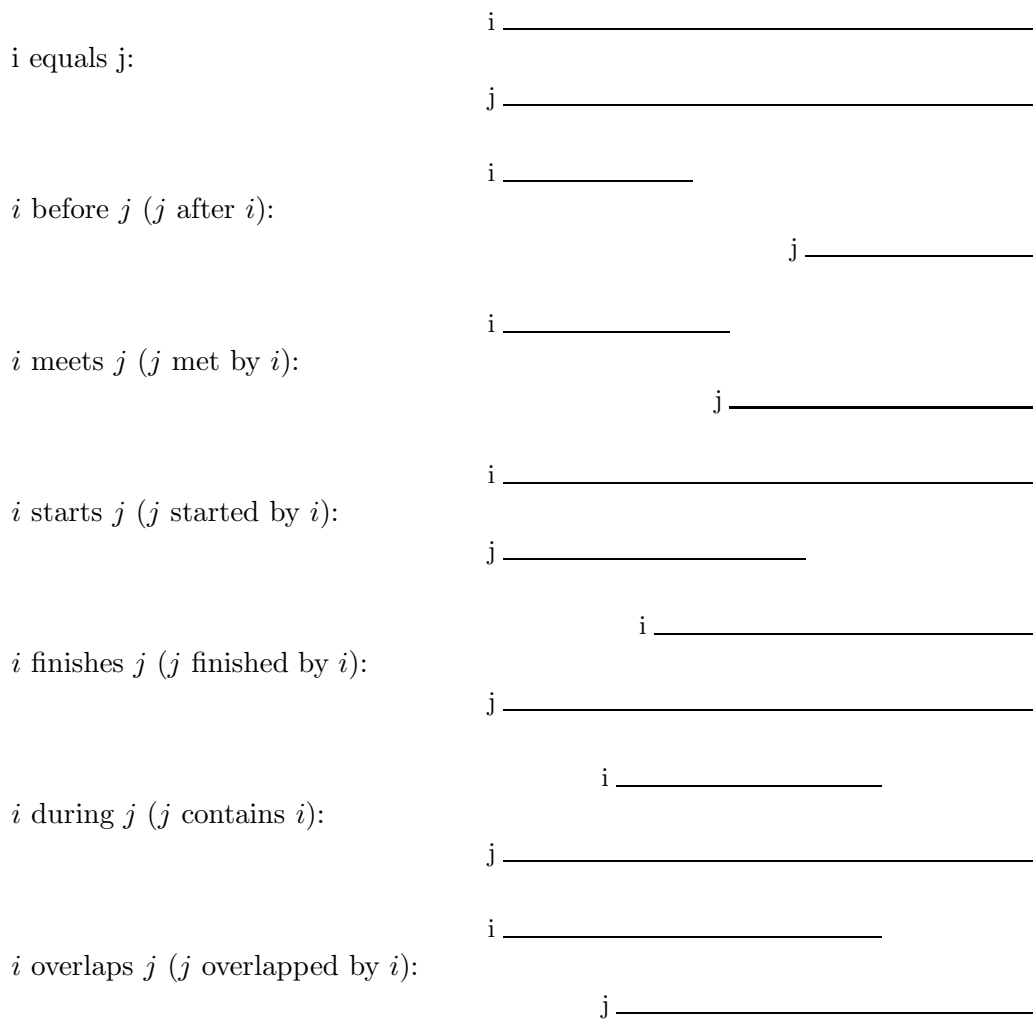

Figure 1: The 13 basic relations between two intervals i and j.

changes like a "100-meter sprint". Although this argumentation is not accepted in unison, time models integrating points and intervals have been proposed [Vil82, Boc90, Gal90, VS96]. The pioneer work in this tradition by Vilain [Vil82] has combined Allen's interval calculus [All83] with points and (primitive) relations between points and between points and intervals, yielding in 26 basic relations holding between points and/or intervals. Vila and Schwalb [VS96], whose work has been inspired by the works of Bochman [Boc90] and Galton [Gal90], have defined points from intervals such that an interval is defined by an ordered pair of ending points. A many-sorted first-order axiomatization is proposed for this calculus [VS96] with sorts for points and intervals and three primitive relations "$<$" (less) over intervals and "*begin*" and "*end*" over intervals and points. Dense and discrete time is differentiated by adding the denseness axiom over points.

3.2.3.2 Generalized Intervals. Motivated by planning problems [LMF86, NS92, TC95, KM99, Ter00] that require an abstraction from the number of times an event repeats, or from the number of times an event occurs in a temporal relation, e.g. to schedule one's office hours, time models for not necessarily convex intervals, so-called *generalized intervals* have been proposed [LMF86, Lad87, DC91, NS92, AK94, Lig98]. Generalized intervals are intended to represent (i) periodic events such as "John has breakfast every morning", (ii) interrupted events such as "John interrupts his breakfast to answer the phone", and (iii) events that consist of several related sub-events such as activities related to a planned event.
Since generalized time intervals are nothing more than mere collections of (possibly infinite many) intervals, the proposed models [LMF86, Lad87, DC91, NS92, AK94, Lig98] generalize Allen's interval algebra [All83] by defining specific relations between such interval collections. The proposals differ in the description of the generalized intervals as well as in the basic relations defined between pairs of them: Ladkin [Lad87] has proposed a general taxonomy of the set of all relations between any pair of generalized time intervals in terms of specific relationships between generalized time intervals. The suggested relations are derived from applications for task descriptions and management and process and action algebras. Ligozat [Lig98] has further extend this work. Khatib's [AK94] framework bases on matrices that represent binary relations between sequences of time intervals. Those matrices contain the time interval relations between any pair of convex components of the considered generalized time intervals.

3.3 Time Granularities Systems

The importance of data modeling and reasoning with data referring to different calendar units like "day", "week", or "teaching term" such as the widespread Gregorian times and dates has been widely recognized in the research areas of Artificial Intelligence and Database Systems. E.g. a person being in Munich from Tuesday to Thursday and in London from Friday to Sunday will meet another person in London on Friday at 8 p.m. One might question how to merge the apparently contradictory information: to relate the temporal conditions of the first person to those of the second person, the informations concerning the first person have to be precised in terms of hours. That means, days and hours cannot be compared by some computer application without having any informations on how to convert days to hours and vice versa. If the reasoner has some additional information about the temporal extend of the trip from Munich to London, one might reason about the time the first person have to leave Munich at the latest to be in London in time.

In the research areas of Artificial Intelligence and Database Systems one of the most promising approaches handling such problems has been the introduction of the concept of *time granularity* [Mon96, Je98, BJW00, Euz01]. Time granularities may be used to specify either the occurrence time of different classes of events (modeled for example by points or intervals) with time granularities or the temporal relations such as Allen's interval relations between such events. For example, the temporal event of a flight departure may be given in the time granularity "minute", and the temporal event of a student's examination in that of "day", or the temporal relation "during" may be given in the time granularity of "day", describing the stay of some person in London. Thus, time granularities can be used to specify both quantitative and qualitative temporal reasoning frameworks. To apply time granularities in such frameworks, not only the granularities themselves, but also meaningful relations between them are needed: if in some application events are expressed in different time granularities, the system must "compare" them in a meaningful manner to reason on relations formulated over such "differently grained" events.
With a common understanding [Mon96, Je98, BJW00, Euz01], time granularities are discretizations of a totally ordered (continuous) time line isomorphic to the real numbers. A time granularity is a (partial) partition[2] of such a time line into a (infinite) countable set of aggregates of points (of such a time line), so-called *granules*, isomorphic to the integers. Examples of granules are "02.01.2005" of time granularity day and "the first week in the year 2005" of time granularity week. The granules of each time granularity are non-overlapping, and they preserve the order of the underlying time line. Granules may be considered either as points or as intervals.
Numerous different formalizations of the concept of time granularity, various relationships between them, and several temporal operations on them have been proposed. The formalizations follow set-theoretic, logic-based, automata-based, and programming language-based approaches. Well-known approaches in each of those traditions are surveyed in the following.

3.3.1 Set-theoretic Time Granularity Systems

A lot of work on set-theoretic approaches to time granularities has been done in Artificial Intelligence [MMCR92, Fre92, CEMP93, Euz93, Her94, GPP95, BCDE00] and in Database Systems [And83, CR87, WJL91, WJS95, Wan95, Sno95, BJW00, NWJ02].
In Artificial Intelligence, proposals for quantitative time granularities based on a hierarchy of strictly constrained time granularities with upward and downward conversions on points and intervals have been made [MMCR92, CEMP93]. In [Euz93], a less-constrained quantitative formalization has been proposed. Qualitative time granularities have been proposed in [Fre92, Her94, GPP95]. In this approach, a separate set of relationships is defined at any granularity level. A qualitative, interval-based framework that uses the same representation formalism at each granularity level is proposed in [Euz93]. In [Euz93], conversions between time granularities are defined on the relations rather than on objects between those relations might hold. Note that these qualitative proposals can only reason in homogeneous networks where all relations are given in the same time granularity. The network can be then converted only as a whole. A qualitative formalization for time granularities where (point-based) objects are converted is proposed in [BCDE00].
In Database Systems, a large amount of work on time granularities exists. This approach to time granularities is inherently point-based and quantitative: conversions which introduce the

[2] in the mathematical sense of a partition of a set

(unsolved) problem of indeterminacy when converting to a finer time granularity is performed on point-based objects, i.e. granules are considered as points with time granularity. This approach suffers from some not only theoretically but also practically relevant problems such as the equivalence problem of time granularities. Furthermore, it only partially works out the problem of time granularity conversions. This problem of the quantitative, point-based approach is discussed in [FM01]. A comprehensive presentation of this quantitative, point-based approach to time granularities is given in [BJW00].
Initially in [And83], the need to support different time granularities in temporal databases has been addressed. Based on Anderson's work [And83], Clifford and Rao [CR87] have proposed a theoretical model of time granularities underlying a total order which introduces most of the features of the set-theoretic approach. They introduce granularity conversions along a *finer than* relation on this total order (e.g. hours are finer than days), defining a time granularity in terms of its next finer time granularity. Wiederhold, Jajodia, and Litwin [WJL91] have advanced this model by adding some specific semantics. These semantics allow for temporal comparisons regarding to the complete ordering of time granularities. Their model allows for handling mixed time granularities (i.e. temporal aspects stated in more than one time granularity). In sequencing papers, Wang and others [WJS95, Wan95] have generalized this totally ordered set of time granularities to a partially ordered one allowing for finer and coarser relations between the time granularities comprised in a particular finite set of partially ordered time granularities. In addition to this finer than relationship, several further relationships between time granularities have been thoroughly investigated. A collection of such relations is given in [BJW00]. Furthermore, a calendar algebra [NWJ02] has been proposed to symbolically represent time granularities as expressions form other (previously defined time granularities) using algebraic operators.

3.3.1.1 Time Granularities. According to the temporal structure proposed for time granularities in [CR87] which has been generalized in [BJW00], a time granularity G is a function from an index set into the power set over a linear (dense or discrete) time line. This function is order preserving and to conserving. The elements of a time granularity do not overlap. The *origin* of a time granularity is the smallest element according to the order relation and it is index by 0. The *image* of a time granularity is the union of its granules, and the *extend* of a time granularity is the smallest interval of the underlying time line that contains the time granularity's image.
For time granularities G and H, Clifford and Rao [CR87] have introduced a *finer than* relation as follows: a time granularity G is finer than a time granularity H iff $\forall i \in \mathbb{Z} \exists j \in \mathbb{Z}$ such that $G(i)$ is a subset of $H(j)$, denoted $G \preceq H$. In [BJW00], several additional relations between time granularities have been suggested:

$G \trianglelefteq H$	iff $\forall j \exists S \;\; H(j) = \bigcup_{i \in S} G(i)$	*groups into*
$G \sqsubseteq H$	iff $\forall i \exists j$ such that $G(i) = H(i)$	*sub-granularity of*
	$G \trianglelefteq H$ and $G \preceq H$	*partitions*
$G \Subset H$	iff the image of G is contained in the image of H	*covered by*
	$G \trianglelefteq H$ and their exists a periodic repetition of the grouping pattern of granules of G into granules of H	*groups periodically into*
$G \leftrightarrow H$	iff $\exists k \forall i$ such that $G(i) = H(i+k)$	*shift-equivalent*

Apart from the relation for shift-equivalence, all those relations define time granularities in a

way such that G is a "more precise" time granularity than H. Calendars are defined in this framework as sets of time granularities that contain a time granularity G such that each of the time granularities groups into the time granularity G [BJW00].
In [BJW00], specific sets of time granularities are defined within which the granules of one time granularity can be converted into granules of another time granularity. In this framework, however, conversions are not always defined: in particular, a conversion from a granule into a granule of a coarser time granularity is only defined, if the corresponding time granularities are related according to the "finer than" relationship [CR87]. A conversion from a granule into granules of a finer time granularity is only defined, if the corresponding time granularities are related according to the "group into" relationship [BJW00]. Thus, a universally valid solution to the problem of time granularity conversion is not provided in the framework of [BJW00]. The problem of time granularity conversion in the quantitative, point-based approach to time granularities is discussed in [FM01]. Less restricted time granularity systems have been considered for example in [Sno95].
In [NWJ02], a formal algebra to always define coarser time granularities from finer ones is proposed. The suggested algebraic operators directly refer to and relay on the afore mentioned relations between time granularities. The afore mentioned problem of time granularity conversion also appears within this calendar algebra.

3.3.1.2 Qualitative Time Granularity. Time granularity operators for qualitative time representation have been first proposed in [Euz93, Euz95]. Those time granularity operators are defined in the context of relational algebras, applicable to both point and interval algebras. In principle, a qualitative algebra [All83, VKvB90] is initially augmented with a neighborhood structure [Fre92]. Subsequently, an interval algebra with qualitative granularity in terms of conversions of the relations according to some coarser and/or finer granularity is constructed [Hir96]. In [Euz93, Euz95], the author provides with a framework where situations are described by a set of possible relationships holding between time points or time intervals. In particular, each layer of the qualitative and relational language represents a situation in the unalternated language (of the chosen) relational algebra. The qualitative time granularities are defined by a set of operators. The operators are used to convert the representation of a (whole) situation into a finer or coarser representation of the same situation (as a whole). The operators apply to the relationships that might hold between the temporal primitives (in some situation). The operators transform each of the relationships that hold in a situation into a coarser (using some upward conversion) or a finer (using some downward conversion) time granularity.
One of the important aspects of the work by Euzenat [Euz01] is the definition of a set of (generic) constraints which should be satisfied by any system of granularity conversion operators:

$r \in \rightarrow r$	*self-conversion*
$\forall r, \forall r', r'', \exists r_1, \ldots r_n$:	*neighborhood compatibility*
$r_1 = r', r_n = r''$ and $\forall i \in \{1, .., n-1\} N(r_i, r_{i+1})$ $\rightarrow r^{-1} = (\rightarrow r)^{-1}$	*conversion reciprocity distributivity*
$r \in \bigcap_{r' \in \uparrow r} \downarrow r$ and $r \in \bigcap_{r' \in \downarrow r} \uparrow r$	*inverse compatibility*
$\uparrow\uparrow r = \uparrow r$ and $\downarrow\downarrow r = \downarrow r$	*idempotence*
$\rightarrow r = \Leftarrow \rightarrow \Rightarrow r$ and $\rightarrow r = \Rightarrow \rightarrow \Leftarrow r$	*representation independence*

where $\uparrow$ denotes upward conversion, $\downarrow$ denotes downward conversion, $\rightarrow$ denotes a neutral operator (if a time granularity change between two layers is addressed, but it is not necessary to know which one is the coarser), $\Leftarrow$ and $\Rightarrow$ denotes an interval relational space, x and y denote time primitives, and r denotes a relation between time primitives. *Self-conversion* means that, independed of the conversion, a relationship must belong to its own conversion. *Neighborhood compatibility* contains the conversion of a relation to form a conceptual neighborhood. *Conversion reciprocity distributivity* denotes symmetry. It states that the conversion of the relation between a first object and a second object must be the reciprocal of the conversion of the relation between the second one and the first one. *Inverse compatibility* states that the conversion operators are consistent with each other. Finally, *representation independence* states that the conversion must not dependent on the representation of the temporal primitive, i.e. whether a relation between points or intervals is converted. From these constraints, the possible conversion operators for a particular relational algebra for points and/or intervals such as the algebras proposed in [Vil82, All83, VKvB90] can be generated. Examples of such relational algebras with conversion operators can be found in [Euz01].

3.3.2 Logic-based Time Granularity Systems

Hobbs [Hob85] has introduced a very general logic-based concept of granularity. He has considered predicates extracted from a global theory (given in some logic formalism) which are relevant to some present situation described at a specific granularity. Extracting relevant predicates leads to an indistinguishability relation: variables are indistinguishable to the situation at hand if no relevant predicates distinguishes between them. In this concept of granularity, a granularity is characterized *after* some theory has been defined. This work has been enhanced by Greer and McCalla [GM89].
Different frameworks to deal with time granularities in Linear Temporal Logics (LTL) has been suggested: for example, in [FM94], Fiadeiro and Maibaum represent each granularity in the same (classical) temporal logic. Conversions are defined between these different representations. Representations with mixed time granularities are not possible in this framework. Combi, Franceschet and Peron [CFP04] have defined time granularities as models of LTL formulas using appropriate propositional symbols to mark endpoints of granules. Demri [Dem04] has generalized the framework of Combi et al. and the automata-based approach (cf. Section 3.3.3) of Montanari and Dal Lago [LM01] to a first-order language. This language allows for defining time granularities by means of periodic constraints specified in a first-order logic.
In [CEMP93, Mon96], Montanari and others have proposed a quantitative temporal logic with granularities. The authors of this work show how to extend syntax and semantics of temporal logics to cope with quantitative temporal properties possibly expressed at different time granularity. This work results in a quantitative and layered temporal logic [Mon96]. This layered logic is based on the idea of a logic of positions, i.e. a topological or metric logic [RG68, RU71]. Such a metric logic is then generalized to a many-layered metric temporal logic, embedding the notion of time granularity. The main features of the logic proposed in [Mon96] are three different operators: a contextual operator (to associate different granularities with different formulas), a displacement operator (to move within a given granularity), and a projection operator (to move across granularities).
Alternative temporal logics with time granularities are proposed in [Mon96, MP96, MPP99]. The logics introduced in those works define various theories of time granularity as extensions of monadic second-order theories. They provide with suitable restrictions to such languages

for time granularities to get decidable temporal logics with time granularities.
Yet another alternative has been proposed in [OG98]. The authors of this work have integrated set-theoretic operators for defining time granularities in terms of predicates into a modal first-order language with modality operators like "next" with time granularity.

3.3.3 Automata-based Time Granularity Systems

Wijsen [Wij00] has proposed a string-based approach to represent time granularities, in particular, to represent infinite periodic time granularities like days or weeks. Such time granularities are modeled as (infinite) words over an alphabet that consists of three symbols: ■ (fillers), □ (gaps), and ≀ (separators). A granule is constructed form fillers and gaps and delimited by a separator. A finite set of granules is used to describe the repeating pattern of a time granularity. Periodic time granularities can be identified with ultimately periodic strings, and they can be finitely represented by specifying a (empty) finite prefix and a finite repeating pattern in terms of granules modeled by combination of the three symbols ■, □,and ≀.
This string-based approach has been further extended by Montanari and Dal Lago [LM01] to an automata-based approach: a specific subclass of Büchi automata has been introduced that recognizes languages only consisting of periodic words (described by the afore mentioned symbols of Wijsen's string-based approach).

3.3.4 A Programming Language Approach to Time Granularities

Bry, Rieß, and Spranger [BRS05] have proposed a programming language approach to modeling temporal and calendric data using time granularities and calendars specified as types: types complement data with machine readable and processable semantics, thus, enabling (meaningful) annotation of data. Such features become evident when considering the following example. An appointment scheduler inferring an appointment for a phone conference of two persons (where one is in Munich and the other in Tel Aviv) refers not only to several time constraints formulated by the conference attendees but also to various temporal and calendric data of different types (possibly interpreted w.r.t. different calendars). Types give such data their intended semantics, e.g. that some data refer to days or that some weekend refer to Saturdays and Sundays and some other weekend to Fridays and Saturdays, as it would be the case in Israel. Essentially, the calendric data involved in such applications are most often rather complex, sometimes involving different calendars (e.g. cultural calendars like the Gregorian and the Islamic and professional calendars) with various regulations and lots of irregularities (e.g. leap years). Furthermore, calendar data such as dates are probably more than any other data domain a subject to user interpretation: e.g. the date "12/02/2005" is interpreted in France as 12^{th} February 2005 while it is interpreted as 2^{nd} December 2005 in the US. The programming language approach to time granularities proposed in [BRS05] provides with declarative language constructs to model both regulations and irregularities and various date formats for such data.
The basic principle of the programming language approach to model calendric data is *predicate subtyping*. Predicate subtyping with predicate types is a stronger form of typing and subtyping enabling to encode more information in types, because the elements of a predicate type are described by a predicate set. Predicate sets are used to declaratively define (possibly infinite) sets. Predicate types have been widely investigated in type theory, logics, proof assistents,

and theorem proving [ROS98]. The typing approach to time granularities and calendars uses predicate types in a different manner and not for theoretical, but instead practical purposes: predicate types are used to define time granularities like "month", "working day", "teaching term", or "exam week" as calendric types. Furthermore, the definition of predicate types is restricted to *aggregations* and *inclusions* of time by providing type constructors which are limited to aggregations and inclusions always define subsets isomorphic to the integers.[3] Finally, predicate types are used as a means to define conversions between any pair of calendric types. Thus, the approach of [BRS05] essentially provides a solution to the problem of time granularity conversion that has been addressed in [FM01].

The type language CaTTS [BRS05] is a realization of this programming language approach. CaTTS is a generic modeling language for data modeling and reasoning with calendars. CaTTS consists of two languages, a *type definition language*, CaTTS-DL, and a *constraint language*, CaTTS-CL, of a (common) parser for both languages, and of a language processor for each language. CaTTS-DL provides with CaTTS-TDL (for type definition language), a tool to define calendars and CaTTS-FDL (for format definition language), a tool to define calendar data, in particular dates to give calendar data well-defined meanings. CaTTS-CL provides a means to express a wide range of temporal constraints over calendar data referring to the types defined in calendar(s) specified in CaTTS-DL.

In CaTTS-DL, one can specify in a rather simple manner more or less complex, cultural and professional calendars. Irregularities like leap seconds or Hebrew leap months can be easily expressed in CaTTS-DL. In particular, CaTTS-DL provide a means to define time granularities as *calendric types* by means of predicate types. E.g. the Gregorian calendar can be modeled in CaTTS-DL as follows:

```
calendar Gregorian =
 cal
   type second;
   type minute = aggregate 60 second @ second(1);
   type hour = aggregate 60 minute @ minute(1);
   type day = aggregate 24 hour @ hour(1);
   type week = aggregate 7 day @ day(1);
   type month = aggregate
     31 day named january,
     alternate month(i)
      | (i div 12) mod 4 == 0 &&
        ((i div 12) mod 100 != 0 || (i div 12) mod 400 == 0) -> 29 day
      | otherwise                                             -> 28 day
     end named february,
     31 day name march,
     30 day named april,
     31 day named may,
     30 day named june,
     31 day named july,
     31 day named august,
     30 day named september,
     31 day named october,
     30 day named november,
     31 day named december
    @ day(1);
   type year = aggregate 12 month @ month(1);
   type weekend_day = select day(i) where
```

[3]Note that the elements of each calendric type can be conveniently represented by integer sets.

```
            relative i in week >= 6 && relative i in week <= 7;
  type working_day = day\weekend_day;
end
```

The above calendar specification binds a calendar (between the keywords `cal` and `end`) to the identifier `Gregorian`. This CaTTS-DL calendar specification consists of a set of type definitions (each identified by the keyword `type` followed by an identifier). The first type defined is `second`. It has no further properties. The type `minute` is defined from the type `second` by specifying a predicate. The CaTTS language processor interprets this recursive type definition as an *aggregation subtype* of the type `second` such that each of its elements comprises 60 seconds[4] (denoted `aggregate 60 second`) and that the minute that has index 1, i.e. `minute(1)` comprises all seconds between `second(1)` in `second(60)` (denoted `@ second(1)`). Any further type definition follows the same pattern. The definitions are straightforward following the rules of the Gregorian calendar [DR01]. Since Gregorian months have different lengths, a CaTTS type `month` is defined with a repeating pattern of the twelve months. The months February, which is one day longer in each Gregorian leap year is defined by an additional pattern which specifies the leap year rule for the Gregorian calendar using the CaTTS language construct `alternate...end`. The type definition of the type `weekend_day` is derived from that of the type `day`. The CaTTS language processor interprets this type definition as an *inclusion subtype* of the type `day` such that each of its elements must be relatively to a week either the 6^{th} or the 7^{th} day (denoted `relative i in week >= 6 && relative i in week <= 7`). The type `working_day` is also and inclusion subtype of day, selecting those days which are not weekend days (denoted `day\weekend_day`).
The above exemplified CaTTS-DL calendar specification defines a calendar as a "type" that can be used, in principle, in *any* language (e.g. SQL, XQuery, XSLT, XML Schema, OWL), using calendar data enriched with type annotations after this CaTTS-DL calendar. CaTTS' type checker is used to check the calendar data typed after a CaTTS-DL calendar in such programs or specifications, thus, providing a means to interpret such data.
Note further that particularities like time zones can be easily expressed in a CaTTS-DL calendar definition. Calendar definitions of other cultural calendars in CaTTS-DL, in particular the Islamic and Hebrew calendars and variations of the Gregorian calendar like the Japanese calendar as well as date format specifications using CaTTS-FDL are given in [BRS05, BS04, BHRS05, Spr06].

4 Temporal Constraint Satisfaction

In the following, formalisms for *Temporal Constraint Satisfaction Problems (TCSPs)* based upon relating points and/or intervals are surveyed. For all representations in this section, time is assumed being linear, dense, and unbounded which is common with most proposals. Two principle temporal constraint formalisms have been introduced in the literature:

1. *qualitative* temporal constraints, in particular in Allen's interval calculus [All83, vBC90, VKvB90, KL91, Mei96], and

2. *quantitative* temporal constraints, mainly based on disjunctive linear relations between time points [KL91, DMP91, Kou92, Mei96] to manipulate dates and durations.

[4] In CaTTS-DL, it is possible to define a type `minute` that considers leap seconds, as well.

Combined (qualitative and/or quantitative) formalisms, based on points and/or intervals, and binary relations between them are proposed in [Vil82, Mei96].

Recall that a *Constraint Satisfaction Problem (CSP)* is defined by a finite sequence of variables $\mathcal{X} := X_1, \ldots, X_n$ where $n > 0$, with respective domains $\mathcal{D} := D_1, \ldots, D_n$, i.e. variable X_i ranges of the domain D_i together with a finite set $\mathcal{C}$ of constraints, each defined on a subsequence of $\mathcal{X}$. A *constraint* C on $\mathcal{X}$ is a subset of $D_1 \times \ldots \times D_n$. If C equals $D_1 \times \ldots \times D_n$, then C is *solved.* A CSP is *solved,* if all its constraints are solved and no domain of it is empty, and *failed,* if it either contains the false constraint, usually denoted $\perp$, or some of its domains are empty. In a TCSP, variables represent points and/or intervals and constraints represent temporal relations between them.
A *solution* to a CSP is a sequence of legal values $(d_1, \ldots, d_n) \in D_1 \times \ldots \times D_n$ if for every constraint $C \in \mathcal{C}$ on the variables $X_{i_1}, \ldots, X_{i_m}$, $(d_{i_1}, \ldots, d_{i_m}) \in C$, i.e. if for all variables of the CSP all constraints are satisfied. If a CSP has a solution, it is *consistent*, otherwise *inconsistent.*
Most approaches to TSCPs have focused on satisfiability problems, i.e. *consistency*, the problem of deciding whether a set of temporal formula (modeled by a set of constraints) has a model or not. In the context of Allen's calculus, the problem of *entailed* relations is also considered.
Consider the following temporal reasoning problem.

Example 4.1 *John, Mary, and Tom work in the same bank, but in different affiliations. They plan a phone conference. They have decided to phone in the morning before the bank's affiliations open at 9:00. John leaves home between 7:50 and 8:00. It takes him 15 to 20 minutes to reach the office. When he arrives, he can wait 30 to 40 minutes. Mary can phone John 5 to 10 minutes after she arrives at the office or between 8:40 and 8:45. Tom arrives before the bank's affiliation opens.*

Reasoning about such information is the principal task of temporal reasoning systems. One wants to answer queries like "is the contained temporal information consistent" (i.e. if there is no conflict between the temporal information modeled), or "what is the earliest/latest possible time for the phone conference to take place".

Properly analyze of such problems leeds to an abstract analyze of *events* (i.e. activities that take time) like being present at work, walking to the office, receiving a phone call, etc. Such events can be identified with (different) time primitives: *points*, e.g. `8:20`, the time when Tom is soonest at the office, or the time, Mary leaves home, and *(generalized) intervals*, `[7:50,8:00]` the time when John leaves home, the time it takes John between leaving his home and arriving at the office, or the time Mary is able to phone John.
Such events may be either related by *quantitative* constraints or by *qualitative* constraints. Quantitative (time) constraints are dates and durations between two events, placing absolute bounds on events, for example 9:00, the time the bank opens according to Example 4.1. Qualitative (time) constraints are relative positions of events, for example John's presence at the office overlaps with Mary's presence at the office or Tom is at the office before the affiliation opens according to Example 4.1.

Note: Quantitative and qualitative constraints serve to model inherently different temporal reasoning problems. To solve quantitative and qualitative temporal reasoning problems, different constraint programming techniques are applied. In particular, quantitative constraints can be modeled by disjunctive linear relations, solved using *backtracking* (and *bounds/arc consistency*) techniques [Apt03] while qualitative constraints are modeled by the (transitivity property of) temporal relations themselves, solved using *path consistency* techniques [Apt03].

4.1 Quantitative Temporal Constraints

In the simplest case, temporal information is available in terms of dates, durations, or other precise numeric form. In this case, quantitative temporal constraints can be formulated, referring to absolute distances between points. Temporal information can be therefore represented numerically such that it may be easily computed, merely subtracting numeric values. Such temporal information can be modeled in so-called *acyclic directed graphs*. In such a graph, (temporal) events are represented by nodes, and the distances between these events are represented by the graph's edges, obtaining a partial order on temporal events. A known event occurrence may then be represented by a constraint. Such an event constraint is represented by a pair of its earliest and latest appearance in time. Those times are computed by adding up distances between the known event and some other (known) event. If the graph contains different possible paths between (unknown) events, its earliest (resp. latest) appearance are modeled by minimal (resp. maximal) path distances.
In the common (more complex) case, the precise numeric information is not available and/or information about distances is not precise, but expressed as a range of possible distances. These distances may be either precise distances or completely qualitative distances like ranges.

Dechter, Meiri, and Pearl [DMP91] have applied the technique of CSPs to model quantitative temporal information, aiming at a TCSP where each variable represents a point in time. Two types of constraints are introduced:

- *unary constraints* to represent point to date information, i.e. a point t is represented by a finite domain $\{i_1, ..., i_n\}$ with $n \geq 1$ and $t \in i_1 \vee ... \vee t \in i_n$, and
- *binary constraints* to represent point to point information, i.e. the distance between two points t_1, t_2 is represented by a finite domain $\{i_1, ..., i_n\}$ with $n \geq 1$ and $t_2 - t_1 \in i_1 \vee ... \vee t_2 - t_1 \in i_n$.

Turning attention back to the introducing Example 4.1, the temporal event that John leaves home between `7:50` and `8:00` may be formulated by an unary constraint t:

$$t \in \{(50, 60)\}$$

where `7:00` is chosen as relative origin in time and minute is the chosen time unit.
The temporal event 'Mary has time to phone John' may be specified by a binary temporal constraint $t_2 - t_1$:

$$t_2 - t_1 \in \{(65, 70), (100, 105)\}$$

where `7:00` is chosen as relative origin in time expressed in the time unit minute, as well.

Let t and s be two quantitative constraints with finite domains i and j. Then two operations over the set of quantitative temporal constraints are defined as follows:

- *Intersection:* The intersection of two quantitative temporal constraints t and s, denoted $t \bigoplus s$, is the set-theoretic intersection of their domains:

$$t \bigoplus s = \{x | x \in i \wedge x \in j\}$$

- *Composition:* The composition of two quantitative temporal constraints t and s, denoted $t \bigotimes s$, is defined as distance addition on their domains:

$$t \bigotimes s = \{z | \exists x \in i, \exists y \in j . x + y = z\}$$

Most of the algorithms proposed to manipulate quantitative temporal constraints are based on *Disjunctive Linear Relations (DLRs)*, and in particular, most tractable quantitative temporal formalisms [KL91, DMP91, Kou92] are subsumed by the Horn DLR constraint framework [Apt03].
Known tractable formalisms for quantitative temporal constraints to express statements such as "an event e happens 5 time units before event e" which are neither expressible in Allen's (qualitative) interval calculus [All83] nor subsumed by the Horn DLR framework are discussed in [Mei96, DJ97].
In [BJW00] a framework and a constraint solver for simple, i.e. point-based quantitative temporal constraints based on DLR Horn clauses with time granularities has been proposed. In this framework, each variable (i.e. each time point) is additionally associated with a time granularity. To propagate a temporal constraint on such time points with different time granularities, a conversion between time granularities has been introduced. In Bettini's and others framework, this conversion is only defined, if the corresponding time granularities are related according to the "group into" relationship.

4.2 Qualitative Temporal Constraints

Given two temporal primitives, i.e. points and/or intervals p and q, a *qualitative temporal constraint* $(p, q) : R$ says that p and q are supposed to stand in *one* of the relations $R \in \mathcal{R}$, where $\mathcal{R}$ is a finite set of *basic temporal relations* that may hold between the two temporal primitives p and q. The set $\mathcal{R}$ of basic temporal relations between points and/or intervals is given in the following:

- basic *point to point* relations [McD82, Vil82] possibly holding between two points, illustrated in Table 1,
- basic *interval to interval* relations [All83] possibly holding between two intervals, illustrated in Table 2, and
- basic *point to interval* and *interval to point* relations [Vil82, Mei96] possibly holding between a point and an interval, illustrated in Table 3.

Relation	Inverse	Relation on Endpoints
t before s	s after t	$t < s$
t equals s		$t = s$

Table 1: The basic binary relations between two points t and s.

Relation	Inverse	Relation on Endpoints
i equals j		$i^- = j^-, i^+ = j^+$
i before j	j after i	$i^+ < j^-$
i during j	j contains i	$j^- < i^-, i^+ < j^+$
i starts j	j started_by i	$i^- = j^-, i^+ < j^+$
i finishes j	j finished_by i	$j^- < i^-, i^+ = j^+$
i meets j	j met_by i	$i^+ = j^-$
i overlaps j	j overlapped_by i	$i^- < j^- < i^+ < j^+$

Table 2: The basic binary relations between two intervals $i = [i^-, i^+]$ and $j = [j^-, j^+]$.

Relation	Inverse	Relation on Endpoints
t before i	i after t	$t < i^-$
t after i	i before t	$t > i^-$
t during i	i contains t	$i^- < t < i^+$
t starts i	i started_by t	$t = i^-$
t finishes i	i finished_by t	$t = i^+$

Table 3: The basic binary relations between a point t and an intervals $i = [i^-, i^+]$.

Let us turn attention back to the introducing Example 4.1. Let further p and q be two temporal primitives. An example of a qualitative temporal constraint between p and q, describing the temporal relation of the fact when Mary and John arrive at the office is:

$$(p, q) : \{before, meets, overlaps\}$$

This qualitative temporal constraint means p *before* q or p *meets* q or p *overlaps* q, i.e. a qualitative temporal constraint describes a disjunction of the possible basic temporal relations between a pair of temporal primitives.

In the point algebra, 2^3 subsets of possible combinations of basic temporal relations (i.e. qualitative temporal constraints) exist, in the interval algebra, 2^{13}, and in the point-interval and interval-point algebra 2^5, each, qualitative temporal constraints exist. Two binary operations are defined over the set of these qualitative temporal constraints:

- *Intersection:* the intersection of two qualitative temporal constraints R_1 and R_2, denoted $R_1 \bigoplus R_2$, is the set-theoretic intersection $R_1 \bigcap R_2$.
- *Composition:* the composition of two qualitative temporal constraints R_1 and R_2, denoted $R_1 \bigotimes R_2$, is defined by a transitivity table [All83], i.e. the 3-elemental transitivity: from $(p, q) : R'$ and $(q, r) : R''$, $R' \bigotimes R''$ is the least restrictive relation between p and r.

Since the constraint propagation algorithm for computing the closure of a set of propositions (i.e. to check for consistency) in the interval algebra is sound but not complete, time intervals are represented by their ending points in many qualitative temporal reasoning applications, in particular when points and intervals are "first-class citizens" of the respective time model, benefiting from the computational advantages of the point algebra [Mei96].
Note that only a part of Allen's interval algebra [All83], that of convex intervals, may be expressed in the point algebra, using the three point relations *before, equals,* and *after*. A complete classification of the computational complexity of Allen's interval calculus and its tractable subclasses (in particular those subclasses of the interval algebra which are subsets of the ORD-Horn algebra) is presented in [KJJ03]. Note further that integrating point-to-point, interval-to-point, and point-to-interval relations into tractable subclasses of Allen's interval calculus does not change its computational complexity [Mei96] in a qualitative temporal reasoning framework.

4.2.1 Quantitative and Qualitative Constraints Combined

The two kinds of temporal constraints, qualitative and quantitative, have been combined into a single representation scheme accepting both kinds of temporal information. Mainly two different approaches exist:

1. The qualitative and quantitative temporal constraints are kept and processed separately. Subsequently the individual parts are composed to a global solution. This approach has been followed by Kautz and Ladkin [KL91].
2. The qualitative and quantitative temporal constraints are integrated into a single *general temporal network* where temporal variables represent points (resp. intervals) along with a set of unary and binary constraints which may be qualitative or quantitative.

Algorithms solving such temporal networks involve different constraint satisfaction techniques [Apt03], in particular *path consistency* and *backtracking*. This approach has been followed by Meiri [Mei96].

Different algorithms for TCSPs in which the relationships among temporal constraints can be modeled by directed graphs have been developed. Algorithms that operate over such so-called *constraint networks* are for example question answering and consistency checking. In those algorithms, constraint propagation is performed on a network of intervals and/or points. The nodes represent intervals and/or points, and the arcs are labeled with sets of binary relations (cf. Table 1, Table 2 and Table 3) on intervals and/or points. The labels represent constraints on the relations among the nodes. A good survey on constraint propagation algorithms for TCSPs is given in [FGV05].

4.3 Multi-Calendar Temporal Constraints

As already mentioned, in [BJW00] an approach to simple, i.e. point-based metric temporal reasoning with time granularities have been proposed. This framework allows for modeling and reasoning with simple temporal constraints on points (with time granularity) and distances between points (with time granularity) in a DLR Horn framework. In this framework, one can express constraints like "at time t (in time granularity g), person A is in London", but neither "an event e (in time granularity g) happens during a task t (in time granularity h)" nor "an event e (in time granularity g) happens 5 time units (in granularity h) before an event e' (in time granularity k". To model and solve such metric temporal constraints, a more expressive framework is required; in particular, one must not only refer to time points but also to time intervals. Starting from the point-based model of [BJW00], interval can be represented by an ordered pair of points with the first point less than the second. To ensure that such intervals can meet, i.e. only having one endpoint in common, the endpoints of the intervals must be *less*, i.e. for an interval i with endpoints i^- and i^+ $i^- < i^+$. This can be achieved when the intervals are closed in their starting points and open on their ending points (or vice versa). However, this requirement points out that a time model based on points does not correspond with our intuitive notion of time. The modeling of temporal knowledge is significantly complicated, because both, points and intervals have to be considered. Furthermore, having e.g. an event e holding sometime during an interval i which in turn is during an interval i', then e holds during i', as well. Thus, relations between intervals and events may be "carried forward" such that reasoning can be kept local. It is not clear to maintain those properties when considering intervals modeled by endpoints in a point-based time model.
Alternatively, to overcome the problems of point-based time models, an approach to model and solve Multi-Calendar Constraint Satisfaction Problems, Multi-Calendar CSPs for short, that is based purely on intervals is proposed in [Spr06]. This approach is realized using the calendar modeling language CaTTS [BRS05]. Representing time intervals using finite domains with domains typed after some calendric type, one may also handle quantifications such as "an event e (in time granularity g) happens 5 time units (in granularity h) before an event e' (in time granularity k". Furthermore, this interval-based approach opens us with the possibility to additionally incorporate generalized (i.e. finite, non-convex) and (infinite) periodic intervals, as well.
To solve Multi-Calendar CSPs, in [Spr06], a *coercion semantics* for subtyping that compiles away subtyping during type checking a CaTTS-CL program against a CaTTS-DL calendar

specification has been defined. In particular, this coercion semantics translates a CaTTS-CL program into an equivalent program in the constraint system *typedFD*. The constraint system typedFD is an extension of the constraint system Finite Domains [FA03] with (1) typed calendar domains and (2) the conversion constraint: $\forall \alpha, \beta \in \mathcal{C}.\ \alpha \leq \beta,\ X^{\alpha} \simeq Y^{\beta}.\ \alpha, \beta$ are calendric types defined in a CaTTS-DL calendar specification $\mathcal{C}$.[5] Using this constraint system typedFD, Multi-Calendar CSPs with finite domain variables can be solved. This extension takes advantage of existing local consistency algorithms approximating a solution to such problems which are known to be NP-hard. To solve Multi-Calendar CSPs, bounds consistency [vHSD92] is used, because the calendar domains are represented by a set of interval domains (i.e. one for the starting time and one for the duration).
Note that in contrast to temporal reasoning with time granularities in [BJW00], in a Multi-Calendar CSP, conversion between *any* pair of calendric types (time granularities) defined in CaTTS-DL is always possible.

5 Conclusion

This article has surveyed various approach to temporal knowledge representation and reasoning from Artificial Intelligence, Database Systems, and Web and Semantic Web research. The considered approaches are classified first, according to their modeling facilities and second, according to their reasoning facilities. Although most of the reviewed time models are rather theoretic, the article has motivated the practical impact of time and calendar modeling and reasoning by Web applications. One of the reviewed approaches, the calendar modeling language CaTTS together with its constraint reasoner for Multi-Calendar CSPs points out to be rather practical, and thus, applicable in such Web and Semantic Web applications.

References

[ABH+02] A. Ankolekar, M.H. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, S.A. McIlraith, S. Narayanan, M. Paolucci, T.R. Payne, K.P. Sycara, and H. Zeng. DAML-S: Web Service Description for the Semantic Web. In *Proceedings of the* 1^{st} *International Semantic Web Conference*, LNCS 2342, pages 411–430. Springer-Verlag, 2002.

[AK94] L. Al-Khatib. *Reasoning with Non-convex Intervals.* PhD Thesis, Florida Institute of Technology, 1994.

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

[All91] J. Allen. Planning as Temporal Reasoning. In *Proceedings of the* 2^{nd} *International Conference on Principles of Knowledge Representation and Reasoning*, pages 3–14. Morgan Kaufmann Publishers, 1991.

[And83] T. Anderson. Modeling Events and Processes at the Conceptual Level. In *Proceedings of the* 2^{nd} *International Conference on Databases*, pages 151–168. Wiley Heyden Ltd., 1983.

[5] X^{α} denotes a calendar domain expression of calendric type α where α is defined in some CaTTS-DL calendar specification $\mathcal{C}$.

[Apt03] K.R. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003.

[BAF98] L. Bertossi, M. Arenas, and C. Ferretti. SCDBR: An Automated Reasoner for Specifications of Database Updates. *Journal of Intelligent Information Systems*, 10(3):235–280, 1998.

[Bak91] A. Baker. Non-monotonic Reasoning in the Framework of the Situation Calculus. *Artificial Intelligence*, 49(1-3):5–23, 1991.

[BCDE00] G. Becher, F. Cléin-Debart, and P. Enjalbert. A Qualitative Model for Time Granularity. *Computational Intelligence*, 16(2):138–169, 2000.

[BCF+98] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The Interactive Museum Tour-guide Robot. In *Proceedings of the* 15^{th} *National Conference on Artificial Intelligence/*10^{th} *Conference on Artificial intelligence/Innovative Applications of Artificial Intelligence*, pages 11–18. American Association for Artificial Intelligence, 1998.

[BDD95] K.van Belleghem, M. Denecker, and D. DeSchreye. Combining Situation Calculus and Event Calculus. In *Proceedings of the International Conference on Logic Programming, MIT Press*, pages 83–97, 1995.

[BHRS05] F. Bry, J. Haußer, F.-A. Rieß, and S. Spranger. Cultural Calendars for Programming and Querying. In *Proceedings of the* 1^{st} *Forum on the Promotion of European and Japanese Culture and Traditions in Cyber Society and Virtual Reality*, 2005.

[BJW00] C. Bettini, S. Jajodia, and S.X. Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning.* Springer-Verlag, 2000.

[Boc90] A. Bochman. Concerted Instance-interval Temporal Semantics: Temporal Ontologies. *Notre Dame Journal of Formal Logic*, 31(3):403–414, 1990.

[BRS05] F. Bry, F.-A. Rieß, and S. Spranger. CaTTS: Calendar Types and Constraints for Web Applications. In *Proceedings of the* 14^{th} *International World Wide Web Conference*, pages 702–711. ACM Press, 2005.

[BRST00] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, High-level Agent Programming in the Situation Calculus. In *Proceedings of the* 17^{th} *National Conference on Artificial Intelligence and* 12^{th} *Conference on on Innovative Applications of Artificial Intelligence,*, pages 355–362. AAAI Press / The MIT Press, 2000.

[Bru72] B. Bruce. A Model for Temporal References and its Applications in a Question Answering Program. *Artificial Intelligence*, 4:1–25, 1972.

[BS04] F. Bry and S. Spranger. Towards a Multi-calendar Temporal Type System for (Semantic) Web Query Languages. In *Proceedings* 2^{nd} *International Workshop Principles and Practice in Semantic Web Reasoning*, LNCS 3208, pages 69–83. Springer-Verlag, 2004.

[CCM95] I. Cervesato, L. Chittaro, and A. Montanari. A Modal Calculus of Partially Ordered Events in a Logic Programming Framework. In *Proceedings of the* 12^{th} *International Conference on Logic Programming*, pages 299–313. MIT Press, 1995.

[CEMP93] E. Ciapessoni, E.Corsetti, A. Montanari, and P. San Pietro. Embedding Time Granularity in a Logical Specification Language for Synchronous Real-time Systems. *Science of Computer Programming*, 20(1–2):141–171, 1993.

[CFP04] C. Combi, M. Franceschet, and A. Peron. Representing and Reasoning about Temporal Granularities. *Journal of Logic and Computation*, 14(1):51–77, 2004.

[CMP93] I. Cervesato, A. Montanari, and A. Provetti. On the Non-monotonic Behavior of Event Calculus for Deriving Maximal Time Intervals. *Interval Computations*, 3(2):83–119, 1993.

[CR87] J. Clifford and A. Rao. A Simple General Structure for Temporal Domains. In *C. Rolland, and M. Leonard (eds.), Temporal Aspects of Information Systems*, pages 17–28. Elsevier Science Publishers, 1987.

[DAR02] DARPA Agent Markup Language. *A DAML Ontology of Time*, 2002.

[DC91] W. Davis and J. Carnes. Clustering Temporal Intervals to Generate Reference Hierarchies. In *Proceedings of the* 2^{nd} *International Conference on Principles of Knowledge Representation and Reasoning*, pages 111–117. Morgan Kaufman, 1991.

[Dem04] S. Demri. LTL over Integer Periodicity Constraints (Extended Abstract). In *Proceedings of* 7^{th} *International Conference on Foundations of Software Science and Computation Structures*, LNCS 2987, pages 121–135. Springer-Verlag, 2004.

[DJ97] T. Drakengren and P. Jonsson. Eight Maximal Tractable Subclasses of Allen's Algebra with Metric Time. *Journal of Artificial Intelligence Research*, 7:25–45, 1997.

[DMB92] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal Reasoning with Abductive Event Calculus. In *Proceedings of the* 10^{th} *European Conference on Artificial Intelligence*, pages 384–388. John Wiley and Sons, Chichester, 1992.

[DMP91] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1–3):61–95, 1991.

[Dow79] D. Dowty. *Word Meaning and Montague Grammar*. Kluwer Academic Publishers, 1979.

[DR01] N. Dershowitz and E.M. Reingold. *Calendrical Calculations: The Millennium Edition*. Cambridge University Press, 2001.

[Esh88] K. Eshghi. Abductive Planning with Event Calculus. In *Proceedings of the* 5^{th} *International Conference on Logic Programming, MIT Press*, pages 562–579, 1988.

[Euz93] J. Euzenat. Représentation Granulaire du Temps. *Revue d'Intelligence Artificielle*, 7(3):329–361, 1993.

[Euz95] J. Euzenat. A Categorical Approach to Time Representation: First Studies on Qualitative Aspects. In *Proceedings of the IJCAI Workshop on Spatial and Temporal Reasoning*, pages 142–152, 1995.

[Euz01] J. Euzenat. Granularity in Relational Formalisms with Applications to Time and Space Representation. *Computational Intelligence*, 17(3):703–737, 2001.

[FA03] T. Frühwirth and S. Abdennadher. *Essentials of Constraint Programming*. Cognitive Technologies. Springer-Verlag, 2003.

[FGV05] M. Fisher, D. Gabbay, and L. Vila. *Handbook of Temporal Reasoning in Artificial Intelligence*. J. Hendler, H. Kitano, B. Nebel (eds.), Foundations of Artificial Intelligence: Volume I. Elsevier, 2005.

[FHN72] R. Fikes, P. Hart, and N.J. Nilsson. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[FM94] J.L. Fiadeiro and T. Maibaum. Sometimes "Tomorrow" is "Sometime": Action Refinement in a Temporal Logic of Objects. In *Proceedings of the 1st International Conference on Temporal Logic*, LNCS 827, pages 48–66. Springer-Verlag, 1994.

[FM01] M. Franceschet and A. Montanari. A Combined Approach to Temporal Logics for Time Granularity. In *Workshop on Methods for Modalities*, 2001.

[FN71] R. Fikes and N. Nilsson. STRIPS: A new Approach to Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[Fre92] C. Freska. Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, 54(1):199–227, 1992.

[FWP97] A. Fernandes, M. Williams, and N. Paton. A Logic-based Integration of Active and Deductive Databases. *New Generation Computing*, 15(2):205–244, 1997.

[Gal90] A. Galton. A Critical Examination of Allen's Theory of Action and Time. *Artificial Intelligence*, 42(2-3):159–188, 1990.

[Gan99] S. Gançarski. Database Versions to Represent Bitemporal Databases. In *Proceedings of the 10th Conference on Database and Expert Systems Applications*, LNCS 1677, pages 832–841. Springer-Verlag, 1999.

[GLL97] G. De Giacomo, Y. Lespérance, and H.J. Lévesque. Reasoning about Concurrent Execution, Prioritized Interrupts, and Exogenous Actions in the Situation Calculus. In *Proceedings of the* 15th *International Joint Conference on Artificial Intelligence*, pages 1221–1226. Morgan Kaufmann Publishers, 1997.

[GLR91] M. Gelfond, V. Lifschitz, and A. Rabinov. What are the Limitations of the Situation Calculus? In *Automated Reasoning, Essays in Honor of Woody Bledsoe, (ed.)S. Boyer*, pages 167–181. Kluwer Academic Publishers, 1991.

[GM89] J. Greer and G. McCalla. A Computational Framework for Granularity and its Application to Educational Diagnosis. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 477–482. Morgan Kaufmann Publishers, 1989.

[GPP95] M. Grigni, D. Papadias, and C. Papadimitriou. Topological Inference. In *Proceedings of the* 14th *International Joint Conference on Artificial Intelligence*, pages 901–907. Morgan Kaufmann Publishers, 1995.

[HA89] P.J. Hayes and J.F. Allen. Moments and Points in an Interval-based Temporal Logic. *Computational Intelligence*, 5(4):225–238, 1989.

[Ham72] C.L. Hamblin. Instants and Intervals. In *J.T. Fraser, F.C. Haber, and G.H. Müller (eds.), The Study of Time*, pages 324–328. Springer-Verlag, 1972.

[Har79] D. Harel. *First-order Dynamic Logic.* LNCS 68. Springer-Verlag, 1979.

[Har84] D. Harel. Dynamic Logic. In *D. Gabby et al. (eds.), Handbook of Philosophical Logic, vol. II, Extensions of Classical Logic, Publishing Company, Dordrecht (NL)*, 1984.

[Her94] D. Hernández. Qualitative Representation of Spatial Knowledge. LNAI 804. Springer-Verlag, 1994.

[Hir96] R. Hirsh. Relational Algebras of Intervals. *Artificial Intelligence*, 83(2):267–295, 1996.

[Hob85] J. Hobbs. Granularity. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 432–435. Morgan Kaufmann Publishers, 1985.

[Je98] C. Jensen and C. Dyreson (eds.). *The consensus glossary of temporal database concepts - February 1998 version*, 1998.

[KG77] K. Kahn and G.A. Gorry. Mechanizing Temporal Knowledge. *Artificial Intelligence*, 9:87–108, 1977.

[KJJ03] A. Krokhin, P. Jeavons, and P. Jonsson. The Tractable Subalgebras of Allen's Interval Algebra. *Journal of the ACM*, 50(5):591–640, 2003.

[KL91] H.A. Kautz and P. Ladkin. Integrating Metric and Temporal Qualitative Constraints. In *Proceedings of the* 9th *National Conference on Artificial Intelligence*, pages 241–246. MIT Press, 1991.

[KM99] L. Khatib and R. Morris. Generating Scenarios for Periodic Events with Binary Constraints. In *Proceedings of the* 6th *International Workshop on Temporal Representation and Reasoning*, pages 67–72. IEEE Computer Society, 1999.

[Kou92] M. Koubarakis. Dense Time and Temporal Constraints with $\neq$. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 24–35. Morgan Kaufmann Publishers, 1992.

[Kow92] R. Kowalski. Database Updates in the Event Calculus. *Journal of Logic Programming*, 12(1–2):121–146, 1992.

[KS86] R. Kowalski and M. Sergot. A Logic-based Calculus of Events. *New Generation Computing*, 4(1):67–95, 1986.

[KS94] R. Kowalski and F. Sadri. The Situation Calculus and the Event Calculus Compared. In *Proceedings of the 1994 International Symposium on Logic Programming*, pages 539–553. MIT Press, 1994.

[LÉ96] H. Lévesque. What is Planning in the Presence of Sensing? In *Proceedings of the* 13^{th} *National Conference on Artificial Intelligence*, pages 1139–1146. AAAI Press, 1996.

[Lad87] P.B. Ladkin. *The logic of Time Representation.* PhD Thesis, University of California, 1987.

[Lif91] V. Lifschitz. Towards a Metatheory of Action. In *Proceedings of the* 2^{nd} *International Conference on Principles Knowledge Representation and Reasoning*, pages 376–386. Morgan Kaufmann Publishers, 1991.

[Lig98] G. Ligozat. Generalized Intervals: A Guided Tour. In *Proceedings of Workshop on Spatial and Temporal Reasoning*, 1998.

[LLR99] H. Lévesque, Y. Lespérance, and R. Reiter. A Situation Calculus Approach to Modeling and Programming Agents. In *In A. Rao and M.Wooldridge (eds.), Foundations and Theories of Rational Agency*, pages 275–299. Kluwer Academic Publishers, 1999.

[LM01] U. Dal Lago and A. Montanari. Calendars, Time Granularities, and Automata. In *Proceedings of the* 7^{th} *International Symposium on Advances in Spatial and Temporal Databases*, LNCS 2121, pages 279–298. Springer-Verlag, 2001.

[LMF86] B. Leban, D. McDonald, and D. Foster. A Representation for Collections of Temporal Intervals. In *Proceedings of the* 5^{th} *National Conference on Artificial Intelligence*, pages 367–371. Morgan Kaufmann Publishers, 1986.

[LPR98] H. Lévesque, F. Pirri, and R. Reiter. Foundations for the Situation Calculus. *Linkping Electronic Articles in Computer and Information Science*, 3(18):159–178, 1998.

[LS95] F. Lin and Y. Shoham. Provably Correct Theories of Actions. *Journal of the ACM*, 42(2):293–320, 1995.

[McC02] J. McCarthy. Actions and other Events in Situation Calculus. In *Proceedings of the* 8^{th} *International Conference on Principles and Knowledge Representation and Reasoning*, pages 615–628. Morgan Kaufmann Publishers, 2002.

[McD82] D. V. McDermott. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science*, 6:101–155, 1982.

[Mei96] I. Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. *Artificial Intelligence*, 87(1–2):343–385, 1996.

[Mey99] J. Meyer. Dynamic Logic Reasoning about Actions and Agents. In *Workshop on Logic-Based Artifical Intelligence*, 1999.

[MH87] J. McCarthy and P. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Readings in Non-monotonic Reasoning*, pages 26–45. Morgan Kaufmann Publishers, 1987.

[Mis91] L. Missiaen. *Localized Abductive Planning with the Event Calculus*. PhD Thesis, Department of Computer Science, K.U. Leuven, 1991.

[MMCR92] A. Montanari, E. Maim, E. Ciapessoni, and E. Ratto. Dealing with Time and Granularity in the Event Calculus. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 702–712. IOS Press, 1992.

[Mon96] A. Montanari. *Metric and Layered Temporal Logics for Time Granularity*. ILLC Dissertation Series 1996-02, University of Amsterdam, 1996.

[Moo85] R. Moore. A Logic of Knowledge and Action. In *Hobbs, J.R. and Moore, R.C. (eds.) Formal Theories of the Common-Sense World*, pages 319–358. Ablex, 1985.

[MP96] A. Montanari and A. Policriti. Decidability Results for Metric and Layered Temporal Logics. *Notre Dame Journal of Formal Logic*, 37:260–282, 1996.

[MPP99] A. Montanari, A. Peron, and A . Policriti. Theories of Omega-Layered Temporal Structures: Expressiveness and Decidability. *Logic Journal of the IGPL*, 7(1):79–102, 1999.

[MPP02] P. Mateus, A. Pacheco, and J. Pinto. Observations and the Probabilistic Situation Calculus. In *Proceedings* 8^{th} *International Conference on Principles of Knowledge Representation and Reasoning*, pages 327–338. Morgan Kaufmann Publishers, 2002.

[MS99] R. Miller and M. Shanahan. The Event Calculus in Classical Logic — Alternative Axiomatizations. *Linkping Electronic Articles in Computer and Information Science*, 4(16), 1999.

[MSZ01] S. McIlraith, T. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[New36] I. Newton. *The Principia: Mathematical Principles of Natural Philosophy*. University of California Press, 1936.

[NS80] W.H. Newton-Smith. *The Structure of Time*. Routledge & Heagan Paul, 1980.

[NS92] M. Niezette and J.-M. Stevenne. An Efficient Symbolic Representation of Periodic Time. In *Proceedings of the* 1^{st} *Conference on Information and Knowledge Management*, LNCS 752, pages 161–168. Springer-Verlag, 1992.

[NWJ02] P. Ning, S.X. Wang, and S. Jajodia. An Algebraic Representation of Calendars. In *the Annuals of Mathematics and Artificial Intelligence*. Kluwer Academic Publishers, 2002.

[OG98] H.J. Ohlbach and D. Gabbay. Calendar Logic. *Journal of Applied Non-classical Logics*, 8(4):291–324, 1998.

[ÖS95] G. Özsoyoglu and R. Snodgrass. Temporal and Real-time Databases: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, 1995.

[PH04] F. Pan and J.R. Hobbs. Time in OWL-S. In *Proceedings of AAAI Spring Symposium on Semantic Web Services*, pages 29–36, 2004.

[PR93] J. Pinto and R. Reiter. Temporal Reasoning in Logic Programming: A Case for the Situation Calculus. In *Proceedings of the 10^{th} International Conference on Logic Programming*, pages 203–221. MIT Press, 1993.

[Pra76] V. Pratt. Semantical Considerations on Floyd-Hoare Logic. In *Proceedings of the 17^{th} IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[Ram00] J. Ramos. *The Situation and State Calculus: Specification and Verification*. PhD Thesis, IST, Universidade Técnica de Lisboa, 2000.

[Rei92] R. Reiter. On Formalizing Database Updates. In *Proceedings of the 3^{rd} International Conference on Extending Database Technology*, LNCS 580, pages 10–20. Springer-Verlag, 1992.

[Rei93] R. Reiter. Proving Properties of States in the Situation Calculus. *Artificial Intelligence*, 64(2):337–351, 1993.

[Rei95] R. Reiter. On Specifying Database Updates. *Journal of Logic Programming*, 25(1):53–91, 1995.

[Rei01] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[RG68] N. Rescher and J. Garson. Topological Logic. *Journal of Symbolic Logic*, 33:537–548, 1968.

[ROS98] J. Rushby, S. Owre, and N. Shankar. Subtypes for Specifications: Predicate Subtyping in PVS. *IEEE Transactions on Software Engineering*, 24(9):709–720, 1998.

[RS90] H. Reichgelt and N. Shadbolt. A Specification Tool for Planning Systems. In *Proceedings of the 9^{th} European Conference on Artificial Intelligence,*, pages 541–546, 1990.

[RU71] N. Rescher and A. Urquhart. *Temporal Logic*. Library of Exact Philosophy. Springer-Verlag, 1971.

[SG88] Y. Shoham and N. Goyal. Representing Time and Action in AI. Revised Version of: Problems in Formal Temporal Reasoning. *Artificial Intelligence*, 36(1):49–61, 1988.

[SGdMM96] C. Sierra, L. Godo, R. López de Màntaras, and M. Manzano. Descriptive Dynamic Logic and its Application to Reflective Architectures. *Future Generation Computer Systems*, 12(2–3):157–171, 1996.

[Sha89] M. Shanahan. Prediction is Deduction but Explanation is Abduction. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1055–1060. Morgan Kaufmann Publishers, 1989.

[Sha90] M. Shanahan. Representing Continuous Change in the Event Calculus. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 598–603, 1990.

[Sha95] M. Shanahan. A Circumscriptive Calculus of Events. *Artificial Intelligence*, 75(2):249–284, 1995.

[Sho87] Y. Shoham. Temporal Logics in AI: Semantical and Ontological Considerations. *Artificial Intelligence*, 33(1):89–104, 1987.

[SK95] F. Sadri and R. Kowalski. Variants of the Event Calculus. In *Proceedings of the 12th International Conference on Logic Programming*, pages 67–81. MIT Press, 1995.

[Sno95] R. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.

[Spr06] S. Spranger. *Calendars as Types – Data Modeling, Constraint Reasoning, and Type Checking with Calendars*. PhD Thesis. Herbert Utz Verlag, München, 2006.

[SWM95] P. Spruit, R. Wieringa, and J. Meyer. Axiomatization, Declarative Semantics and Operational Semantics of Passive and Active Updates in Logic Databases. *Journal of Logic and Computation*, 5(1):27–70, 1995.

[TC95] A. Tuzhilin and J. Clifford. On Periodicity in Temporal Databases. *Information Systems*, 30(5):619–639, 1995.

[Ter00] P. Terenziani. Integrated Temporal Reasoning with Periodic Events. *Computational Intelligence*, 16(2):210–256, 2000.

[vB91] J. van Benthem. *The Logic of Time*. D. Reidel Publishing Company, 1983; revised and expanded edition, 1991.

[vBC90] P. van Beek and R. Cohen. Exact and Approximate Reasoning about Temporal Relations. *Computational Intelligence*, 6(3):132–144, 1990.

[vHSD92] P. van Hentenryck, V. Saraswat, and Y. Deville. Constraint Processing in cc(FD). Technical Report, unpublished Manuscript, 1992.

[Vil82] M.B. Vilain. A System for Reasoning about Time. In *Proceedings of the 2nd National (US) Conference on Artificial Intelligence*, pages 197–201. AAAI Press, 1982.

[VKvB90] M.B. Vilain, H.A. Kautz, and P. van Beek. Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report. In *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann Publishers, 1990.

[VS96] L. Vila and E. Schwalb. A Theory of Time and Temporal Incidence based on Instants and Periods. In *Proceedings of the 3^{rd} Workshop on Temporal Representation and Reasoning*. IEEE Computer Society, 1996.

[Wan95] S.X. Wang. Algebraic Query Languages on Temporal Databases with Multiple Time Granularities. In *Proceedings of the 4^{th} International Conference on Information and Knowledge Management*, pages 304–311. ACM Press, 1995.

[Web90] J. Weber. On the Representation of Concurrent Actions in the Situation Calculus. In *Proceedings of the 8^{th} Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 28–32. Morgan Kaufmann Publishers, 1990.

[Wij00] J. Wijsen. A String-based Model for Infinite Granularities. In *Proceedings of the AAAI Workshop on Spatial and Temporal Granularities*, pages 9–16, 2000.

[WJL91] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with Granularity of Time in Temporal Databases. In *Proceedings of the 3^{rd} International Conference on Advanced Information Systems Engineering*, LNCS 498, pages 124–140. Springer-Verlag, 1991.

[WJS95] S.X. Wang, S. Jajodia, and V. Subrahmanian. Temporal Modules: An Approach Toward Federated Temporal Databases. *Information Sciences – Informatics and Computer Science: An International Journal*, 82(1–2):103–128, 1995.

[YS02] P. Yolum and M. Singh. Flexible Protocol Specification and Execution: Applying Event Calculus Planning Using Commitments. In *Proceedings of the 1^{st} International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 527–534. ACM Press, 2002.