

INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67 D-80538 München

————— **LMU**
Ludwig ———
Maximilians—
Universität —
München ———

Representation of Temporal Knowledge for Web-based Applications

Stephanie Spranger

Diplomarbeit

Beginn der Arbeit: 13. Mai 2002
Abgabe der Arbeit: 07. Oktober 2002
Betreuer: Prof. Dr. François Bry

Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

München, den 07. Oktober 2002

Abstract

The representation of temporal information, and reasoning about such temporal information is important for many (intelligent) computer systems. This has led to the development of a wide variety of temporal formalisms with a concise syntax, proper semantics, and computational inference rules. An overview of such temporal reasoning and data modeling formalisms is given. Any temporal formalism is usually developed with regard to its intended application – be it planning, forecasting, learning, or scheduling since any application’s particular temporal information crucially varies from the one’s of some other application.

Subsequent to the overview, Web-based planning systems and scheduling are considered. Such systems ask for representation of temporal knowledge. An automated appointment scheduling system is described to such an extent that it can be easily implemented, and its temporal aspects are thoroughly investigated. This is achieved by investigating the structure of appointments an appointment scheduler must be able to reason about, and by suggesting a possible hierarchy of calendar systems for modeling knowledge bases. Throughout this discussion, an available specification language for modeling temporal concepts of calendar systems (i.e., a particular temporal formalism) is used. The discussion provides means for designing a set of knowledge bases containing temporal concepts of different calendar systems, and an automated appointment scheduler.

Zusammenfassung

Die Darstellung zeitlicher Informationen und logisches Schließen über solche Informationen ist für viele intelligente Computersysteme von großer Bedeutung. Für solche Systeme wurden eine ganze Reihe von logik-basierten Beschreibungsformalissen zur Darstellung der Zeit (kurz: Zeitformalissen) entwickelt. Diese Arbeit bietet einen umfassenden Überblick über Zeitformalissen an. Es wird verdeutlicht, dass es keine universell einsetzbaren Zeitformalissen geben kann. Demzufolge sind Zeitformalissen insbesondere von ihrer jeweils intendierten Anwendung abhängig.

Im Anschluß an den Überblick über Zeitformalissen werden web-basierte Planungs- und Terminabsprachesysteme betrachtet. Solche Systeme erfordern die Modellierung zeitabhängigen Wissens. Ein automatisches Terminabsprachesystem wird beschrieben und zwar so, dass es leicht zu realisieren ist. Es wird hinsichtlich seiner Zeitaspekt analysiert. Im besonderen wird die Struktur von Terminen, wie sie bei Terminabsprachen auftreten können, und eine Hierarchie von Kalendersystemen zur Modellierung von Wissensbasen analysiert. Für diese Analyse wird eine Spezifikationsprache zur Modellierung von Kalenderausdrücken (also ein spezieller Zeitformalismus) verwendet. Aufbauend auf die Analyse können eine Hierarchie von Wissensbasen, die automatisch zugreifbare und wiederverwendbare Kalenderausdrücke beinhalten, und schließlich ein automatisches Terminabsprachesystem entwickelt werden.

Contents

1	Introduction	3
1.1	Motivation	4
1.2	Overview of the Thesis	5
2	Temporal Formalisms: A Survey	7
2.1	Change-based Formalisms	8
2.1.1	Situation Calculus	8
2.1.1.1	The Language of the Situation Calculus	9
2.1.1.2	Review and Extensions	12
2.1.2	Event Calculus	13
2.1.2.1	The Simplified Event Calculus	13
2.1.2.2	Review and Extensions	16
2.1.3	Dynamic Logic	17
2.1.3.1	Propositional Dynamic Logic	18
2.2	Time-based Formalisms	19
2.2.1	The Temporal Structure	21
2.2.1.1	Point-based Formalisms	22
2.2.1.2	Interval-based Formalisms	24
2.2.1.3	Point- and Interval-based Formalisms	27
2.2.1.4	Non-convex Intervals	30
2.2.2	Multidimensional Time	33
2.2.2.1	Temporal Dimensions	34
2.2.2.2	Applying Temporal Dimensions to the Web	39
2.2.3	A Representation Scheme for Temporal Formalisms	42
2.2.3.1	Calendar Systems	43
2.2.3.2	XML Schema's Means for Representing Time and Calendars	49
3	Three Scenarios	54
3.1	Appointment Scheduling	55
3.1.1	Scenario 1: Automated Appointment Scheduling via the Web	55
3.1.2	The Scenario's Temporal Aspects	56
3.2	Planning and Managing Events	58
3.2.1	Scenario 2: Automated Event Planning and Management via the Web	59
3.2.2	The Scenario's Temporal Aspects	60

<i>Representation of Temporal Knowledge for Web-based Applications</i>	2
3.3 Budgeting	62
3.3.1 Scenario 3: Automated Budgeting via the Web	62
3.3.2 The Scenario's Temporal Aspects	64
4 Time in a Calendar and Appointment Scheduling System	66
4.1 Overviewing the System's global Structure	67
4.2 The System's Temporal Aspects	72
4.2.1 The basic Functioning of an Appointment Scheduler	72
4.2.2 The Structure of Appointment Times	75
4.2.2.1 Appointments	75
4.2.2.2 User-defined Temporal Constraints	76
4.2.3 A Hierarchy of Calendar Systems	81
5 Conclusions	91
5.1 Results	91
5.2 Future Works	93
References	95

1 Introduction

In one way or another, any dynamic system that requires intelligence has to do with time. Planning systems reason about a sequence of actions, achieving one goal after another, to meet deadlines and further activities. Appointment systems schedule time slots for appointments among different people, meeting several specified temporal constraints. Furthermore, domains such as explanation, database updates and transactions, learning, robotics, and natural language understanding involves time.

The phenomena which happen in such dynamic systems occur over time. That means, anything occurs associated with its temporal references. Different actions occur temporally assigned (e.g., before, after) to different phenomena which are, in turn, referenced with time. Consequently, a certain time-dependent phenomenon remains in a system's state until an action that changes this phenomenon occurs. That means, if a change occurs, then the phenomenon passes over into its next state. Once we say something has changed, some state of a phenomenon in the system's modeled reality has changed. Thus, different states of phenomena referenced with time exist, and we are looking forward to specify the relations between such different states in time. These relations are temporal. They can be either explicit or implicit.

Time seems to be a fundamental issue that is associated with any time-dependent phenomenon in any dynamic system. As a consequence, we need means for representing time. The representation of time allows us for describing changes of the modeled reality by some system, and therefore, it is essential for reasoning about action and change. These and further considerations gave birth to a large field of research that can be summarized to the effort of developing a general framework for temporal reasoning. Temporal reasoning comprises a formalization of the aspects of time and means for representing and reasoning about the temporal aspects of the intended knowledge. For this reason, we need a language to represent these temporal aspects, and a method for reasoning about the required knowledge using such a language. This challenge has led to the development of several temporal formalisms (e.g., temporal logics, temporal models, and temporal representation schemes). To turn back, the specification of and reasoning about dynamic systems is the aim of temporal reasoning.

In general, the representation of time, and reasoning about time ask for knowledge-based representation formalisms (i.e., temporal formalisms). Any knowledge representation formalism needs a concise syntax, proper semantics, and computational inference rules. Therefore, any formalism equivalent to full first-order logic that allows for the use of context is an eligible candidate as knowledge representation formalism. Regarding to dynamic systems, a logic-based specification language for temporal representation and reasoning is required. Such a language need to capture the semantics of time phenomena a dynamic system is referenced with. That means, temporal phenomena in a dynamic system are modeled by using a properly fitting logic-based specification language for different aspects of time (i.e., a particular temporal formalism).

After having properly described the knowledge of some system's domain of discourse (e.g., time in a dynamic system) using a knowledge representation formalism, another difficulty arises: Employing the power of automated temporal reasoning, even in distributed dynamic systems. That means, we must instruct automated access to the required information sources containing the semantics of some dynamic system's necessary information about time. Thus, a machine-processable representation of the information sources is required. The specified

knowledge – using a knowledge representation formalism – must be stored in such a way that it becomes machine-processable, and in a form making it possible for some dynamic system to operate with it. To put it in a nutshell, for some considered domain of discourse (in this context time) of a particular dynamic system, a logic-based specification language and a knowledge-based repository for the required knowledge that allows for automated temporal reasoning is needed.

Throughout this thesis we will see, that there exists no universally applicable temporal formalisms for representing and reasoning about time. In fact, temporal formalisms are always designed with respect to an intended application. Therefore, this thesis is concerned with different temporal formalisms usable for different kinds of temporal reasoning systems whereby one system, an automated appointment scheduling system, is discussed in detail.

1.1 Motivation

So far, the importance of (knowledge-based) temporal formalisms reasonably representing time for temporal reasoning systems is addressed. Furthermore, it is mentioned, that temporal formalisms are always designed with regard to an intended application. Therefore, a suitable application for temporal knowledge representation and reasoning must be chosen. In order to discuss temporal reasoning with regard to future developments, a dynamic system is singled out that can be settle in the Web.

Initially, the Web has been developed as a medium for human-readable documents rather than for data and information automatically processable. Recently, several attempts have been made for having Web contents handled automatically by intelligent systems. Since time undoubtedly plays a fundamental role in any intelligent information system, some proposals are made for applying temporal formalism to the Web. For these reasons, this thesis investigates temporal formalisms additionally with regard to their applicability to the Web, exemplified by automated appointment scheduling.

Appointment scheduling is a good candidate for automation because it is often tedious, iterative, and time-consumption for people. Appointments include detailed temporal constraints on time slots, and on associated priorities and preferences of the participating people. Unlike previous proposals for automated appointment scheduling in the literature, e.g., [49, 122, 179, 178, 162, 200], we envision appointment scheduling as flexible and convenient as possible among an arbitrary number of people which use arbitrary electronic calendars that possibly base on different calendar systems. That is, scheduling and negotiating non-overlapping appointments among arbitrary people living in different cultural, legal, and business environments, frequently in different time zones, and possibly using different calendar systems (e.g., Gregorian calendar or Islamic calendar) shall be automated. People usually feature wide variations in managing calendars: They use different calendar systems and languages, several individual calendar entries, and a wide variety of cultural, legal, and business depending definitions of time spans covering temporal concepts such as lunch-time or easter and other public holidays. Furthermore, the level of privacy and accessibility rights, user-defined preferences and priorities on appointments and on certain registered events, the number of rescheduled or canceled appointments, and additionally, the method of storing, querying, and inserting appointments and/or related information into a calendar vary.

From this briefly introduced difficulty of automating appointment scheduling it becomes clear, that appointment scheduling is a good candidate for discussing knowledge representation of

temporal aspects. On the one hand, both the temporal concepts of different calendar systems and the different structures for specifying appointments must become expressible by a temporal formalism. On the other hand, appointment scheduling is usually concerned only with one time slot dated sometime in the future. Therefore, this reasoning system is not as complex as, for example, a planning system. It will be shown, that calendar and appointment scheduling systems that enforce flexible temporal constraints, and the managing of different calendar systems become possible with temporal knowledge representation techniques. In particular, a logic-based specification language for temporal knowledge representation (i.e., a temporal formalism) that is able to properly represent the various temporal concepts of different calendar systems and the temporal aspects of appointments is required. Furthermore, a large-scaled set of knowledge bases containing the required knowledge about calendar systems expressed in the afore mentioned temporal formalism in a machine-processable form is required.

1.2 Overview of the Thesis

This thesis aims at two objectives: First, a widespread survey of temporal formalisms for knowledge representation in temporal reasoning systems is given. Second, Web-based planning and scheduling systems are considered. In particular, an automated calendar and appointment scheduling system is described, and its temporal aspects are investigated in that the system can be easily realized. This system asks for a temporal formalism for expressing the required temporal knowledge. Since Ohlbach and Gabbay have proposed such a temporal formalism, a calendar logic [144], we use this language for our discussion.

Chapter 2 presents a survey of several temporal formalisms for knowledge representation that have been proposed in the literature, as well from the Artificial Intelligence (AI) community as from the database community. Their ontology, axioms, and models are discussed. Furthermore, their underlying intuitions are considered by their applicability to practice. Generally, two traditions can be figured out, the change-based formalisms and the time-based formalisms. The fundamental difference between these two traditions is the representation of time: In time-based temporal formalisms the representation of time is concerned with nothing more than the mere flow of time, i.e., a constant change unaffected by anything else. In change-based temporal formalisms the representation of time takes place by means of entities (so-called change-indicators) which indicate that some change has occurred. Three well-known representatives of change-based temporal formalisms are considered, the Situation Calculus, the Event Calculus, and Dynamic Logic. Time-based temporal formalisms are presented with regard to their temporal structures, a multidimensional representation for timestamping different time-dependent objects and their associated properties, and representation schemes for calendar systems. Subsequently, we focus on the representation of, and reasoning about time in certain time-dependent systems. Thus, chapter 3 proposes three different scenarios describing different time-dependent applications in the context of the Web. Each of them asks for a temporal formalism usable for temporal knowledge representation and reasoning. Particularly, appointment scheduling, event planning, and budgeting are considered. One of the systems mentioned in chapter 3, the appointment scheduling system, is detailed discussed in chapter 4. In particular, we thoroughly investigating its temporal aspects aiming at the description of a temporal reasoner (i.e., an appointment scheduler), and the description of knowledge bases containing temporal knowledge the appointment scheduler needs for reasoning. The temporal aspects are merely discussed by investigating the basic functioning of an

appointment scheduler, by classifying the structure of appointments for specifying the temporal aspects the appointment scheduler must be able to reason about, and by suggesting a hierarchy of calendar systems for specifying the temporal knowledge that need to be made machine-processable for the appointment scheduler. The temporal formalism used for this discussion is Ohlbach's and Gabbay's calendar logic [144]. This discussion can be used to develop machine-processable knowledge bases containing information sources about calendar systems, and to realize an automated appointment scheduler. We conclude this thesis in chapter 5 by briefly synthesizing the main objectives and results of this thesis. Furthermore, the realization of the discussed appointment scheduling system, and future research aspects are addressed.

2 Temporal Formalisms: A Survey

This chapter surveys several knowledge representation formalisms of time (i.e., temporal formalisms), and discusses their possible fields of application; also on the Web.

The ability to model temporal aspects of the real world is essential for many applications such as appointment scheduling, financial budgeting, disease surveillance, forecasting and planning tasks. Knowledge representation that supports such applications must therefore enable the modeling of time-dependent information and data.

For integrating the temporal dependencies in various applications, and thus, formally characterizing time, there has been extensive research activity on logical formalisms for time, ontological primitives, the representation of action and change, representing different temporal dimensions (i.e., temporal semantics), and temporal representation schemes for calendar systems both in Artificial Intelligence (AI), e.g., [1, 2, 126, 106, 63, 184, 137, 109] and in Database research, e.g., [173, 27, 68, 174, 48, 14]. Since knowledge representation undoubtedly plays an important role on the Web – as in any other information system – recent approaches for adapting several temporal formalism to the Web has been made, e.g., [42, 71, 72, 29, 22, 85, 45, 84, 202].

The major goal of this chapter is to establish an extensive survey and an adequate classification of important temporal formalisms, comprising all crucial aspects of knowledge representation for time-dependent information and data.

This chapter is structured as follows: In the first section, we consider change-based temporal formalisms which focus on time-dependent entities that indicate some change in time has occurred in the modeled reality. Three well-known representatives of such change-based temporal formalisms, the Situation Calculus, the Event Calculus, and Dynamic Logics are recalled. For each of them, we present a possible axiomatization, and discuss their limitations and extensions provided in the literature.

In the second section, time-based temporal formalisms which consider time itself independent of anything that could happen in it are presented. In a first stage, several possible definitions for a formal temporal structure are discussed along with their main advantages and disadvantages. The proposed formalisms mainly differ in the considered (temporal) ontological primitives (i.e., timepoint and time interval, or point and interval for short), and furthermore, how these primitives are defined. Such a temporal structure is the fundamental building block of any time-based temporal formalism. All further considerations base on such a temporal structure as an underlying formal characterization of time. Subsequently, we consider the challenge of preserving multiple past states of objects stored within time-dependent applications such as temporal databases. This leads to the identification of a maximal set of three temporal dimensions (i.e., transaction time, valid time, and event time), and as a consequence thereof, the need for a multidimensional representation of time. Additionally, we review the definition of an object's history (i.e., how an object evolves over time) by using at least one of the above mentioned temporal dimensions. We conclude by considering formal representations of calendar systems that can be formalized by defining each calendar system's associated time units (i.e., temporal granularities), any relationship between pairs of their different included time units. Additionally, combinations of different calendar systems into one conjoint representation scheme are outlined.

Whenever a certain temporal formalism is adapted to the Web, the respective concept is

(briefly) presented.

2.1 Change-based Formalisms

Change-based formalisms for representing and reasoning about time consider time only as important because the world around us changes constantly. That means, within these temporal formalisms the entities which indicate that some change has occurred have primary consideration. These entities are called *change-indicators* [170]. Situation Calculus and Event Calculus in Artificial Intelligence (AI), and furthermore, Dynamic Logic in the field of program verification are well-known change-based formalisms. *Actions* in the Situation Calculus, and *programs* in Dynamic Logics are the respective change-indicators of these calculi. In a similar way, *events* are the change-indicators in the Event Calculus. In general, a change-indicator changes the truth-value of a particular proposition with regard to its domain of discourse. That is, there are starting and ending timepoints during which a particular proposition is true and false, respectively in the underlying domain of discourse.

These three representatives of change-based formalisms (i.e., Situation Calculus, Event Calculus, and Dynamic Logic) are commonly used in AI for purposes such as predicting the effects of actions on a system's state, planning actions, and analyzing the operations of programs performing some task.

Although change-based formalisms have much intuitive appeal, several limitations have been revealed [170]. To overcome these limitations, a wide variety of extensions of the three above mentioned formalisms have been developed by various researchers such as McCarthy, Reiter, Pinto, Shanahan, Kowalski, and Harel. Furthermore, several authors have suggested richer, mostly time-based formalisms for representing time, action, and change such as McDermott, Allen, Vilain, Vila, Galton, Shoham, and Ladkin. Due to time-based formalisms centered within our field of interest, we are not going into detail of change-based formalisms. Only the basic issues along with some well-established refinements and extensions of the above mentioned calculi are considered.

This section is structured as follows: In the first subsection, we present the basics of the Situation Calculus along with a possible axiomatization. Moreover, a review including some further developments on the Situation Calculus is presented. Subsection 2 deals with the Event Calculus. We consider a possible axiomatization of the Simplified Event Calculus, and outline the richer semantics of the (original) Event Calculus in comparison with the Simplified Event Calculus. Additionally, reviews and extensions of this calculus are addressed. Finally, we briefly survey the main aspects of Dynamic Logic. This formalism is more commonly used for program verification than for knowledge representation.

2.1.1 Situation Calculus

The Situation Calculus introduced by McCarthy and Hayes in 1969 [125] has long been a foundation to AI research for representing and reasoning about time, action, and change. It is the calculus of choice for investigating various problems arising when actions and their consequences are considered, e.g., [65, 108]. Later on, it has been taken as a basis for practical work in planning, e.g., [73, 56, 55, 116, 66], explanation, e.g., [166], control simulation, e.g., [53], database updates, e.g., [154, 156, 13] and database transactions, e.g., [96], and in agent programming and robotics, e.g., [168, 60, 88, 119, 21, 117, 19, 120, 84]. Actually, this calculus

is one of the most studied temporal formalisms for making statements about the world in change.

The Situation Calculus is a first-order predicate calculus. A sentence in a first-order predicate calculus is written, for example, in the form $P(x)$. In this case, P is a predicate, and x is the subject represented as a variable. A first-order logic can be useful for creating computer programs, and is additionally of interest to research in AI. Although first-order logic is sufficient for various reasoning tasks, several more powerful forms of logic have been proposed.

In the remainder of this subsection, we consider the basic functioning of the Situation Calculus along with a possible axiomatization. Additionally, we outline its usefulness by briefly illustrating a (toy) application. Finally, a few proposed variances and extensions of this calculus to overcome some drawbacks discussed in the literature are reviewed.

2.1.1.1 The Language of the Situation Calculus To put it in a nutshell, the Situation Calculus represents a *changing world* [165] using a strictly ordered discrete sequence of situations. The Situation Calculus represents a changing world since a situation at a unique timepoint can be changed by an action leading to a succeeding situation in the modeled world. Each situation represents the complete state of the world at a unique timepoint. Thus, one of the basic entities in this calculus is the *situation* viewed by McCarthy and Hayes as a *snapshot* [125] of the world at a certain timepoint. Following this perception, a situation is defined as a complete state of the world at a particular timepoint. Various facts can hold in a certain situation. Such a fact and a set of facts, respectively cannot fully describe the situation within which they are true. An example for such a fact is $shine(\sigma)$ meaning that the sun is shining in situation σ . Furthermore, it is implicitly assumed that functions, mapping situations to situations, represent executable actions rather than making mere assertions about possibly existing times (i.e., timepoints and intervals, respectively), as it is done in time-based temporal formalisms.

The Situation Calculus can be formalized by a specific many-sorted predicate calculus with some reserved predicate symbols and function symbols. The world is formalized in terms of *situations*, *actions*, and *objects* which are the sorts of this calculus. In doing so, situations and actions cover the (application) domain independent sorts. In contrast, objects are used for everything else except for actions and situations, i.e., they represent (application) domain dependent sorts.

Situations are first-order terms denoting finite sequences of actions. In almost the same manner as situations, actions are first-order terms consisting of action functions along with their arguments. Actions are the cause of state transitions. They are the fundamental instrument of change in the Situation Calculus since the Situation Calculus is specifically designed for representing and reasoning about dynamically changing worlds. In this way, any change of the modeled world is a result of some labeled action.

Describing the world at each situation is realized by using *fluents*. Any property of the modeled world that can change over time is a fluent. Fluents are functions and relations, respectively whose truth values vary from situation to situation, (i.e., their truth values depend on a particular situation). The domain of a fluent is a set of situations. Fluents are denoted by function symbols and predicate symbols, respectively having as their last argument some situation term. Furthermore, fluents can be distinguished between fluents whose range is the set of truth values, i.e., $\{\text{true}, \text{false}\}$, so-called *propositional fluents*, and *situational fluents*

whose range is the set of situations itself.

For representing situations, the special binary function symbol *do* is used:

$$do : action \times situation \longrightarrow situation,$$

i.e., $do(\alpha, \sigma)$ performs action α in situation σ .

$do(\alpha, \sigma)$ signifies the resulting sequence from adding some action α (when action α is performed) to the (action) sequence σ . That is, the successor's situation of σ is specified by $do(\alpha, \sigma)$. The starting point of a temporal sequence is denoted by the initial situation σ_0 . Consequently, all possible world expansions start at the situation σ_0 . That means, σ_0 denotes an empty sequence of actions, i.e., all possible world expansions start at the situation σ_0 , and moreover, in the situation σ_0 no action has yet taken place. In this way, a situation corresponds to the history of actions leading from σ_0 to it.

In addition to the reserved binary function *do*, the binary predicate *poss* is reserved:

$$poss : action \times situation,$$

i.e., $poss(\alpha, \sigma)$ expresses that the action α is possible within the situation σ .

Thus, *poss* (i.e., short hand for possible) is a predicate for action preconditions.

Furthermore, a partial order on situations can be defined for the Situation Calculus:

$$< : situation \times situation.$$

$\sigma < \sigma'$ means, that σ' is reachable from σ by a sequence of executable actions, i.e., σ is a proper sub-history of σ' . $<$ has the intended interpretation of successive situations as action histories.

Let us consider the foundational axioms Σ for the discrete, (sequential) Situation Calculus (taken from [111, 155]):

- (1) $\sigma_0 \neq do(\alpha, \sigma)$
- (2) $do(\alpha_1, \sigma_1) = do(\alpha_2, \sigma_2) \Rightarrow (\alpha_1 = \alpha_2 \wedge \sigma_1 = \sigma_2)$
- (3) $\forall P[(P(\sigma_0) \wedge \forall \alpha, \sigma(P(\sigma) \Rightarrow P(do(\alpha, \sigma)))) \Rightarrow \forall \sigma P(\sigma)]$
- (4) $\neg(\sigma < \sigma_0)$
- (5) $\sigma < do(\alpha, \sigma') \iff (poss(\alpha, \sigma') \wedge \sigma < \sigma')$

Axioms (1) and (2) are unique names assumptions for situations and actions. These two axioms forbid finite cycles and merging. The third axiom denotes a second-order induction. Putting this axiom into words, every situation is either the initial situation σ_0 , or every situation must be obtained by successively applying the function *do*. A detailed discussion on the use of the third axiom is given in [155].

Axioms (4) and (5) inductively define $<$. Axiom (4) formalizes that σ_0 cannot be the successor of any situation σ . The last axiom (5) expresses that the action α , if it is executable in the situation σ' (i.e., $poss(\alpha, \sigma')$), leads to the succeeding situation $do(\alpha, \sigma')$. Then $do(\alpha, \sigma')$ is the successor of the situation σ . Additionally, the situation σ must be before the situation σ' regarding to the (action) sequence. That means, an action α is only executable in a particular situation σ , if the action α is possible within this situation. That is, an action need not always be executable in any situation.

In [111] some propositions directly following from Σ are stated. That is, situations are transitive, irreflexive, and furthermore all names for given situations are unique.

The axiomatization of the Situation Calculus, as considered so far, is not the only possible one. It meets the view of situations in Reiter's formalism [158] rather than McCarthy's and Hayes's snapshot view [125]. Considering the presented axiomatization, situations are regarded as the nodes of a tree basing on an initial situation σ_0 . The actions which can occur in some situation σ are the edges branching from the situation σ in this tree.

Further axiomatizations (partly differing from the previously presented one) are, for example, given in [8, 9, 51, 146, 163]. Moreover, modified axiomatizations that have varying emphasis for actions are provided, for example, in [118, 147]. These proposals explicitly specify the domain of situations. In [123], McCarthy gives an axiomatization for the Situation Calculus featuring events as primary case, and usual actions as special case.

Besides these axiomatizations, some abductive logic programming formalization of the Situation Calculus is possible, e.g., [10, 102].

In the introduction of this subsection, several applications of the Situation Calculus are addressed. In order to understand the calculus' utilizability in practice, we exemplarily consider its use in agent programming and robotics as it is, for example, done in [168, 60, 88, 119, 21, 117, 19, 120, 127, 84]. Typical applications for computational agents are, for example, information retrieval, automation of common user activities, and intelligent robotics and Web agents such as search engines.

Let us consider the example 2.1 for an agent system supporting users' scheduling appointments (taken from [117]).

Example 2.1 *Each user has a SCHEDULEMANAGER agent that knows something about the user's scheduling. The underlying theory of simple actions for these SCHEDULEMANAGER agents is the Situation Calculus. For example, someone wants to organize a meeting. Thereupon, an ORGANIZER agent contacts the SCHEDULEMANAGERS of the participating users, and tries to find a matching for an appointment. Adding an appointment to a user's SCHEDULEMANAGER can be expressed in the Situation Calculus as follows by using the axiomatization we have considered previously.*

The considered action is ADDTOSCHEDULE.

$$\begin{aligned} & \text{poss}(\text{ADDTOSCHEDULE}(\text{agent}, \text{user}, \text{period}, \text{activity}, \text{organizer}), \sigma) \iff \\ & \text{agent} = \text{SCHEDULEMANAGER}(\text{user}) \wedge \\ & \neg \exists \text{activity}', \text{organizer}' \text{ SCHEDULE}(\text{user}, \text{period}, \text{activity}', \text{organizer}', \sigma) \end{aligned}$$

This formalization expresses, that it is possible for the agent agent to add the activity activity to the user's schedule in the situation σ assumed that agent is the user's SCHEDULEMANAGER and there is no other appointment at the user's schedule for the considered period σ . Additionally, it can be specified how the action ADDTOSCHEDULE affects the current state of the modeled world. This can be expressed in the Situation Calculus as follows:

$$\begin{aligned} & \text{poss}(\text{ADDTOSCHEDULE}(\text{agent}, \text{user}, \text{period}, \text{activity}, \text{organizer}), \sigma) \implies \\ & \text{SCHEDULE}(\text{user}, \text{period}, \text{activity}, \text{organizer}, \\ & \text{do}(\text{ADDTOSCHEDULE}(\text{agent}, \text{user}, \text{period}, \text{activity}, \text{organizer}), \sigma) \end{aligned}$$

In almost the the manner as in the example 2.1, intelligent Web agents are realized using the Situation Calculus for exploiting Web service markup in [127]. That is, the Situation

Calculus is used within Web applications which ask for intelligent agent technology.

2.1.1.2 Review and Extensions Several extensions and refinements for the Situation Calculus have been proposed. Due to space and time restriction, we only consider a sample of them, and refer to [123] for further ones. In this paper, i.e., [123], McCarthy provides a tentative view on the extensions proposed for the Situation Calculus.

Although the Situation Calculus is applicable to a wide variety of problems having to do with actions and their effects, Shoham and Goyal [170] have figured out some limitations: They mention that situations are timepoints. That means, the Situation Calculus is applicable as long as only one action happens at a particular timepoint. But it cannot be used if the world can change spontaneously, if actions have some duration (i.e., if actions are not considered to be instantaneous), if actions are concurrent, or if overlapping actions occur. Furthermore, in the Situation Calculus, the flow of time is always discrete. That means, there is an atomic space between any two adjacent timepoints which does not allow further division (i.e., there are no points between two adjacent atomic timepoints). Having a discrete flow of time, fluents are always assigned to different timepoints. However, if we want to express, for example, ‘turning on the water tap, then the level of water in the bathtub is growing steadily’, it is not expressible in the Situation Calculus since in this proposition a continuous process rather than a discrete one is described.

In order to overcome some of the above mentioned limitations specified in [170], Weber [194] presents, for example, an extension of the Situation Calculus allowing for concurrent actions and dates.

Further criticisms on the Situation Calculus are asserted by Kowalski [101]. He sees some disadvantages of the Situation Calculus in the fact that actions need to be totally ordered, and moreover, the considered information must be assimilated in chronological order. Therefore, he also provides some extensions in [101]. Yet, another suggestion dealing with similar argumentation to the one made in [101] has been made, for example, by Lin and Shoham in [113].

According to Raimos’ review of the Situation Calculus [149], another drawback of the Situation Calculus concerns the fact, that any situation is completely characterized by a set of fluents, each of them, in turn, holding in a particular situation. The limitations discussed within [149], are tendentially overcome by introducing *states* (like situations and actions) as (application) domain independent sorts of the Situation Calculus’ language. This extension is called the Situation and State Calculus [149]. Furthermore, Raimos provides a set of axioms relating this newly introduced sort with the already existing sorts.

An interesting extension of the Situation Calculus has been suggested by Pinto and Reiter [145] by adding a (virtual) time-line along with an explicit notion of time to the Situation Calculus. This extension allows for the same representational features as the ones of some time-based temporal formalisms.

Yet another extension for the Situation Calculus is Pinto’s and others’ suggested Probabilistic Situation Calculus [121]. This extension addresses the problem of having domains in which actions can occur non-deterministically with some probabilistic outcomes.

For all that, it is not at all clear, that the Situation Calculus suffers from the drawbacks and limitations we have previously recalled. Other researchers, e.g., [65, 112, 124, 145], have figured out that its expressiveness is considerably richer than it seems from the previous

discussion. In fact, they argue that the Situation Calculus, as it is, is suitable to handle most of the above considered cases.

To summarize the considerations within this subsection, the Situation Calculus is a first-order predicate calculus (with some additionally second-order features) which allows for expressing dynamically changing worlds. All changes of the modeled world are results of labeled actions. That means, the world changes when an action transforms one situation (of the modeled world) into another. In this way, a possible world history is formed. Such a history is nothing more than a mere sequence of actions represented by the given situations. Beyond the basic functioning of the Situation Calculus, we have reviewed a discussion on the Situation Calculus concerning its possible limitations along with a few proposed extensions. Furthermore, we have seen that it is however a calculus of choice for agent programming and robotics. The Situation Calculus is nowadays rediscovered for agent programming on the Web, e.g., [127, 84]. In [127], semantic Web services using agent technology that exploits the proposed Web service markup are discussed. The agent technology employed in [127] is realized by utilizing the Situation Calculus.

2.1.2 Event Calculus

Another change-based formalism for representing and reasoning about *actions* (or *events*) and their effects is the Event Calculus. It has been originally introduced by Kowalski and Sergot in 1986 [103]. Initially, this calculus has been formulated as a logic program [103], i.e., it uses the Horn clause subset of the predicate calculus augmented with negation-as-failure [103]. However, the calculus is now formulated in a number of different, but related languages, some of them using logic programming whereas others are presented in terms of classical, modal, or specialized logics. We refer to Miller's and Shanahan's paper [131] which presents various axiomatizations for the Event Calculus basing on classical logic.

Informally, the basic idea of the Event Calculus is that *fluents* (i.e., time-dependent properties of the world) are true at particular timepoints. That means, such a fluent must have been initiated by some action occurrence at an earlier timepoint, and not been terminated by another action occurrence in the meantime. Similarly, a fluent is false at a particular timepoint, if it has been previously terminated, or not initiated afore. For signifying which actions initiate and terminate which fluents underlying various situations, and to state which actions occur when, (application) domain dependent axioms are provided. In the Event Calculus, action occurrences are often referred to as *events*.

In this subsection, we consider the basics of the (original) Event Calculus, as introduced in [103]. Furthermore, a possible axiomatization for the Simplified Event Calculus [101] is presented. We outline its differences from the original calculus, and additionally emphasize its utilizability in practice by outlining a possible use case scenario. Finally, some variants and extensions of the Event Calculus are discussed.

2.1.2.1 The Simplified Event Calculus Turning attention to the basics of the Event Calculus, its main feature is a temporal structure which is independent of any action occurrence and event occurrence, respectively. The flow of time of its temporal structure is usually stated being linear. Nonetheless, the considered flow of time can also be branching (i.e.,

partially ordered). However, it has been pointed out, e.g., [37, 132], that a non-linear flow of time can lead to various problems. For example, particular events can be left unrelated.

In addition to the temporal structure, the ontological primitives, as introduced in the original Event Calculus [103], and in its further variants [161], are events initiating and terminating time intervals over which fluents hold. The Event Calculus allows for representing simultaneous events. Furthermore, this calculus contains rules that enable the derivation of implied events from incomplete information about explicitly given events. That means, the original Event Calculus allows for managing incomplete information. Nonetheless, all considered changes in the Event Calculus are discrete. Therefore, this calculus has been extended for representing continuously changing events, e.g., [157, 165].

Because of its great expressiveness, the Event Calculus originally presented in [103] turns out to be too complicated. A much simpler variant of the original Event Calculus, the Simplified Event Calculus has been suggested [101]. It was initially used for database updates. This simplified version of the Event Calculus has been shown to be more useful in practice. Actually, the Simplified Event Calculus has been applied to problems, such as planning, e.g., [37], database updates, e.g., [101, 54], explanation, e.g., [164], and temporal reasoning, e.g., [146].

Let us consider an axiomatization of the Simplified Event Calculus (taken from [161, 164, 165]). Further details of the original Event Calculus are given in [103, 161].

For describing the Simplified Event Calculus, five different predicates besides *equals* are necessary:

(1) *HoldsAt* : *fluents* \times *timepoints*

whereby *HoldsAt*(ϕ, θ) means, that fluent ϕ is true at the timepoint θ .

(2) *Happens* : *events* \times *timepoints*

whereby *Happens*(η, θ) means, that event η occurs at the timepoint θ .

(3) *Initiates* : *events* \times *fluents* \times *timepoints*

whereby *Initiates*(η, ϕ, θ) means, if event η occurs at the timepoint θ , then it initiates the fluent ϕ .

(4) *Terminates* : *events* \times *fluents* \times *timepoints*

whereby *Terminates*(η, ϕ, θ) means, if event η occurs at the timepoint θ , then it terminates the fluent ϕ .

(5) *Clipped* : *timepoints* \times *fluents* \times *timepoints*

whereby *Clipped*(θ_1, ϕ, θ_2) means, that the fluent ϕ terminates between the timepoint θ_1 and the timepoint θ_2 .

The corresponding definitional axioms are:

(6) $HoldsAt(\phi, \theta) \leftarrow Happens(\eta, \theta_1) \wedge Initiates(\eta, \phi, \theta_1) \wedge (\theta_1 < \theta) \wedge \neg Clipped(\theta_1, \phi, \theta)$

(7) $Clipped(\theta_1, \phi, \theta) \leftarrow Happens(\eta, \theta_2) \wedge Terminates(\eta, \phi, \theta_2) \wedge \neg(\theta < \theta_2) \wedge \neg(\theta_2 < \theta_1)$

These two axioms can also be expressed with one core axiom, as it is done in [161].

In general, these two axioms state that some fluent ϕ initiated by an event η persists until it is terminated by some other event η_1 . Let us have a closer look at the axioms (6) and (7): Within this calculus, timepoints are ordered by a strict ordering relation (i.e., $<$). The predicate *HoldsAt*(ϕ, θ) expresses that the fluent ϕ holds in the timepoint θ . *Initiates*(η, ϕ, θ_1) expresses that the event η initiates at timepoint θ_1 a time interval during which the fluent

ϕ holds. Finally, $Terminates(\eta, \phi, \theta_2)$ states that event η terminates at the timepoint θ_2 an ongoing time interval over which the fluent ϕ holds. The occurrence of the event η at a certain timepoint θ_1 is represented by $Happens(\eta, \theta_1)$. To summarize, an event η can initiate and terminate, respectively a fluent ϕ depending on the action joint with the event η . Furthermore, negation-as-failure is expressed by the \neg (not) operator. In this way, the first axiom (i.e., $HoldsAt(\phi, \theta)$) expresses that the fluent ϕ is true at the timepoint θ . In particular, the fluent ϕ is initiated by a certain event η at a timepoint θ_1 preceding the timepoint θ . Furthermore, ϕ is not terminated between the timepoints θ_1 and θ (i.e., $\neg Clipped(\theta_1, \phi, \theta)$).

Turning attention to the second axiom, the predicate $Clipped(\theta_1, \phi, \theta)$ expresses that the fluent ϕ is terminated between the two timepoints θ_1 and θ . Using negation-as-failure in $Clipped$'s definition guarantees a correct working of $HoldsAt$, even if events and timepoints are only partially ordered, or the mapping is only partly known. In principle, the predicate $Clipped$ expresses the terminating behavior of some considered fluent ϕ .

In addition to the axioms (6) and (7), the predicates $<$, $Initiates$, $Terminates$, and $Happens$ must be axiomatized. The definition of $<$ satisfies the integrity constraints irreflexivity, transitivity, and antisymmetry, i.e., $<$ a strict order relation. Furthermore, $Initiates$, $Terminates$, and $Happens$ are defined by (application) domain dependent axioms. Rather than giving some definition for $<$ and $Happens$, alternatively, these predicates can be left undefined.

It has to be mentioned, that the aboved given axiomatization of the Simplified Event Calculus is not the only one. Further axiomatizations are, for example, provided in [50, 164, 101, 132, 131].

The Simplified Event Calculus, as described above, is sufficient for dealing with complete information about the participating events. That is, all event occurrences are explicitly and completely given. In [161], Kowalski and Sadri illustrate that this formulation of the calculus can yield incorrect results when having only incomplete information about events. In contrast, the original version of the Event Calculus introduced in [103] can handle incomplete information about explicitly given events. Thus, the original calculus enables the handling of incomplete information, but the simplified version does not allow for this. Another main difference between these two calculi exists: The Simplified Event Calculus is explicitly concerned with fluents holding over timepoints whereas in the original Event Calculus fluents hold as well over timepoints as over time intervals. A detailed comparison of the Simplified Event Calculus with the original Event Calculus is presented in [161]. In this article, Sadri and Kowalski also give some theoretical theorems explaining why the simplified form of the Event Calculus has gradually replaced the original one in practice [161].

So far, we have considered the theoretical foundations of the Simplified Event Calculus, and have addressed its applicability in practice. In the introduction of this subsection, some references to applications of the Event Calculus are given. This calculus is, for example, applicable to network protocol specification and execution (i.e., a communication language for intelligent agents), as it is proposed in [202]. In the example 2.2, some commitments using the Event Calculus are given. Example 2.2 is taken from [202].

Example 2.2 *The following commitment protocols are stated in the context of some electronic commerce application discussed in [202]. They are specified in the Event Calculus. Initially, a base-level commitment $C(\text{MERC}, \text{CUST}, \text{DELGOODS})$ is given, specifying that the merchant MERC is committing to the customer CUST that goods will be delivered (DELGOODS). Additionally, the conditional commitment $CC(\text{MERC}, \text{CUST}, \text{PAIDMONEY}, \text{DELGOODS})$*

is given, specifying that the merchant MERC will commit to send the goods only if the customer CUST commits to pay the demanded money.

Several formal representations in terms of the Event Calculus for the operations that can be performed to create and manipulate the above described commitments are given in [202]. In the following, the formalization of $Create(e(x),c)$ establishing that the commitment c is reproduced, i.e., when x performs the event e , the commitment c is initiated (x and y denote agents, e denotes an event, and $e(x)$ denotes that event e is performed by agent x) is illustrated:

$$Create(e(x),C(x,y,p)):\{Happens(e(x,t) \wedge Initiates(e(x),C(x,y,p),t))\}$$

When considering the example 2.2 it becomes clear, that the Event Calculus can actually be employed for specifying a communication language for intelligent agents – for example, within Web-based multi-agent systems.

2.1.2.2 Review and Extensions Having considered the basics of the Simplified Event Calculus along with a possible axiomatization, we now turn attention to a few variants and extensions both of the Simplified Event Calculus and of the Event Calculus.

The most frequently cited criticisms of the Event Calculus are the problems that have been pointed out by Pinto and Reiter [146]. The discussed drawbacks are simply listed below without further annotations. A detailed consideration is presented in [146]. Pinto and Reiter argue that some problems arise because the Event Calculus allows for handling time intervals. Further problems are caused by the use of negation-as-failure, especially, when applying negation-as-failure in the presence of incomplete information about the temporal relations between events. Furthermore, concurrency of events can lead to problems.

Several extensions are provided by Shanahan, e.g., [165, 167]. Within these proposals, the problems described in [146] are avoided by eliminating time intervals from the calculus altogether. In his former paper [165], Shanahan introduces a variant of the Simplified Event Calculus along with a suitable extension allowing for continuous changes. As well the Event Calculus of Kowalski and Sergot [103] as the simplified version merely deal with discrete change. In this way, it is not at all clear how to formalize in the Event Calculus that a certain event invariably follows another certain event, or that a certain event occurs when some fluent holds. Shanahan argues that handling this and similar situations (i.e., continuous change) become necessary for formally representing commonsense physics (e.g., filling and emptying a vessel). In his later paper [167], Shanahan suggests an Event Calculus for handling domain constraints, concurrent events, and events with non-deterministic effects. However, this extension is based on an Event Calculus loosely corresponding to the original Event Calculus. Further criticisms on the Event Calculus are made by Miller [130]. This criticism refers to planning problems. In order to allow for planning, Miller presents a domain independent axiomatization of the Event Calculus by using a sorted predicate calculus.

Kowalski himself together with Sadri have proposed a further variant of the Event Calculus, the so-called New Event Calculus [161]. This version of the Event Calculus is basically the Simplified Event Calculus augmented with integrity constraints. For having the special case in which all event occurrences are explicitly and completely given, they have chosen the Simplified Event Calculus as a basis for their new proposal. Furthermore, they explain that the newly introduced integrity constraints can be used for dealing with information about event occurrences without knowing their absolute time. Thus, they avoid the drawbacks of

the Event Calculus (concerning its complexity), and they enable incomplete information handling in the simplified version by additionally making integrity constraints available.

Besides the considered extensions, further investigations on variants and extensions of the Event Calculus are proposed by Montanari and others in [24, 26, 30]. Along with these considerations, they provide a modal extension of the Event Calculus for computing current, necessary, and possible maximal time intervals appearing in a possible relative, partial, or incremental ordering of events [25].

To summarize, the original Event Calculus is a quite powerful formalism for representing and reasoning about events and actions affecting some change since it allows for dealing with incomplete information and partially ordered events. Its vocabulary is primarily concerned with time intervals over which fluents hold rather than with timepoints. For practical reasons, the Event Calculus has been gradually replaced by some simplified version, the Simplified Event Calculus. The Simplified Event Calculus allows for handling complete information of events only rather than incomplete information. However, handling incomplete information of events is possible in the Event Calculus. In this subsection, we have considered a possible axiomatization of the Simplified Event Calculus. Additionally, the practical use of the Simplified Event Calculus has been addressed by considering its possible use as a communication language for intelligent agents. That is, the Event Calculus is used for developing multi agent systems in distributed environments such as the Web for tasks such as planning and electronic commerce, e.g., [91, 202]. Finally, discussions along with proposed extensions as well of the Event Calculus as of the Simplified Event Calculus have been recalled.

2.1.3 Dynamic Logic

Dynamic Logic is a composition of three (complementary) classical formalisms: Propositional and predicate logic, modal logic, and the algebra of regular events. Dynamic Logic enables verification of imperative programs and program specifications. It has been introduced by Pratt [148] (emphasizing the modal nature of program interactions), Harel [77, 78], and Moore [135].

Modeling programs as modal operators whereas programs are integrated in an appropriate language is the basic idea of Dynamic Logic. In Dynamic Logic, explicit syntactic constructs called *programs* are introduced. Programs change the values of variables which causes changes of the truth value of some formula. For example, the program $x := x + 1$ over the integer numbers changes the truth value of the formula ‘ x is even’. Consequently, the change-indicators of Dynamic Logics are programs [170]. That is, applying some change-indicator to a program’s state has the effect of changing it into another state regarding to the underlying program.

Although Dynamic Logic has turned out to be a formalism for reasoning about some actions of (natural or artificial) systems in general, it has been originally introduced as a formalism for program verification. Actually, it is the only logical formalism the research on program verification also used in AI. In AI, Dynamic Logic has been adapted for tasks such as the description of actions in the common-sense world, or for specifying some particular reasoning system. In [129], Meyer gives a survey on the usefulness of Dynamic Logic for AI. Among the numerous formalisms for (formal) reasoning about programs, Dynamic Logic enjoys the singular advantage of being strongly related to classical logic.

In this subsection, we briefly survey Dynamic Logics by considering a simple representative of this logical formalism, Propositional Dynamic Logic.

2.1.3.1 Propositional Dynamic Logic In the following, we give a brief overview on the elementary form of Propositional Dynamic Logic proposed by Fisher and Ladner [57]. The Propositional Dynamic Logic is a simple, but widely employed formalism as well for program verification as for knowledge representation, e.g., [177, 171].

Informally, Propositional Dynamic Logic provides means for reasoning about programs and actions in general allowing for changing situations. For example, if α denotes an action, we can express that performing α in a particular situation always results in ϕ being true by the formula $[\alpha]\phi$. These situations and actions can be visualized by graphs. The nodes of this graph represent situations and states of the world, respectively, and the edges of this graph denote actions. Consider the following graph:

$$\sigma_1 \text{---} \alpha \text{---} \sigma_2.$$

That means, performing action α , σ_2 can be reached from state σ_1 .

Beyond this, Propositional Dynamic Logic does not only allow for handling atomic actions with no internal structure. It also provides various operators for building complex actions out of simpler ones (as it is explained in definition 2.1).

In the following, we give a formal specification of Propositional Dynamic Logic (cf. definitions 2.1 and 2.2), as defined by Fisher and Ladner [57]. The language of Propositional Dynamic Logic contains two sorts of expressions, programs and formulas. For each of them, in turn, a set of basic (or atomic) expressions is introduced. Let ϕ_0 denote the set of atomic formulas (i.e., propositional variables), and let σ_0 denote the set of atomic programs (i.e., indivisible statements in a programming language). For simplicity the constant \perp (false), and \top (true) are included.

Definition 2.1 *The set of programs Σ is inductively defined as follows:*

1. *Atomic programs are programs.*
2. *If α and β are programs, and ϕ is a formula, then*
 - (a) *$(\alpha ; \beta)$ (sequential composition),*
 - (b) *$(\alpha \cup \beta)$ (nondeterministic choice),*
 - (c) *α^* (iteration), and*
 - (d) *$\phi?$ (test)*

are programs.

- (a) The action which consists of first executing α and then β .
- (b) Denotes a nondeterministically choosing between doing α or β .
- (c) Denotes an arbitrary finite number of α actions, i.e. doing α , 0, or more times.
- (d) We want to be able to test whether certain things hold in a given situation. Thus, for some formula ϕ , $\phi?$ denotes the act of testing for ϕ .

Definition 2.2 *The set of formulas Φ is inductively defined as follows:*

1. *Atomic formulas, \top , and \perp are formulas.*
2. *If ϕ and φ are formulas, and if α is a program, then $(\phi \vee \varphi)$, $\neg \phi$, and $\langle \alpha \rangle \phi$ are formulas.*

To put it into words, $\langle \alpha \rangle \phi$ means that it is possible to run program α , and the run of the program α results in assertion ϕ being true.

It has to be mentioned, that Propositional Dynamic Logic, as previously formalized, is just a simple version of Dynamic Logic. There are by now quite a number of books and surveying papers on Dynamic Logic. The reader is referred to [79] which contains a recent survey on various Dynamic Logic formalisms.

In this subsection, we have outlined the logical foundations of Propositional Dynamic Logic by giving a formal definition of programs and formulas. It is a simple representative of Dynamic Logics.

Up to this point, we have considered change-based temporal formalisms for representing and reasoning about time and change. The most notable issue on these temporal formalisms is that time is not explicitly introduced. In fact, time is implicitly described by change-indicators affecting a transition from one state in the modeled world to another. In this section, we have reviewed the basic concepts of three well-established representatives of change-based temporal formalisms: the Situation Calculus, the Event Calculus, and Dynamic Logic. At first, the Situation Calculus, and the Event Calculus have been considered. For each of them, a formal axiomatization has been presented, and each of them has been discussed regarding to further refinements and extensions. Furthermore, the usability of these two calculi for Web applications – mainly for developing intelligent agents – has been addressed. Finally, a brief survey of the basic functioning along with a formal representation of a simple representative of Dynamic Logics, Propositional Dynamic Logic, has been presented. Dynamic Logic is more common in use for program verification than in use for knowledge representation.

We now turn our attention to temporal formalisms dealing with an explicit notion of time over which various assertions can be made (i.e., time-based formalisms). The time-based formalisms have a more general framework for representing knowledge about time, action, and change.

2.2 Time-based Formalisms

Philosophy has the longest tradition in developing time-based temporal formalisms. A survey on these formalisms is given in [11]. In AI, the best known temporal formalisms are the ones introduced by McDermott [126] and Allen [1]. They establish the two main temporal ontological primitives, the timepoint and the interval (i.e., a period of time). Since these first proposals much work has been done both in AI, for example, by Shoham, Ladkin, Vila, and Vilain and in the database community, for example, by Snodgrass, Soo, and Lin. Time-based temporal formalisms are nowadays applied, for example, to planning, e.g., [3, 126, 151, 76]

(more references are given by the AAI organization¹), and to (temporal) databases, e.g., [173, 205, 64, 22].

The fundamental difference between change-based temporal formalisms and time-based temporal formalisms is the representation of time: Starting in a time-based temporal formalism with a precisely defined temporal structure, change is manifested when propositions become true or false at different times with regard to the flow of time. In this way, change is simply specified by the fact that the truth value of a proposition changes over time. The time-based temporal formalisms are concerned with nothing more than the mere flow of time, i.e., a constant change unaffected by anything else.

Although most proposals for temporal formalisms support the needs of specific applications, or groups of similar applications, on a more abstract level, the functionalities provided by these temporal formalisms have indeed noteworthy similarities comprising the following temporal features, e.g., [68, 100]:

- At first, every temporal formalism has a **temporal structure** providing the underlying ontology and domains for time. Within the temporal structure, different components such as the temporal ontological primitives (e.g., timepoints and intervals) and the flow of time (e.g., if time is considered been linear or branching) have to be specified. The temporal structure describes the fundamental issues of time by defining its various aspects such as the characteristics of the flow of time. Therefore, the temporal structure forms the ontological basic building block of any temporal formalism.
- The second building block of a temporal formalism is a **temporal history**. In general, a temporal history of an object is a representation of different versions (i.e., states) of an object as this object evolves over time. A version represents the actual appearance of an object at a certain time. Properties of any obtained version in an object's history can be, in turn, associated with a temporal dimension that can be independent of the version's temporal dimension. This leads to a multidimensional temporal model. Any temporal dimension is specified by a particular temporal structure. These two temporal building blocks (i.e., the temporal structure and the temporal history) together allow for a wide variety of means for specifying the occurrence of time. They provide temporal frameworks within which it is possible to represent all sorts of things that can go on in time, but they do not on their own provide means for representing those things themselves.
- The third building block of a temporal formalism is a **representation scheme** for calendar systems in which the temporal structure can be made human readable and usable by formally describing the characteristics of calendar systems. This is, for example, done in [175, 28, 180, 17, 143, 98, 142].

In this section, we give an overview on the proposals concerning the three temporal building blocks of a temporal formalisms, i.e., the temporal structure, the temporal history, and a representation scheme for calendar systems. In the first subsection, we review various temporal structures that mainly differ on the ontological primitives they support. That is, some temporal formalisms only support timepoints, some other only support time intervals, and again others support as well timepoints as intervals. The second subsection deals with temporal

¹<http://www.aaai.org/AITopics/html/planning.html>

dimensions, in particular, transaction time, valid time, and event time that can be associated with any modeled object, leading to a formal representation of an object's history. Additionally, we look at the usefulness of these temporal dimensions for managing time-dependent data and information on the Web. Finally, we present possible representation schemes for calendar systems. They aim at temporal formalisms which catch the semantics of temporal concepts of calendar systems (e.g., Gregorian calendar, Islamic calendar).

2.2.1 The Temporal Structure

The temporal structure has generated much debate, primarily in AI, e.g., [83, 106, 185, 110, 186] mainly with regard to the ontological primitives used for the representation of time from quite theoretical domains such as temporal logics. In this subsection, we present the most important results of this discussion.

For constructing a temporal formalism, several decisions have to be made about the temporal structure. Every temporal formalism refers to a set of temporal elements related by one, or a set of temporal relations. For the temporal elements different ontological primitives (i.e., basic ontological concepts of time) can be considered. The main candidates of (temporal) ontological primitives are timepoints and intervals (i.e., periods of time). Furthermore, the temporal structure determines the desired properties (e.g., the ordering, dense vs. discrete, bounded vs. unbounded) of the chosen ontological primitives referring to the flow of time. After having decided on the ontological primitives together with their basic relations (e.g., $before(x, y)$ is a basic relation between two timepoints expressing that the timepoint x is temporally before the timepoint y), some structure can be defined hereon. That means, axioms describing the behavior of the temporal relations have to be provided. In this way, the considered ontological primitives along with their relations and an appropriate axiomatization form a temporal structure.

Early work on temporal formalisms have centered around two structural models of time [11]: In one model, a linear flow of time is assumed. That means, time progresses from the past to the future in a totally ordered manner. In the other model, one allows for a branching flow of time with many possible future models, i.e., time is only partly ordered.

Subsequent discussions on the temporal structure mainly refers to the ontological primitives available in a temporal formalism: What can be denoted as a timepoint, and the relationships between timepoints and intervals seem to be the most notable and controversial issues. In particular, the discussion is about whether or not one can imagine a timepoint as some infinitesimal interval, whether or not intervals are sets of timepoints, and whether or not propositions can be true at some single timepoint [80].

In addition to the ontological primitives, further decisions for the representation of temporal knowledge must be made [100]: Is time anchored, or not? (Anchored temporal information can be exactly located on the underlying time-line, e.g., 20 June 2002 whereas unanchored temporal information has a known length, but no specific starting and ending anchor points, e.g., 'A lasted for a week'). Are indefinite temporal relations enabled in the temporal formalism (e.g., 'A occurred before B')? Can we represent metric temporal relations (e.g., 'A starts two minutes after B')?

In this subsection, we review a couple of temporal formalisms differing from the ontological primitives they support. In a first paragraph, possible temporal structures for point-based

temporal formalisms are presented. The second paragraph contains a discussion on interval-based temporal formalisms starting from Allen's temporal formalisms [1]. In the third paragraph, an overview of formalisms having as well timepoints as intervals as ontological primitives is given. Finally, we address to non-convex intervals allowing for the representation of recurrent intervals, and intervals with gaps.

2.2.1.1 Point-based Formalisms The idea that timepoints should be the primitives in a temporal ontology has been influenced by physics. In physics it is common to model time as an unbounded, ordered continuum of timepoints that is structured as the set of the real numbers [140], that is unbounded, and that has an order relation (i.e., $<$). The researchers in this tradition are McDermott [126], Shoham [169], and Galton [63]. Not only researchers from the AI community, but also researchers in temporal databases uses this assumption, as it is, for example, done in [128].

There are some simple, but yet powerful arguments for this view. That is, timepoints are important for modeling continuous change, e.g., [126, 63]. For example, if one throws a ball into the air, then a point of its trajectory depends on a timepoint. In addition, if intervals become necessary, they can be introduced as ordered pairs of timepoints, as it is done, for example, in [126, 106].

Following this argumentation, McDermott considers time as a left-linearly and right-partially ordered (i.e., branching into the future), unbounded, and dense collection of timepoints [126]. Two kinds of temporal properties are introduced in this temporal formalisms, *facts* and *events*. Facts are interpreted over timepoints, and events are interpreted over intervals. McDermott takes timepoints as ontological primitives, and defines intervals as ordered pairs of timepoints (i.e., an interval is defined by its two ending points). Thereupon, one can define the holding of a fact over a particular timepoint, or the happening of an event over a particular interval. For this purpose, some predicates are introduced to reasoning about facts and events. This branching time model is especially successful when applied to planning problems, e.g., [126, 36, 192]. We refer to [20] for further considerations of the point-based formalism over partially ordered (i.e., branching) time.

Let us consider the basics of point-based temporal formalisms. These formalisms are defined over a set of timepoints together with an ordering relation. Timepoints can be defined as some durationless pieces of time. Thus, a timepoint is a unique point in time. Alternatively, a timepoint can be identified as some time not distinguished between its beginning time and its ending time.

Formally, a point-based temporal formalism is defined over a structure $\langle \mathbf{T}, < \rangle$. In this case, \mathbf{T} denotes a set of timepoints along with the attributive ordering relation $<$. Additionally, this temporal structure has the following properties:

- **Ordering.** For timepoints ordering there must be at least a partially ordered set consistent with the following properties:

$$\begin{aligned} (\textit{irreflexivity}) \quad & \neg(x < x) \\ (\textit{antisymmetry}) \quad & ((x < y) \wedge (y < x)) \implies (x = y) \\ (\textit{transitivity}) \quad & ((x < y) \wedge (y < z)) \implies (x < z) \end{aligned}$$

Additionally, one can impose a linear flow of time either only towards the past:

$$(\textit{left-linear}) \quad ((y < x) \wedge (z < x)) \implies ((y < z) \vee (y = z) \vee (z < y))$$

or in both directions (i.e., towards the past and the future):

$$(linear) \quad (x < y) \vee (x = y) \vee (y < x)$$

- **Boundness, or unboundness.** The timepoints ordering structure may have a beginning and an end (i.e., it is bounded), or have no end towards the past and the future (i.e., unbounded):

$$(right-bounded) \quad \forall x \exists y (y < x)$$

$$(left-bounded) \quad \forall x \exists y (x < y)$$

- **Dense or discrete.** If there is always a timepoint between any two timepoints then the flow of time is dense:

$$\forall x, y (x < y) \implies \exists z (x < z < y)$$

Discreteness can be expressed as follows:

$$\forall x, y ((x < y) \implies \exists z ((x < z) \wedge \neg \exists w (x < w < z)))$$

$$\forall x, y ((x < y) \implies \exists z ((z < y) \wedge \neg \exists w (z < w < y)))$$

A dense ordering structure of timepoints can be important for modeling continuous changes. Consequently, any extent of time can be partitioned into sub-intervals. This property is of interest for modeling planning problems. In planning systems, tasks are frequently decomposed into subtasks. Another consequence is, in a dense time structure it is not possible to refer to some previous and to some next timepoint, respectively. In contrast, in a discrete temporal structure some next timepoint can be expressed. It has to be mentioned, that any finite strict partial order is automatically discrete.

The properties introduced above are sufficient to achieve a certain level of completeness. Van Benthem shows, that both the *unbounded dense linear* formalism and the *unbounded discrete linear* formalism are syntactically complete [11]. That means, assertions using one of these formalisms can be deduced from the axioms related this formalisms.

Let \mathbf{T} be a set of timepoints, and $<$ a strict total order on the set \mathbf{T} . This set consists of three natural relations between pairs of timepoints: $<$ (before), $=$ (equals), and $>$ (after). These are the three basic relations between pairs of timepoints which allow for expressing eight (i.e., two timepoints participate on one relation: $2^3 = 8$) disjunctive relations between pairs of timepoints. The eight producible disjunctions out of the three basic relations result in the following relations: $!$, $<$, $=$, \leq , $>$, \neq , \geq , $?$. $!$ is the relation never holding between two timepoints, and $?$ is the relation always holding between two timepoints. These relations correspond to different degrees of knowledge on the involved pairs of timepoints.

Furthermore, a conversion function (i.e., $\mathbf{Conv}: \mathbf{T} \longrightarrow \mathbf{T}$, maps a relation onto its inverse), and a composition function (i.e., $\mathbf{Comp}: \mathbf{T} \times \mathbf{T} \longrightarrow \mathbf{T}$, the composition operation corresponds to the transitive relation entailed by a sequence of two timepoint relations) can be adopted to these eight elements. In this way, a point-based temporal formalism can be represented by a point algebra. This point algebra allows for relating two timepoints by using the three binary relations (i.e., $<$, $=$, $>$). It is described in detail in [189].

As we have seen previously, within point-based temporal formalisms time is described by timepoints located on the underlying time-line which characterizes the flow of time. The flow of time can be specified by various properties (i.e., the ordering, bounded vs. unbounded, dense vs. discrete), and thus, allowing for different representations of time depending on particular modeling intensions. Furthermore, there are three basic relations between pairs of timepoints which can be combined.

However, point-based temporal formalisms do not provide any concept of duration on a time-point although time usually has some duration. Therefore, a general criticism against point-based formalisms is their artificial character [2].

2.2.1.2 Interval-based Formalisms The second ontological primitive over which a formalism of time can be formulated is arbitrarily called time period, time interval, or interval, for short. Examples for temporal intervals are ‘the 18th century’, ‘during the 2002 Pentecost holidays’, and ‘10:00 to 11:35 on May, 13th 2002’. They can be specified as particular pieces of time, located on the underlying time-line. Intervals are main concepts for temporal representation and reasoning since they are the extents of things. That is, propositions are true during them, and an interval describes the lifetime of an object. In general, intervals are not totally ordered, and furthermore, several relations can hold between pairs of intervals. Beyond this, the distinction between dense time and discrete time also runs through interval-based temporal formalisms. Although most of them can be extended by axioms restricting either to discrete time or to dense time, they are often consistent with the assumption that time is dense in some places and discrete in others [80].

Furthermore, an interval can either be anchored or unanchored. For example, ‘Pentecost 2002’ is an anchored interval because it can be exactly located on the underlying time-line whereas ‘one week’, or ‘as long as it takes for the kettle to boil’ are examples for unanchored intervals. Any unanchored temporal information has a known length, but no specific starting point and ending point on the underlying time-line. Thus, an unanchored interval, a so-called duration, is a constant quantity of time that can be compared, added, and scaled [75]. A duration is a property of an interval rather than a property of a timepoint since the duration of a timepoint is zero. Any axiomatization for durations can be added to other time-based formalisms. Various axioms for durations are given in [80]. Including durations into a temporal formalism, it becomes important to distinguish between a dense flow of time and a discrete flow of time since in dense time a moment (i.e., a non-dividable interval) has no duration, but in discrete time it has some.

Interval-based temporal formalisms have been of interest to several researchers in AI such as Allen, Hayes, Hamblin, Ladkin, and Leban. In the beginning, Allen [1] has proposed an interval-based temporal formalism that is considered as fundamental in reasoning about actions and plans in AI. For this reason, we first survey Allen’s interval-based temporal formalism. Subsequently, proposals taking Allen’s formalism as a starting point are addressed.

Allen’s Interval Formalism Allen introduces an interval-based temporal formalism with thirteen binary relations between intervals. These relations describe every possible way of relating two convex intervals over a linearly ordered set [1, 2]. Allen’s temporal formalism is exclusively based on (convex) intervals. Intervals are proposed for providing an intuitive description for any occurrence of events in time. Such an event has a duration, and it holds on an unbounded, linear flow of time. Whenever a temporal relation between two events is uncertain, disjunctions of some of the thirteen basic relations can be used to represent what is known.

Let us now consider a formal representation of Allen’s interval formalism. In this formalism an initial structure $\langle \mathbf{P}, \mathbf{A} \rangle$ is given. \mathbf{P} denotes a set of intervals, and \mathbf{A} denotes the set of the thirteen basic relations on intervals. Intervals are considered as pairs, (a, b) of real

numbers with $a < b$. That is, there is an ordering ($<$) between a and b .

A convex interval i can be depicted as follows (with time going from left to right):

i _____

Then Allen's relations (i.e., the set \mathbf{A}) are illustrated as follows:

- i *equals* j (strict identity of intervals i and j)

i _____
 j _____
- i *meets* j (i is before j and there is no gap between them)

i _____
 j _____
- i *precedes* j (i is before j and there is a non-empty interval between them)

i _____
 j _____
- i *overlaps* j (i starts before j , and j ends after i)

i _____
 j _____
- i *starts* j (i and j have the same starting point, but i ends strictly earlier than j)

i _____
 j _____
- i *during* j (i has a strictly later starting point and a strictly earlier ending point than j)

i _____
 j _____
- i *finishes* j (i and j have the same ending point, but i starts strictly later than j)

i _____
 j _____

Additionally, there are six further relations, the inverse relations to all the above illustrated ones, except for *equals* that is self-inverse. That is, if a binary relation R is given, then the inverse relation R^{\sim} is defined as follows:

$$aR^{\sim}b \iff bRa.$$

These thirteen relations are disjoint, and except for *equals* they are also disjoint from their inverses. That means on the one hand, no pair of intervals is in more than one of these relations, and on the other hand, each pair of intervals is in exactly one of these relations.

In addition to the intervals and their basic relations, a set of axioms, Σ , specifying the behavior of the basic relations is given in [1]:

Let \mathbf{P} be a set of intervals, and \mathbf{A} the set of the thirteen basis relations.

1. Given any interval, for each relation in \mathbf{A} there exists another interval related by this relation to the first interval.
 $\forall p \in \mathbf{P}, r \in \mathbf{A} \exists p'(r(p, p'))$

2. The relations in \mathbf{A} are mutually exclusive.

$$\forall p, p' \in \mathbf{P}, r, r' \in \mathbf{A} (r \neq r' \wedge r(p, p') \implies \neg(r'(p, p')))$$

3. A *transitivity table* is given.

Interval relations are transitive in the following sense:

If intervals a and b are in relation r_1 , and

if intervals b and c are in relation r_2 ,

then we can restrict the set of possible relations for a and c .

The transitivity table gives an overview over all possible inferences between interval relations following the previously described scheme.

The intuition of the above depicted basic relations can also be formalized in another framework, considering a linearly ordered set of timepoints. Consequently, intervals can be constructed out of ordered pairs of distinct and ordered ending points [106].

However, the fundamental idea of Allen's temporal formalism is to treat intervals directly as first-class primitives rather than using some underlying point-set over which intervals can be defined.

Review and Development of Allen's Interval Formalism Allen's definition of the considered thirteen basic relations between pairs of intervals definable by using the natural ordering $<$ on the real numbers has also been studied by other researchers. For example, van Benthem [11] and Ladkin [105, 106], both independently formulated an interval-based formalism over unbounded, dense, and countable categories (i.e., there is only one countable model, up to isomorphism). Van Benthem, on the one hand, formulated his interval formalism simply using two basic relations, *precedes* and *contained-in* which are consistent with the structure of intervals over the rational numbers. His *contained-in* relation is the union of Allen's *starts*, *during*, and *finishes* relations. Ladkin, on the other hand, explicitly defines each of the thirteen basic relations as basing on intervals in terms of the relation *precedes*. Indeed, van Benthem's and Ladkin's formalism are explicitly definable in the other [106]. Thus, these formalisms are mutually inter-definable having almost the same models except for the different set of basic relations.

Another redefinition of Allen's formalism has been provided by Allen himself, and Hayes, as some formal theory of intervals formalized in first-order logics. In [81, 82], they illustrate that all the above illustrated thirteen basic relations can be formulated in terms of the single relation *meets*. For this purpose, they have introduced five axioms. These five axioms define what a (durationless) *meeting-place* between two intervals is, that *meeting-places* are linearly ordered, and ensure that the underlying time-line is unbounded at both ends.

Ladkin relates Allen's and Hayes' *meets*-formalism to other interval-based formalisms, completely characterizes its axiomatizations. Moreover, he proposes a completion in order to obtain an axiomatization of this interval-based formalism. This extension consists of an additional axiom expressing the timepoint's density. In doing so, Ladkin shows that Allen's and Hayes's *meets*-formalism is logically weaker than the formalisms proposed by van Benthem [11] and Ladkin [106]. Ladkin's various reports from the Krestel Institute and his Ph.D. Thesis [106] contain further details of both Allen's and Hayes' *meets*-formalism and Ladkin's previously outlined considerations.

To summarize, intervals are fundamental for any temporal formalism since they are the extent of things. Allen can be seen as the pioneer of interval-based temporal formalisms. Several improvements have been made, mainly by Ladkin, van Benthem, Allen, and Hayes.

Beyond the previous discussion, it has to be mentioned, that as a consequence of his merely interval-based formalism, Allen argues against timepoints [2]. He emphasizes that timepoints are not necessary since they can be represented as very short intervals, so-called *moments*, as introduced in [83]. Moments are very short intervals having some duration, but they are not decomposable. Thus, Allen completely refuses to represent any timepoint.

However, if we have timepoints expressing instantaneous events, then this becomes a difficult task in purely interval-based formalisms. That is, timepoints cannot be represented as mere types of intervals since timepoints and intervals have different properties. Furthermore, timepoints cannot be represented as very short intervals since a short interval does not divide an interval into two meeting points. Allen's and Hayes' proposal for modeling timepoints as moments fails for the same reason. In order to overcome this problem, intervals and timepoints must have the same status as ontological primitives in a temporal formalism. For this reason, we now turn our attention to formalisms treating timepoints equivalent with intervals as ontological primitives.

2.2.1.3 Point- and Interval-based Formalisms Allen's formalism with intervals as the only ontological primitives appears to be more intuitive than point-based formalisms that have been criticized because of their artificial character. Nevertheless, the notion of timepoint is present in our common-sense of time. That is, many situations such as talking about a general proposition being true at a certain timepoint require for an instantaneous representation of time. In doing so, timepoints should have the same ontological status as intervals in a temporal formalism. Another important argument for including timepoints as ontological primitives different from intervals is the fact, that some transition from one state of the modeled reality to another state is usually considered as being instantaneous. Furthermore, expressing accomplishing events (e.g., 'to close the door') [138] and continuous change is only possible in temporal formalisms that explicitly contain timepoints as ontological primitives different from intervals [63]. In order to catch these and similar semantics, we consider in the following a few important temporal formalisms joining the ideas of timepoints and of intervals.

Vilain [188] chooses Allen's interval-based temporal formalism as a starting point, and then he extends this formalism with timepoints, point-to-point, and point-to-interval relations. In particular, this formalism consists of a structure $\langle \mathbf{P1}, \mathbf{P2}, \mathbf{AV} \rangle$. $\mathbf{P1}$ denotes a set of timepoints, $\mathbf{P2}$ a set of intervals, and \mathbf{AV} a set of the following 26 relations:

- Allen's thirteen relations between pairs of intervals,
- five basic relations between a timepoint and an interval denoted with *.precedes*, *.starts*, *.during*, *.finishes*, and *.after*, and the inverses to these relations, and
- $<$, $>$, and $=$, the three basic relations between timepoints.

So far, the resulting formalism and its related problems are not analyzed. Nevertheless, Vilain's paper [188] is the pioneer formalism on mixing an interval-based temporal formalism with a point-based one.

Another revision of Allen’s formalism with the objective to introduce timepoints as ontological primitives in addition to intervals is proposed by Galton [63]. Galton first remarks that formalisms treating timepoints as parts of intervals and intervals as sets of timepoints, respectively raise several problems such as the *Divided Instant Problem*, as illustrated in example 2.3.

For understanding the *Divided Instant Problem*, and therewith, the discussion on temporal formalisms conjoining timepoints and intervals, let us consider example 2.3:

Example 2.3 *Suppose the proposition A represents the fact ‘John is having breakfast’, and $\neg A$ represents the fact ‘John is not having breakfast’. Furthermore, we suppose that John is having breakfast during the period P1. Then John stops at the timepoint i, and he is not having breakfast during the next period P2. Thus, the intervals P1 and P2 obviously meet. If timepoints are parts of intervals and intervals are sets of timepoints, respectively, then we have the problem to decide, whether A is true at timepoint i, or not. That means, we have to decide, if the timepoint i belongs to either*

1. P1, or
2. P2, or
3. both (i.e., P1 and P2), or
4. neither to P1 nor to P2.

(1) and (2) cause no problems. In case (3) A and $\neg A$ would be true at i, but that is inconsistent. The same holds for (4) since as well A as $\neg A$ would be false.

In order to overcome problems such as the *Divided Instant Problem*, the basic idea of Galton’s formalism is that a timepoint only appears at the point when two intervals meet. In return, he postulates the existence of timepoints and intervals, both having the same ontological status. As relations between pairs of intervals he retains Allen’s thirteen basic relations. Additionally, two point-to-interval relations, *Within* and *Limits* are introduced. Thereupon, he provides a rather loose axiomatization also involving the relation *In*. The additional interval relation *In* is defined as a disjunction out of Allen’s relations *during*, *starts*, and *finishes*. In particular, Galton defines the following axioms (timepoints are written with lower case letters, and intervals are written with upper case letters):

- (1) $\forall I \exists i \text{ Within}(i, I)$
- (2) $(\text{Within}(i, I) \wedge \text{In}(I, J)) \implies \text{Within}(i, J)$
- (3) $(\text{Within}(i, I) \wedge \text{Within}(i, J)) \implies \exists K (\text{In}(K, I) \wedge \text{In}(K, J))$
- (4) $(\text{Within}(i, J) \wedge \text{Limits}(i, J)) \implies \exists K (\text{In}(K, I) \wedge \text{In}(K, J))$

This axiomatization provides a temporal formalism with timepoints and intervals, both with an appropriate degree of commitment. For example, holding of some proposition on an interval does not require to decide whether it holds at the limiting timepoints or not since the corresponding limiting timepoint is described by a timepoint independent of an interval. In this way, Galton obviates the *Divided Instant Problem*. However, some criticisms on Galton’s

formalism have been made. For example, Vila complains that this formalism is too weak for a common-sense temporal formalism since it actually accept too many models [183, 184, 186].

Yet another suggestion for a temporal formalism having both, timepoints and intervals as ontological primitives is addressed to Vila [184]. This formalism is inspired by Bochman [18] and Galton [63]. Within this temporal formalism, timepoints are defined from intervals according to the intuition that some interval can be described by its ordered pair of ending points. Thus, some interval is considered as a piece of time on the underlying time-line, beginning at a certain timepoint, and ending at some later timepoint.

Formally, this formalism consists of two different ontological primitives, **I** (i.e., the sort of timepoints), and **P** (i.e., the sort of intervals) constituted by two infinite disjoint sets of symbols, and three primitive binary relation symbols, $<$: $\mathbf{I} \times \mathbf{I}$, $begin$: $\mathbf{I} \times \mathbf{P}$, and end : $\mathbf{I} \times \mathbf{P}$. Vila provides the following first-order logical axiomatization characterizing his formalism **IP**. The subsequent axioms are taken from [184] (timepoints are written with lower case letters, and intervals are written with upper case letters):

- IP1** $\neg(i < i)$
- IP2** $(i < i') \implies \neg(i' < i)$
- IP3** $((i < i') \wedge (i' < i'')) \implies (i < i'')$
- IP4** $(i < i') \vee (i' < i) \vee (i = i')$
- IP5.1** $\exists i'(i' < i)$
- IP5.2** $\exists i'(i < i')$
- IP6** $(begin(i, P) \wedge end(i', P)) \implies (i < i')$
- IP7.1** $\exists i begin(i, P)$
- IP7.2** $\exists i end(i, P)$
- IP8.1** $(begin(i, P) \wedge begin(i', P)) \implies (i = i')$
- IP8.2** $(end(i, P) \wedge end(i', P)) \implies (i = i')$
- IP9** $(i < i') \implies \exists P(begin(i, P) \wedge end(i', P))$
- IP10** $(begin(i, P) \wedge end(i', P) \wedge begin(i, P') \wedge end(i', P')) \implies (P = P')$

Axioms **IP1** to **IP4** describe the properties of $<$ (i.e., the ordering relation). The pair of axioms (**IP5.1** and **IP5.2**) effects unboundness on this ordering relation, and **IP6** orders the ending points of some interval. The pair of axioms (**IP7.1** and **IP7.2**) and (**IP8.1** and **IP8.2**) formalize that the beginning timepoint and the ending timepoint, respectively of an interval always exists and that they are unique. For this purpose, **IP9** and **IP10** ensure the existence and uniqueness, respectively of intervals closing the direct connection between timepoints and intervals.

Vila's above described temporal formalism is convenient for decomposability of intervals, representing instantaneous holding of states, and differ among open/closed intervals. Furthermore, Vila emphasizes, that the *Divided Instant Problem* is adequately obviated.

A development on Vila's formalism is provided by Vila himself, and Schwalb allowing for a distinction between continuous phenomena and discrete phenomena occurring in time [187]. For this purpose, they define a formalism of temporal incidence upon Vila's **IP** formalism. In doing so, a temporal incidence denotes domain-independent properties for the truth-values

of fluents (i.e., propositions describing the state of the world), and events (i.e., propositions describing the occurrences that make the world change) anywhere in time.

Generally speaking, this formalism is based on two basic ideas: It allows fluents to hold at timepoints, and furthermore, a distinction between continuous fluents and discrete fluents is made. For realizing these two basic ideas, two predicates, *holds* and *occurs*, are introduced.

So far, we have seen that there are various temporal formalisms (although we have only considered a few of them) integrating timepoints and intervals, both having the same ontological status. On a more abstract level, two alternatives for this overall concept appear. That is, on the one hand, we can start with an arranged point-interval formalism (i.e., defining intervals as pairs of timepoints), or on the other hand, we can define timepoints from intervals. Against the former choice some semantic problems have been revealed such as the *Divided Instant Problem* [11, 83] leading several researchers turning onto the second way. However, in [183], ontological arguments are stated against those timepoint constructions on interval-based formalisms, and in [184], Vila proposes an integrating formalism following the first concept. That is, we are ending up with a non-homogenous picture of time, having some technical glitches and the certainty that not only intervals but also timepoints are necessary ontological primitives in any temporal formalism. In this way, timepoints are usable for expressing instantaneous events, and intervals are usable for expressing durable events. Furthermore, either point relations or interval relations can be used depending on what is more efficient.

2.2.1.4 Non-convex Intervals Up to now, we have considered convex (i.e., contiguous) intervals containing all their subintervals, and having no gaps. In the above considered formalisms, intervals are represented by ordered pairs, $(a, b) : a < b$ generally of real numbers. For example, the event (i.e., in general a proposition with some duration) ‘John having breakfast’ can be represented by a convex interval. But events can recur (e.g., ‘John has breakfast every morning’), or some event can be interrupted (e.g., ‘John stops his breakfast to answer the phone, and then continues his breakfast’). Furthermore, some event can consist of many related sub-events, i.e., unions of events. Thus, it can be useful to additionally represent gaps for separating either different occurrences of the same interval, or for representing some interruption of a single interval. Numerous examples of temporal reasoning require an abstraction from the number of times an event recur, or the number of times an event occurs in a temporal relation (e.g., scheduling one’s office hours). That is, the main difference between repeating events and single events is the available information such as the number of recurrences, or the interval of an event can be indefinite. In fact, one has to manage more information about non-convex intervals than about convex ones. In particular, planning and scheduling tasks ask for axioms within which often neither the number nor the ordering of events is specified. Various potential applications that require the representation of non-convex intervals are given in [137, 95].

In the following, a survey of the main aspects on representing and managing temporal information about non-convex intervals is given. Furthermore, some work in the research on non-convex intervals is addressed.

Since the idea of explicitly introducing intervals as ontological primitives at least within AI research has been first presented with Allen’s interval calculus over convex intervals [1], Koomen [99], and also Davis and Carnes [35] enhance Allen’s formalism by allowing for a proper representation of periodicity in time. For example, Koomen proposes an application

in which it is necessary to incorporate recurrence of some information into a planning system. More recent work on reasoning with non-convex intervals, and moreover, a model for the temporal structure of repeating events is mainly addressed to Morris and Khatib [136, 93, 137, 94, 95]. Formal applications and systems representing and reasoning about recurrences, and collections of intervals can be found in the context of calendar information used for temporal databases, e.g., [107, 141, 182, 180, 181].

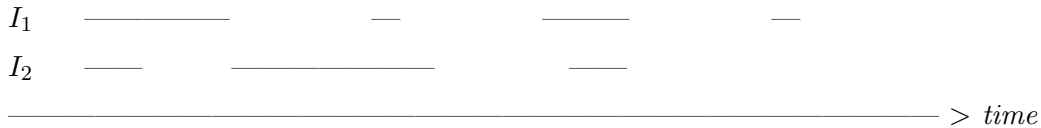
Following the basic ideas on non-convex intervals, Allen's interval calculus is generalized to model non-convex intervals since non-convex intervals are nothing more than mere collections of many continuous intervals. Having some representation for non-convex intervals, we can then reason about gapped events (i.e., single, several times interrupted events), recurring events (i.e., repeatedly occurring events), and unions of events (i.e., events consisting of any related sub-events). The number of events, and therefore, the number of intervals within a non-convex interval can be either finite or infinite. When dealing with finite non-convex intervals, each of them can be seen as a finite collection of (convex) intervals with the following form:

$$I = (a_1, b_1), (a_2, b_2), \dots, (a_n, b_n) : a_j < b_j \leq a_{j+1}, \forall j = 1, \dots, n.$$

The non-convex interval I represents a sequencing set of n convex sub-intervals, a so-called n -interval. The pairs (a_i, b_i) , $\forall i \in \{1, \dots, n\}$ within the interval I are ordered pairs of real numbers; in fact, the (convex) intervals introduced by Allen [1]. Furthermore, this representation enforces a strict linear ordering on I . Thus, each non-convex interval I in a temporal domain is a sequence of (convex) sub-intervals containing gapped events, or interrupted events.

The interval relations between sub-intervals of two non-convex intervals can be modeled within a $n \times m$ - matrix whereby n and m are the numbers of sub-intervals contained in the comprising non-convex intervals [94, 93], as it is illustrated in the example 2.4.

Example 2.4 Let I_1 and I_2 be two non-convex intervals that can be illustrated as follows:



That is, the matrix consists of three columns (the size of I_2) and four rows (the size of I_1). The entry in the second row and the second column is, for example, the 'Allen'-relation between the second subinterval of I_1 and the second subinterval of I_2 which is during, and the entry in the third row and the first column, for example, is the Allen relation between the third subinterval of I_1 and the first subinterval of I_2 which is precedes \sim . Consequently, the matrix relation between I_1 and I_2 has the following form (i.e., it forms a 3×4 - matrix):

$$\begin{pmatrix} \text{starts}\sim & \text{meets} & \text{precedes} \\ \text{precedes}\sim & \text{during} & \text{precedes} \\ \text{precedes}\sim & \text{overlaps}\sim & \text{finishes}\sim \\ \text{precedes}\sim & \text{precedes}\sim & \text{precedes}\sim \end{pmatrix}$$

(whereby the relations within this matrix are some of the relations introduced by Allen; \sim denotes the particular inverse relation)

From the example 2.4 it becomes clear, that within such a matrix binary relations between sequences of intervals are represented. Some entry in row i and column j is the relation between the i^{th} component of one sequence of intervals and the j^{th} component of the other one. For example, the entry *meets* in the first row and the second column inside the matrix in the example 2.4 is the relation between I_1 's first subinterval and the second subinterval of I_2 . Since the components of the sequences in example 2.4 are convex intervals, the resulting entry within the corresponding matrix is a relation between two convex intervals, as defined by Allen [1]. The relations between subintervals contained in non-convex intervals that can be illustrated using a matrix are called internal relations [136]. Besides the internal relations, also external relations can be formalized [136]. The external relations are relations between two non-convex intervals that can be computed by the associated internal relations. For example, the external relation between the two non-convex intervals I_1 and I_2 in example 2.4 is *starts*[~]. Usually, it is sufficient for a pair of non-convex intervals of size n and m , respectively only to consider the positions $(1,1)$ and (n,m) of the related matrix in order to compute the non-convex intervals' relations [136]. Beyond this, Morris and Khatib provide a framework how recurrent relations can be built from ordinary interval relations [136, 93, 137].

Non-convex intervals as ontological primitives have been first introduced by Ladkin in [105], and later in [106]. In order to deal with non-convex intervals, Ladkin proposes a general taxonomy of the set of all relations between any two non-convex intervals in terms of specific relations for general non-convex intervals. Several relations between non-convex intervals are suggested and illustrated by Ladkin [106] derived from applications for task description and management, action formalism, and process formalism. For example, Ladkin suggests the relation *always-meets* guaranteeing that each convex component of a (non-convex) interval J is always met by a certain component of a (non-convex) interval I (e.g., 'after John has had breakfast, he always brushes his teeth'). Further examples for those relations are *disjoint-from* denoting two intervals without any point in common, and *bars* expressing that the union of two sets of points is convex.

Later on, Ligozat expresses Ladkin's relations in the language of generalized intervals subsuming timepoints, intervals, and non-convex intervals in order to allow for reasoning about complex events (i.e., events with more than two crucial timepoints) [109]. However, we do not want to detail Ligozat's work, and therefore refer the interested reader to [109].

From the previous brief introduction into non-convex intervals it becomes clear, that representing gapped events, recurring events, and unions of events is often useful for various applications, in particular when planning and scheduling problems are addressed. For this purpose, non-convex intervals can be regarded as a generalization of convex intervals since non-convex intervals are (finite) collections of (convex) intervals.

There are indeed a good deal of further considerations, discussions, and proposals for a temporal structure not only in the AI community, as in [80, 61, 4, 5, 76, 141] but also within the research on temporal databases: In [196], a bibliography on temporal database research is given, and several links concerning temporal database research are gathered together ². Yet, throughout this survey on a possible temporal structure for temporal formalisms, we have recalled the most important representational issues for representing time-dependent data and information, and reasoning about such data and information. The considered proposals

²http://www.scism.sbu.ac.uk/cios/paul/Research/tdb_links.html

mainly differ from the ontological primitives (i.e., timepoints and intervals) they support. The range on crucial different formalisms in the literature has been discussed. It is widely agreed that not only intervals but also timepoints need to be represented within a temporal formalism. Integrating both timepoints and intervals in a temporal formalism can be done by a conjoint point-interval-based formalism. However, such a formalism may be too complex for many applications, and therefore, the representation of intervals as pairs of timepoints describing their ending points is widely accepted. Furthermore, many applications ask for the representation of non-convex intervals. This is primarily important when planning tasks are addressed.

2.2.2 Multidimensional Time

So far, we have considered various ontological aspects of time (i.e., possible temporal structures) allowing for temporal knowledge representation that is useful for many time-dependent applications such as financial budgeting, medical history, versions in CAD/CAM applications, scheduling, and airline reservations. For such time-dependent applications further temporal dimensions due to preserving multiple past states of some stored object, and thus, allowing for version management are needed. Objects contained in time-dependent applications can be associated with such further temporal dimensions, in particular *transaction time*, *valid time*, and *event time*.

At least two relevant temporal dimensions are well known within the research on temporal databases, valid time and transaction time [90]. The valid time of a proposition denotes the time when this proposition is true in the modeled reality. The transaction time of an object denotes the time the object is current in the underlying time-dependent (physical) application (usually a database). In particular, an object is stored at some timepoint. The particular object is then ‘current’ until it is logically deleted. Consequently, transaction times have some duration rather than being instantaneous. While valid time can only be associated with propositions (i.e., statements that can be true or false), transaction times can be associated with any time-dependent object that is considered in the underlying application. Since valid time and transaction time have varying semantics, they are orthogonal [173]. Therefore, they are referred to as (different) temporal dimensions. Any temporal dimension is specified by a temporal structure serving the semantics of the referenced temporal dimension.

These two temporal dimensions are considered to be a starting point for further investigations concerning temporal database semantics. It has been figured out that they are not sufficient to fully comprise the semantics of every kind of time-dependent information and data [90]. For this reason, additional temporal dimensions, in particular, event time and decision time are proposed, e.g., [27, 139, 90, 68]. In general, event time denotes the time a particular event occurs in the modeled reality. As an effect of such an event, it (abruptly) changes the truth values of certain propositions. Decision time is defined in the same manner.

The above mentioned temporal dimensions lead to a formal representation of the evolution of objects over time. The actual appearance of an object at a certain time is usually called version. A set of such versions forms the temporal history of an object. This facilitates the handling of past versions of an object. Furthermore, an object’s history allows for a straightforward representation of a linear flow of time. In this way, version management for time-dependent applications becomes possible.

Version management is thoroughly investigated within the research on temporal databases,

e.g., [97, 203, 23, 6, 12, 159, 160, 68, 64]. Recently, the above mentioned temporal dimensions have been adapted for a more sophisticated resource management on the Web, e.g., [71, 72, 29, 22, 85, 45].

In this subsection, we first recall the crucial temporal dimensions (i.e., valid time, transaction time, and event time) that have been proposed for modeling time-dependent information in the context of temporal databases. Additionally, the representation of an object's history (i.e., its evolution over time) is addressed. We conclude by describing some approaches that use the above mentioned temporal dimensions for managing multiple versions of Web resources, and furthermore, for manipulating time-dependent information and data contained in Web documents.

2.2.2.1 Temporal Dimensions Numerous and indeed confusing temporal dimensions are proposed for representing additional temporal dimensions, e.g., [34, 115, 172, 173, 62, 92, 201, 90, 27, 86] allowing for several tasks, in particular, for version management.

In the following, the most important temporal formalisms for different temporal dimensions (i.e., different temporal semantics) are briefly reviewed. Additionally, a table (cf. table 1) containing some frequently used terms and synonyms for the considered temporal dimensions is given. Subsequently, we consider how objects evolve over time.

Review Maier and Copland [34] introduce *event time* and *transaction time* as two different types of time. Event time denotes the time an event causes some change of a real-world object. The transaction time is described as the time some change is recorded in a database.

Lum and others [115] propose two other types of time for associating an object with various temporal dimensions, *logical time* and *physical time*. In principle, the logical time is the time with which a change of an object can be described. That is, the logical time is similar to the event time introduced in [34]. Although the physical time is intended to be related with the logical time in the sense that one is allowed to model retroactive updates (i.e., a proposition can become valid before the time at which the proposition is generated by an update) and proactive updates (i.e., a proposition can become valid at a later time than the time at which the proposition is generated by an update), it is used as the recording time of real-world entities. In this way, Lum and the others introduce semantically richer temporal dimensions discerning between retroactive updates and proactive updates in an underlying time-dependent application.

Early attempts on clarifying the terms of various temporal dimensions enriching the semantics of temporal databases, and thus, allowing for multidimensional time representation are made by Snodgrass and Ahn [172]. They propose three temporal dimensions, *valid time*, *transaction time*, and *user-defined time*. These three dimensions base on a classification of time-related databases into historical databases, rollback databases, and temporal databases. That is, the valid time is employed for a historical database allowing for storing a database object's history. Therefore, valid time can be in the past and in the future, respectively, and it can be changed freely. Furthermore, valid time is supposed to underlie a linear flow of time bounded into the past and into the future. (In other formalisms valid time is assumed to be branching, e.g. [64].) The transaction time is described as the temporal dimension used by rollback databases (i.e, a collection of all the timestamped databases). Thus, transaction time is useful for rolling back a state of a database to some previous timepoint, and moreover,

for storing different versions of database objects. In such a situation some transaction time often assumes a branching flow of time. However, the transaction time of an object cannot extend beyond the current time since it is unknown, if the object will remain current in the database in the future. For the same reason, it is not possible to change what has been previously stored in the database. Consequently, if a change on a recorded object (timestamped with a transaction time) is passed, then a new version of this object must be stored without discarding the altered version.

Starting from the previous descriptions, Snodgrass and Ahn argue that transaction time and valid time are orthogonal since these two temporal types have different semantics [173]. That means, each of them can be independently recorded (or not), and each of them has specifically associated properties. Nevertheless, their usages within a certain time-dependent application can produce significant interactions between them [90].

Compared to the timestamping dimensions of valid time and transaction time, user-defined time is an attribute whose domain consists of time values. Thus, user-defined time is treated as a data value that can be considered as any ontological primitive (e.g., as an interval). In that way, it is more concerned with some general ontology of time rather than with timestamped (database) objects, and therefore out of scope of the discussion in this paragraph.

Raising the confusion, Gadia and Yeung [62] adopt the transaction time and valid time dimension from [172] for their n-dimensional (symmetric) time model. However, within this time model [62], the valid time is defined similarly to the event time in [34] rather than to Snodgrass' and Ahn's definition of valid time [172].

Beyond the above briefly introduced formalisms for catching the different semantics of different temporal dimensions, other researches propose similar dimensions at most for transaction time and event time as introduced in [34], frequently using other designations, e.g., [92, 201].

Later on, several researchers have pointed out that simply associating two different temporal dimensions with each time-dependent object, the one (tendentiously) called valid time indicating when some proposition is true in reality, and the other (tendentiously) called transaction time denoting when some object is current in the database, is insufficient to fully capture the semantics of time-dependent information, e.g., [27, 139, 90, 86]. For example, transaction time and valid time combined are inadequate to model an abrupt change (i.e., event) of an object's state. That means, when an event occurs, it actually changes certain properties of recorded objects. Additionally, there can be situations in which the time at which a property assigned to a particular object becomes valid differs from the time at which the causing event occurs.

In order to allow for such semantics, Chakravarthy and Kim [27] propose a three-dimensional model including *transaction time*, *valid time*, and *event time*. In this formalism, the valid time is, in turn, differentiated between *valid period* and *sort of validity*. The valid time itself of an object's property is the holding of this property regarding to the flow of time. The valid period denotes an interval and a set of intervals, respectively over which a certain property is valid, and sort of validity denotes the totality of properties in the property set of its comprising object. (An object can have more than one associated property.) The transaction time is the time at which a property is stored in a database. It simply allows for completing the valid time to preserve multiple past states of some stored object. Finally, the event time is the time at which an event occurs. This temporal dimension is, in turn, differentiated into *one-time events*, *retroactive events*, and *proactive events*. A proposition generated by some one-time event becomes valid immediately after the event occurs. A proposition generated

As a conceptual basis for the following classification we use the *Consensus Glossary of Temporal Databases* edited by Jensen and Dyreson [89]:

Authors	transaction time	valid time	event time
Maier and Copland (1984)	transaction time		event time
Lum and others (1984)	physical time		logical time
Snodgrass and Ahn (1985)	transaction time	valid time	
Gadia and Yeung (1988)	transaction time		valid time
Yearsly and others (1994)	database time		event time
Kemp and Kowalczyk (1994)	database time		world time
Chakravarthy and Kim (1994)	transaction time	valid time	event time
Nascimento and Eich (1995)	transaction time	valid time	decision time
Imfeld (2000)	transaction time	valid time	event time

Table 1: Classification of temporal dimensions for modeling multiple temporal dimensions.

by a retroactive event becomes valid before the time the event happens, and a proposition generated by a proactive event becomes valid after the time the event occurs. Within this formalism, transaction time, valid time, and event time can be combined depending on the needs of the underlying application. For this purpose, Chakravarthy and Kim discuss how multidimensional time can be applied to objects in time-dependent applications. They have figured out three different possibilities for doing so. That is, one can combine (i) valid time and transaction time allowing for preserving multiple histories and error correction, (ii) valid time and event time allowing for preserving multiple histories, and (iii) valid time, transaction time, and event time as a maximal set of temporal dimensions allowing for preserving all multiple past states generated by these operations.

Nascimento and Eich [139] introduce a three dimensional temporal model similar to the one proposed in [27], except for the denotation (i.e., transaction time, valid time, and decision time). Decision time is in fact the same as Chakravarthy’s and Kim’s introduced event time. However, they do not differentiate between proactive events and retroactive events. Furthermore, transaction time is stated to be instantaneous rather than having some duration, as it is defined within the other formalisms.

For briefly classifying the presented temporal dimensions for modeling multidimensional time, table 1 surveys them.

To summarize the previous review, and thus, to settle on a consistent terminology, some simplifications and unifications emerge in more recent works, e.g., [27, 90, 46, 44, 74, 31, 43, 86]. That is, we figure out at most three different temporal dimensions, transaction time, valid time, and event time, as illustrated in table 1. Valid time is applicable only to propositions that can be true or false in the modeled reality. Transaction time can additionally be associated with any modeled object. That means, any stored object can be timestamped using a transaction time dimension. Transaction time denotes the time an object is current, for example, in a database. Valid time is the time a proposition is true in the modeled reality. In most proposals as well valid time as transaction time are defined as durable rather than as instantaneous. Finally, event time is the time a particular event occurs in reality, and thus, enforces a change on a certain property associated with an object, or on a set of properties all

associated with the same object. Furthermore, event time can be differentiated allowing for additionally handle the semantics of proactive events and of retroactive events. Events are particularly related with the valid time of a property and of a set of properties, respectively which are, in turn, associated with a particular object. Any property associated with an object can have more than one event time.

In order to understand the necessity of the three previously discussed temporal dimensions, we conclude with example 2.5:

Example 2.5 *Considering the weather forecast for the next day. It is associated with several dates, e.g.,*

- *the date, a weather station is commissioned,*
- *the date, some barometric pressure measurements are made,*
- *the date, some change in the barometric trend is stated,*
- *the date, some temperature measurements are made,*
- *the date, some wind speed is measured,*
- *the date, some satellite photos are made,*
- *the date, a forecast is made,*
- *the date, the previously forecast meets the actually reached weather conditions, and*
- *the date, when each of these dates are stored in the weather database*

(Certainly, there are much more data related with weather forecasting. However, these are sufficient to understand the temporal dimensions.)

In addition to the propositions itemized in example 2.5, let us consider the proposition ‘X makes the weather forecast for the following 24 hours’. The valid time of this proposition is the time the person makes the weather forecast. The transaction time of this proposition is the time when the stated forecast is stored in the weather forecast database until deleted, and replaced by a more recent forecast. If we consider the proposition ‘X is educated to make the weather forecast’, the valid time is the interval that begins when the person X starts its training, and it terminates when the person’s training is finished. In the same way, we can consider many other propositions within this example, and relate each of them with its valid time. Thus, there are many interrelated propositions, each having a different valid time.

Furthermore, example 2.5 provides an insight into the correlation between valid time and event time. Assuming the proposition ‘the weather forecast for the following day is made’. Many different events (e.g., the ones stated in example 2.5) take place before this forecast is made. These events are actually different event times of our proposition, influencing its truth value. (Various kinds of propositions can have various numbers of event times.) Furthermore, the event to forecasting the weather for the following day is proactive due to the fact that whenever a weather forecast is made, the forecast is not valid until some later time (in this case one day later).

From the example 2.5 it is clear, that a multidimensional representation of time can be useful, mainly when planning systems and forecasting systems are addressed. To summarize, the valid time is good for stating that a certain proposition associated with an object is true in the modeled reality, and the transaction time (that is applicable to any modeled object) completes the valid time allowing for retaining multiple past states (i.e., versions) of some stored object. Different versions (i.e., states of some stored object) can, for example, be generated by updates and error corrections in the underlying database. Beyond this, the event time enables the handling of different decisions (i.e., events) that result in an abruptly change of the related valid time, timestamping a certain proposition associated with an object in the modeled reality.

The Temporal History Temporal dimensions enable a formal representation of real-world objects as they evolve over time. The set of the evolving versions of a particular object forms a temporal history of this object [68]. Once an object's history is defined, it allows for preserving its multiple past states (i.e., versions). In that way, information concerning the history of objects, composite objects, and actions are maintained. That is, past versions of any object become accessible and able to reactivate. In this context, a version denotes the actual appearance of an object at a certain time. Furthermore, the evolving history informs the user of the latest version of any object, and can be used to create some new version. That is, this so-called version management, or versioning, for short, deals with the need to fully retain current and past versions (i.e., states of some stored object), facing changes of the modeled reality.

Multidimensional models of time with their ability for version management, and therefore, preserving multiple past states of the modeled reality are well-investigated in the database community. Various suggestions for multidimensional database version models are made, e.g., [97, 203, 23, 6, 12, 159, 39, 160, 68, 64].

In general, the temporal history of an object can be defined as an order relation between its different stored versions. That means, each history has a certain temporal ordering. This ordering can either be linear or branching. A linear ordering may or may not allow for overlapping of its anchored ontological primitives. Furthermore, each ordering can have a temporal structure comprising some (or all) ontological primitives. That is, the temporal history of an object is an ordering relation over its various versions. What is more, if in an object's history two different versions are unrelated, then these two versions are parallel.

A temporal history is a sequence of versions which result from some changes in the modeled reality. Assuming a linear flow of time, a temporal history can be defined as follows (take from [31]):

Definition 2.3 *A temporal history over a time-dependent object O is a sequence $H = (H_0, \dots, H_n, H_{n+1}, \dots)$ of different versions, so that:*

- *All the versions $H_0, \dots, H_n, H_{n+1}, \dots$ share the same object O (i.e., the versions, $H_0, \dots, H_n, H_{n+1}, \dots$, all are different states (i.e., versions) of the same stored object O),*
- *H_0 is the initial version of the object O , and*
- *H_i results from applying a change-operation to $H_{i-1} \geq 1$.*

A finite history is similarly defined except for the case, that a finite history has a final version. However, histories are usually considered being infinite.

So far, such a temporal history has one temporal dimension, the transaction time (i.e., the time an object's version is current). However, the status of a certain version contained in the history can have more than one possible temporal dimension. For example, a version is already true in reality, but not yet stored in the underlying database. In this case, the version has two different temporal dimensions: The former is its valid time, and the later is its transaction time. That means, a particular version of an object is timestamped by a transaction time. Such a version contains various temporal properties that can be represented by a pair (\mathbf{P}, \mathbf{v}) [27]. \mathbf{P} is the property itself, and \mathbf{v} denotes \mathbf{P} 's valid time. An object can have several properties during its lifetime. The valid times associated with any property belonging to the same object do not overlap one another. That is, they are disjoint since at a given timepoint at most one property is valid. By moving this timepoint along the underlying time-line of the object's history, one can get properties of an object at various timepoints. The valid time can be in the past and in the future.

Similarly to the previously described association of an object's history with its transaction times, and its properties' valid times, event time and valid time can be combined [27]. This becomes necessary when we, for example, want to model the semantics comprised within the example 2.5 illustrating the temporal dimensions in a weather forecasting system. This is due to the fact, that within such a system unpredictable events occur.

To summarize, we have first recalled some formalisms dealing with various temporal dimensions allowing for a multidimensional representation of time. Furthermore, we have sketched the consensus of the considered formalisms among each other. In addition, the notion of temporal history has been outlined. Beyond this, we have seen, that the introductory one-dimensional history can be extended to two-dimensional representations of timestamped objects either by combining transaction time and valid time or valid time and event time. Actually, a three-dimensional representation of an object's history is also possible, e.g., [27, 139]. In this way, multiple past states of time-dependent applications can be preserved, and version management of multiple past states becomes possible.

2.2.2.2 Applying Temporal Dimensions to the Web As we have seen previously, the management of time-dependent information is a thoroughly investigated research issue in the field of temporal databases. Due to this effort, various temporal dimensions an object can be associated with are developed. These temporal dimensions allow for describing an object's history in time. That is, it becomes possible to manage different versions of an object. Recently, some attempts are made for applying these temporal dimensions to the World Wide Web since time undoubtedly plays a fundamental role in any kind of information system. In fact, the challenge of managing and storing multiple versions of Web resources is crucial in many Web applications such as financial budgeting, medical histories, and weather forecasting systems.

Primarily, the temporal dimensions of transaction time and valid time are adapted for the World Wide Web, timestamping time-dependent information. This allows for version management of Web resources. In this context, proposals are made for temporal Web repositories supporting the versioning of Web resources, and for extending XML [52] allowing for timestamping time-dependent data contained within XML documents, e.g., [193, 41, 40, 42, 71,

72, 29, 22, 85, 45, 197].

In this paragraph, we outline some suggestions for temporal Web repositories and Web servers allowing for versioning of remote Web resources. Furthermore, we consider a proposed extension of XML that allows for timestamping time-dependent data within XML documents. This is due to modeling the semantics of time-dependent informations contained within XML documents.

On the one hand, several formalisms are proposed for supporting resource versioning of any kind of Web related data. In particular, standardization efforts are made extending the HTTP protocol. These extensions support search facilities for Web repositories, versioning, and configuration management, e.g., [193, 41, 40, 42, 29, 22, 85, 45]. In these papers, versioning is considered from the system's side. That is, maintaining mechanisms of reflecting a resource's modification over time are proposed. The temporal dimension involved here is the transaction time. In the context of the Web, transaction time concerns the evolution of data regarding to the system (i.e., a Web repository, or a Web server) within which the data is stored.

The development of versioning systems for the Web has its beginning as well in the research of temporal databases (i.e., the use of transaction time) as in Computer Aided Design (CAD) systems. In general, in CAD systems data is collected at different times. The collected data can be linked, and it can be combined allowing for complex objects such as Web pages. For this purpose, a version model that provides semantical means for managing and structuring time-dependent data is required.

The Web-based Distributed Authoring and Versioning system (WebDAV) [193] is similar to a CAD system. WebDAV is intended as a basic infrastructure for authoring that can be used for accessing remote resources on the Web. In principle, WebDAV is an extension of the HTTP protocol, introducing additional concepts to the ones provided by HTTP. These extending concepts concern properties (similar to properties of database objects), collections of URIs, namespace operations, and a resource locking mechanism.

The WebDAV protocol is, in turn, extended by the DeltaV protocol [85] allowing for versioning on the Web. This protocol enables remote access to a central versioning repository. That means, DeltaV introduces the concept of evolving resources to maintain all modified versions of any resource in a history. Whenever a modified version of a resource is created, a new, unique URL is created by the underlying server. Subsequently, the modified version can be accessed by a client via the new URL. Additionally, DeltaV provides mechanisms allowing for obtaining metadata (i.e., information about recorded resources).

Some other formalisms for Web repositories allowing for versioning on the Web are AIDE [40, 42], WebGUIDE [41], and the Versus model [22]. AIDE enables users to follow changes on Web pages. In return, this system is supported by a centralized versioning repository recording various versions of arbitrary documents for reverting on them in the future. In almost the same manner, WebGUIDE is a system for exploring the changes which are stored in this system. Additionally, it offers a navigation tool to realize differences among various versions of a Web document over time. Finally, the Versus model supports object versioning, and provides distributed operations for versions. Among other things, it enables the reuse of document contents, and views on past states of the repository's data. That is, this repository provides a time dimension on stored data.

Beyond these proposals, Dyreson suggests a transaction time server [45]. In this context, transaction time is defined as the modification time of a Web resource (e.g., an XML document). This server is good for archiving different versions of a resource. Furthermore, it

supports querying of Web resources specified by their transaction time.

However, we do not want to go into detail about these systems, and therefore, refer to [29] comparing various version management schemes.

On the other hand, Web documents can also contain temporal information changing over time whereby the validity of the information is a significant part of the information itself. That means, information within a certain Web document needs to be timestamped. The temporal dimension used in this context is the valid time denoting the time a certain proposition (stated within a Web document) is true in reality.

Grandi and Mandreoli [71, 72] propose a temporal extension of XML along with an XSL stylesheet. This extension allows for a selective processing of HTML/XML documents containing timestamped information, and using the temporal dimension of valid time. In particular, they suggest two new XML-tags, denoted with `<valid>` and `<validity>`, for timestamping time-related information of XML documents, or XML documents themselves. `<valid>` defines the validity context comprising the time-dependent contents, and `<validity>` specifies the period within which the associated content is true in reality. This validity period is described by the ending points of an interval. For representing timestamped data, XML Schema's dating and timing format [198] is used. Beyond this, they provide an XML Schema document that enables the definition of well-formed timestamped XML documents. The XSL stylesheet allows for a selective filtering of such documents.

To summarize the above outlined formalisms, using the temporal dimensions transaction time and valid time for modeling and obtaining time-dependent data and information on the Web, it allows for solving problems such as the *broken link problem*. That is, a broken link is either a link to a no longer existing resource or to a resource that has been modified to such an extent that its content is no longer relevant. Furthermore, it becomes possible to retain the validity of any information contained in a Web document by using the semantics of valid time. In contrast to valid time, transaction time concerns the evolution of Web resources regarding to the system within which this data is recorded. It observes the lifetime of each associated version of any recorded Web resource.

In order to realize transaction time and valid time semantics on the Web, we have recalled some proposals for Web repositories and Web servers containing a version model due to the semantics of transaction time. Furthermore, we have sketched a formalism for including the semantics of valid time within an XML document.

In this subsection, we have considered several temporal dimensions allowing for managing the evolution of objects varying over time. These dimensions have been mainly developed for needs in temporal databases.

First, we have recalled the temporal dimensions of transaction time, valid time, and event time providing semantics for timestamping time-dependent objects. This leads to different dimensional considerations of the timestamped objects in the modeled reality. Additionally, the considered temporal dimensions allow for a formal representation of an object's history. A Temporal history enables version management that is indeed important for many applications such as weather forecasting. Finally, we have indicated that according to some recent proposals the considered temporal dimensions can be applied to the Web. This underlines the importance of timestamped information and version management on the Web.

2.2.3 A Representation Scheme for Temporal Formalisms

In the two previous subsections, we have first recalled the many possibilities in which one can formalize time. Subsequently, further semantics of temporal dimensions which time-stamp objects and their associated properties are considered. Such temporal formalisms provide temporal frameworks within which it is possible to represent the temporal aspects of time-dependent entities, but they do not on their own provide means for representing those time-dependent entities themselves. Therefore, we need a representation scheme within which the various aspects of any temporal formalism can be interpreted in a human readable and usable manner. Usually, this is obtained by a representation scheme describing the characteristics of *calendar systems*. Common calendar systems are the Gregorian calendar, the Hebrew (Jewish) calendar, the Japanese calendar, and the Islamic (Moslem) calendar. Beyond these well established calendar systems, educational institutions use Academic calendars, and furthermore, for financial budgeting some Fiscal calendar is used. Since many time-dependent applications ask for a human usable interpretation depending on different cultural, legal, and even business aspects of the underlying temporal structure, a representation scheme for calendar systems must allow for combining different calendar systems into one conjoint model. Actually, calendar systems are ubiquitous not only in natural languages, but also for computer applications using Web resources and databases. That is, the handling of human activities, and just as any communication concerned with such activities need to deal with an explicit or implicit temporal context. Such a temporal context is normally expressed in terms of a convenient temporal unit (i.e., temporal granularity) of some underlying calendar system. This leads to the origin of calendar systems.

In general, any calendar system consists of a set of *temporal granularities*, and a set of *conversion functions* between different temporal granularities. A temporal granularity is a time unit for a calendrical datum, belonging to a particular calendar system. For example, holidays are usually measured in the granularity of days, and train schedules in that of minutes. That is, temporal granularities define the available time units (e.g., day, week, month) that are, in turn, usable in relation with a particular calendar system. Additionally, any temporal granularity contains a set of *granules*. For example, in the Gregorian calendar the temporal granularity day has the granules Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday. Likewise in a (German) Academic calendar the temporal granularity semester has the granules Winter and Summer. The associated conversion functions between temporal granularities specify the interrelations among the considered temporal granularities regarding to one calendar system, or a combination of different calendar systems. That means, the conversion functions allow for mappings between finer and coarser temporal granularities in an underlying calendar system. Moreover, they enable some change-over from one calendar system to another calendar system. In addition to its temporal granularities and its conversion functions, any calendar system comprises a wide variety of calculations determining its particular characteristics. For example, any calendar contains various secular and religious holidays (i.e., public holidays) that must be computed regarding to a complete specification of a particular calendar system.

The abilities of representing temporal granularities along with the particular conversions among them, and the computation of each calendar system's characteristics are crucial for many applications such as appointment scheduling or financial budgeting. Therefore, it is not particularly remarkable that numerous proposals have been made for defining both different temporal granularities and the relationships between them, designing calendar logics, calen-

dar algebras, and algorithms for calendrical calculations, e.g., [32, 152, 28, 190, 134, 204, 16, 114, 44, 150, 48, 144, 15, 98, 143, 133, 14, 70, 142, 58, 59, 153, 175, 198].

In this subsection, we first review the general issues and approaches concerning temporal granularities along with their conversion functions proposed in the literature. Additionally, a proposed illustration for the interrelations among temporal granularities using a *granularity graph* is presented. Such a set of temporal granularities and their relationships representable in a granularity graph form a calendar system. Subsequently, we consider how different calendar systems can be related into a conjoint representation of calendar systems. Finally, we consider XML Schema's means for representing temporal aspects [198]. XML Schema provides a wide variety of means for defining structure, content, and semantics of XML documents. The Extensible Markup Language (XML) [52] supports structured documents on the Web.

2.2.3.1 Calendar Systems A calendar system interpretes the various aspects of the underlying temporal structure with human's specified meaning. It provides a human readable representation and interpretation of time. That means, calendar systems allow for a translation between human readable and usable time units and the underlying temporal structure. In the following, we first consider temporal granularities, and how they can be illustrated using a granularity graph. Additionally, various relationships between any two temporal granularities in a granularity graph of one calendar system are mentioned. Thereupon, we sketch the crucial problem of handling multiple calendar systems in an application that can indeed become important when tasks depending on different cultural, legal, or business aspects are addressed. For example, a university institute uses Academic calendar notions for expressing time tables, or a government uses a Business calendar underlying a financial year consisting of semesters or quarters. That is, a lot of today's applications ask for supporting conversions among different calendar systems. For this purpose, it is shown how different calendar systems can be integrated into one conjoining representation scheme for calendar systems.

Multiple Temporal Granularities In the following, crucial developments concerning the relevant terminology, and a general framework for temporal granularities for calendar systems are summarized.

Various definitions for temporal granularities, and thereupon, semantics for temporal operations with operands at different granularities have been proposed in the literature. The best known operations on different temporal granularities are the ones from the T-SQL query language [174]. Yet, Anderson [7] first emphasizes the need for supporting different temporal granularities. Hereon, Clifford and Rao [33] have proposed a theoretical model of temporal granularities underlying a total ordering of temporal granularities. Furthermore, they introduce granularity conversions along a *finer than* relation on this total ordering (e.g., hours are finer than days). Wiederhold, Jajodia, and Litwin [195] advanced this model by adding some specific semantics. These semantics allow for temporal comparisons regarding to the complete ordering of temporal granularities. Their model allows for handling mixed temporal granularities (i.e., temporal aspects stated in more than one temporal granularity). In sequencing papers, Wang and others [191, 190] have generalized this totally ordered set of temporal granularities to a partially ordered one (i.e., a *lattice*) allowing for finer and coarser relations between the temporal granularities comprised in a particular lattice. In [191], Wang

and the others have provided a formal definition (cf. definition 2.4) that is simplified and generalized, for example, in [114, 48, 17] for time units (i.e., temporal granularities). In this definition, granules are defined as pieces of the underlying time-line ‘grouped’ into aggregations (i.e., temporal granularities). For example, the granule Monday is grouped into the Gregorian temporal granularity day. (The following definition is taken from [48].):

Definition 2.4 *A temporal granularity is a mapping Γ from the integer numbers (i.e., the index set) to granules (i.e., the subset of the underlying time-line), so that*

1. *The origin of Γ , $\Gamma(0)$, is non-empty.*
2. *If $i < j$, and if $\Gamma(i)$ and $\Gamma(j)$ are non-empty, then each element of $\Gamma(i)$ is less than each element of $\Gamma(j)$.*
3. *If $i < k < j$, and if $\Gamma(i)$ and $\Gamma(j)$ are non-empty, then $\Gamma(k)$ is non-empty.*

The first condition of definition 2.4 states that the origin of a temporal granularity Γ is always not empty. The second condition states that the granules in a particular temporal granularity do not overlap, and their index set’s order is the same as the order of their underlying time-line. The third condition states that a subset of the index set mapping to non-empty subsets of the underlying time-line is contiguous. This definition implies that each set of granules of any temporal granularity is non-overlapping and totally ordered. The underlying ordering is derived from the ordering of the integer numbers. Furthermore, it implies that the set of integers mapping to non-empty granules is contiguous. Using this definition, one can express common temporal granularities such as days or weeks, bounded granularities such as the years in the 20th century, temporal granularities with non-contiguous granules such as working days, and temporal granularities with gaped granules (i.e., temporal granularities with non-convex intervals as granules) such as working months.

Considering definition 2.4, in the model of Wang and the others [191], a temporal granularity Γ is specified by providing a function $\Gamma()$ mapping indexes (i.e., integers) to granules (i.e., subsets of the underlying time-line) with regard to the temporal granularity Γ . In this way, any temporal granularity together with its including granules is formalized as a partition of the underlying time-line. Furthermore, temporal granularities are considered to be composed out of indivisible time units, so-called *chronons* such as microseconds [191]. These chronons are usually illustrated by \perp (i.e., the finest temporal granularity in the set of the represented temporal granularities). These chronons are referred to as integers. Furthermore, in Wang’s and the others formalism [191], mappings between different temporal granularities are always defined with regard to the defined chronon. That means, whenever some temporal granularity should be converted into another one, it must be first converted into the finest temporal granularity, and this one subsequently into the desired temporal granularity.

Although the specification in [191] of temporal granularities and its further generalizations, e.g., [16, 17, 142] provide a mathematical framework for considering multiple temporal granularities, it does not provide any practical solutions for modeling calendar systems. For several reasons, such a specification of temporal granularities is insufficient for practical applications using some description of a calendar system [114]: First, this specification refers temporal granularities to as integers (i.e., granules expressed by integers). However, not only humans but also many time-dependent applications (e.g., appointment scheduling systems) refer time to as dates based on a particular calendar system rather than as integers. Second, these

models require a smallest temporal granularity for every calendar system, but users often do not know what the smallest temporal granularity is. In fact, they build further calendar systems basing on well-known calendar systems (e.g., an Academic calendar can, for example, be build on the Gregorian calendar). Third, considering all the characteristics of human made calendar systems (e.g., leap years and leap seconds in the Gregorian calendar), the mappings from granularities to the finest time unit (i.e., the temporal granularity denoted by \perp) are mostly complex. Finally, this model has not taken into account the different anchors of different calendar systems. For example, the Business calendar was starting on the Gregorian date October 1, 1990. In this way, Wang and the others do not present calendar systems. Starting on the one hand with the drawbacks concerning this theoretical formalism, and on the other hand with practical application needs, several researchers provide more practical definitions for means of calendar systems, considering directly the concerns and characteristics of calendar systems, e.g., [38, 176, 152, 174, 47, 204, 104, 114, 69, 144, 48, 150, 44, 143, 70, 142, 153]. They have used Wang's definition as a stating point. Considering these formalisms, temporal granularities directly define the reasonable temporal granularities (e.g., day, week, year) that can be used in conjunction with a calendar system whereby any calendar system is comprised of various temporal granularities. That means, temporal granularities classify the underlying time-line into different levels formed by the different time units (i.e., temporal granularities). More precisely, a temporal granularity consists of a specific name and a linearly ordered (finite or infinite) set of granules [104]. For example, the temporal granularity month of the Gregorian calendar is the temporal granularity that comprises a set of twelve granules, these are January, February, . . . , December, and for the Gregorian year, the set of granules is infinite and it comprises all integer numbers. Furthermore, each temporal granularity can be defined with regard to its next finer temporal granularity, e.g., [174, 47, 114, 48, 44]. That is, within the Gregorian calendar, we can, for example, specify days in terms of hours, weeks and months in terms of days, and years in terms of months. For this purpose, conversions between pairs of temporal granularities must be provided. That is, temporal granularities are specified by relationships to other temporal granularities (e.g., a mapping between a Gregorian day and hour) rather than directly by using their index functions, as it is done in [191]. That is, temporal granularities are related in the sense, that the granules in one temporal granularity can be aggregated in order to form larger granules belonging to a *coarser than* temporal granularity. For example, every Gregorian year can be considered as an aggregation of 365 or 366 (if the considered year is a leap year) Gregorian days. Similarly, Gregorian days are in a *finer than* temporal granularity with Gregorian years. The coarser/finer relations of temporal granularities can be formalized as follows (taken from [48]):

Definition 2.5 *Let Γ and Υ be two temporal granularities.*

- *Then Υ is **coarser than** Γ ($\Upsilon \preceq \Gamma$), and Γ is **finer than** Υ ($\Gamma \preceq \Upsilon$), if for each granule $v \in \Upsilon$, there exists a set of granules $\Sigma \subseteq \Gamma$, so that $v = \bigcup_{\gamma \in \Sigma} \gamma$.*
- *If Γ is finer than, or coarser than Υ , then the two temporal granularities are **comparable**.*

The definition 2.5 states that a temporal granularity is finer than another, if every granule in the finer temporal granularity is a subset of some granule in the coarser temporal granularity. For example, the Gregorian week is a coarser temporal granularity than the Gregorian day

since any week is composed of a set of seven days, and by this means, days are finer than weeks. Otherwise, Gregorian months are neither finer nor coarser than weeks since some months are not completely composed of a set of weeks.

In addition to this finer than relationship, there are several further relationships between temporal granularities which are thoroughly investigated in [14]. In the following, we want to consider some further of these basic relationships between pairs of temporal granularities (taken from [14]).

Definition 2.6 *A temporal granularity Γ groups into a temporal granularity Υ ($\Gamma \trianglelefteq \Upsilon$), if for each granule $v \in \Upsilon$, there exists a (possible infinite) set of granules $\Sigma \subseteq \Gamma$, so that $\Upsilon(v) = \bigcup_{\gamma \in \Sigma} \Gamma(\gamma)$.*

The definition 2.6 states that a temporal granularity Γ groups into another temporal granularity Υ , if any granule in Υ is the union of some set of granules in Γ . It has to be mentioned, that $\Gamma \preceq \Upsilon$ does not imply $\Gamma \trianglelefteq \Upsilon$, nor the other way round.

Definition 2.7 *A temporal granularity Γ is a subgranularity of Υ ($\Gamma \sqsubseteq \Upsilon$), if for each granule $\gamma \in \Gamma$, there exists a granule $v \in \Upsilon$, so that $\Gamma(\gamma) = \Upsilon(v)$.*

Definition 2.8 *A temporal granularity Γ partitions a temporal granularity Υ , if $\Gamma \trianglelefteq \Upsilon$ and $\Gamma \preceq \Upsilon$.*

The difference of the definition 2.8 according to the groups into definition (cf. definition 2.6) is that any granule in Γ must have a counterpart in Υ . Many common Gregorian calendar granularities are related by the partitions relationship such as day partitions week and month partitions year.

For determining mappings between related temporal granularities two converting operations mostly in the context of database query languages denoted with *scale* and *cast* are provided and formalized, e.g., [174, 47, 114, 48, 17, 70]. This leads to a specification of temporal granularities by providing the conversion functions between appropriate pairs of temporal granularities. The set of conversion functions along with a temporal granularity order relationships (i.e., \trianglelefteq , \preceq , or \sqsubseteq) of temporal granularities in a calendar system can be described in a directed graph, a so-called *granularity graph* [48]. In doing so, a calendar system can be defined as follows (taken from [14]):

Definition 2.9 *A calendar system is a set of temporal granularities over a particular timeline that includes a smallest temporal granularity \perp with respect to \trianglelefteq (groups into).*

In the following, we consider a granularity graph of Gregorian calendar granularities along the finer than relationship. Due to correctness, some restrictions on such a granularity graph are required [48]:

1. The granularity graph must contain a finest temporal granularity, denoted by \perp .
2. For every temporal granularity Γ in the granularity graph, there must exist a path along the finer than relationships of the involved temporal granularities from Γ to \perp .

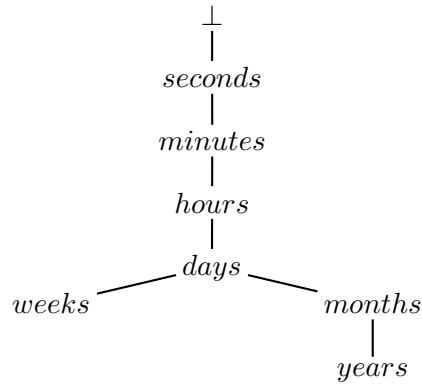


Figure 1: Example of a granularity graph.

3. For every temporal granularity Γ in the granularity graph, there must exist a path along the coarser than relationships of the involved temporal granularities from \perp to Γ .

The temporal granularities in a certain granularity graph form a calendar system [176]. In this way, some temporal granularities of the Gregorian calendar form the granularity graph illustrated in figure 1. This representation allows for calendrical calculations among the considered temporal granularities, for example, using the scale and cast operations. It has to be mentioned that the mappings between pairs of temporal granularities in some calendar system can either be *regular* or *irregular*, e.g., [47, 48]. For example, the mapping from Gregorian days to month must consider both months having varying numbers of days and leap years. Thus, a mapping from days to months is irregular. That means, if each granule of some coarser temporal granularity can be composed of a fixed number of granules of some finer temporal granularity, then such a mapping is said to be regular. Otherwise, it is irregular.

To summarize, a calendar system is a specification that combines the different levels of its temporal granularities, and furthermore, it describes the mappings between them. In addition to this specification, any calendar system contains a wide variety of particular characteristics such as leap years, holidays, and cycles of days, months, and years. Descriptions and accurate computations of the characteristics of twenty-five calendar systems and their relationships to one another are given in [153].

Conjoining different Calendar Systems If we wish to compare granules of temporal granularities that are not within the same calendar system, then the granularity graph illustrated in figure 1 that only includes the Gregorian calendar must be enlarged. Due to this effect, only one temporal granularity in each calendar system must be directly related either to a temporal granularity in some other calendar system or to the basic temporal unit common for all integrated calendar systems [48]. Additionally, several further mappings between the temporal granularities on the different calendar systems must be provided. Since different calendar systems can be combined by converting a particular temporal granularity from the one calendar system into that one of another, calendar systems can be developed separately, and subsequently combined. For example, we can combine the Gregorian calendar with the

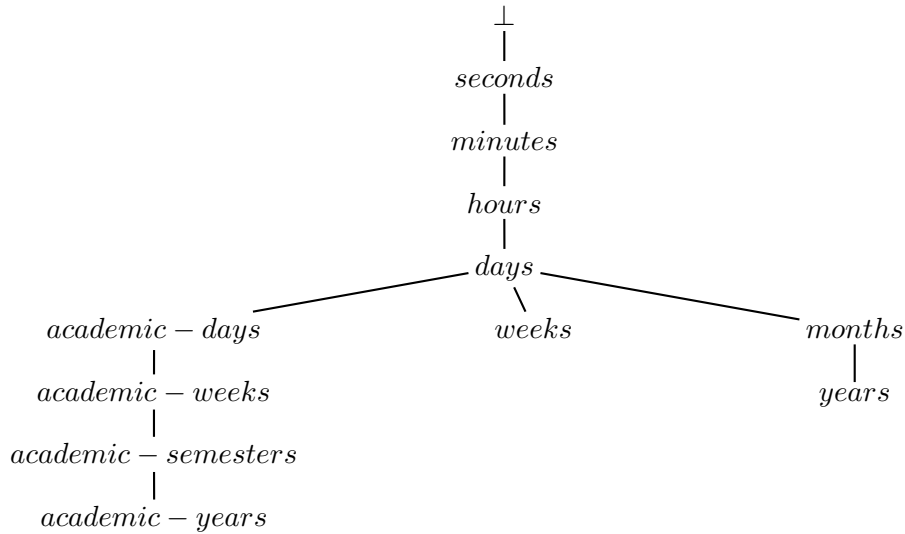


Figure 2: Example of a granularity graph comprising different calendar systems.

Academic calendar illustrated in the granularity graph of figure 2. This illustration presents some of their containing temporal granularities.

Usually, a calendar system underlies a precisely formalized structure of time determining the temporal ‘behavior’ of various temporal aspects expressed by means of calendar systems. Several researchers (some of them choose other formalisms than the previously presented one) have considered the integration of different calendar systems that is frequently required for time-dependent applications such as appointment scheduling, e.g., [175, 104, 69, 48, 144, 98, 153]. Another formalism for integrating different calendar systems is the one proposed by Reingold and Dershowitz [153]. They do not convert calendar systems using such a granularity graph underlying the representation of calendar systems by means of the included temporal granularities. Rather, they define a fixed date 1 (i.e., Monday, January 1, 1 (Gregorian)), and additionally provide two functions doing the conversions for different calendar systems. That is, for each calendar system x they define a function `fixed-from-x(x-date)` converting the date x -date on its underlying calendar system to the corresponding fixed date. The function `x-from-fixed(date)` does the inverse operation. Combining these two functions any conversion between any two calendar systems can be calculated.

Yet another formalism for describing calendar systems – a calendar logic – is provided by Ohlbach and Gabbay [144]. This calendar logic allows for specifying temporal concept of different calendar systems. In this formalism, temporal logic is linked to real time calendar systems over finite temporal intervals. They provide an abstract (logic-based) specification language for defining commonly used temporal concepts such as Islamic weeks, Academic semesters, or public holidays. Time units (i.e., temporal granularities) are defined as independent partitions of different lengths of an underlying linear time-line. For specifying the underlying linear time-line, and thus, locating any considered time unit, an origin of time (e.g., 1.1.1970, Greenwich Mean Time (GMT)) is chosen and moved to the coordinate 0. For any time unit two functions must be realized. The one gives the number of seconds elapsed from the chosen origin, and the other maps such a number back to a common date format in

a particular calendar system. All informations of a particular calendar system (e.g., Gregorian leap years, different lengths of months) must be encoded within these functions. That is, any time unit has its own coordinate system isomorphic to the integer numbers. These two functions allow for converting back and forth between named calendar systems and a corresponding time unit. Based on this, definitions for terms denoting sets of timepoints are proposed along with algorithms for deciding whether a given timepoint is within the set of timepoints denoted by a given temporal unit or not. Within this calendar logic some built-in functions are provided: U_within_V ; U and V are specified time units. This constructor allows for, for example, specifying days in weeks or months, or names of weekdays. This constructor can handle the problem that time units are not always regular according to one another such as weeks and months, or weeks and years. For this purpose, a differentiation between inclusive interpretations and exclusive interpretations, respectively regarding to, if, for example, a part of a week is count as the first week, say of a month or not. The constructors $begin_U$ and end_U (U denotes a specific time unit, and the function takes an instance of another time unit) are provided in order to convert either from finer time units to coarser ones or the other way round from coarser time units to finer ones. The constructor U_s (U denotes a specific time unit) turns individual coordinates into sets such as $day_s(w)$ denotes the set of all days in the week w . For this constructor also an inclusive variant and an exclusive variant are provided. Furthermore, sets for temporal concepts (e.g., all Mondays in a specific year), and periods of time can be specified. Specifying periods, it is up to the user to specify the precise meanings, say what is the beginning time for temporal concepts such as ‘for the next year’. Additionally, arithmetics for temporal concepts specifying, for example, yesterday or tomorrow are provided. Furthermore, conditional concepts with arithmetic comparisons are possible, thanks to the construct $c?a:b$, determining if c holds then a , otherwise b . That is, this calendar logic is quite powerful since it allows for defining several commonly used temporal concepts. Nevertheless, temporal concepts such as easter cannot be computed in this calendar logic since this logic is not as powerful as a programming language such as Prolog. However, implementing this calendar logic as a logic program (e.g., in Prolog) such functions can be directly encoded in this program.

To summarize, calendar systems can be formalized either by defining their associated time units (i.e., temporal granularities) as subsets of a specified time-line or by providing converting functions to and from a specified fixed date. Several specific semantics of calendar system (e.g., various lengths of months, leap years, holidays) must be additionally considered.

2.2.3.2 XML Schema’s Means for Representing Time and Calendars As we have seen in the previous paragraph, the representation of temporal aspects in a human readable and usable manner is important since many time-dependent applications are used by humans such as appointment scheduling systems, or financial budgeting and management systems. Newly, many of these applications are adapted for the Web. These time-dependent Web applications ask for modeling and describing their included time dependent information in a human readable and usable manner. For this purpose, the Extensible Markup Language (XML) [52] supporting structured documents on the Web, and allowing among other things for data-exchange on the Web, need a temporal representation scheme. Mainly XML Schema [198] provides some means for describing and representing temporal aspects which are included in Web documents. In principle, XML Schema provides various means for defining structure, content, and semantics of XML documents.

This paragraph is concerned with XML Schema's possibilities for representing time and dates of a calendar system.

In general, XML Schema [199] provides a rudimentary framework of build-in datatypes for representing time and date in the Gregorian calendar using the ISO-8601 norm for representing date and time [87]. Time zones and leap years are taken into account. However, XML Schema does not support other calendar systems than the Gregorian. In addition to the datatypes for representing time and dates, an algorithm for adding some temporal durations to a certain timepoint and interval, respectively in time is provided. Each timepoint and interval, respectively is represented by specifying, for example, the year, month and day of a particular date within the Gregorian calendar.

In particular, XML Schema provides the following simple build-in datatypes for modeling temporal aspects:

- **duration** allows for representing unanchored time intervals whereby a duration can be specified by a Gregorian year, month, day, hour, minute, and second. In addition to positive durations, negative ones can be specified.
- **dateTime** allows for representing some Gregorian date (i.e., century, year, month, day), and additionally for specifying the particular time (i.e., hour, minute, second) of the represented date.
- **time** allows for representing the time (i.e., hour, minute, second) of a particular day.
- **date** allows for representing Gregorian calendar dates (i.e., century, year, month, day).
- **gYearMonth** allows for representing a particular Gregorian month in a Gregorian year.
- **gYear** allows for representing a Gregorian calendar year.
- **gMonthDay** allows for representing annually recurring Gregorian dates.
- **gDay** allows for representing monthly recurring Gregorian days.
- **gMonth** allows for representing annually recurring Gregorian months.

XML Schema's previously listed build-in simple datatypes are all expressed in the ISO-8601 norm [87] for representing time and date. In particular, the datatype **duration** is convenient for representing a time interval, and **dateTime** for representing a specific timepoint regarding to a (thought) time-line. Each of the other provided and previously listed XML Schema's datatypes allows for representing only a clipping out of the enabled time format (i.e., Gregorian century, year, month, day, hour, minute, and second). That is, **time** allows for the representation of some recurrent timepoint of time represented as some time of a day. **Date**, **month**, and **Year** are initially considered as non-periodic timepoints, but they are additionally allowed for a consideration as temporal intervals whereas **gMonthDay**, **gDay**, and **gMonth** are initially considered as annually and monthly, respectively recurrent (i.e., periodic) timepoints. Similar to the non-periodic datatypes, they can be considered as non-periodic time intervals. The ordering relations of each of the considered temporal types are partial order since there is not always an ordering relation between any two temporal types (e.g., between one month and 30 days).

In addition to XML Schema's means for representing temporal aspects, XML Schema allows for adding some `duration` to a Gregorian date or time, using a particular algorithm that is provided for this purpose. Such a computation can, for example, be used to decide whether a specific time interval includes a particular `dateTime` or not. Some `duration` can similarly be added to `date`, `gYearMonth`, `gYear`, `gDay`, or `gMonth`.

To summarize, XML Schema enables the representation of anchored and unanchored time intervals and timepoints that can be as well periodic as non-periodic, using as a representation scheme the Gregorian calendar. Dates and times of this calendar system must be represented by using the ISO-8601 norm for representing date and time. However, XML Schema does not allow for representing the semantics of different calendar systems. Furthermore, no converting algorithms between different temporal granularities are given since it is only possible to add a duration to a specified Gregorian date or time.

In this subsection, we have considered representation schemes for calendar systems usable in addition to any temporal structure. In a first paragraph, temporal granularities (i.e., temporal units of an underlying calendar system) have been described as partitions of the underlying time-line. Temporal granularities are related in that some temporal granularities are coarser or finer than others. Subsequently, it has been explained that a particular set of temporal granularities form a calendar system. Furthermore, we have seen that different calendar systems can be combined. The relations between different temporal granularities are specified by particular computable conversions between them. Beyond this, we have sketched that any calendar system contains a wide variety of characteristics, for example, determining its specifics of lengths of months or leap years. In fact, any calendar system contains many different computational aspects determining the totality of its characteristics. Beyond this, some further approaches for specifying a representation scheme for calendar systems, in particular Reingold's and Dershowitz' calendrical calculations [153], and Ohlbach's and Gabbay's calendar logic [144] have been presented.

In a second paragraph, we have considered XML Schema's means for representing temporal aspects. XML Schema provides only means for representing Gregorian dates and time, and some basic calculations among the presentable temporal aspects.

To summarize this section, time-based temporal formalisms are primarily concerned with a thoroughly defined temporal structure that is reviewed in the first subsection. The crucial issue of any temporal structure concerns the assumed ontological primitives (i.e., timepoints and/or intervals). As we have seen in the presentations of this section, there are many different temporal formalisms proposed in the literature, and there are difficulties concerned with each of them: On the one hand, point-based formalisms are said to be artificial, but on the other hand Allen's interval formalism has trouble with the *Divided Instant Problem*. To overcome these problems, temporal formalisms integrating timepoints and intervals, both having the same ontological status have been provided in the literature. We have reviewed and discussed the most important conjoining formalisms. These formalisms differ from, what kind of timepoints (e.g., if timepoints are defined having no duration or an indivisible duration), and what kind of intervals (e.g., if intervals are convex or non-convex, or differ between open and closed intervals) are assumed. Since non-convex intervals are important for applica-

tions dealing with recurrent events, we have considered their basic characteristics in the last paragraph of the first subsection.

In the second subsection, temporal dimensions enriching the semantics of an underlying temporal structure due to timestamping time-dependent entities in applications such as temporal databases have been considered. We have figured out mainly three different temporal dimensions, denoted with transaction time, valid time, and event time that can be variously combined in a particular application. Since these temporal dimensions are orthogonal, we talk of different temporal dimensions. The crucial issue of temporal dimensions is that they allow for representing an object's history (i.e., how a particular object evolves over time). Subsequently, we have considered how these temporal dimensions can be adapted for the Web: Transaction time can be integrated into the semantics of Web servers storing different versions of Web resources, and valid time can be used for describing the validity time of temporal information included in some Web document.

In the last subsection, representation schemes allowing for human readability and usability of any temporal formalism by means of calendar systems have been presented. Any calendar system is composed of different temporal granularities (i.e., time units) and particular conversions between them. Moreover, each calendar system comprises a wide variety of characteristics such as leap years, its particular beginning, and cycles of days, month, and years. Subsequently, it has been shown how different calendar systems can be related into one conjoint representation scheme allowing for conversions among the included calendar systems. Finally, XML Schema's means for representing time and dates have been presented since it is a powerful knowledge representation formalisms on the Web.

This chapter has surveyed a wide variety of temporal formalisms for knowledge representation and temporal reasoning. It becomes clear, that the means of any temporal formalism must always be chosen with regard to a particular application. That means, it is neither possible nor worthwhile to develop some temporal formalism for knowledge representation including all phenomena of time. Rather, temporal formalism are developed with regard to a certain group of intended applications. Therefore, this survey simplifies the finding of a proper temporal formalism when a time-dependent application shall be realized.

In a first step, temporal formalisms have been differentiated between change-based formalisms and time-based formalisms. This distinction has been made since in time-based temporal formalisms the representation of time is explicit, in change-based temporal formalisms it is implicit.

In the first section, we have considered three well-known representatives of change-based temporal formalisms which concentrate on change producing entities, so-called change-indicators: The Situation Calculus, the Event Calculus, and Dynamic Logics. We have discussed how the change-indicators of these three temporal formalisms (i.e, actions, events and programs, respectively) change the truth-values of particular propositions in the underlying domain of discourse by considering for each of them a possible axiomatization. Additionally, we have addressed to the diversity of discussions, extensions and refinements of these temporal formalisms. Subsequently, it has been shown how these formalisms can be applied in practice (particularly on the Web).

In the second section, we have considered time-based temporal formalisms which concentrate

on the flow of time. The three main building blocks (i.e., the temporal structure, multidimensional time, and representation schemes for calendar systems) of such temporal formalisms have been addressed. At first, an overview on a wide variety of formalisms for thoroughly defining a temporal structure has been given. These formalisms mainly differ from each other therein, how the ontological primitives timepoint and/or interval are defined. The assets and drawbacks of the considered formalisms have been discussed. Furthermore, we have shown that there is no universal solution that can be used for any time dependent application. Rather, a temporal structure should be defined depending on the needs of the intended time-dependent application. But it is widely agreed, that a temporal formalism must provide means for representing and reasoning about both ontological primitives, timepoints and intervals. Subsequently, temporal dimensions have been reviewed. Each temporal dimension bases on a specific temporal structure. These dimensions allow for timestamping various time-dependent objects in the modeled reality. Furthermore, they allow for preserving multiple past states of an object. Since an object can be assigned to different temporal semantics (i.e., temporal dimensions), a multidimensional representation of time becomes possible. When using temporal dimensions, it becomes possible to represent the evolution of a time-dependent object over time (i.e., to represent an object's history). Due to human readability and usability of a temporal formalism, we have finally recalled some possible representation schemes for calendar systems that can be linked to any temporal structure. We have considered the components of any calendar system, its different temporal granularities (i.e., time units), and how these are related to each other. Additionally, temporal formalisms integrating different calendar systems into one combined representation scheme have been discussed. We have emphasized the usefulness of the formal specification of calendar systems in addition to a formally specified temporal structure mainly when time dependent applications are related with human interaction.

Furthermore, we have reviewed – whenever a cause was given – to what extended temporal formalisms have been adapted to the Web.

3 Three Scenarios

In this chapter, we discuss the use of temporal formalisms for knowledge representation within particular time-dependent Web applications by making up three scenarios. These scenarios are particularly constructed to provide a focus point for discussions around the application of temporal formalisms for well directed investigation and development of machine-processable temporal formalisms usable within temporal reasoning systems. Even though the three considered systems (described within the following three scenarios), all ask for a temporal (representation) formalism, the required means of such a temporal formalism however differ from application to application.

The reasoning tasks we are focusing on in the following scenarios build around scheduling and planning tasks in a distributed environment such as the Web. In particular, this chapter specifies usage scenarios on appointment scheduling, event planning, and budgeting systems. One aspect, when realizing such systems is a specification of the temporal aspects and temporal concepts of calendar systems, how to model their temporal extents, and how their meanings can become machine-processable by an appropriate temporal reasoner. That is, we focus on temporal formalisms for knowledge representation each of the considered systems is concerned with. An appropriate formalism for representing temporal knowledge and means for reasoning about this knowledge would make it easier for machines to automatically find, access, and use application-relevant temporal knowledge. That is, such a temporal formalism would reduce the system's actual computation tasks due to a temporal reasoning system.

The problems considered in these scenarios are as far as possible relevant to academia and to industry. We try to illustrate the needs for knowledge-based representations of temporal aspects and temporal concepts of calendar systems (i.e., particular temporal formalisms) required for planning and scheduling systems. The three subsequently described scenarios are not an exhaustive list covering all possible use cases for knowledge-based temporal formalisms. In fact, they are a representative cross-section of interesting challenges for investigating the different aspects of time-related entities. The applications described in the following scenarios are advancements and generalizations, respectively of today's well-established planning and scheduling systems; in particular, appointment scheduling, event planning, and budgeting. They are invented due to point out mainly time-dependent knowledge representation problems of calendar systems along with their underlying temporal structures when realizing such applications. The first scenario is more concerned with scheduling problems whereas the two others are more concerned with planning problems. Considering the time-related problems within the following scenarios, in the first scenario, temporal concepts of different calendar systems depending on cultural, legal, business, and individual dependencies have primary consideration. The second scenario emphasizes the need for modeling possible developments in time (i.e, a branching flow of time) since it should enable automated planning and managing of a potential large and complex set of event plans. The last scenario is so as to pointing out the problem of representing different temporal dimensions since, for example, a history of past budgets for facilitating extrapolation tasks is required.

This chapter is structured as follows: At first, we recall the possibilities of currently available planning and scheduling systems in the context of each of the described scenarios. In particular, new challenges concerning the representation of temporal aspects of time-related entities occurring in appointment scheduling, event planning, and budgeting systems are considered.

The respective scenario advances and generalizes today's well-established appointment and planning systems. Finally, we outline each scenario's important temporal aspects and temporal concepts that need to be adequately modeled by a temporal formalism so that they can be reasonably used, and thus, enabling a temporal reasoning system.

3.1 Appointment Scheduling

Appointment scheduling is a problem faced daily by many people in enterprises, organizations, and governmental agencies. In many cases, it is solved using natural communication by phone, e-mail, or fax. Recently, cooperating agent systems have been developed, partially automating this task. Actually, there are various systems in the market allowing for automated appointment scheduling. For example, Appointment Quest's online scheduling software ³, NetAppointments scheduler ⁴, Net Simplicity's meeting manager ⁵, and Time Trade's real-time appointment scheduler ⁶. Such services can be used as follows: An enterprise, say a hair-cutting company, provides such an online appointment scheduling services. In doing so, this company has installed one of the available appointment scheduling systems on its Web page. The company's customers who are using this service can see in real time what appointment times are available, they can set up or cancel appointments, and some of these systems send the service's user an appointment reminder per e-mail. Many of these commercial application systems offer some kind of calendar manipulation, but they are mostly limited to one calendar system and a particular representation scheme. Furthermore, they neither provide conversions between different calendar systems nor the ability for specifying complex user-defined temporal constraints about planned appointments.

The appointment scheduling system described in the following scenario allows for (partly) automated appointment scheduling among arbitrary people on the Web. This systems allows for flexible and convenient specification of temporal constraints for any planned appointment. Furthermore, each participant has an electronic calendar, can use temporal concepts of different calendar systems, and specify temporal constraints in the calendar system he is used to.

3.1.1 Scenario 1: Automated Appointment Scheduling via the Web

Bob is an employee at a software enterprise in the USA that is producing high value IT solutions. This enterprise is developing most of its software in India whereas most of its management and distribution is settled in the U.S. Bob himself is working in the enterprise's sales department in San Francisco. He is responsible for the distribution of (new) software products.

Recently, a banking establishment in Frankfurt has put a request on a new banking software. For this purpose, Bob has to organize a first meeting with the chief software developer from New Delhi specialized on this field of functions, and furthermore with a representative from the banking establishment. This meeting shall take place in Frankfurt in the following month. At this meeting, a system specification, beginning and completing, and the costs for this

³<http://www.appointmentquest.com/>

⁴<http://www.netappointment.com/>

⁵<http://www.netsimplicity.com/>

⁶<http://www.timetrade.com/>

software project shall be discussed. For scheduling an appointment for this meeting, Bob instructs his Web agent through a Web browser using his electronic calendar associated with his personal calendar agent. In doing so, he determines several temporal constraints for the planned appointment such as that the meeting must take place in August (in the current year), and that the meeting at most lasts three hours. Furthermore, Bob has the ability to determine different priorities on some of the stated temporal constraints. In addition to the appointment's temporal constraints, Bob determines his meeting partners (with which an appointment time must be scheduled) by specifying their names on his calendar agent. In the same manner as on some temporal constraints, priorities on the participants' planned activities and appointments can be specified. Additionally, he determines a timepoint until which an appointment time must be agreed. Bob's agent then tries to find a match on available time slots for the planned appointment according to the specified temporal constraints and the participants' free and busy appointments, and furthermore, their preferences on different appointments (e.g., the German does not want to have a meeting during his lunch-time). After a few minutes, the agent has found several free time slots for the planned appointment whereby the agent considers various temporal concepts such as shiftings due to time zones or daylight saving times. Subsequently, any participant gets an (individually specified) list of the possibly free time slots along with each participant's relevant information such as correlating public holidays or personal calendar entries depending on each participant's cultural, legal, and business environment. Bob, the software developer, and the representative from the banking establishment subsequently make a compromise on one of the proposed appointments.

Since the meeting in Frankfurt was successful, the participants need to schedule some further appointments for discussions on the development of the ongoing project, further inquiry, and in the final stage some acceptance tests. Furthermore, they need to come to an agreement for the commissioning date. Additionally, a monthly meeting shall take place during the period of time the planned project is running. For these reasons, Bob instruct his Web agent through his handheld Web browser (by specifying the discussed temporal constraints for the further meetings). The agent subsequently tries to find the requested number of appointment times. In doing so, the agent is supplied by temporal information of the participant's online calendars, and by some additional machine-processable temporal information that is related with the planned appointments such as concerned public holidays and information about different time zones. After Bob's agent has computed several possible free time slots for the planned appointments, he send e-mails, or voice mails to the handhelds and cell phones, respectively of the participants, including a list of possible appointment times also showing possible temporal conflicts. Finally, Bob, the software developer, and the representative from the banking establishment discuss on the provided appointments, and come to a compromise which is communicated to the involved calendar agents.

3.1.2 The Scenario's Temporal Aspects

In this subsection, the temporal structure of the mentioned appointment scheduling system are considered.

Considering the problem of (partly) automated appointment scheduling in the previous scenario, appointments and temporal constraints specifying temporal intervals within which an appointment must lie are usually specified using temporal concepts of calendar systems such as weekdays or names of months. Furthermore, appointments are frequently specified by tempo-

ral concepts such as names of public holidays, or lunch-time. Beyond this, the participants can use temporal concepts of different calendar systems as it is the case in the previous scenario: Both the German and the US American use a Gregorian calendar with slightly different temporal concepts, and the Indian probably uses a Hindu calendar. From these considerations it becomes clear, that (partly) automated appointment scheduling needs knowledge-based representation of temporal concepts of different calendar systems (i.e., an abstract temporal formalism). For this purpose, each calendar system's time units (e.g., hour, week, year) must be specified along with their particularities (e.g., leap years, varying lengths of months), and how they are interrelated. Furthermore, temporal concepts such as public holidays or day-times (e.g., afternoon, evening) must be representable in such a machine-processable calendar model. Additionally, shiftings between time zones, daylight saving times, and the fact the days begin depending on different calendar systems on different daytimes (e.g., the Gregorian day begins at midnight whereas the Islamic day begins at sunset) must be considered. For enabling flexible, (partly) automated appointment scheduling in a distributed environment such as the Web, a calendar model for representing, and automatically reasoning about the various temporal concepts of different calendar systems becomes necessary.

Considering the temporal aspects in the context of appointment scheduling, time is always linear since two or more people usually specifying the temporal location of a single appointment sometime in the future. That means, for planning an appointment, we need not to consider other temporally related appointments with the planned one that, in turn, can affect the planned appointment itself. This however would be the case when considering the planning of events since in this case, time frequently branches into the future.

Since appointments and temporal constraints for locating appointments are defined in terms of calendars, the underlying time-line is divided into time units of different lengths whereby the smallest practical time unit in the context of appointment scheduling is either minute or second. Therefore, time can be considered being discrete.

Furthermore, when dealing with temporal concepts of calendar systems and appointments, the temporal intervals involved are always bounded to the past and to the future. This makes sense since usually one does not specify the temporal location of an appointment with 'sometimes in the future'.

Following ontological primitives need to be handled: Timepoints and the relationships (i.e., equals, before, and after) between pairs of timepoints since appointments are often specified by timepoints, e.g., the meeting begins today at 3.30 p.m. Temporal intervals which are anchored on the underlying time-line are specified by a set of timepoints, particularly by two timepoints, a beginning time and an ending time. Additionally, relationships between pairs of intervals must be provided allowing for reasoning about the constraint intervals. Up to now, we have timepoints and intervals. One must be able to check whether a given timepoint lies within a specified interval, or not. For example, some user can specify that an appointment must fall into the interval specified by the ending points 'today' and the 'following weekend'. In this way, relationships between timepoints and intervals can be considered. In addition to anchored intervals specified by sets of timepoints, appointments are linked to unanchored intervals, i.e., free floating intervals of time, specifying the particular length of an appointment such as the meeting lasts 3 hours, or specifying the temporal location of a timepoint. Such free floating intervals have a time unit and a length specified by a number. The handling of such intervals can be integrated into the appointment scheduler by formulating constraints in the context of Allen's interval formalism [1, 2] (cf. also 2.2.1.2). This interval formalism

describes proper temporal intervals (which can be unanchored), and all possible relationships between pairs of intervals by specifying 13 basis relations. Additionally, it is desirable to have the ability to add or subtract temporal durations to timepoints, or to specify the duration of an interval by subtracting the interval's beginning time from its ending time. Beyond this, intervals can be composed of a set of intervals specified by a recurring event. That means, some intervals comprise several (convex) intervals with gaps between them. Therefore, the temporal formalism's underlying temporal structure must provide means for handling regular non-convex intervals.

To summarize this subsection, a proper temporal structure satisfying the needs of appointment scheduling specifies a linear, discrete, and bounded time-line. Furthermore, single timepoints, sets of timepoints, i.e., anchored (non-convex) intervals specified by their ending points, and free floating, i.e., unanchored temporal intervals must be considered along with the possible relationships between pairs of each of them.

In this section, we have first briefly recalled the features of currently available (Web-based) appointment scheduling systems. Subsequently, a to-be scenario on (partly) automated appointment scheduling has been described. In this scenario, appointment scheduling in a distributed environment among arbitrary people having all their individual electronic calendars that vary in language, structure, and possibly in the used calendar system has been described. The challenge for the scenario becoming reality is to find means allowing for comparing, combining, and automatically accessing and reasoning about the relevant temporal aspects and temporal concepts of the different calendar systems required for this system. One stage towards automated communication and interoperability between different electronic calendars enabling (partly) automated appointment scheduling is a machine-understandable abstract calendar model. For finding a reasonable temporal formalism for automated appointment scheduling in a distributed environment, we have finally sketched the temporal aspects relevant in this scenario.

3.2 Planning and Managing Events

In the following scenario, we consider how to perform (Web-based) event planning by using an electronic calendar that allows for managing and reasoning about plans. There is a main difference between event planning and appointment scheduling: Appointment scheduling is only concerned with one time dependent event (i.e., the planned appointment itself) whereas event planning is concerned with several time dependent events related with the actually planned one. That means, event planning must consider several pre-planning activities important for the planned event to be performed. These activities, in turn, can be ordered on an underlying time-line describing the temporal relations within the planning process of the planned event. That means, for planning and managing an event some performing computer program must manage an ongoing planning process. This process includes several activities related with the planned event which are specified relatively to each other regarding to the flow of time. Therefore, the underlying time-line in the context of event planning is usually complex since, for example, alternatives in such a planning process must be considered. The

related events in a set of plans are only partially ordered to one another. Therefore, event planning differs from personal calendar management and appointment scheduling in that the planning task is characterized by a more complex temporal structure. Additionally, competing events regarding to the planned one must be considered. In this way, an event planning system need a complex temporal structure. The planning system needs to maintain proper representations of this structure, and must be able to reason over them.

Several (mostly domain-independent) planning systems comprising automated methods for generating and reasoning about plans and schedules has been developed within the AI community. These systems are mainly concerned with actions (what they do and when they do something), and changes which are caused by actions. For this reason, many planning domains require a rich notion of time describing, for example, properties of actions such as that they can overlap, and that they can have different or no durations. A large list of such automated planning systems can be found on Robert St.Amant's Web page containing AI planning resources⁷. Furthermore, within the AI community several event management tools have been developed⁸.

The event planning and management system described in the following scenario is a knowledge-based reasoning system. It is intended to assist a user in planning and managing a potential large and complex set of event plans. The key idea is to apply the afore mentioned AI technology for modeling and reasoning about plans and processes to the task of automated support for event planning and managing activities in a distributed environment such as the Web. For this purpose, the system needs to provide means for finding and accessing information about some planned event's related activities, locating and handling the semantics of mutually exclusive events, and the handling of incomplete information.

In the reminder of this section, the event planning scenario is described. Subsequently, we provide a brief discussion on the temporal formalisms required for event planning and managing.

3.2.1 Scenario 2: Automated Event Planning and Management via the Web

Maria is working at the municipality in London. She is responsible for planning, managing, and surveying cultural events in the city. Usually, she is planning such events for the following summer season and winter season, respectively one year in advance. The entire set of planned events along with their related activities, i.e., each planning process need to be managed and checked for consistency. Furthermore, Maria needs large to-do lists, reminders, lists of particular event's related activities, and required services necessary for some event. In order to accelerate these tasks and advance its flexibility and consistency, Maria uses an event planning and managing calendar system usable through a Web interface. This planning system is managing the various planning process for each planned event. Maria's planning system contains a graphical user interface allowing for interaction with the system such as requesting pre-planning activities for a planned event, or a survey on planned events for the following week. That means, the event planning system manages a potentially large set of plans along with each plan's planning process. In addition to the graphical user interface, Maria's planning system contains a planning agent that can manage a planning process, even if the required information is only incomplete. This is due to the fact, that in planning tasks

⁷http://www.csc.ncsu.edu/faculty/stamant/planning_resources.html

⁸<http://dmoz.org/Business/Industries/Hospitality/Software/Event.Planning>

several information are usually not available just from the beginning. The agent can locate and handle the semantics of mutually exclusive events. Furthermore, it can find and access libraries (or knowledge bases) containing descriptions of activities and their related services such as renting a catering service (and a list of possible catering services), or inviting a pianist (and a list of pianists). Of course, many of these tasks would themselves decompose into structured activities (i.e., sub-tasks): Renting a catering service could involve calling a catering service, scheduling an appointment, arrange the service, and entering the appointments into the planning calendar.

Imagine, Maria has to arrange for a piano concert of a not yet chosen celebrated pianist that should be dated in the next summer season. Maria's planning agent immediately posts commitments to different tasks required for the planned event in Maria's internal knowledge base – her 'planner'. Additionally, the agent updates the calendar and a to-do list contained within Maria's graphical user interface (associated with her electronic planner). For the newly planned event (i.e., the piano concert), the posted commitments of Maria's planning agent might include inviting pianists which come into question, naming the sponsors, obtaining and managing the available funds, determining the expected number of spectators, organizing a suitable location and set ups, and finding and arranging a proper catering service beginning two hours before the planned event starts. Once Maria has specified the time and date the event might take place, say August, 27th, 8 p.m., the temporal information associated with the planned event and its related activities in the planning process are immediately updated. For example, a calendar entry then reminds Maria in organizing the catering so that this services begins at 6 p.m on August, 27th. Furthermore, if this event is just one in a chain of cultural events in London in this summer, Maria's planning system notices conflicts with mutually exclusive events, and notify Maria, suggesting an alternative appointment time which would avoid a possible conflict. Furthermore, conflicts with other (temporally related) events such as public holidays are noticed, and the planning agent notifies Maria about them. Additionally, the planning agent might suggest Maria that the location and set ups should be scheduled now, even if the actual deadline for reservations has not yet occurred due to rising the flexibility in planning, i.e., handling the situation if the location is already reserved.

Let us consider some tasks the event planning and managing system might automatically manage for Maria. Within Maria's electronic calendar all completely planned events are listed. Complete events have a specific time and date occurrence, and the related activities in their planning processes have specified temporal constraints. Furthermore, a to-do list related with each planned event, containing activities Maria has committed, but which not yet have a specific time schedule are itemized. Beyond this, Maria can add and delete, respectively new activities to an event's planning process. Furthermore, she can view details of existing planning processes. Additionally, Maria's event planning and managing system holds various reminder functions.

Generally, when time passes, the event planning and managing system observes the execution of any planned event's activities, and reminds the user when deadlines are approaching.

3.2.2 The Scenario's Temporal Aspects

Considering the scenario above, we initially need a temporal formalism for describing and representing the planning calendar, and for specifying temporal constraints on planned events and their related activities. The explicit temporal constraints for specifying a particular time

and date for a planned event and related activities comprise beginning time, ending time and durations. Temporal concepts for specifying earliest or latest beginning and ending times, and maximal and minimal durations should probably be enabled within the event planning and managing system. Regarding to these temporal constraints, a convenient and flexible temporal formalism for representing calendar systems is required.

Considering the underlying temporal structure for event planning and management, it is essentially more complex than for appointment scheduling since (1) partially ordered activities, (2) incomplete information, and (3) non-convex intervals (i.e., planning processes) must be considered. Furthermore, the system must provide (4) consistency checks among correlating planning processes and planned events.

1. The partially ordered activities associated with a planned event within its planning process ask for a branching representation of time since, for example, alternatives between possible choices need to be representable and reasonable. That is, conditional branches in time must be considered.
2. The planning process contains both, a planned event's associated activities and the planned event itself. It can be represented by a non-convex interval comprising the different (convex) temporal intervals associated with each of the activities in a particular planning process.
3. When planning an event, a user frequently commits to perform a particular activity without yet specifying the exact time and date at which it will occur. Therefore, a temporal formalism for event planning must have abilities to describe incomplete information. Furthermore, such an incomplete commitment can be completed by specifying a particular time or plan for a planned event. For this purpose, the system must be able to propagate this new information to all affected parts of some planned event.
4. Turning our attention to the consistency check, that is, the system must check whether all input constraint – as well the temporal constraints as all others – are consistent with each other. That means, when a new event or activity associated with a planned event is added by the user, the system carries out a consistency check. In this way, new events and activities are consistent with the user's previous ones in the set of planned events.

The flow of time with regard to event planning is usually continuous since planning tasks (i.e., a particular extent of time) are frequently partitioned into sub-tasks.

To summarize the previous discussion, the underlying temporal structure of a calendar system for event planning and managing is bounded at both ends (planning processes have a beginning time and an ending time), continuous, branching into the future, and additionally, it must provide means for handling non-convex intervals. That is, a temporal formalism for event planning and management must provide a rich, yet complex temporal structure managing also incomplete information.

From the previous discussion it becomes clear, that a (logic-based) temporal formalism properly fitting the needs of a planning and managing system has a quite complex temporal structure. That is, alternatives, incomplete information, and temporally correlating activities must be considered. Furthermore, events must be classified for discovering mutually exclusive

events.

To summarize, we have initially recalled the characteristics of AI planning systems. Subsequently, a (partially) automated event planning and management system operating in a distributed environment has been described. Finally, the temporal aspects of event planning and managing problems have been briefly discussed.

3.3 Budgeting

Recently, several Web-based budgeting software solutions have been proposed. For example, the Adaytum system by Adytum⁹, the Budget 2000 system by EPS Software¹⁰, the SmartStreamBudgetSystem by Geac Corp.¹¹, the Helmsman system by Helmsman Group Inc.¹², and the Budget Maestro by Planet Corp.¹³. In the majority of cases, these systems provide a set of plan screens, each for automating tasks such as personal, capital, and revenue planning. These systems are a further development of spreadsheet budgeting tools. Such newly developed budgeting systems operate on a centralized multirelational and multidimensional, respectively database collecting the budgeting system's required data. Some of these systems allow for use on the Web. Spreadsheets are a commonly used tool for analysis in data mining, operating on OLAP-based database systems. OLAP-based systems provide richer means for querying than conventional (i.e., relational) databases such as drill-down, roll-up, cube, slice, and dice operators. Such database tools are especially used for purpose of analysis.

The following scenario builds upon such self-contained budgeting system. We assume an electronic calendar system that manages the (time-based) semantics of budgeting tasks. That is, this system manages a history of past budgets, the various aspects of the current budget, and the various aspects of the current budgeting (with regard to the current budget).

3.3.1 Scenario 3: Automated Budgeting via the Web

Rose is headmaster at a public school somewhere in the USA. At the end of each school year, she has to plan next year's budgeting for the school. The budget is the sum of the public budget the school gets from the government every year, depending on the public revenues each fiscal year minus the sum of the expenses for the school. The government guidelines the distribution of the school's annual budget.

Rose uses a Web-based budgeting system that guides her through the process of creating a budget she and the teaching staff can manage, see into, balance, and compare with any previous budget of the school, the budgeting, and the actual budget for the running school year. Additionally, the school's budgeting system can use, for example, for prognoses on the national financial developments the government's online budgeting, and for reports and data contained in Web-based financial knowledge bases.

For planning a new budgeting, Rose initially lists all incoming budgets for the following year that is depending on the estimated public revenues for the considered year. Subsequently, she determines various funds for the following school year. For this purpose, Rose considers,

⁹<http://www.adaytum.com/>

¹⁰<http://www.epssoftware.com/>

¹¹<http://www.geac.com/>

¹²<http://www.helmsmangroup.com/>

¹³<http://www.budgetnow.com>

funds such as renovations, supply and disposal for electric current, heating and water costs, and further expenses. These expenses vary depending on the school's terms. For example, the expense on electric current is during some vacation less than while the school is in session. Such aspects must be considered for the budgeting. Rose subsequently distributes the incoming budget and the determined funds for the school assorted, for example, by teaching aids, office supplies, school trips, and further resorts. (For this purpose, she probably considers the budgeting from other schools for the current school year.) For each of these resorts, she determines a particular fixed percentage from the incoming budget, considering seasonal and school's term depending changes on the available funds. Her budgeting system computes, and subsequently stores the budget for any of the determined resorts together with several constraints Rose has stated on each resort's budget. Her budgeting system provides an automated method for distributing these information throughout her teaching staff, informing each of the teachers for his/her terms of references per e-mail.

Since each teacher of the terms of references he/she is responsible for is working self-contained, Rose gets monthly a report from the school's budgeting system. This monthly report includes absolute and relative deviation compared to the running budgeting, and to extrapolate it regarding to the previous year. Additionally, the report lists as well the running budget for each terms of references separately as the total momentary disposable budget. Of course, the budgeting can be anytime seen into by the teaching staff. That is, Rose or any other teacher of the teaching staff can request for particular information on the development of the budget at any time.

Robert is responsible for the school's library. Recently, he has recognized that he needs to purchase a set of new geography books for the 9th and the 10th grad since the available ones are obsolete. In doing so, Robert calculates the costs for this new purchasing. Subsequently, he makes a request over the calculated sum on the school's budgeting system. The system responds by determining that this purchasing would exceed the still available budget for the school's library regarding to the running budgeting. Since this purchasing seems to be important, Robert instructs the budgeting system to send Rose an e-mail including the budgeting system's calculations. Rose makes a request on the budgeting system for a possible surplus from the last school year. Moreover, she request for possible further purchasings in the running budget which are inescapable. Subsequently, Rose instructs the budgeting system to calculate some possible developments on the school's running budget when making this purchase. The budgeting systems starts some calculations using recurrent purchasings for the school by considering required funds for the running budget, budgeting over the past years, the government's prognoses, and further considerations. After a few minutes, the system provides some possible developments of the running budget when the requested purchasing is done, listening detailed *when-if* scenarios on the budget's development. This can be, for example, that the agent proposes a decrease of the previous year's surplus, or a redistribution of the running budgeting's fund. After Rose has chosen one of the proposed solutions, she subscribes through the budgeting system the decision on the budgeting calendar. The renewed running budgeting is subsequently announced, and all concerned teachers get an e-mail including the newly valid budgeting for the running school year together with its changings with regard to the previously valid budgeting.

3.3.2 The Scenario's Temporal Aspects

In the following, the temporal aspects of a temporal formalism properly fitting the (time-dependent) needs of budgeting are summarized. The subsequently discussed temporal aspects must be linked to a representation of calendar systems.

A budgeting system must manage different temporal dimensions. In particular, budgeting is concerned with three temporal dimensions, namely transaction time, valid time, and event time. That means, a budgeting system initially manages a history of past budgets, each of them timestamped with its transaction time. The transaction time is the time, a budget is current, and subsequently, stored in the system's underlying database. Additionally, the current budget's properties (e.g., the total amount of the available fund), and the associated budgeting's properties (e.g., planned funds for electric current, for heating and water costs, and for renovations) are timestamped and stored, each with a valid time. Of course, both the total running budgeting and the associated budget are timestamped with a valid time. Beyond this, the semantics of a further temporal aspect need to be considered: Budgeting is a task usually concerned with sudden events leading to unforeseen expenses (e.g., a water damage in the school's building). Such events need to be modeled by timestamping, and subsequently storing them with the semantics of event time. That means, a temporal history containing timestamped events with their particular event times need to be realized. Since the event time of unaccounted expenses for the running budget semantically differs from the temporal semantics of the valid times of the budget and the budgeting, it is orthogonal to them.

That is, we have three different temporal dimensions, transaction time, valid time and event time associated with the considered properties of some budget and its related budgeting, and the budget itself. Therefore, budgeting asks for a multidimensional calendar model that allows for the discussed timestamps of different semantics aiming at automated reasoning tasks on budgeting.

Considering the brief discussion above, we need a temporal history for both, the annual budget and the annual budgeting along with each of their relevant contained property's valid time and event time. Each considered budgeting-related property's valid time can be used to timestamp the particular temporal period within which this object is valid. Additionally, some temporal semantics is required to timestamp the validity of a budget and budgeting, respectively as a whole. Each considered budgeting-related property's event time can be used for timestamping unforeseen events, leading to changings of both the running budget and the running budgeting. Furthermore, the semantics of transaction time can be used to timestamp past versions (from the past years' budgets), and store them in a temporal history. This allows for extrapolating the current budget with some previous budget.

Beyond this, the temporal structure of each of the temporal dimensions need to be discussed.

In this section, a to-be scenario about budgeting in a distributed environment, operating with many time-dependent properties of some budget has been described. Due to the information required for combining, comparing, and extrapolating budgets, an electronic budgeting system must manage different temporal semantics of any budget's properties. That is, the time-dependent data and information for budgeting must be related with temporal components (i.e., transaction time, valid time, and event time). This information must be stored in a

budget's temporal history managing the timestamped information and data.

At first, budgeting systems operating on multirelational (or multidimensional) centralized databases have been recalled. Subsequently, an advanced budgeting system that is able to operate on the Web has been described. Its working has been described within a scenario. Finally, we have summarized the temporal aspects of different temporal dimensions required for an electronic budgeting system that manages any budget's particularities in an electronic calendar.

In this chapter, we have provided a focus point for discussions around applying and designing temporal formalisms for knowledge representation within certain dynamic systems. Three scenarios generalizing and advancing, respectively today's scheduling and planning systems have been described. These scenarios are particularly concerned with Web-based appointment scheduling, event planning, and budgeting tasks. Each of the systems described in the scenarios ask for an electronic calendar system enabling temporal reasoning tasks. However, the particular temporal aspects each of the required temporal formalisms must provide vary. That is, the appointment scheduling system asks for a sophisticated temporal formalism managing all the aspects of arbitrary calendar systems each in its cultural, legal, business, and individual context. The event planning system asks for a temporal formalism that manages alternatives in time, incomplete information, and non-convex intervals (i.e., planning process). The budgeting system asks for a temporal formalism that manages different temporal semantics (i.e., temporal dimensions) leading to a multidimensional temporal formalism. Initially, we have recalled for each of the described scenarios the currently available software solutions. Subsequently, a respective scenario has been described, leveraging or advancing currently available systems. Within each of the described scenarios, we have emphasized the representation and reasoning about each scenario's related temporal aspects. Finally, each scenario's temporal aspects have been briefly discussed. This is due to work out the characteristics of temporal formalisms properly fitting the requirements of the considered reasoning tasks. Such temporal formalisms would make the considered systems more flexible, easier to communicate, and to interoperate.

4 Time in a Calendar and Appointment Scheduling System

The main objective of this chapter is to describe a time-dependent system operating in a distributed environment such as the Web, and to discuss its various temporal aspects. This chapter builds upon the first scenario of the previous chapter (cf. 3.1.1), thoroughly investigating the structure of appointments an appointment scheduler must be able to reason about, and the hierarchical structuring of calendar systems providing a starting point for developing a set of knowledge bases that contain temporal knowledge required for appointment scheduling. This investigation aims at a proper description of an appointment scheduling system. Therefore, an abstract (logic-based) specification language for calendar systems (i.e., a temporal formalism), and a set of knowledge bases containing predefined temporal concepts of different calendar systems are required. Since Ohlbach and Gabbay have developed a calendar logic [144] for representing the temporal concepts of arbitrary calendar systems, we use this language within the description of an automated appointment scheduling system. This description is conducted through the (temporal) requirements of an automated appointment scheduler. For this purpose, the temporal aspects of appointments are classified (i.e., the temporal aspects an appointment scheduler must be able to reason about). The appointment scheduler is described to such an extent that it can easily be realized. Furthermore, the design of the required knowledge bases is merely discussed by suggesting a hierarchy of calendar systems. However, a realization of these knowledge bases can be easily deduced from the proposed hierarchy of calendar systems.

Calendars are important tools for everyday office tasks. One of the most common activities that take place in every day business life is scheduling and negotiating non-overlapping appointments among arbitrary people. This task becomes even harder since nowadays many companies are operating world-wide. Consequently, appointment times need to be scheduled among arbitrary people living in different cultural, legal, and business environments, frequently in different time zones, and possibly using different calendar systems (e.g., Gregorian calendar or Islamic calendar). Usually, people feature wide variations in managing calendars. The variation begins with the underlying calendar system someone is using, several individual calendar entries such as individually relevant birthdays, and a wide variety of cultural, legal, and business depending definitions of time spans covering temporal concepts such as lunch-time or easter and other public holidays. Furthermore, the level of privacy and accessibility rights, user-defined preferences and priorities on appointment times and on certain registered events, the number of rescheduled or canceled appointments, and additionally, the method of storing, querying, and inserting appointments and/or related information into a calendar vary. Thus, automatically computing possibly free time slots for a planned appointment (i.e., appointment scheduling), and thereby allowing for flexible and convenient usage of personal electronic calendars is a difficult task.

Since there is a desire for automating this task, several electronic calendars with reminder facilities and other features have been developed, and integrated into office computer systems. Additionally, several proposals for automated calendar and appointment systems have been made in the last decade, e.g., [122, 49, 179, 162, 67, 178, 200]. These systems are usually realized as multi-agent systems. In such systems, any user has a personal agent that can cooperate with other agents. However, these systems suffer from a lack of flexibility and power since most of them are stand-alone systems, and they do not contain facilities for scheduling appointments among multiple participants using different calendar systems – although this

additional capability would be a clear advantage of electronic calendars over traditional paper calendars. For the purpose of (partly) automated appointment scheduling becoming an open task comprising arbitrary people which use different personal calendars depending on cultural, legal, and business environments, an advanced appointment scheduler is necessary. An appointment scheduler asks for a wide variety of information about time and calendar systems. Therefore, the various temporal concepts of arbitrary calendar systems must be defined in an abstract specification language for calendar systems. Subsequently, this information must be stored (e.g., in different knowledge bases serving the different calendar systems), and made accessible for the appointment scheduler. For example, temporal concepts such as public holidays, names of weekdays or months, lunch-time, or school vacation should be usable by any user of an electronic calendar, and automatically understandable and processable by the appointment scheduler.

In this chapter, the structure of appointments and a hierarchical structure of calendar systems are thoroughly investigated. This investigation is oriented on the temporal needs of an automated calendar and appointment scheduling system operating in a distributed environment such as the Web. In the first section, the possible structure and working of a calendar and appointment scheduling system by determining its main software components along with their interrelations is described. This is due to present a real application for a temporal formalism for knowledge representation of calendar systems. For the calendar and appointment scheduling system described within this chapter, Olbach's and Gabbay's calendar logic [144] is used. In the second section, the temporal aspects of the calendar and appointment scheduling system are investigated. This section contains three subsections: At first, the basic functioning of an appointment scheduler is described. Subsequently, the structuring of appointments and of temporal constraints for specifying appointment times the described appointment scheduler must be able to reason about is classified. Finally, we discuss a possible hierarchy of different calendar systems that allows for designing a set of machine-processable knowledge bases containing temporal knowledge that is meaningful for the described appointment scheduler.

4.1 Overviewing the System's global Structure

In this section, we describe the (rough) structure of a distributed calendar and appointment scheduling system in order to figure out the temporal aspects required for (partly) automated appointment scheduling. In this system each user has a personal calendar agent that keeps his/her personal electronic calendar. Such an electronic calendar contains, among other things, the user's scheduled appointments (e.g., conference Monday, 1st August, 10.30), preferences (e.g., no meetings on Friday afternoon), and individually varying temporal terms (e.g., user X defines lunch-time as an interval sometime between 12.30 a.m. and 2 p.m. with a duration of one hour). The personal calendar agents can communicate with each other. For scheduling an appointment among two or more persons on behalf of the associated users, the corresponding calendar agents cooperate for finding possible appointment times. For this tasks, the calendar agents either use a centralized appointment scheduler or each uses an internal copy of such an appointment scheduler. The second case requires an appropriate negotiation strategy for the operating calendar agents. Some user can initiate the calculating of a free time slot for a planned event by using his personal calendar agent. After having calculated free time slots for a planned appointment, each participant receives an ordered list

of possible appointment times. This list includes additional information such as personally stored calendar entries or public holidays in each user's personal cultural, legal, and business environment.

In the following overview, the required software components and temporal concepts that enable (partly) automated appointment scheduling are addressed. A possible realization of such a system is out of the scope of this work.

For motivating the following description of an automated appointment scheduling system, let us start again with the first scenario of chapter 3 (cf. 3.1.1). Three persons, i.e., a US American from San Francisco, an Indian from New Dehli, and a German from Frankfurt want to find a three-hour long time slot in August for a particular meeting. The central problem is to find an appointment time that satisfies the user-defined temporal constraints (i.e., three hours sometime in August), each participant's preferences (e.g., the German does not want to have a meeting during his lunch-time), and conflicts with each participant's busy appointment times. This task becomes more difficult because the three participants live in different states which are settled in different time zones. That is, an automated appointment scheduler must consider time changes between time zones, and manage the temporal concepts of different calendar systems. After having computed possible appointments, the automated appointment scheduler should provide each participant an ordered list of possible time slots along with each participant's corresponding conflicts relating to personal calendar entries, and the public holidays in each user's cultural and legal context.

Considering this scenario, we are initially faced with the problem of specifying non-overlapping appointment times among arbitrary people. Furthermore, several temporal concepts of different calendar systems varying from participant to participant such as time zones, public holidays, or daytimes must be considered. In this way, the design and implementation of a useful (partly) automated calendar and appointment scheduling system that is usable in a distributed environment represents a challenge in many aspects. To be actually usable, such a system has to fulfill several requirements: Each user should have the ability to use the calendar system he is used to, to formulate individual requests (e.g., display my scheduled appointments for the following week), and to specify preferences (e.g., no meetings on Friday afternoon) on his/her personal electronic calendar. Such an appointment scheduler must manage the various temporal concepts of different calendar systems. Furthermore, it must be able to reason about several aspects of user-defined temporal constraints for a particular appointment, and to understand the informations stored in each participant's personal calendar. That means, the calendar and appointment scheduling system must understand and reason about the temporal concepts of arbitrary calendar system. For this purpose, the required information must be specified by a suitable temporal formalism, and made automatically accessible and processable by the appointment scheduling system.

In order to understand the global structure and working of a Web-based appointment scheduling system, let us consider the illustration of a calendar and appointment scheduling system in figure 3. Each user has access to the facilities offered by the system through his personal calendar agent that can communicate with other personal calendar agents.

Considering figure 3, each user maintains its *personal calendar* comprising a *dialog manager* with a graphical user interface for accessing one's own calendar, a *database*, and a *personal calendar agent*. Additionally, each user has the ability to determine individually tuned read and write access rights on his personal calendar.

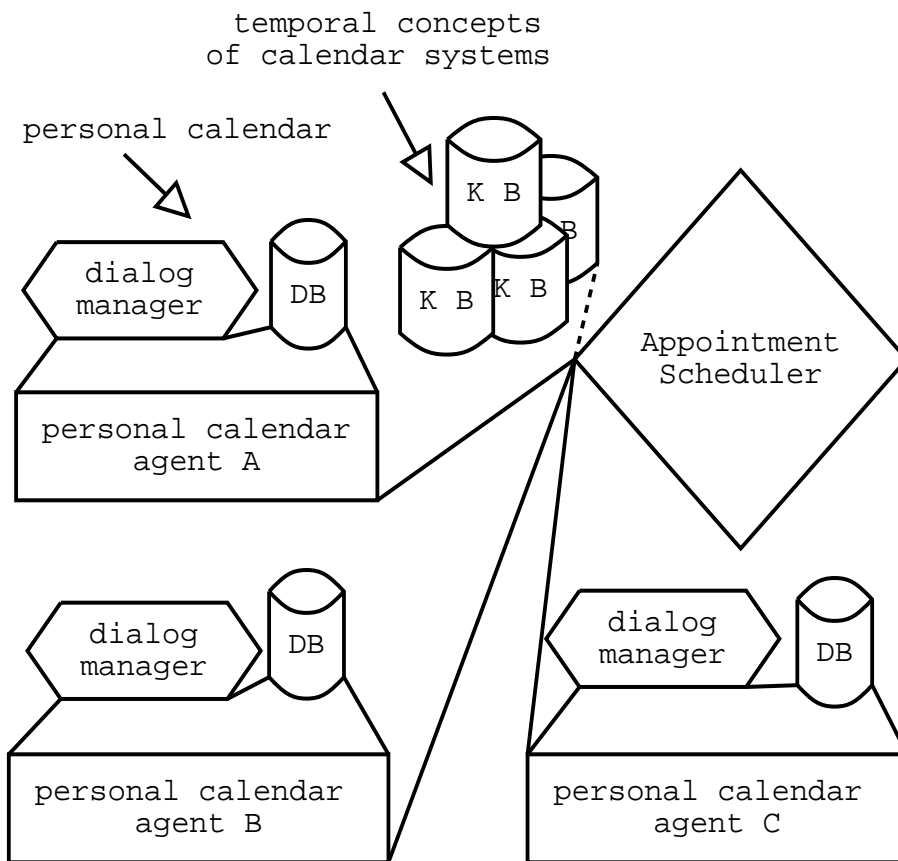


Figure 3: The global structure of a calendar and appointment scheduling system.

Each user's individual calendar entries contain free and busy appointment times and preferences on them. The registered events and scheduled appointments are stored in the user's personal calendar database. For example, the user has registered that he has a French course every Monday evening between 7 p.m. and 9 p.m. from May to November. Furthermore, any user determines preferences for particular times and priorities on different scheduled appointments. When a new appointment is scheduled, or an already fixed appointment is changed, this information is also stored in the user's calendar database. That is, the owner of an electronic calendar can register all relevant appointments, and further events in his calendar which he formerly might have registered in his personal paper calendar.

Each user's personal calendar is managed by a personal calendar agent. This calendar agent serves as the user's personal assistant. The user interacts with his/her personal calendar agent through a graphical user interface (i.e., the dialog manager). Cooperation among different calendar agents must be realized by communicating messages. This is because a calendar agent has no direct access rights to the personal calendars of other participants. Additionally, the calendar agent has access to its user's personal calendar. In this way, a calendar agent provides management functions. Among other things, it provides on request the addition of its user's context-dependent holidays to the personal calendar, the addition of new private appointments (involving no other participants), the rescheduling or deletion of already scheduled appointments, and the querying of the user's calendar contents.

Starting again with the illustration in figure 3, besides the human users and their personal calendars, additional components in a calendar and appointment scheduling system are pictured.

Initially, there is a component denoted as an *appointment scheduler*. This is a software program that can be realized as a constraint solver, containing some predefined constraints about temporal aspects in the context of appointment scheduling (e.g., relations between temporal intervals or mappings from one calendar system to another). Additionally, the appointment scheduler must reason about several user-defined temporal constraints in order to allow each user for defining several temporal constraints for specifying any desired appointment (e.g., a meeting next week on an afternoon). If a user initiates to schedule an appointment, then he initially determines the appointment's participants, and defines some temporal constraints specifying the time the particular appointment must take place. The appointment scheduler subsequently checks the initiator's and the participant's personal calendars (i.e., free and busy appointment times, and user-defined preferences) for reasoning about possible time slots. Furthermore, the scheduler checks, if there is any temporal information related with the participants and the possible time slots. For example, a time change since two participants are living in different time zones, or public holidays corresponding with each participant's cultural and legal context.

The appointment scheduler can either be a global component accessible for each of the personal calendar agents or each of the agents can have a local copy of such an appointment scheduler. If each calendar agent has a local copy of the appointment scheduler, the calendar agents need a possibility to cooperate and communicate among each other by using an agent communication language. The task of such an agent communication language is to provide communication facilities between an agent and his user. Furthermore, it allows for transmitting requests and responses among different agents. That means, agents translate messages from one agent's communication form to another agent communication form by using an agreed standard. The agent can interpret and standardize incoming messages, and subse-

quently send them back to other agents. That is, the agents accept messages from various agents, translate them into its used language, compute something, and finally translate the results back to the standard protocol to communicate the computed results. Such communication among calendar agents can be established by a *negotiation protocol*. A negotiation protocol specifies what happens, if a user is initiating to schedule an appointment among two or more participants. That is, the reaction on non-free time slot for a specified appointment, on user-defined preferences and priorities, or on tentatively scheduled appointments are specified. However, a proper scheduling and negotiation strategy among the cooperating calendar agents for scheduling an appointment among more than one user is out of this work's scope. The interested reader is, for example, referred to [67, 122, 162, 178].

For scheduling appointments, or querying one's personal calendar, a user frequently uses *temporal concepts* of different calendar systems such as public holidays or names of weekdays and months. Such temporal concepts are defined in terms of properly fitting temporal formalism. Afterwards, they are reasonably stored for communicating them within the calendar and appointment scheduling system. Any calendar system comprise a wide variety of such temporal concepts. The meanings of different temporal concepts specify the semantics of some calendar system. This comprises the semantics of universal usable time units of commonly used calendar systems (e.g., Gregorian years, Islamic, months) along with their particularities (e.g., leap years, lengths of months). Furthermore, temporal concepts vary depending on cultural, legal, business, and individual aspects (e.g., public holidays, working days, lunch-time). These dependencies leads to different calendar systems probably used by a single user. For example, terms of universities and schools dependent on a universal calendar system (e.g., Gregorian calendar), and additionally on some Academic calendar used in a particular cultural and legal context. Additionally, the semantics of time zones, daylight saving times, and seasons with respect to different states and cities must be considered.

To summarize this consideration, the various temporal concepts of different calendar systems must be specified using a temporal formalism (within the described system Ohlbach's and Gabbay's calendar logic [144]). Afterwards, this information must become accessible and reasonable for the appointment scheduling system. That means, this information must be defined and stored in a set of knowledge bases, each facing the various temporal aspects of a particular calendar system.

In this section, the basic software components of a calendar and appointment scheduling system have been presented. That is, each user has a personal electronic calendar containing all kinds of individually relevant calendar information such as free and busy appointment times and preferences associated with particular appointment times. Additionally, each user has a personal calendar agent that manages the user's personal calendar, and that can communicate with other calendar agents due to negotiating and scheduling appointments on its user's behalf. For this reason, an appointment scheduler for computing free time slots for a requested appointment on user-defined temporal constraints which are possibly formulated in temporal concepts of different calendar systems is required. We have mentioned that various temporal concepts of different calendar systems such as lunch-time, easter, or time zones need to be automatically accessible and processable for the appointment scheduler. For this purpose, the appointment scheduling system requires machine-processable temporal information resources (i.e., knowledge bases) for managing and reasoning about the various temporal concepts of

different calendar systems.

4.2 The System's Temporal Aspects

In this section, the temporal aspects occurring in an appointment scheduling system are investigated. In particular, we thoroughly investigate the structure of appointments an appointment scheduler must be able to reason about, and the hierarchical structuring of calendar systems providing a starting point for future developments of a set of knowledge bases that contain temporal knowledge required for appointment scheduling. These temporal aspects need to be expressed in a logic-based specification language for calendar systems (i.e., a temporal formalism). Since Ohlbach and Gabbay have proposed a specification language for calendar systems, a calendar logic [144], we use this language. Furthermore, some temporal information must be predefined and stored in that it becomes machine-processable within the referenced system. For this purpose, knowledge bases need to be developed.

Besides the management functionalities of a personal calendar agent concerning operations on its user's personal electronic calendar, the major task of a calendar and appointment scheduling system is to allow for automatically negotiating and scheduling appointments involving any number of users of electronic calendars. Automatically scheduling an appointment among two or more persons is a non-trivial task because each of the participants has different preferences on appointment times, uses different individual temporal concepts such as lunch-time, has different free and busy appointments, and possibly uses a calendar system differing from the one of some other user. Furthermore, appointments are frequently rescheduled or deleted. In this way, the process of appointment scheduling is crucially dynamic. In addition, a user of a (partly) automated appointment scheduling system prefers to specify temporal constraints for a planned appointment as flexible and convenient as possible. For example, some user prefers to specify appointments relative say to a public holiday such as two weeks after easter. This section attends to these time-dependent problems that should become representable and manageable within the described appointment scheduling system.

This section deals with an investigation of the temporal aspects required for appointment scheduling. It is structured as follows: In the first subsection, the basic functioning of the calendar and appointment scheduling system's automated appointment scheduler is described. That is, the working of this software component linked with a temporal formalism, in our case, Olbach's and Gabbay's calendar logic [144] is considered. In the second subsection, the structure of the user-defined temporal constraints the appointment scheduler is reasoning about, and the structure of appointments are classified providing examples expressed in the mentioned calendar logic. In the third subsection, we discuss a possible hierarchical structuring of different calendar systems that can be realized as a set of machine-processable knowledge bases. These knowledge bases shall contain definitions of various temporal concepts expressed in the mentioned calendar logic.

4.2.1 The basic Functioning of an Appointment Scheduler

For understanding the necessity of an abstract (logic-based) specification language for calendar systems usable within a calendar and appointment scheduling system, we briefly survey the possible working of the system's appointment scheduler in the following.

Considering the problem of (partly) automated appointment scheduling, certain tasks associated with temporal and further (in the context of this work irrelevant) preconditions which need to be executed, arise. In particular, a non-overlapping appointment time for two or more people should be found. Thereby, all user-defined temporal preconditions (i.e., temporal constraints), and each participant's free and busy appointment times and preferences must be considered. The task of scheduling appointments among arbitrary people in a calendar and appointment scheduling system is solved by an appointment scheduler (cf. figure 3, q.v. 4.1). This can either be a global software component with which the calendar agents can communicate or each calendar agent has a local copy of an appointment scheduler. In the latter case, negotiation about a possible appointment time is organized via a specific communication strategy (i.e., a negotiation protocol) for the calendar agents.

However, an appointment scheduler is a software component that can be realized as a constraint solver. When realizing such a constraint solver, beginning time, ending time, and duration of appointments, and additionally, different appointments' relationships (e.g., overlapping of two appointments) are described.

An appointment scheduler would usually reason about abstract ontological primitives (i.e., timepoints and intervals of time) associated with a properly defined temporal structure. The ontological primitives are specified by (integer) numbers. Precisely because appointments are usually defined in terms of calendars, a temporal structure that properly fits the problem of appointment scheduling must be linked to calendar systems. That means, an automated appointment scheduler must provide a convenient and flexible way for managing the characteristics of temporal concepts belonging to different calendar systems. An abstract (logic-based) specification language for different calendar systems (i.e., a temporal formalism) is Ohlbach's and Gabbay's calendar logic [144] (see also 2.2.3.1). This calendar logic links a temporal logic with the semantics of temporal concepts of calendar systems. In doing so, time units of different calendar systems along with each of their particularities (e.g., leap years in the Islamic calendar) can be specified as temporal intervals of different lengths on the underlying linear time-line. This allows for temporal reasoning about the meanings of calendar systems in an automated appointment scheduler. Furthermore, this logic allows for specifying many temporal concepts (e.g., lunch-time, tee-time). Beyond this, this logic allows for checking whether a given timepoint lies within a specified temporal interval or not, and if an inference rule holds between two temporal concepts, or not.

Let us consider the usefulness of this calendar logic for automated appointment scheduling more detailed. (Further features of Ohlbach's and Gabbay's logic are collected in section 2.2.3.1.) Time units (i.e., temporal granularities) such as Gregorian weeks or Islamic months are defined as independent partitions (of different lengths) of an underlying linear time-line. They can be mapped to the underlying linear time-line and back to some time unit by providing two functions for each considered time unit. One function that gives the number of seconds (or another basic time unit) elapsed from a chosen origin in time to the given date. The other function can map back this number to a common date format. On the underlying time-line, time units are referred to as integers. Using these two functions, conversions between different calendar systems become possible. For this reason, this calendar logic is adequate for defining temporal constraints (translated into intervals of integer numbers by the named functions) that can be handled by an appointment scheduler. Initially that means, any user can specify temporal constraints, and can use convenient temporal concepts (e.g., daytimes, public holidays) in the calendar system the user is familiar with. Furthermore, the appointment scheduler can reason about ontological primitives (i.e., timepoints and inter-

vals) on an underlying linear time-line using integer numbers, thanks to the calendar logic's mapping functions. In this way, a constraint solver for temporal reasoning and another for planning timetables, both developed at the Institute for Computer Science at the Ludwig-Maximilians university in Munich ¹⁴ can be exploited together with the mentioned calendar logic for the attempt of designing an appointment scheduler. That means, the proposed calendar logic need to be implemented within such a constraint solver. Initially, an origin in time (e.g., 1.1.1970, Greenwich Mean Time) need to be specified. The origin is moved to point 0 on the underlying time-line. That is, 0 is the coordinate of the origin. Similarly, any time unit gets an own coordinate system relative to the chosen origin. Subsequently, for any time unit of any considered calendar system two functions need to be implemented: One that maps a time unit to the underlying linear time-line by specifying half-open intervals located on this time-line, and the other that maps such an half open interval back to a common time unit. Within these functions, all characteristics of any considered time unit associated with a calendar system such as leap years, irregularities on months, or time zones need to be implemented. We exemplarily show this in example 4.1.

Example 4.1 *Let us consider how functions mapping a time unit to the underlying linear time-line, and back to a common time-unit can be realized. In the following, Islamic years are considered. (The following example is given in an abstract syntax, abutted to Prolog.): An arbitrary date is moved to the 0 coordinate. The computing time unit is day (i.e., day is the basic time-unit).*

```
islamic - year[[(Year) ->
[begin - isalmic - year([1, 1, Year], in_days), end - isalmic - year([1, 1, Year], in_days)].
begin - isalmic - year([1, 1, Year], in_days) ->
in_days is
(Year - 1) * 354 + non-leap days in prior years
(11 * Year + 3) div 30 leap days in prior years
+/- days before/after start of calendar (with respect to the 0 coordinate).
end - isalmic - year([1, 1, Year], in_days) ->
Year1 is Year + 1 the end of the year Year
Absolute_date is
(Year1 - 1) * 354 + non-leap days in prior years
(11 + Year1 + 3) div 30 leap days in prior years
+ days before start of calendar

islamic - yearN([b, e]) ->
islamic - year(in_days_b, Year).
islamic - year(in_days_b, Year) ->
Approximation is (in_days_b - days before start of calendar) div 354.
```

In this subsection, we have briefly addressed the working of an automated appointment scheduler. This appointment scheduler can be realized by using already existing reasoners (i.e., constraint solvers) for time and planning tasks in conjunction with a logic-based specification language of calendar systems (i.e., a temporal formalism). For the described appointment

¹⁴<http://www.pst.informatik.uni-muenchen.de/personen/fruehwir/buch.html>

scheduler, Ohlbach's and Gabbay's calendar logic [144] is used. This calendar logic allows for expressing temporal constraints by time units and other temporal concepts of different calendar systems. The semantics of time units are specified by functions that map them back and forth to a linear time-line. On this time-line, the time units are specified by intervals of integers. The appointment scheduler is reasoning about these intervals. These functions can be implemented in conjunction with the constraint solver since this calendar logic can be realized using a logic-based programming language, say Prolog. To put it in a nutshell, the appointment scheduler is a combination of a proper temporal reasoner, realized as a constraint solver, and the means of the referenced calendar logic.

4.2.2 The Structure of Appointment Times

In this subsection, the structure of user-defined temporal constraints about which the described appointment scheduler should be able to reason, and the structure of appointment themselves are investigated. For this purpose, we initially discuss the structure of appointments, and subsequently, we analyze the possible structure of user-defined temporal constraints derived from the structure of appointments. This discussion aims at a proper classification of the different structuring features of user-defined constraints on appointments. Examples of the different structural classes are expressed in Ohlbach's and Gabbay's calendar logic [144].

4.2.2.1 Appointments In this paragraph, the fundamental structure of appointments is discussed.

An appointment is a convex interval that has essentially three characteristics: a *beginning time*, an *ending time* and a particular *duration*. For example, a meeting on Friday, August, 1st, 2002 between 10.30 and 11.45 is an appointment that begins at 10.30 on the specified day (i.e., August, 1st, 2002) and ends at 11.45 on the same day. The appointment's duration is 1 hour and 15 minutes. That is, as well the beginning time as the ending time of a particular appointment are usually defined by a date and a time with respect to a particular calendar system (in this example the Gregorian calendar). In our common sense of time, both the beginning time and the ending time are referred to as timepoints, and the beginning time is temporally before the ending time. The appointment itself is an interval of time anchored on an underlying time-line that is fully described by two timepoints, the beginning time and the ending time. In such an interval both the beginning time and the ending time is comprised. Therefore, such an interval is left and right closed. Furthermore, any appointment has a specific duration (e.g., 1 hour and 15 minutes). A duration is a time span that is not anchored on the underlying time-line (i.e., an unanchored interval). A duration is usually related to an appointment as a whole. Compared to durations, some beginning time and some ending time, respectively are only related to a particular timepoint of some appointment's interval. Durations are expressible in Allen's interval formalism (cf. 2.2.1.2) describing all possible relations between pairs of intervals. Furthermore, arithmetic operations can be applied to durations since they can be added and scaled.

So far, this paragraph has summarized the structure of appointments which are convex intervals. These convex intervals are composed of two timepoints, one specifying the beginning and the other the ending of an appointment, and a specific duration.

4.2.2.2 User-defined Temporal Constraints If specifying temporal constraints for a planned appointment, people frequently use in addition to times (e.g., 3 p.m.), names of weekdays (e.g., Monday), names of months (e.g., August), names of holidays (e.g., easter), daytimes (e.g., afternoon), deictic times (e.g., next Monday), and even vague temporal terms (e.g., late in the afternoon). (Vague temporal terms are however out of the scope of this work). These temporal concepts depend on both a particular calendar system and on the user's individual, cultural, legal, and business environment. We assume that the semantics of these temporal concepts of calendar systems are specified in terms of the referenced calendar logic and stored, so that they are reasonable for the previously described appointment scheduler. That is, we emanate from a sharable, reusable, and communicatable temporal formalism used by the personal calendar agents and their associated appointment scheduler.

In principle, temporal constraints in the context of appointment scheduling can be specified by a single timepoint (e.g., August, 6th, 12 a.m.), or by a set of timepoints specifying temporal intervals within which an appointment must lie (e.g., between April and September), or by a temporal duration (e.g., two weeks) associated with an appointment. Temporal constraints for specifying a duration can additionally be used for describing the location of a particular timepoint (e.g., five days after new-year). Additionally, it must be possible to determine an appointment that takes just some hours of a day, or an appointment that protects several days such as a conference. Beyond this, regularly recurring appointments such as a professor's consulting hour are frequently specified.

In this paragraph, the structure (i.e., the different appearances) of user-defined temporal constraints is investigated. For this purpose, such temporal constraints are properly classified. Such temporal constraints are good for specifying appointments in the context of automated appointment scheduling. Therefore, the appointment scheduler must be able to reason about these temporal constraints. Initially, durations of appointments are classified. Subsequently, a classification of timepoints for specifying appointments is given. Examples are expressed in Ohlbach's and Gabbay's calendar logic [144].

Durations In the following, we discuss how durations of an appointment can be specified.

1. **Duration:** In the majority of cases, a user specifies the possible duration of a planned appointment by a number and one or more time units, for example, the meeting lasts maximal 2 hours and 30 minutes, i.e., the duration of the planned meeting is 2:30 hours.
2. **Range of durations:** Sometimes, it can become necessary to specify a range of the possible duration of a planned appointment (e.g., the conference lasts two to three days) since no more information is available at the time when specifying the appointment. For such a duration the user specifies two boundaries. That is, the range of the duration in the previous example is 2 days to 3 days, i.e., a duration that is specified by two boundaries, each comprising a number and a time unit.
3. **Open range of durations:** In addition to the previously considered durations which are specified by two boundary points, people sometimes specify appointments only by determining one boundary point such as the meeting lasts less than an afternoon. In such open durations, the specified boundary point can either be exclusive (i.e., not lying in the specified duration), as it is the case for less-than and more-than, or it can be inclusive (i.e., lying in the specified duration), as it is the case for at-least and at-most

(e.g., the meeting should last at most one afternoon). Such temporal constraints are difficult to express, since it is not possible to specify the amount of time that must be added and subtracted, respectively to the exactly specified temporal interval. That is, such temporal constraints are vague. (They are out of the scope of this work.)

4. **Disjunction of durations:** Furthermore, people sometimes specify the duration of an appointment by naming a disjunction of possible durations such as the meeting lasts one morning or one afternoon. That is, it can become necessary to deal with alternatives.

Since durations are time spans which are not anchored on an underlying time-line, they cannot be specified by using Ohlbach's and Gabbay's calendar logic [144]. However, we can use Allen's interval formalism for specifying unanchored intervals (i.e., free floating intervals) [1] (see also section 2.2.1.2). Allen's relations between such free floating intervals can be specified along with the previously described appointment scheduler.

Timepoints For describing the temporal location of a particular appointment, numerous temporal constraints are possible. In the following, we discuss the possible specifications of single timepoints and sets of timepoints, respectively when someone specifies the temporal location of an appointment. Specifying the temporal location of an appointment, people use a wide variety of temporal concepts such as lunch-time, afternoon, names of public holidays, or names of weeks and months. In the following, it is simply assumed that such temporal concepts of different calendar systems are defined in terms of the referenced calendar logic, stored, and made reasonable through knowledge bases.

In the easiest case, a person specifies the date and a duration for the planned appointment on the specified date. For example, a person specifies a meeting on Monday, August 5th in 2002 that has a maximal duration of half an hour.

Turning our attention to more complex specifications of temporal constraints for a planned appointment, we can figure out eight different classes. In order to have a better conceivability of such user-defined temporal constraints, and since the described appointment scheduler is associated with a specification language for calendar systems, some examples are expressed in Ohlbach's and Gabbay's calendar logic [144]. The examples chosen in the following classification are all in the context of the Gregorian calendar. However, such temporal constraints can be similarly formulated in any other calendar system using the referenced calendar logic.

1. **Intervals specified by time units:** Examples for such temporal constraints are:

- this year in August; i.e., in calendar logic (assuming that the function $August(x_{year})$ is somewhere defined and stored):

$$[2002, year]:August(x_{year})$$

- in the 33rd week

That is, appointments can be specified by determining the interval of a particular time unit. That is, a particular instance of a time unit such as the Gregorian month August as an instance of the time unit Gregorian month can be used. Usually, the interval's specific beginning time as well as the interval's specific ending time lies within the interval.

2. **Intervals with two specified ending points:** Examples for such temporal constraints are:

- between Monday and Thursday (this week); assuming that this week is the 33rd week in the year 2002, i.e., in calendar logic:

$$\{Monday(\text{week_within_year}(33,2002)) \cup \dots \cup Thursday(\text{week_within_year}(33,2002))\}$$

- today between 2 p.m. and 6 p.m.
- between today and the following weekend

That is, appointments can be specified by determining two timepoints between which the appointment must be temporally located. In this case, the interval's specific beginning time and the interval's specific ending time usually lies within the specified interval.

3. **Intervals with one specified ending point:** Examples for such temporal constraints are:

- tomorrow after lunch-time; assuming that today is August, 8th, 2002 i.e., in calendar logic:

assuming that tomorrow is defined as follows:

$$tomorrow(d) = d + 1$$

and that lunch-time between 13 and 14 o'clock is defined as follows:

$$lunch-time(d) = \text{hour_within_day}(d,13)$$

These two functions are provided by the calendar and appointment scheduling system. That means, they are specified in terms of the referenced calendar logic, and they are stored in some machine-processable knowledge bases.

Then the user-defined constraint is:

$$begin_hour(tomorrow(\text{August, 8th, 2002})) = lunch-time(tomorrow(\text{August, 8th, 2002}))$$

- before this week's Thursday
- through the following weekend

That is, the temporal location of appointments can be specified by determining only the beginning time and ending time, respectively within which an appointment must be scheduled. Furthermore, the beginning time and the ending time, respectively can be either contained within the specified interval or not contained within the specified interval.

4. **Shiftings:** Examples for such temporal constraints are:

- three weeks ago; assuming that this week is the 33rd week, i.e., in calendar logic:

$$weeks-ago(33) = 33 - 3$$

- two weeks after easter
- in one month

That is, appointments can be specified by a timepoint and a (positive or negative) duration of time that must be added and subtracted, respectively to the specified timepoint.

5. **Countings:** Examples for such temporal constraints are:

- the first Friday in September; is the first Friday after the beginning of September; i.e., in calendar logic:

$$\begin{aligned} &Friday(\text{begin_week}(\text{September}(2002))) < \text{September}(2002) ? \\ &Friday(\text{begin_week}(\text{September}(2002)) + 1) : \\ &Friday(\text{begin_week}(\text{September}(2002))) \end{aligned}$$

- the second week in the summer semester
- the first week after easter

That is, appointments can be specified by counting instances of a time unit relative either to another (usually comprising time unit such as month comprises weeks) or to a temporal concept such as a public holiday, a season, or an academic semester.

6. **Relational temporal constraints:** Examples for such temporal constraints are:

- the weekend around August, 16th 2002; i.e., in calendar logic: initially, the week containing August, 16th is defined as follows (the function week_N maps referencing timepoints to the week's coordinates, and the function $\text{day}_{[]}$ maps the day's coordinate to the underlying time-line):

$$\text{week}_N(\text{day}_{[]}^b(\text{August}, 16\text{th}, 2002))$$

Then the user-defined constraint is:

$$\begin{aligned} &\text{weekend}(\text{week}_N(\text{day}_{[]}^b(\text{August}, 16\text{th}, 2002))) = \\ &\{\text{Saturday}(\text{week}_N(\text{day}_{[]}^b(\text{August}, 16\text{th}, 2002)))\} \cup \{\text{Sunday}(\text{week}_N(\text{day}_{[]}^b(\text{August}, \\ &16\text{th}, 2002)))\} \end{aligned}$$

- in the week before easter
- the Friday in two weeks

That is, the temporal location of an appointment can be specified by determining a particular timepoint and interval, respectively such as easter or August, 16th. Furthermore, a space of time (e.g., weekend) within which the planned appointment must be located relatively to the specified timepoint is given. The difference between shiftings and relational temporal constraints is, for relational temporal constraints a space of time within which the appointment must lie is additionally specified.

7. **Deictic times:** Examples for such temporal constraints are:

- this week; assuming that today is August, 8th, i.e, in calendar logic:

$$\text{week}_N(\text{day}_{[]}^b(\text{August}, 8\text{th}, 2002))$$

- the next month
- last year

That is, an appointment can be specified by deictic temporal constraints using temporal concepts such as next, this, or last. Such temporal concepts can be handled similarly to countings where the duration of the specified time unit is added or subtracted to the specified instance of this time unit. In the case of deictic temporal constraints, we need functions that specify temporal concepts such as next.

8. **Periodic temporal constraints:** Examples for such temporal constraints are:

- every Monday in the summer semester 2002; assuming that a function specifying the summer semester of a particular university is given that take a year coordinate and gives back an interval between two timepoints, i.e., in calendar logic:

$$\text{all-Mondays}(\text{summer-semester}(2002)) = \text{Monday}(\text{week_s}(\text{summer-semester}(2002)))$$

- the following three Mondays
- every Monday except for the first Monday in August

That is, an appointment that is recurring several times is specified by its particular recurring period of time (or periods if the temporal constraint is nested).

So far, a classification of the possible structure of user-definable temporal constraints, except for vague temporal expressions, is given.

A problem remains to the handling of open and/or closed intervals specified by one or two ending points. It is desirable to map all kinds of intervals to those which are closed either to the right or to the left side. Closed intervals are not desirable since subsequent closed intervals are not disjoint and problems such as the *Divided Instant Problem* (cf. example 2.3) arise. Beyond this, Ohlbach's and Gabbay's calendar logic allows only for half open intervals [144]. That is, if temporal constraints are expressed in this logic, intervals are always treated as being half open. However, using, for example, seconds as the basic time unit within the calendar logic, a difference between open and closed intervals no longer exists. That is, the problem of dealing with either closed or open intervals is irrelevant.

In this paragraph, the structure of user-defined temporal constraints is investigated by introducing eight different classes. This classification is deduced from the temporal aspects of appointments.

To summarize the previous discussion, the structure of appointments and constraints for specifying appointments has been classified. The different occurrences of appointments and temporal constraints are recognized by means of merely structural characteristics. For the examples in the previously suggested classification, the Gregorian calendar is used.

Initially, the structure of appointments has been discussed. Subsequently, we have investigated the structure of temporal constraints, an automated appointment scheduler must be able to reason about. Eight different structural classes have been discovered. In each class, an example has been expressed in a calendar logic proposed by Ohlbach and Gabbay [144]. This calendar logic, in turn, is linked with the described appointment scheduler. Throughout the classification several temporal concepts of calendar system (e.g., lunch-time, Christmas) have been used. These temporal concepts shall be specified in terms of the referenced calendar logic, and they shall be machine-processable for the described appointment scheduler.

Form the suggested classification it becomes clear, that such a wide variety of user-defined

temporal constraints only becomes possible for automated appointment scheduling if they can be formulated in a logic-based language (as the one suggested by Ohlbach and Gabbay [144]). Such a language must provide means for capturing the various temporal concepts of calendar systems, and conversions among them. Furthermore, such a language should be easily to merge with a constraint solver for temporal reasoning, aiming at a (partly) automated appointment scheduler since the appointment scheduler, in turn, must be able to reason about the previously addressed temporal constraints. This can be achieved by using the referenced calendar logic along with the afore described appointment scheduler.

4.2.3 A Hierarchy of Calendar Systems

Now we turn attention to the temporal concepts of calendar systems frequently used when specifying an appointment. For example, someone is planning a meeting in the week after Shroven Tuesday, then ‘Shroven Tuesday’ need to be afore specified in the referenced calendar logic so that some user simply can use this function and the appointment scheduler can reason about this its information. They shall be specified in terms of the referenced calendar logic. Furthermore, they must be stored in that they become easily processable by the described appointment scheduler. For this purpose, a possible hierarchical structuring of calendar systems. In particular, each calendar system comprises several temporal concepts, specified through cultural, legal, business, and individual aspects. We consider how such temporal concepts can actually become available for the appointment scheduler by providing a possible hierarchy of calendar systems.

So far, we have discussed the global structure of a distributed and dynamic calendar and appointment scheduling system. Furthermore, the structure of appointments and user-defined temporal constraints for appointment scheduling among arbitrary people aiming at a (partly) automated appointment scheduler are investigated. The previously assumed appointment scheduler allows for temporal reasoning about user-defined temporal constraints for some planned appointment. These temporal constraints can be formulated by using various temporal concepts of any calendar system, if the all-embracing calendar and appointment system (cf. figure 3) provides means for machine-processable knowledge bases of temporal concepts of any considered calendar system. The goal is to have a calendar and appointment scheduling system within which each owner of an electronic calendar simply uses the temporal concepts of calendar systems, and the appointment scheduler automatically processes them. That means, the temporal concepts of calendar systems must become accessible, reusable, and reasonable by the appointment scheduler and the users themselves. This section addresses this problem: Since people frequently use more than one calendar system, and since the calendar systems used are usually interrelated, we specify the interrelationships among different calendar systems. That means, a hierarchical structuring of calendar systems is proposed, so that they can be used by the described automated appointment scheduler. The goal of this subsection is to provide an abstract hierarchical structuring of calendar systems for temporal knowledge representation and reasoning within appointment scheduling. The subsequently proposed hierarchy can be realized by a machine-processable set of knowledge bases.

In this subsection, a hierarchized model of calendar systems aiming at a hierarchy of knowledge-bases providing machine-processable temporal concepts of calendar systems is discussed. The discussed hierarchy is motivated by an example.

An introducing Example Let us initially consider example 4.2 approaching the required knowledge about temporal concepts of calendar systems used by some person.

Example 4.2 *John is a scientific assistant at a university in Munich, Bavaria, Germany. He uses a personal electronic calendar system that allows (among other things) for scheduling appointments with any other person holding an electronic calendar. The following (and possibly further) temporal concepts are diarized in his electronic calendar:*

- *common calendar system in use in Germany: Gregorian calendar with German names for weekdays and months*
- *dates are usually itemized in a day-month-year format (e.g, 13.08.2002), and time in a 24 hour format (e.g, 16:06)*
- *the public holidays of the region or even the community John is living in, for example, Assumption Day (i.e., 15th August)*
- *beginning time and ending time of German daylight saving time*
- *begining time and ending time of the university's summerterm and its winterterm*
- *the time while the university is in session (and vacations during this time period for the running semester)*
- *the weekly recurring – possibly with some exceptions – courses John is holding in the running semester*
- *conferences John is going to join in in the running year*
- *further job-related calendar events such as exams John has to prepare or to join*
- *several personal registered calendar events such as birthdays John wants to be reminded at, the weekly time and date (with some exceptions) of a language course John is visiting in the current year, John's planned vacations, etc.*

On John's request, these and further temporal concepts can be automatically accessed, and subsequently diarized into his electronic calendar by his personal calendar agent, thanks to a machine-processable (knowledge-based) temporal formalism.

Starting with example 4.2, John uses time units (e.g., hours, months, years) of an underlying calendar system (i.e., the Gregorian calendar). These time units are implicitly used by John along with all their particularities in the calendar system's peculiarities such as varying lengths of different months or leap years. Furthermore, time units are used in a geographical context. For example, the local time depending on John's particularly used time zone and the daylight saving time. Time changes depending on time zones must be considered, for example, when John is planning an appointment with a colleague from New York via an automated appointment scheduler.

Additionally, John is using a time and date format which is common in Germany. Therefore, knowledge about cultural depending time and date formats is required for automated appointment scheduling.

Yet another problem arises due to the fact, that John probably specifies appointment times using German names for weekdays and months (e.g., Montag, or Juli). It would be desirable for him just to use these terms instead of specifying such functions any time he uses his electronic calendar or schedules appointments via an automated appointment scheduler.

So far, these are the basic temporal concepts of the Gregorian calendar and the local time John is (implicitly) using when he schedules appointments. Additionally, he is interested in the public holidays valid for the region or even community (i.e., in Bavaria, Munich) he is living and working in. Public holidays are not only concerned with a calendar system, but also with a particular state, region, or even city or community. That is, the semantics of public holidays are rather concerned with someone's cultural and legal scope than with a calendar system, although they are expressed in terms of a particular calendar system – usually the cultural community's commonly used one (in the previous example: the Gregorian calendar). In addition to the Gregorian calendar, and the geographically depending local times specifying John's context-depending time units, John is using further – more specialized – calendar systems. Since he is working at a university, he uses several temporal concepts such as an university's session or particularities of some lesson. Thus, in addition to the Gregorian calendar, John uses several academic-based temporal concepts specific in the context of his place of employment, i.e., a university in Munich. That means, John is using an Academic calendar fitting the semantics of lessons, semesters, the university's vacations, and further temporal concepts in the context of the university John is employed.

The last point in the previous example's list contains several personal temporal concepts such as birthdays. That is, in addition to the underlying calendar systems (i.e., in the example the Gregorian calendar and an Academic calendar), John uses several individual temporal concepts.

The Hierarchy From example 4.2 is clear, that a person usually uses more than one calendar system. That means, a person uses a particular calendar system common in the person's cultural, legal, and geographical environment in addition with the person's business and individual time-based particularities. The used calendar systems vary from quite common calendar systems to very specialized ones. For example, a Business calendar contains business-related temporal concepts such as working days and working months. These temporal concepts can be specialized by a medical practitioner specifying, for example, his individual consultation hours (i.e., the working days in this doctor's practice). Usually, time units of more special temporal concepts are defined in terms of more general temporal concepts. For example, a working week is a subset of a common calendar week excluding the weekend and any holidays. A particular doctor's consultation hours, in turn, can be defined in terms of working weeks or working days. That means, a Business calendar can be defined as a specialization of a universal calendar system, say the Gregorian calendar. And the Business calendar can be, in turn, further specified. This paragraph aims at a proper hierarchy of calendar systems describing their relationships.

In the following, we discuss a proper hierarchy of calendar systems. Such a hierarchy can be realized by a set of knowledge bases within which the temporal concepts of calendar systems are predefined. These temporal concepts shall be specified in terms of Ohlbach's and Gabbay's calendar logic [144]). In this way, the temporal concepts of calendar systems become easily to access, to understand, and to process by the described automated appointment scheduler. This is due to enable the use of temporal concepts of arbitrary calendar systems

as flexible and convenient as possible for specifying requests on one's own electronic calendar, and specifying temporal constraints for automated appointment scheduling among arbitrary people. That is, after having classified temporal aspects of appointments and user-defined constraints on appointments, the second step towards a calendar and appointment scheduling system, is to discuss a possible hierarchy of different calendar systems explicitly showing their interrelationships. This hierarchy leads to the development of a set of machine-processable knowledge bases.

In this paragraph, a complete hierarchy of calendar systems, underlying the assumption that the use of calendar systems initially vary depending on a cultural, legal, business, and individual temporal concepts is illustrated. This illustration (cf. figure 4) is subsequently discussed. From example 4.2 is clear, that temporal concepts initially depend on a particular calendar system (e.g., Gregorian calendar, Islamic calendar) and a local time. Subsequently, several temporal concepts are expressed in a cultural and legal context (i.e., the state, region, and community a person is living in). Furthermore, any person's place of employment determines a wide variety of temporal concepts. Finally, any person uses several individual temporal concepts. That particularly means, we can figure out a kind of hierarchy of different calendar systems in use by any person. The considered calendar systems are:

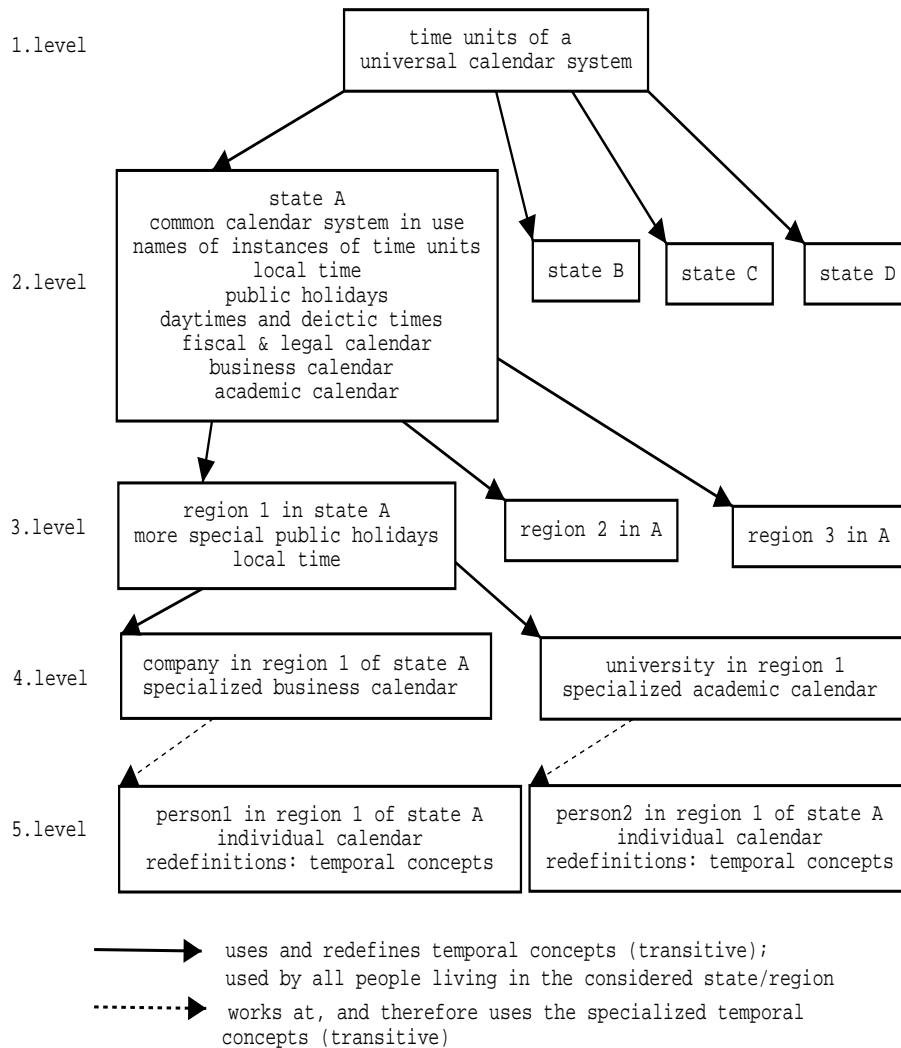
- universal calendar systems such as the Gregorian calendar or the Islamic calendar
- Legal calendars, for example, contain the definitions of public holidays within a particular state
- Fiscal calendars, for example, contain deadlines for paying taxes
- Business calendars, for example, contain definitions for working days, or weekend
- Academic calendars, for example, contain definitions for semesters and lessons
- individually defined temporal concepts within a particular calendar system

Basically, calendar systems are hierarchized from quite common ones to very specialized ones. Common, i.e., universal, calendar systems comprise temporal concepts valid for numerous people (e.g., time units of the Gregorian calendar). More specialized calendar systems comprise temporal concepts valid for a small group of people or even an individual person (e.g., academic cycles in a particular university).

For understanding the interrelationships of the different calendar systems, and how they can be hierarchized such that they become easily machine-processable by the described appointment scheduler, let us consider the illustration in figure 4.

Initially, the time units of any considered calendar system such as Islamic months, or Gregorian years along with all their particularities (e.g., leap years) must be specified. This is obtained by means of the appointment scheduler (cf. 4.2.1). Beyond this, the suggested hierarchy of temporal concepts of arbitrary calendar systems primarily depends on the cultural (e.g., commonly underlying calendar system with specific names for instances of time units), legal (e.g., public holidays) and geographical (e.g., time zone) particularities of any nation state. Therefore, the suggested hierarchy shown in figure 4 is conducted through the dependencies of some nation state.

In this way, the first level in figure 4 contains *time units of universal calendar systems*. That



Temporal concepts of higher levels can be redefined further below
Temporal concepts of lower levels have higher priority

Figure 4: Hierarchical structuring of calendar systems.

is, the essential time units (e.g., hours, Islamic months, Gregorian years) of any considered calendar system along with all their particularities (e.g., beginning of a day, varying lengths of different months, leap years, time zones) need to be initially defined and made accessible and processable. Using Ohlbach's and Gabbay's calendar logic [144] for specifying time units in our calendar and appointment scheduling system, time units along with all their particularities are implemented in the functions mapping time units back and forth between different calendar systems and the underlying linear time-line. In particular, designing an automated calendar and appointment system, these functions can be implemented in the appointment scheduler itself. The appointment scheduler, in turn, provides interfaces for these functions, so that they can be easily used when specifying temporal constraints for a planned appointment or specifying individual temporal concepts for one's electronic calendar.

Considering the second level in figure 4, labeled with *state*, temporal concepts should be separately defined for each state (e.g., Germany, USA, France) since most temporal concepts depend on the used language (e.g., names for instances of time units such as Tuesday or September, and names for daytimes and deictic times such as afternoon or tomorrow) and on cultural, legal, and geographical dependencies such as public holidays, financial deadlines for income taxes, or definitions for daytimes (e.g., evening). Obviously, the temporal concepts on this level – be it for working days, working weeks, and working months, or for public holidays in the particular state's legal and cultural context – are defined using time units of a particular calendar system commonly used in the particular state. Thus, particular definitions of time units from the first level in the illustrated hierarchy (cf. figure 4) are used for specifying the temporal concepts on this level (2). This level contains all calendar systems (e.g., Academic calendar, Financial calendar, Working calendar) which slightly differ from the associated state's commonly used calendar system (e.g., Gregorian calendar), but which are defined in terms of the common calendar system's time units. For example, a working week, say in Germany, is a Gregorian week excluding public holidays, leisure time, and the weekend.

Taking a step beneath to the third level, regions within a particular state are illustrated. That means, some cultural, legal, or geographical temporal concepts are valid only for a particular region, community and city, respectively in a particular state. For example, public holidays are not equal throughout a whole state. They are rather valid only for a particular region (although there is usually a subset of public holidays valid for a particular state as a whole). In several states (e.g., Germany), school vacations differ depending on the state's associated regions. Furthermore, several states such as the CIS have more than one local time (i.e., more than one time zone). That is, on this level, temporal concepts in addition to the ones defined on the associated state's level (2), or with a time-related semantics differing from some level further above are used.

The fourth level contains temporal concepts valid for a certain company, organization, institute, university, and others using slightly different definitions of temporal concepts defined further above in the illustrated hierarchy (cf. figure 4). For example, a bank probably has working hours slightly differing from the working hours defined on the associated state's level. Furthermore, some temporal concepts such as the time a particular university is in session cannot have a certain semantic until this level.

On the lowest level (i.e., the fifth level) in figure 4, there are different individual persons living in a particular region of the associated state. That is, such a person uses all the temporal concepts on the hierarchy (following the pointers in the illustration above) in his/her cultural, legal, geographical, and business context. For this purpose, the transactions, improvements,

and redefinitions of temporal concepts from a more general to a more special level (i.e., from the first level to the fifth level) must be transitive, so that any person can use the knowledge essential in his/her personal environment. Furthermore, any person defines individual temporal concepts (e.g., birthdays of some acquaintances, or individual planned vacations), and redefines some temporal concepts from higher levels. For example, someone has a slightly deviating perception of what time period comprises an evening.

Recapitulating the illustration and discussion above, all universally valid time units of any considered calendar system along with their particularities (i.e., the semantics contained in the first level of figure 4) should be implemented directly with the appointment scheduler since they are not redefinable. Furthermore, these definitions must contain means for converting back and forth among arbitrary calendar systems. A possible solution is suggested in Ohlbach's and Gabbay's calendar logic [144]. The subsequently discussed temporal concepts, each specified in a certain cultural, legal, geographical, business, or individual context, are defined in terms of universally valid time units of a particular calendar system. These temporal concepts can be hierarchized by considering some nation state's cultural, legal, geographical, and business context. Furthermore, both regions (or communities and cities) and organizations, companies, universities, and other institutions of the respective state must be considered regarding to the semantics of their used temporal concepts. On the lowest level, individual temporal concepts are specified which are only meaningful for a single person. However, any person uses several temporal concepts from the different levels above regarding to the state or even region the person is associated with.

The discussed hierarchy leads to a hierarchical ordered set of knowledge bases containing temporal concepts of arbitrary calendar systems following the discussed differentiation in figure 4. Within this hierarchy, temporal concepts can be redefined on lower levels. As usually, accessing temporal concepts comprised within different levels, the related temporal concept on the lowest level is chosen.

A concluding Example After having discussed a possible hierarchy of different calendar systems aiming at a set of knowledge bases, let us turn attention back to John's electronic calendar (cf. example 4.2). To recall this example's background, John is a scientific assistant at a university in Munich, Germany. Let us consider the hierarchy of temporal concepts of the different calendar systems John is using:

Example 4.3 *John, scientific assistant at a university in Munich, Bavaria, Germany (i.e., cultural, legal, geographical, and business context):*

1. level: *universal time units (along with all their particularities) of the Gregorian calendar and the local time in Munich, e.g.,
Gregorian weeks (containing seven days and begins on Monday),
Gregorian months (having different lengths),
Gregorian years (leap years).*
2. level: *temporal concepts commonly used in Germany:
instances of Gregorian calendar units such as Montag, or September,
Gregorian calendar: particular temporal concepts for Germany, e.g., daytimes such as afternoon or night,*

*Legal calendar, e.g., public holidays such as Christmas,
 Fiscal calendar, e.g., deadlines for advanced payments of different taxes,
 Academic calendar, e.g., semesters of universities, school year,
 Business calendar, e.g., working year, working month, working week, working day, working hour, weekend.*

3. level: *temporal concepts additionally used in Bavaria, e.g.,
 Legal calendar, e.g., public holidays such as Assumption Day,
 Academic calendar, e.g., particularities such as the time slots of Bavarian school vacations and terms of universities.*
4. level: *temporal concepts used in the university John is employed at, i.e., a specialization on Germany's common Academic calendar, e.g.,
 time the university is in session,
 lesson,
 redefinitions of working month, working week, working day, working hour in the universities context.*
5. level: *temporal concepts individually specified by John in terms of the Gregorian calendar, e.g.,
 birthdays of acquaintances,
 appointment at John's dentist,
 planned vacations,
 redefinition of weekend in John's personal context.*

We assume, that the automated calendar and appointment scheduling system used by John is able to access and reason about the context-dependent necessary time units, and further temporal concepts, provided by the appointment scheduler and the distributed knowledge bases.

Considering the previous example, it becomes clear, that we need a hierarchy of knowledge bases containing different calendar systems depending initially on cultural and legal semantics, valid in a certain state. It seems unhandy if, for example, Germans and Frenchmen access the same knowledge base of temporal concepts, say for public holidays or daytimes. In fact, temporal concepts become manageable for automated calendar and appointment scheduling systems, if they are from the beginning stored in their related semantical context. The context, in turn, mainly depends on some nation state's cultural, legal, and business realities, as founded in the previous discussion.

This paragraph contains an illustration of hierarchized temporal concepts of different calendar systems. We have shown, that we can deduce such a hierarchy from cultural, legal, geographical, business, and individual semantics of temporal concepts used by any person. The purpose of this illustrated and discussed hierarchy of temporal concepts is the development of a set of knowledge bases allowing for machine-processable temporal concepts of calendar systems which are frequently used by automated appointment scheduling in a distributed environment. In order to underline the proposed hierarchy of different calendar systems, we have illustrated an example.

In this subsection, a possible hierarchy for a large-scaled repository of knowledge bases containing the various temporal concepts of calendar systems has been proposed. The differ-

ent temporal concepts that must be stored within the knowledge bases can be expressed in Ohlbach's and Gabbay's logic-based specification language for calendar systems [144] that is usable for appointment scheduling in a distributed environment such as the Web. In this way, a hierarchy of knowledge bases, serving the needs of an automated calendar and appointment scheduling system can be realized.

In this section, the temporal aspects of a calendar and appointment scheduling system have been thoroughly investigated. For realizing the described appointment scheduling system, a logic-based specification language for calendar systems (i.e., a particular temporal formalism), and a set of machine-processable knowledge bases containing knowledge about temporal concepts of calendar systems is required. Since Ohlbach and Gabbay have proposed such a language, a calendar logic [144], we have used this language within the previous discussion. In particular, the temporal aspects of appointments an appointment scheduler must be able to reason about, and a hierarchy of calendar system that provides a starting point for developing a set of knowledge bases containing temporal knowledge that is meaningful for the appointment scheduler have been discussed.

In the first subsection, the basic functioning of a (partly) automated appointment scheduler has been addressed. The appointment scheduler is a temporal reasoner that is linked to the referenced calendar logic. In the second subsection, the structure of appointments and of temporal constraints specifying appointment times has been investigated. We have shown that appointments can be specified by the temporal location of their beginning time, ending time, and their particular duration. The different possibilities of specifying the beginning time, ending time, and the duration have been itemized. Different classes of temporal constraints have been suggested. The third subsection contains an analysis of the interrelationships of calendar systems. We have shown that different calendar systems are related to one another in that they build a hierarchy of calendar systems. In this hierarchy, calendar systems are sorted from their general to their specific usability depending on cultural, legal, business, and individual environments. The suggested hierarchy leads to the development of a set of machine-processable knowledge bases.

To summarize chapter 4, a Web-based calendar and appointment scheduling system has been described, and its temporal aspects have been investigated. Such an intelligent dynamic system asks for a logic-based specification language (i.e., a temporal formalism), and a large-scaled repository of knowledge bases containing the required knowledge about time. The temporal formalism used for the described appointment scheduling system is Ohlbach's and Gabby's calendar logic [144]. In particular, we have investigated the structure of appointments and of temporal constraints for specifying appointments an appointment scheduler must be able to reason about, and a possible hierarchy of calendar systems suggesting a starting point for developing a set of knowledge bases that contain predefined temporal concepts of calendar systems. The required means are derived from the meanings of calendar systems, and the requirements of (partly) automated appointment scheduling.

In the first section, a calendar and appointment scheduling system has been introduced. The system consists of personal electronic calendars, an appointment scheduler, and a set of

knowledge bases comprising definitions of temporal concepts of arbitrary calendar systems. Furthermore, the cooperation and communication among the main software components has been (briefly) explained. In the second section, the system's temporal aspects are investigated. For discussing temporal knowledge representation that properly fits the means of automated appointment scheduling, initially, the working of the appointment scheduler has been considered. Subsequently, the structure of appointments and of temporal constraints for specifying appointments have been investigated. Finally, a possible hierarchy of calendar systems have been suggested.

So far, the temporal characteristics of an appointment scheduling system have been investigated. Future works would contain the implementation of the described appointment scheduler. Furthermore, a set of knowledge bases containing predefined temporal concepts of different calendar systems conducted by the discussed hierarchy of calendar systems must be realized.

5 Conclusions

This thesis serves two objectives: First, an extensive survey on important temporal formalisms for representing and reasoning about time has been given, conjoining the ideas from the AI community with the ideas from the database community. This survey can be used simply as a work of reference. Furthermore, this survey helps whenever the properties of a temporal representation formalism for a particular time-dependent application need to be decided. Second, a temporal reasoning system, a (partly) automated appointment scheduling system, has been described. Means have been discussed for representing and reasoning about the system's temporal aspects of the required knowledge. The investigated temporal aspects are, in particular, temporal concepts of calendar systems and temporal constraints on appointments specified by means of temporal concepts of calendar systems. Knowledge representation of temporal information is highly required for such a system: A representation formalism, in particular, a logic-based specification language (i.e., a temporal formalism) for the purpose of expressing knowledge about arbitrary calendar systems and user-defined temporal constraints on appointments, and a large-scaled knowledge base repository containing the desired knowledge about temporal concepts of calendar systems are needed. That is, a temporal formalism for knowledge representation that is usable for appointment scheduling is required. The means such a temporal formalism must provide have been discussed. The temporal formalism used throughout the discussion is Ohlbach's and Gabbay's calendar logic[144]. The discussion can be used to implement an automated appointment scheduler for reasoning about temporally specified appointments, and to realized a large-scaled repository of a set of machine-processable knowledge bases containing the required knowledge about temporal concepts of calendar systems.

In the following, the main results of this thesis are synthesized. Furthermore, an overview on future research issues arising from the results is given.

5.1 Results

Throughout this thesis, the most important formalisms for representation of temporal knowledge in temporal reasoning systems have been presented, and each formalism's representational choices in the literature have been discussed. Approaches both from the AI community and from the database community have been considered. Each temporal formalism's applicability to the Web has been addressed since the Web – as any information system – is highly concerned with time.

Regarding to the considered temporal formalisms, a distinction between change-based temporal formalisms and time-based temporal formalism has been made. They differ in the representation of time: In time-based temporal formalisms the representation of time is concerned with nothing more than the mere flow of time, i.e., a constant change unaffected by anything else. In change-based temporal formalisms the representation of time takes place by means of entities, so-called change-indicators, which indicate that some change has occurred. Three well-known representatives of change-based temporal formalisms have been considered: The Situation Calculus, the Event Calculus, and Dynamic Logic. Within these formalisms, time is implicitly described by change-indicators, in particular, actions in the Situation Calculus, events in the Event Calculus, and programs in Dynamic Logic. Change-indicators affect a transition from one state in the modeled world to another. By this means, they make

statements about the world in change. Situation Calculus and Event Calculus are adapted for modeling, among other things, negotiation protocols for multi agent systems operating in distributed environments such as the Web. Even though Dynamic Logic is more concerned with program verification than with knowledge representation, it is the only temporal formalism for program verification used within AI systems.

Initially, any time-based temporal formalism has a precisely defined temporal structure which specifies the formalism's ontological primitives (i.e., timepoints and/or intervals), the characteristics of the flow of time (i.e., linear or branching, dense or discrete), and if time is bounded to its ends, or not. Starting with a properly specified temporal structure, change is manifested by the fact that the truth value of a proposition changes over time. In this way, time-based temporal formalisms are concerned with nothing more than the mere flow of time. Regarding to the ontological primitives, it seems to be generally agreed that the interval is the fundamental primitive representing the extent of things in time. However, for many applications a representation of timepoints is also required. In addition to a temporal structure, different temporal dimensions, in particular transaction time, valid time, and event time providing semantics for timestamping time-dependent objects have been presented. These semantics lead to a multidimensional representation of timestamped objects. The temporal dimensions allow for describing the evolution of an object over time. This, in turn, enables version management that is actually important for many applications such as budgeting, or data and information management in Web repositories. Finally, representation schemes for calendar systems usable in addition to any temporal structure have been addressed. Such representation schemes are (logic-based) descriptions of the characteristics of temporal concepts (e.g., time units) of different calendar systems. Temporal concepts of calendar systems can be described as independent partitions of an underlying time-line. Representations of calendar systems are required for any time-dependent application with human interactions since people usually express the temporal extents of things by means of some calendar system.

After having considered the most important temporal formalisms for representing and reasoning about time, the main conclusion of this survey is, that it seems unrealistic to ask for a general temporal formalism achieving a satisfactory approach independently of a particularly intended application. Thus, it is at most reasonable to consider the different approaches of temporal representation formalisms as a set over which a sensible decision must be made, and thus, achieving the most properly fitting temporal formalism for representing and reasoning about the temporal information within some intended application.

To accredit and emphasize the result above, Web-based planning and scheduling systems have been considered for addressing the meaningfulness of temporal formalisms in a particular application. Three different scenarios describing the working of different dynamic applications, in particular, appointment scheduling, event planning, and budgeting have been described. Although these three applications, all ask for a logic-based temporal formalism satisfying each of the application's underlying temporal aspects, the required temporal formalisms crucially vary with regard to each application's time-dependent aspects. We have picked out one of the mentioned applications: The appointment scheduling system. This system asks for temporal knowledge representation. In particular, a logic-based specification language (i.e., a temporal formalism) and a set of knowledge bases are required. Initially, the basic functioning of an automated appointment scheduler reasoning about various temporal aspects of appointment has been described in that this software component can easily be implemented. The appointment scheduler can be realized as a temporal reasoner in conjunction with a properly fitting

temporal formalism. The temporal formalism used throughout the discussion about the appointment scheduling system is Ohlbach's and Gabbay's calendar logic [144]. The temporal aspects within a calendar and appointment scheduling system have been thoroughly investigated by discussing the structure of appointments the appointment scheduler must be able to reason about, and a possible hierarchy of calendar systems containing the appointment scheduler's required knowledge about temporal concepts of calendar systems. This investigation has been conducted by the requirements of (partly) automated appointment scheduling in a distributed environment such as the Web. In the course of this investigation, an automated appointment scheduler has been described to such an extent that it can be easily realized. Furthermore, a large-scaled knowledge-based repository of highly expressive knowledge about temporal concepts of calendar systems derived from the suggested hierarchy of calendar systems can be developed. Merging these components, the described calendar and appointment scheduling system can be realized.

5.2 Future Works

Starting again with the fundamentals for building a temporal reasoning system, the notion of time must be formalized by using a (logic-based) language for representing the temporal aspects of the modeled knowledge. Furthermore, means for reasoning about the temporal aspects using such a knowledge representation language must be provided. Thus, for realizing the thoroughly investigated calendar and appointment scheduling system that bases on knowledge representation of temporal information and reasoning about such information, some future tasks must be achieved:

- The described (partly) automated appointment scheduler that is linked with the referenced calendar logic must be implemented so that it provides means for reasoning about arbitrarily defined appointments.
- A large-scaled knowledge-based repository derived from the hierarchy of calendar systems that has been discussed in this thesis need to be developed. This repository must contain machine-processable knowledge about calendar systems specified in the referenced calendar logic which the appointment scheduler can automatically access and process.

These two items – thoroughly described within this thesis – are required for realizing the discussed calendar and appointment scheduling system. For realizing the appointment scheduler, the discussed functioning of this software component must be implemented as a constraint solver. The appointment scheduler is reasoning about user-defined temporal constraints on appointments by exploiting the knowledge that is reasonably provided by a set of machine-processable knowledge bases.

Furthermore, if the described calendar and appointment scheduling system should be realized as a whole, an appropriate communication and negotiation strategy for the mentioned acting calendar agents must be decided and realized. These calendar agents, in turn, must be implemented. However, the calendar and appointment scheduling systems goes far beyond the objectives of this thesis, since temporal representation formalisms for temporal reasoning systems have been discussed rather than multi agent systems and negotiation protocols.

Beyond the calendar and appointment scheduling system, two further applications have been mentioned: An event planning system and a budgeting system. Similar to the provided investigation of temporal knowledge representation in an appointment scheduling system, the aspects of temporal knowledge representation within these two application need to be investigated. Although these two applications also ask for a temporal formalism that enables representation of knowledge about temporal aspects of calendar systems, we have shown that these temporal formalisms must be by far complexer than for automated appointment scheduling. In particular, event planning asks for a temporal formalism that allows for planning and managing a potential large and complex set of event plans. Therefore, a usable temporal formalism must be able to manage arbitrarily ordered non-convex intervals, and to describe a set of activities which are related with a particular plan (i.e., the planning process). Additionally, a branching flow of time must become manageable since planning is usually concerned with possible alternatives in the planning process. The budgeting systems asks for a temporal formalism that allows for managing plans, and additionally, means for representing an object's history in time. Such an object can be timestamped with more than one temporal dimension. That means, budgeting systems require a multidimensional temporal formalism. Thus, automating the tasks of event planning and budgeting, for each of these temporal reasoning systems a (logic-based) specification language (i.e., a temporal formalism) must be provided that catches the rich notion of time appearing in such systems.

References

- [1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
- [2] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.
- [3] J. Allen. Planning as temporal reasoning. In *Proceedings of the KR-91*, pages 3–14, 1991.
- [4] J. Allen. Time and time again: The many ways to represent time. *Journal of Intelligent Information Systems*, 6(4):341–356, 1991.
- [5] J. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.
- [6] J. Andany, M. Leonard, and C. Palisser. Management of schema evolution in databases. In *Proceedings of the Conference on Very Large Databases, Barcelona, Spain*, pages 161–170, 1991.
- [7] T. Anderson. Modeling events and processes at the conceptual level. In *Proceedings of the 2nd International Conference on Databases. Ed. S.M. Deen and P. Hammersley. The British Computer Society. Cambridge, Great Britain: Wiley Heyden Ltd.*, pages 151–168, 1983.
- [8] K. Apt and M. Bezem. Acyclic programs. *New Generation Computing*, 9(3-4):335–363, 1991.
- [9] A. Baker. Nonmonotonic reasoning in the framework of the situation calculus. *Artificial Intelligence*, 49(1-3):5–23, 1991.
- [10] K.van Belleghem, M. Denecker, and D. DeSchreye. Combining situation calculus and event calculus. In *Proceedings of the International Conference on Logic Programming, L.Sterling, editor, MIT Press*, pages 83–97, 1995.
- [11] J.van Benthem. *The logic of time*. D. Reidel Publishing Company, 1983; revised and expanded edition, 1991.
- [12] E. Bertino. A view mechanism for object-oriented databases. In *Proceedings of the International Conference on Extending Database Technology, Vienna, Austria*, pages 136–151, 1992.
- [13] 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.
- [14] C. Bettini, S. Jajodia, and S. Wang. *Time granularities in databases, datamining, and temporal reasoning*. Springer Verlag, Berlin, 2000.

- [15] C. Bettini and R. De Sibi. Symbolic representation of user-defined time granularities. In *Proceedings of the 6th International Workshop on Temporal Representation and Reasoning, Orlando, Florida*, pages 17–28, 1999.
- [16] C. Bettini, X. Wang, and S. Jajodia. Testing complex temporal relationships involving multiple granularities and its application to data mining. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM Press, Montreal, Canada*, pages 68–78, 1996.
- [17] C. Bettini, X. Wang, and S. Jajodia. A general framework for time granularity and its application to temporal reasoning. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):29–58, 1998.
- [18] A. Bochman. Concerted instance-interval temporal semantics: Temporal ontologies. *Notre Dame Journal of Formal Logic*, 31(3):403–414, 1990.
- [19] Boutilier, C. et al. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the 17th AAAI / 12. IAAI 2000: Austin, TX, USA*, pages 355–362, 2000.
- [20] M. Broxvall and P. Jonsson. Towards a complete classification of tractability in point algebras for nonlinear time. In *Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming*, pages 129–143, 1999.
- [21] Burgard, W. et al. The interactive museum tour-guide robot. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.
- [22] J. Campos and M. Silva. *Versus: A temporal Web repository*. Universidade de Lisboa, 2001.
- [23] W. Cellary and G. Jomier. Consistency of versions in object-oriented databases. In *Proceedings of the VLDB Conference*, pages 432–441, 1990.
- [24] I. Cervesato, L. Chittaro, and A. Montanari. A modal calculus of partially ordered events in a logic programming framework. In *Proceedings of ICLP-95: 12th International Conference on Logic Programming, L. Sterling, editor, MIT Press*, pages 299–313, 1995.
- [25] I. Cervesato, L. Chittaro, and A. Montanari. A general modal framework for the event calculus and its skeptical and credulous variants. In *Proceedings of the 1994 Joint Conference on Declarative Programming - GULP-PRODE'94*, pages 336–350, 1994.
- [26] I. Cervesato, A. Montanari, and A. Proveti. On the non-monotonic behavior of event calculus for deriving maximal time intervals. *Interval Computations*, 3(2):83–119, 1993.
- [27] S. Chakravarthy and S.-K. Kim. Resolution of time concepts in temporal databases. *Information Sciences*, 80(1-2):91–125, 1994.
- [28] R. Chandra, A. Segev, and M. Stonebreaker. Implementing calendars and temporal rules in next generation databases. In *Proceedings of the International Conference on Data Engineering, Houston, Texas, USA*, pages 264–273, 1994.

- [29] S.-Y. Chien, V. Tsotras, and C. Zaniolo. *A Comparative study of version management schemes for XML documents*. Time Center, Technical Report, TR-51, 2000.
- [30] L. Chittaro, A. Montanari, and A. Provetti. Skeptical and credulous event calculi for supporting modal queries. In *Proceedings of ECAI-94: 11th European Conference on Artificial Intelligence*, A.Cohn, editor, John Wiley and Sons, pages 361–365, 1994.
- [31] J. Chomicki and G. Saake. *Logics for database and information systems*. Kluwer Academic Publishers, Boston/Dordrecht/London, 1998.
- [32] Ciapessoni, E. et al. Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of Computer Programming*, 20(1–2):141–171, 1993.
- [33] J. Clifford and A. Rao. A simple general structure for temporal domains. In *C. Rolland, and M. Leonard, editors, Temporal Aspects of Information Systems*, Elsevier Science Publishers B.V., IFIP, pages 17–28, 1987.
- [34] G. Copeland and D. Maier. Making smalltalk a database system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Boston, MA*, pages 316–325, 1984.
- [35] W. Davis and J. Carnes. Clustering temporal intervals to generate reference hierarchies. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, J.Allen, et al., editors, Morgan Kaufman, pages 111–117, 1991.
- [36] T. Dean and M. Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36(3):375–399, 1988.
- [37] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus. In *Proceedings of ECAI-92, Vienna, Austria*, pages 384–388, 1992.
- [38] N. Dershowitz and E. Reingold. Calendrical calculations. *Software-Practice and Experience*, 20(9):899–928, 1990.
- [39] Doucet, A., et al. Integrity constraints and versions. In *Proceedings of the 6th International Workshop on Foundations of Models and Languages for Data and Objects, Integrity in Databases, Dagstuhl Castle, Germany*, pages 25–39, 1996.
- [40] F. Dougliis and T. Ball. An internet difference engine and its applications. In *Digest of Papers, COMPCON-96*, pages 71–76, 1996.
- [41] Dougliis, F. et al. WebGUIDE: Querying and navigating changes in web repositories. In *Proceedings of the 5th International WWW Conference, Paris, France, Computer Networks 28(7-11)*, pages 1335–1344, 1996.
- [42] Dougliis, F. et al. The AT&T internet difference engine: Tracking and viewing changes on the web. *World Wide Web*, 1(1):27–44, 1998.
- [43] M. Dumas, M.-C. Fauvet, and P.-C. Scholl. *TEMPOS: A temporal database model seamlessly extending ODMG*. Research Report, 1013-ILSR -7, LSR-IMAG, Grenoble, France, 1999.

- [44] Dumas, M. et al. TEMPOS: managing time and histories on top of OO-DBMS. In *Proceedings of EDBT'98 demo session, Valencia, Spain, 1999*.
- [45] C. Dyreson. Towards a temporal World Wide Web: A transaction-time server. In *Proceedings of the Australian Database Conference, Gold Coast, Australia, pages 290–301, 2001*.
- [46] C. Dyreson and R. Snodgrass. Supporting valid-time indeterminacy. *ACM Transactions on Database Systems*, 23(1):1–57, 1998.
- [47] C. Dyreson, R. Snodgrass, and M. Freiman. *Efficiently supporting temporal granularities in a DBMS*. FTP, Technical Report 95/7, James Cook University, Australia, 1995.
- [48] Dyreson, C. et al. *Efficiently supporting temporal granularities*. Time Center, Technical Report, TR-31, 1998.
- [49] N. Eisenger and N. Elshiewy. *MADMAN - multi-agent diary manager*. ESRC-92-7i Internal Report, 1992.
- [50] K. Eshghi. Abductive planning with event calculus. In *Proceedings of the 5th International Conference on Logic Programming, MIT Press*, pages 562–579, 1988.
- [51] C. Evans. Negation as failure as an approach for the Hanks and McDermott problem. In *Proceedings of the 2nd International Symposium on Artificial Intelligence, Monterrey, Mexico*, pages 1–24, 1989.
- [52] Extensible Markup Language (XML). <http://www.w3.org/XML>. W3C, 2000.
- [53] L. Fangzhen. Applications of the situation calculus to formalizing control and strategic information: The prolog cut. *Artificial Intelligence*, 103(1-2):273–294, 1998.
- [54] A. Fernandes, M. Williams, and N. Paton. A logic-based integration of active and deductive databases. *New Generation Computing*, 15(2):205–244, 1997.
- [55] R. Fikes, P. Hart, and N.J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.
- [56] 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.
- [57] M. Fisher and R. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
- [58] G. FitzPatrick. Calendaring and scheduling with XML-RDF. In *XML Conference and Exposition, Orlando, Fl, 2001*.
- [59] M. Franceschet and A. Montanari. A combined approach to temporal logics for time granularity. In *Workshop on Methods for Modalities 2, Amsterdam, NL. Submitted for publication, 2001*.
- [60] J. Funge. Representing knowledge within the situation calculus using interval-valued epistemic fluents. *Journal of Reliable Computing*, 5(1), 1999.

- [61] D. Gabbay and M. Reynolds. *Temporal Logic: Mathematical foundations and computational aspects*. Oxford University Press, 1994.
- [62] S. Gadia and C. Yeung. A generalized model for a relational temporal database. In *Proceedings of the International Conference on Management of Data, Chicago*, pages 251–259, 1988.
- [63] A. Galton. A critical examination of Allen’s theory of action and time. *Artificial Intelligence*, 42(2-3):159–188, 1990.
- [64] S. Gançarski. Database versions to represent bitemporal databases. In *Proceedings of Database and Expert Systems Applications (DEXA) Conference, Florence, Italy*, 1999.
- [65] M. Gelfond, V. Lifschitz, and A. Rabinov. What are the limitations of the situation calculus? In *Automated Reasoning, Essays in Honor of Woody Bledsoe, edited by S. Boyer, Kluwer Academic Publishers*, pages 167–181, 1991.
- [66] 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 14th International Joint Conference on Artificial Intelligence, Nagoya, Japan*, pages 1221–1226, 1997.
- [67] C. González and J. Escamilla de los Santos. A negotiation protocol for meeting scheduling based on a multi-agent system. In *1st Iberoamerican workshop on distributed artificial intelligence and multiagent systems, Xalapa*, 1996.
- [68] I. Goralwalla, M. Özsu, and D. Szafron. An object-oriented framework for temporal data models. In *Temporal Databases: Research and Practice, LNCS 1399*, pages 1–35, 1997.
- [69] Goralwalla, I. et al. Modeling temporal primitives: Back to basics. In *Proceedings of the 6th International Conference on Information and Knowledge Management (CIKM-97)*, pages 24–31, 1997.
- [70] Goralwalla, I. et al. Temporal granularity: completing the puzzle. *Journal of Intelligent Information Systems*, 16(1):41–63, 2001.
- [71] F. Grandi and F. Mandreoli. *The valid Web: It’s time to go*. Time Center, Technical Report, TR-46, 1999.
- [72] F. Grandi and F. Mandreoli. The valid web: An XML/XSL infrastructure for temporal management of web documents. In *Proceedings of the ADVIS 2000 - International Conference on Advances in Information Systems, Izmir, Turkey, Lecture Notes in Computer Science, Springer Verlag, Berlin*, pages 294–303, 2000.
- [73] C. Green. Theorem proving by resolution as a basis for question - answering systems. In *B. Meltzer D. Michie, editor, Machine Intelligence 4, Edinburgh University Press, New York*, pages 183–205, 1969.
- [74] H. Gregersen and C. Jensen. *Conceptual modeling of time-varying information*. Time Center, Technical Report, TR-35, 1998.

- [75] T. Gruber and G. Olsen. An ontology for engineering mathematics. In *Doyle and Torasso and Sandewall, editors, 4th International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann*, pages 258–269, 1994.
- [76] P. Haddawy. A logic of time, chance, and action for representing plans. *Artificial Intelligence*, 80(1–2):243–308, 1996.
- [77] D. Harel. *First-order dynamic logic*. Lecture Notes in Computer Science, Vol. 68. Springer-Verlag, New York, 1979.
- [78] D. Harel. Dynamic logic. In *D. Gabbay et al., editors, Handbook of Philosophical Logic, vol. II, Extensions of Classical Logic, Publishing Company, Dordrecht (NL)*, 1984.
- [79] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
- [80] P. Hayes. *A catalog on temporal theories*. Technical Report, UIUC-BI-AI-96-01, University of Illinois, 1995.
- [81] P. Hayes and J. Allen. A common-sense theory of time. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA*, pages 528–531, 1985.
- [82] P. Hayes and J. Allen. Short time periods. In *Proceedings of IJCAI-87, Los Angeles, CA*, pages 981–983, 1987.
- [83] P. Hayes and J. Allen. Moments and points in an interval-based temporal logic. *Computational Intelligence*, 5(4):225–238, 1989.
- [84] Hobbs, J. et al. DAML-S: Web service description for the Semantic Web. In *the 1st International Semantic Web Conference (ISWC), to appear*, 2002.
- [85] J. Hunt and J. Reuter. Using the web for document versioning. an implementation report for DeltaV. In *Proceedings of the 23rd International Conference on Software Engineering, Toronto*, pages 507–513, 2001.
- [86] S. Imfeld. *Time, points, and space - towards a better analysis of wildlife data in GIS*. PhD Thesis, Universität Zürich, 2000.
- [87] ISO-8601 Representation of dates and times. <http://www.iso.ch/markete/8601.pdf>. International Organization for Standardization, 2000.
- [88] Jenkin, M. et al. A logical approach to portable high-level robot programming. In *Proceedings of the 10th Australian Joint Conference on Artificial Intelligence, Perth, Australia. Invited paper*, 1997.
- [89] C. Jensen and C. Dyreson (editors). *The consensus glossary of temporal database concepts - February 1998 version*. <http://cs.engr.uky.edu/dekhtyar/685/papers/>, 1998.
- [90] C. Jensen and R. Snodgrass. Semantics of time-varying information. *Information Systems*, 21(4):311–352, 1996.
- [91] Ch. Jung, K. Fischer, and A. Burt. *Multi-agent planning using an abductive event calculus*. DFKI Research Report, RR-96-04, 1996.

- [92] Z. Kemp and A. Kowalczyk. Incorporating the temporal dimension in a GIS. In *Michael Worboys, editor, Innovations in GIS 1. Taylor & Francis, London, UK., Innovations in GIS 1, chapter 7. Taylor and Francis, London, 1994.*
- [93] L. Khatib. *Reasoning with non-convex intervals*. PhD Thesis, Florida Institute of Technology, 1994.
- [94] L. Khatib and R. Morris. Quantitative structural temporal constraints on repeating events. In *Proceedings of TIME-98*, pages 74–80, 1998.
- [95] L. Khatib and R. Morris. Generating scenarios for periodic events with binary constraints. In *TIME-99 (IEEE press), the 6th International Workshop on Temporal Representation and Reasoning, Orlando, FL*, pages 67–72, 1999.
- [96] I. Kiringa. Towards a general theory of advanced transaction models in the situation calculus. In *Proceedings of the 8th International Workshop on Knowledge Representation meets Databases, Rome, Italy, 2001.*
- [97] P. Klahold, G. Schlageter, and W. Wilkes. A general model for version management in databases. In *Proceedings of the 12th VLDB Conference, Kyoto, Japan*, pages 319–327, 1986.
- [98] N. Kline, J. Li, and R. Snodgrass. *Specifying multiple calendars, calendric systems, and field tables and functions in TimeADT*. Time Center, Technical Report, TR-41, 1999.
- [99] J. Koomen. Reasoning about recurrence. *Journal of Intelligent Information Systems*, 6:461–496, 1991.
- [100] M. Koubarakis. *Reasoning about time and change: A knowledge base management perspective*. University of Toronto, Canada, 1990.
- [101] R. Kowalski. Database updates in the event calculus. *Journal of Logic Programming*, 12(1-2):121–146, 1992.
- [102] R. Kowalski and F. Sadri. The situation calculus and the event calculus compared. In *M. Bruynooghe, editor, Proceedings of ILPS-94*, pages 539–553, 1994.
- [103] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.
- [104] S. Kraus, Y. Sagiv, and V. Subrahmanian. *Representing and integrating multiple calendars*. University of Maryland, Technical Report, CS-TR-3751, 1996.
- [105] P. Ladkin. Time representation: A taxonomy of interval relations. In *Proceedings AAAI-86, Philadelphia, PA*, pages 360–366, 1986.
- [106] P. Ladkin. *The logic of time representation*. PhD Thesis, University of California, 1987.
- [107] B. Leban, D. McDonald, and D. Foster. A representation for collections of temporal intervals. In *Proceedings of the AAAI-86*, pages 354–371, 1986.

- [108] V. Lifschitz. Towards a metatheory of action. In *J.Allen, R.Fikes, E.Sandewall, editors, Proceedings of the 2nd International Conference on Principles Knowledge Representation and Reasoning (KR-91), Los Altos, Morgan Kaufmann Publishers*, pages 376–386, 1991.
- [109] G. Ligozat. Generalized intervals: A guided tour. In *Proceedings of the ECAI-98 Workshop on Spatial and Temporal Reasoning, Brighton, UK*, 1998.
- [110] G. Ligozat and L. Vila. *Ontology and theory of time*. Working Draft, 1998.
- [111] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994.
- [112] F. Lin and Y. Shoham. Concurrent actions in the situation calculus. In *Proceedings of the 10th AAAI-92, San Jose, California*, pages 590–595, 1992.
- [113] F. Lin and Y. Shoham. Provably correct theories of actions. *Journal of the ACM*, 42(2):293–320, 1995.
- [114] H. Lin. *Efficient conversion between temporal granularities*. Time Center, Technical Report, TR-19, 1997.
- [115] Lum, V. et al. Designing DBMS support for temporal dimension. In *Proceedings of the ACM SIGMOD International Conference, Boston, MA*, pages 115–126, 1984.
- [116] H. Lévesque. What is planning in the presence of sensing? In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96), Portland, OR*, pages 1139–1146, 1996.
- [117] H. Lévesque, Y. Lespérance, and R. Reiter. A situation calculus approach to modeling and programming agents. In *In A. Rao and M.Wooldridge, editors, Foundations and Theories of Rational Agency*, pages 275–299, 1999.
- [118] H. Lévesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1998.
- [119] Lévesque, H. et al. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3):59–83, 1997.
- [120] Lévesque, H, et al. Ability and knowing how in the situation calculus. *Studia Logica*, 66(1):165–186, 2000.
- [121] P. Mateus, A. Pacheco, and J. Pinto. Observations and the probabilistic situation calculus. In *D. Fensel, et al., editors, Proceedings 8th International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann*, pages 327–338, 2002.
- [122] F. Mattern and P. Sturm. An automatic distributed calendar and appointment system. *Microprocessing and Microprogramming*, 27(1-5):455–462, 1989.
- [123] J. McCarthy. Actions and other events in situation calculus. In *to appear*, 2000.

- [124] J. McCarthy. *Situation calculus with concurrent events and narrative*. Stanford University, 2001.
- [125] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *B. Meltzer and D. Michie, editors, Machine Intelligence 4, Edinburgh University Press, Edinburgh, Scotland*, pages 463–502, 1969.
- [126] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.
- [127] S. McIlraith, T. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.
- [128] E. McKenzie and R. Snodgrass. *An evaluation of algebras incorporating time*. Technical Report, TR 89-22, Dept. of Computer Science, University of Arizona, 1989.
- [129] J. Meyer. Dynamic logic reasoning about actions and agents. In *Workshop on Logic-Based Artificial Intelligence, Washington, DC*, 1999.
- [130] R. Miller. Deductive and abductive planning in the event calculus. In *Proceedings of the 2nd AISB Workshop on Practical Reasoning and Rationality, Manchester, U.K.*, 1997.
- [131] R. Miller and M. Shanahan. The event calculus in classical logic — alternative axiomatizations. *Linköping Electronic Articles in Computer and Information Science*, 4(16), 1999.
- [132] L. Missiaen. *Localized abductive planning with the event calculus*. PhD Thesis, Department of Computer Science, K.U. Leuven, 1991.
- [133] A. Montanari, A. Peron, and A. Policriti. The way to go: Multi-level temporal logics. In *J. Gerbrandy, et al., editors, Liber Amicorum for the Fiftieth Birthday of Johan van Benthem, ILLC*, 1999.
- [134] A. Montanari and A. Policriti. Decidability results for metric and layered temporal logics. *Notre Dame Journal of Formal Logic*, 37(2):260–282, 1996.
- [135] R. Moore. A formal theory of knowledge and action. In *J.R.Hobbs and R.C.Moore, editors, Formal theories of commonsense world. Ablex, Norwood NJ*, pages 319–358, 1985.
- [136] R. Morris and L. Khatib. An interval-based temporal relational calculus for events with gaps. *Journal of Experimental and Theoretical Artificial Intelligence*, 3:87–107, 1991.
- [137] R. Morris, W. Shoaff, and L. Khatib. Domain independent reasoning about recurring events. *Computational Intelligence*, 12(3):450–477, 1996.
- [138] A. Mourelatos. Events, processes, and states. *Linguistics and Philosophy*, 2:415–434, 1978.
- [139] M. Nascimento and M. Eich. Decision time for temporal databases. In *Proceedings TIME-95, Melbourne Beach, FL*, pages 157–162, 1995.

- [140] I. Newton. Mathematical principles of natural philosophy. In *F. Cajori, editor*, 1936.
- [141] M. Niezette and J.-M. Stevenne. An efficient symbolic representation of periodic time. In *Implementing Temporal Reasoning: Workshop Notes, AAAI-92*, pages 130–140, 1992.
- [142] P. Ning, X. Wang, and S. Jajodia. An algebraic representation of calendars. In *the Annals of Mathematics and Artificial Intelligence (Kluwer)*, to appear, 2001.
- [143] H. Ohlbach. About real time, calendar systems and temporal notions. In *H. Barringer, et al., editors, Advances in Temporal Logic, Kluwer Academic Publishers*, pages 319–338, 1999.
- [144] H. Ohlbach and D. Gabbay. Calendar logic. *Journal of Applied Non-classical Logics*, 8(4):291–324, 1998.
- [145] J. Pinto and R. Reiter. Adding a time line to the situation calculus. In *Proceedings of the 2nd Symposium on Logical Formalizations of Commonsense Reasoning*, pages 172–177, 1993.
- [146] J. Pinto and R. Reiter. Temporal reasoning in logic programming: A case for the situation calculus. In *Proceedings of the International Conference on Logic Programming*, pages 203–221, 1993.
- [147] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):261–325, 1999.
- [148] V. Pratt. Semantical considerations on floyd-hoare logic. In *Proceedings of the 17th FOCS, IEEF*, pages 109–121, 1976.
- [149] J. Ramos. *The situation and state calculus: Specification and verification*. PhD Thesis, IST, Universidade Técnica de Lisboa, 2000.
- [150] D. Randall, H. Hamilton, and R. Hilderman. Generalization for calendar attributes using domain generalization graphs. In *Proceedings of the 5th International Workshop on Temporal Representation and Reasoning (TIME-98), Sanibel Island, Florida*, pages 177–184, 1998.
- [151] H. Reichgelt and N. Shadbolt. A specification tool for planning systems. In *Proceedings ECAI-90*, pages 541–546, 1990.
- [152] E. Reingold and N. Dershowitz. Calendrical calculations, II: Three historical calendars. *Software-Practice and Experience*, 23(4):383–404, 1993.
- [153] E. Reingold and N. Dershowitz. *Calendrical calculations: The millennium edition*. Cambridge University Press, 2001.
- [154] R. Reiter. On formalizing database updates. In *Proceedings of the 3rd EDBT-92, Vienna, Austria*, pages 10–20, 1992.
- [155] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64(2):337–351, 1993.

- [156] R. Reiter. On specifying database updates. *Journal of Logic Programming*, 25(1):53–91, 1995.
- [157] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proceedings of Common Sense 96: 3rd symposium on Logical Formalizations of Commonsense Reasoning, Stanford, CA*, pages 2–13, 1996.
- [158] R. Reiter. Knowledge in action. In *Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
- [159] J. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383–393, 1995.
- [160] J. Roddick. A model for schema versioning in temporal database systems. *Aust. Computer Science Communication*, 18(1):446–452, 1996.
- [161] F. Sadri and R. Kowalski. Variants of the event calculus. In *Proceedings of the International Conference on Logic Programming, L. Sterling, editor, MIT Press*, pages 67–81, 1995.
- [162] S. Schmeier and A. Schupeta. *PASHA – personal assistant for scheduling appointments*. German Research Center for Artificial Intelligence, 1996.
- [163] L. Schubert. Monotonic solution of the frame problem in the situation calculus. In *H.E. Kyburg, et al., editors, Knowledge Representation and defesible reasoning, Kluwer, Academic Publishers*, pages 23–67, 2000.
- [164] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1055–1060, 1989.
- [165] M. Shanahan. Representing continuous change in the event calculus. In *Proceedings of the European Conference on Artificial Intelligence*, pages 598–603, 1990.
- [166] M. Shanahan. Explanation in the situation calculus. In *Proceedings IJCAI-93*, pages 160–165, 1993.
- [167] M. Shanahan. A circumscriptive calculus of events. *Artificial Intelligence*, 75(2):249–284, 1995.
- [168] S. Shapiro, Y. Lespérance, and H. Lévesque. Goals and rational action in the situation calculus - a preliminary report. In *Working Notes of the AAAI Fall Symposium on Rational Agency: Concepts, Theories, Models, and Applications*, pages 117–122, 1995.
- [169] Y. Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33(1):37–63, 1987.
- [170] 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.
- [171] Sierra, C. et al. Descriptive dynamic logic and its application to reflective architectures. In *Future Generation Computer Systems 12*, pages 157–171, 1996.

- [172] R. Snodgrass and I. Ahn. A taxonomy of time in databases. In *Proceedings of the International Conference on Management of Data, Austin, Texas*, pages 236–246, 1985.
- [173] R. Snodgrass and I. Ahn. Temporal databases. *IEEE Transactions on Computers*, 19(9):35–42, 1986.
- [174] Snodgrass, R. (editor). *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.
- [175] Snodgrass, R. et al. The multical project. In [http : //www.eecs.wsu.edu/ cdyreson/pub/temporal/multical.htm](http://www.eecs.wsu.edu/cdyreson/pub/temporal/multical.htm), 1993–1997.
- [176] M. Soo and R. Snodgrass. *Mixed calendar query language support for temporal constants*. TempIS, Technical Report 29. Computer Science Department, University of Arizona, 1992.
- [177] 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.
- [178] S.Sen. An automated distributed meeting scheduler. *IEEE Expert*, 12(4):41–45, 1997.
- [179] K. Sycara and J. Liu. Distributed meeting scheduling. In *Proceedings of the 16th Annual Conference of the Cognitive Society*, 1994.
- [180] P. Terenziani. Integrating calendar dates and qualitative temporal constraints in the treatment of periodic events. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):763–783, 1997.
- [181] P. Terenziani. Integrated temporal reasoning with periodic events. *Computational Intelligence*, 16(2):210–256, 2000.
- [182] A. Tuzhilin and J. Clifford. On periodicity in temporal databases. *Information Systems*, 30(5):619–639, 1995.
- [183] L. Vila. Instants, periods and the dividing instant problem. In *IMACS International Workshop on Qualitative Reasoning and Decision Technologies (QUARDET-93)*, 1993.
- [184] L. Vila. IP - an instant-period-based theory of time. In *R.Rodriguez, editor, Proceedings of the ECAI-94 Workshop on Spatial and Temporal Reasoning*, pages 197–201, 1994.
- [185] L. Vila. A survey on temporal reasoning in artificial intelligence. *Artificial Intelligence*, 7(1):4–28, 1994.
- [186] L. Vila. Revisiting time and temporal incidence. In *F. Anger, editor, Proceedings of the AAAI-96 Workshop on Spatial and Temporal Reasoning*, 1996.
- [187] L. Vila and E. Schwalb. *A theory of time and temporal incidence based on instants and periods*. Department of Information and Computer Science, California Univ., Irvine, CA, 1996.
- [188] M. Vilain. A system for reasoning about time. In *Proceedings of the 2nd National (US) Conference on Artificial Intelligence, AAAI-82 , Pittsburgh, PA*, pages 197–201, 1982.

- [189] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In *D. S. Weld and J. de Kleer, editors, Readings in Qualitative Reasoning about Physical Systems*. Kaufmann, San Mateo, CA, pages 373–381, 1990.
- [190] X. Wang. Algebraic query languages on temporal databases with multiple time granularities. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, Baltimore, Maryland, pages 304–311, 1995.
- [191] X. Wang, S. Jajodia, and V. Subrahmanian. Temporal modules: An approach toward federated temporal databases. In *ACM SIGMOD International Conference on Management of Data*, Washington, D.C., pages 227–236, 1993.
- [192] R. Washington and B. Hayes-Roth. Incremental abstraction planning for limited-time situations. In *M. Ghallab and A. Milani, editors, New Directions in Planning*, IOS Press, Amsterdam, pages 91–102, 1996.
- [193] WebDAV. Corp. <http://www.webdav.org>, 2001.
- [194] J. Weber. On the representation of concurrent actions in the situation calculus. In *Proceedings of CSCSI-90*, Ottawa, Ontario, pages 28–32, 1990.
- [195] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with granularity of time in temporal databases. In *Lecture Notes in Computer Science, vol. 498, R. Anderson et al., editors, Springer-Verlag*, 1991.
- [196] Y. Wu, S. Jajodia, and X. Wang. *Temporal database bibliography update*. <http://www.isse.gmu.edu/csis/tdb/bib97/bib97.html>, 1997.
- [197] XFront. *XML Schema versioning*. <http://www.xfront.com/Versioning.pdf>, 2001.
- [198] XML Schema. <http://www.w3.org/XML/Schema>. W3C, 2001.
- [199] XMLSchema Part2. <http://www.w3.org/TR/2001/REC-xmlschema-2>. W3C, 2001.
- [200] Yan, J. et al. *An intelligent meeting scheduler*. Final Report, University of Alberta, 1999.
- [201] Yearsley, C. et al. Computational support for spatial information handling: Models and algorithms. In *Michael Worboys, editor, Innovations in GIS 1, chapter 6*. Taylor and Francis, London, 1994.
- [202] P. Yolum and M. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, to appear, 2002.
- [203] S. Zdonik. Version management in an object-oriented database. In *Proceedings of the IFIP International Workshop on Advanced Programming Environments, Trondheim, Norway*, pages 405–422, 1987.
- [204] R. Zhang and E. Unger. *Calendar algebra*. Kansas State University, Technical Report, 1996.

- [205] G. Özsoyoglu and R. Snodgrass. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, 1995.