

Modeling the Evolution of Objects in Temporal Information Systems

A. Artale¹, C. Parent², and S. Spaccapietra³

¹ Faculty of Computer Science, Free Univ. of Bolzano, I
artale@inf.unibz.it

² HEC/INFORGE, Université de Lausanne, CH
christine.parent@unil.ch

³ Database Laboratory, Ecole Polytechnique Fédérale Lausanne, CH
stefano.spaccapietra@epfl.ch

Abstract. This paper presents a semantic foundation of temporal conceptual models used to design temporal information systems. We consider a modeling language able to express both timestamping and evolution constraints. We conduct a deeper investigation on evolution constraints, eventually devising a model-theoretic semantics for a full-fledged model with both timestamping and evolution constraints. The proposed formalization is meant both to clarify the meaning of the various temporal constructors appeared in the literature and to give a rigorous definition to notions like satisfiability, subsumption and logical implication. Furthermore, we also show how to express temporal constraints using a subset of first-order temporal logic, i.e., DLR_{US} , the description logic DLR extended with the temporal operators *Since* and *Until*. We show how DLR_{US} is able to capture the various modeling constraints in a succinct way and to perform automated reasoning on temporal conceptual models.

1 Introduction

This paper is a contribution to improve modeling of temporal data, building on state of the art know-how developed by the conceptual data modeling community. Analyses of many proposals for temporal models (aiming in particular at helping designing temporal databases) and a summary of results achieved in the area can be found in two good surveys [15, 21]. The main temporal modeling features that we focus on in this paper can be summarized as:

- **Timestamping.** The data model should obviously distinguish between temporal and atemporal modeling constructors. This is usually realized by temporal marking of classes, relationships and attributes. In the database, these markings translate into a *timestamping* mechanism, i.e., attaching lifecycle information to classes and relationship instances, and time-varying values to attributes. In this work we consider just *validity time* (rather than transaction time [20, 27]), thus lifecycle information expresses when an object or a tuple belongs to a class or a relationship, respectively. Time-varying attributes store values together with when they hold.

- **Evolution Constraints.** They apply both to classes (status and transition constraints) and relationships (generation and cross-time constraints). *Status Classes* constraints [11] rule the permissible evolution of an object as member of a class along its lifespan. For example, an object that is an active member of a class may become an inactive member of the same class. *Transition* constraints [16] rule *object migration*, i.e., the possibility for an object to change its class membership from one class to another. For example, an object in the Student class may later migrate to become an object of the Faculty class. Complementary aspects of evolution are modeled through *Generation Relationships* [17] which describe the fact that objects in a class are generated by other objects in another (possibly the same) class. For example, in a company database, splitting of a department translates into the fact that the original department generates two (or more) new departments. Objects participating to *Cross-Time Relationships* [23] may not coexist at the time the relationship is asserted. For example, the grandfather-of relationship can involve a dead grandfather with a leaving grandchild.

This paper presents a semantic foundation for temporal data models, as a possible response to concerns stating: “[...] it is only by considering those conceptual models as a mathematical object with a formal definition and semantics that they can become useful tools for the design of databases schema and applications [...]” [12].

We present a deeper investigation on evolution constraints, eventually devising a model-theoretic semantics for a full-fledged conceptual model with both timestamping and evolution constraints. While timestamping aspects have been extensively discussed [3, 4, 10, 14, 22, 26], a clear formalization of evolution constraints is still missing, despite the fact that in the literature such constraints have been advocated as useful for modeling the behavior of temporal objects [4, 25, 17, 16, 23, 26, 24].

The formalization proposed here builds on previous efforts to formalize temporal conceptual models. Namely, we rely on a previous work to define the \mathcal{ER}_{VT} model [4], a temporal Extended Entity-Relationship (EER) model—i.e., the standard ER modeling language enriched with ISA links, disjoint and covering constraints, and full cardinality constraints—equipped with both a linear and a graphical syntax and based on a model-theoretic semantics. \mathcal{ER}_{VT} captures timestamping constructors along with transition constraints. This work extends \mathcal{ER}_{VT} with status classes, generation relationships and cross-time relationships. Another closely related work is the one of Finger and McBrien [12]. They propose a model-theoretic formalization for the ERT model, an EER model with timestamping but just cross-time relationships (called H-marked relationships by the authors and introduced in a previous paper by McBrien, Seltveit and Wrangler [23]). Our proposal modifies the semantics of cross-time relationships as presented in [12] to comply with a crucial modeling requirement, i.e. snapshot reducibility [22].

The advantage of associating a set-theoretic semantics to a language is not only to clarify the meaning of the language’s constructors but also to give a semantic definition to relevant modeling notions. In our case, given an interpretation function to assign a set-theoretic semantics to the (temporal) modeling constructors, we are able to give a rigorous definition of the notions of: *schema satisfiability* when a schema admits a non empty interpretation which guarantees that the constraints expressed by the schema are

not contradictory (similarly we define the notions of class and relationships satisfiability); *subsumption* between classes (relationships) when the interpretations of a class (relationships) is a subset of the interpretation of another class (relationship) which allows to check new ISA links; *logical implication* when a (temporal) constraint is implicitly true in the current schema thus deriving new constraints. In particular, in this paper we stress both the formalization of a constructor and the set of logical implications associated to such formalization. The obtained logical implications are generally in agreement with those mentioned in the literature on temporal conceptual models. Thus, each constructor's formalization (together with its associated logical implications) can be seen as a set of precise rules on the allowed behavior of objects, in particular regarding their evolution in time. Even if we do not address specific implementation issues, these rules can be turned into explicit integrity constraints in the form of trigger rules to be added to the schema specified by the database designer, thus enabling to check the validity of user actions involving object evolution. Since the rules are the result of a formal characterization we solve what is in our opinion a serious weakness of existing modeling approaches, i.e., without a rigorous foundation there is no guarantee that the proposed model leads to a sound system.

Finally, as a byproduct of the semantic formalization, we also show how (temporal) modeling constraints can be equivalently expressed by using a subset of first-order temporal logic, i.e., the temporal description logic \mathcal{DLR}_{US} [5]. \mathcal{DLR}_{US} is a combination of the expressive and decidable description logic \mathcal{DLR} (a description logic with n-ary relationships) with the linear temporal logic with temporal operators *Since* (S) and *Until* (U) which can be used in front of both concepts and relations. The choice of extending \mathcal{DLR} is motivated by its ability to give a logical reconstruction and an extension of representational tools such as object-oriented and conceptual data models, frame-based and web ontology languages [7, 8, 9, 19]. In this paper, we use \mathcal{DLR}_{US} both to capture the (temporal) modeling constructors in a succinct way, and to use reasoning techniques to check satisfiability, subsumption and logical implication. We show how \mathcal{DLR}_{US} axioms capture the above mentioned rules associated with each constructor's formal semantics while logical implications between \mathcal{DLR}_{US} axioms is a way to derive new rules. Even if full \mathcal{DLR}_{US} is undecidable this paper addresses interesting subsets of \mathcal{DLR}_{US} where reasoning becomes a decidable problem.

The paper is organized as follows. Sections 2 and 4 recall the characteristics of the \mathcal{DLR}_{US} description logic and the \mathcal{ER}_{VT} temporal data model on which we build our proposal. Section 3 shows the modeling requirements that lead us in elaborating the rigorous definition of our evolution framework. Section 5 discusses the evolution constraints we address while Section 6 provides a formal characterization for them together with a set of logical implications and the correspondent \mathcal{DLR}_{US} axioms. Section 7 shows that reasoning on the full-fledged temporal setting is undecidable but provides useful scenarios where reasoning becomes decidable. Section 8 concludes the paper.

2 The Temporal Description Logic

The temporal description logic \mathcal{DLR}_{US} [5] combines the propositional temporal logic with *Since* and *Until* and the (non-temporal) description logic \mathcal{DLR} [7]. \mathcal{DLR}_{US} can be

$$\begin{aligned}
 C &\rightarrow \top \mid \perp \mid CN \mid \neg C \mid C_1 \sqcap C_2 \mid \exists^{\leq k}[U_j]R \mid \\
 &\quad \diamond^+ C \mid \diamond^- C \mid \square^+ C \mid \square^- C \mid \oplus C \mid \ominus C \mid C_1 \mathcal{U} C_2 \mid C_1 \mathcal{S} C_2 \\
 R &\rightarrow \top_n \mid RN \mid \neg R \mid R_1 \sqcap R_2 \mid U_i/n : C \mid \\
 &\quad \diamond^+ R \mid \diamond^- R \mid \square^+ R \mid \square^- R \mid \oplus R \mid \ominus R \mid R_1 \mathcal{U} R_2 \mid R_1 \mathcal{S} R_2 \\
 \\
 \top^{\mathcal{I}(t)} &= \Delta^{\mathcal{I}} \\
 \perp^{\mathcal{I}(t)} &= \emptyset \\
 CN^{\mathcal{I}(t)} &\subseteq \top^{\mathcal{I}(t)} \\
 (\neg C)^{\mathcal{I}(t)} &= \top^{\mathcal{I}(t)} \setminus C^{\mathcal{I}(t)} \\
 (C_1 \sqcap C_2)^{\mathcal{I}(t)} &= C_1^{\mathcal{I}(t)} \cap C_2^{\mathcal{I}(t)} \\
 (\exists^{\leq k}[U_j]R)^{\mathcal{I}(t)} &= \{d \in \top^{\mathcal{I}(t)} \mid \#\{\langle d_1, \dots, d_n \rangle \in R^{\mathcal{I}(t)} \mid d_j = d\} \leq k\} \\
 (C_1 \mathcal{U} C_2)^{\mathcal{I}(t)} &= \{d \in \top^{\mathcal{I}(t)} \mid \exists v > t. (d \in C_2^{\mathcal{I}(v)} \wedge \forall w \in (t, v). d \in C_1^{\mathcal{I}(w)})\} \\
 (C_1 \mathcal{S} C_2)^{\mathcal{I}(t)} &= \{d \in \top^{\mathcal{I}(t)} \mid \exists v < t. (d \in C_2^{\mathcal{I}(v)} \wedge \forall w \in (v, t). d \in C_1^{\mathcal{I}(w)})\} \\
 (\top_n)^{\mathcal{I}(t)} &\subseteq (\Delta^{\mathcal{I}})^n \\
 RN^{\mathcal{I}(t)} &\subseteq (\top_n)^{\mathcal{I}(t)} \\
 (\neg R)^{\mathcal{I}(t)} &= (\top_n)^{\mathcal{I}(t)} \setminus R^{\mathcal{I}(t)} \\
 (R_1 \sqcap R_2)^{\mathcal{I}(t)} &= R_1^{\mathcal{I}(t)} \cap R_2^{\mathcal{I}(t)} \\
 (U_i/n : C)^{\mathcal{I}(t)} &= \{\langle d_1, \dots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid d_i \in C^{\mathcal{I}(t)}\} \\
 (R_1 \mathcal{U} R_2)^{\mathcal{I}(t)} &= \{\langle d_1, \dots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \\
 &\quad \exists v > t. (\langle d_1, \dots, d_n \rangle \in R_2^{\mathcal{I}(v)} \wedge \forall w \in (t, v). \langle d_1, \dots, d_n \rangle \in R_1^{\mathcal{I}(w)})\} \\
 (R_1 \mathcal{S} R_2)^{\mathcal{I}(t)} &= \{\langle d_1, \dots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \\
 &\quad \exists v < t. (\langle d_1, \dots, d_n \rangle \in R_2^{\mathcal{I}(v)} \wedge \forall w \in (v, t). \langle d_1, \dots, d_n \rangle \in R_1^{\mathcal{I}(w)})\} \\
 (\diamond^+ R)^{\mathcal{I}(t)} &= \{\langle d_1, \dots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v > t. \langle d_1, \dots, d_n \rangle \in R^{\mathcal{I}(v)}\} \\
 (\oplus R)^{\mathcal{I}(t)} &= \{\langle d_1, \dots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \langle d_1, \dots, d_n \rangle \in R^{\mathcal{I}(t+1)}\} \\
 (\diamond^- R)^{\mathcal{I}(t)} &= \{\langle d_1, \dots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v < t. \langle d_1, \dots, d_n \rangle \in R^{\mathcal{I}(v)}\} \\
 (\ominus R)^{\mathcal{I}(t)} &= \{\langle d_1, \dots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \langle d_1, \dots, d_n \rangle \in R^{\mathcal{I}(t-1)}\}
 \end{aligned}$$

Fig. 1. Syntax and semantics of \mathcal{DLRUS}

regarded as a rather expressive fragment of the first-order temporal logic $L^{\{\text{since, until}\}}$ (cf. [10, 18]).

The basic syntactical types of \mathcal{DLRUS} are *classes* (i.e., unary predicates, also known as *concepts*) and *n-ary relations* of arity ≥ 2 . Starting from a set of *atomic classes* (denoted by CN), a set of *atomic relations* (denoted by RN), and a set of *role symbols* (denoted by U) we hereinafter define inductively (complex) class and relation expressions as is shown in the upper part of Figure 1, where the binary constructors ($\sqcap, \sqcup, \mathcal{U}, \mathcal{S}$) are applied to relations of the same arity, i, j, k, n are natural numbers, $i \leq n$, and j does not exceed the arity of R .

The non-temporal fragment of \mathcal{DLRUS} coincides with \mathcal{DLR} . For both class and relation expressions all the Boolean constructors are available. The selection expression $U_i/n : C$ denotes an n -ary relation whose argument named U_i ($i \leq n$) is of type C ; if it is clear from the context, we omit n and write $(U_i : C)$. The projection expression $\exists^{\leq k}[U_j]R$ is a generalisation with cardinalities of the projection operator over the argument named U_j of the relation R ; the plain classical projection is $\exists^{\geq 1}[U_j]R$. It is also possible to use the pure argument position version of the model by replacing role sym-

bols U_i with the corresponding position numbers i . To show the expressive power of \mathcal{DLR}_{US} we refer to the next Sections where \mathcal{DLR}_{US} is used to capture various forms of temporal constraints.

The model-theoretic semantics of \mathcal{DLR}_{US} assumes a flow of time $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, where \mathcal{T}_p is a set of time points (or chronons) and $<$ a binary precedence relation on \mathcal{T}_p , is assumed to be isomorphic to $\langle \mathbb{Z}, < \rangle$. The language of \mathcal{DLR}_{US} is interpreted in *temporal models* over \mathcal{T} , which are triples of the form $\mathcal{I} \doteq \langle \mathcal{T}, \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}(t)} \rangle$, where $\Delta^{\mathcal{I}}$ is non-empty set of objects (the *domain* of \mathcal{I}) and $\cdot^{\mathcal{I}(t)}$ an *interpretation function* such that, for every $t \in \mathcal{T}$ (in the following the notation $t \in \mathcal{T}$ is used as a shortcut for $t \in \mathcal{T}_p$), every class C , and every n -ary relation R , we have $C^{\mathcal{I}(t)} \subseteq \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}(t)} \subseteq (\Delta^{\mathcal{I}})^n$. The semantics of class and relation expressions is defined in the lower part of Fig. 1, where $(u, v) = \{w \in \mathcal{T} \mid u < w < v\}$. For classes, the temporal operators \diamond^+ (some time in the future), \oplus (at the next moment), and their past counterparts can be defined via \mathcal{U} and \mathcal{S} : $\diamond^+ C \equiv \top \mathcal{U} C$, $\oplus C \equiv \perp \mathcal{U} C$, etc. The operators \square^+ (always in the future) and \square^- (always in the past) are the duals of \diamond^+ (some time in the future) and \diamond^- (some time in the past), respectively, i.e., $\square^+ C \equiv \neg \diamond^+ \neg C$ and $\square^- C \equiv \neg \diamond^- \neg C$, for both classes and relations. The operators \diamond^* (at some moment) and its dual \square^* (at all moments) can be defined for both classes and relations as $\diamond^* C \equiv C \sqcup \diamond^+ C \sqcup \diamond^- C$ and $\square^* C \equiv C \sqcap \square^+ C \sqcap \square^- C$, respectively.

A *knowledge base* is a finite set Σ of \mathcal{DLR}_{US} axioms of the form $C_1 \sqsubseteq C_2$ and $R_1 \sqsubseteq R_2$, with R_1 and R_2 being relations of the same arity. An interpretation \mathcal{I} satisfies $C_1 \sqsubseteq C_2$ ($R_1 \sqsubseteq R_2$) if and only if the interpretation of C_1 (R_1) is included in the interpretation of C_2 (R_2) at all time, i.e. $C_1^{\mathcal{I}(t)} \subseteq C_2^{\mathcal{I}(t)}$ ($R_1^{\mathcal{I}(t)} \subseteq R_2^{\mathcal{I}(t)}$), for all $t \in \mathcal{T}$. Various *reasoning services* can be defined in \mathcal{DLR}_{US} . A knowledge base, Σ , is *satisfiable* if there is an interpretation that satisfies all the axioms in Σ (in symbols, $\mathcal{I} \models \Sigma$). A class C (or relation R) is *satisfiable* if there is \mathcal{I} such that $C^{\mathcal{I}(t)} \neq \emptyset$ (respectively, $R^{\mathcal{I}(t)} \neq \emptyset$), for some time point t . A knowledge base, Σ , *logically implies* an axiom, $C_1 \sqsubseteq C_2$, and write $\Sigma \models C_1 \sqsubseteq C_2$, if we have $\mathcal{I} \models C_1 \sqsubseteq C_2$ whenever $\mathcal{I} \models \Sigma$. In this latter case, the concept C_1 is said to be *subsumed* by the concept C_2 in the knowledge base Σ . A concept C is *satisfiable*, given a knowledge base Σ , if there exists a model \mathcal{I} of Σ such that $C^{\mathcal{I}(t)} \neq \emptyset$ for some $t \in \mathcal{T}$, i.e. $\Sigma \not\models C \sqsubseteq \perp$.

While \mathcal{DLR} knowledge bases are fully able to capture atemporal EER schemas [7, 8]—i.e., given an EER schema there is an equi-satisfiable \mathcal{DLR} knowledge base—in the following Sections we show how \mathcal{DLR}_{US} knowledge bases can capture temporal EER schemas with both timestamping and evolution constraints.

3 Modeling Requirements

This Section briefly illustrates the requirements that are frequently advocated in the literature on temporal data models.

- **Orthogonality.** Temporal constructors should be specified separately and independently for classes, relationships, and attributes. Depending on application requirements, the temporal support must be decided by the designer.

- **Upward Compatibility.** This term denotes the capability of preserving the nontemporal semantics of conventional (legacy) conceptual schemas when embedded into temporal schemas.
- **Snapshot Reducibility.** Snapshots of the database described by a temporal schema are the same as the database described by the same schema, where all temporal constructors are eliminated and the schema is interpreted atemporally. Indeed, this property specifies that we should be able to fully rebuild a temporal database by starting from the single snapshots.

Orthogonality affects mainly timestamping [25] and \mathcal{ER}_{VT} already satisfies this principle by introducing temporal marks that could be used to specify the temporal behavior of classes, relationships, and attributes in an independent way.

Upward compatibility and snapshot reducibility [22] are strictly related. Considered together, they allow to preserve the meaning of atemporal constructors. In particular, the meaning of classical constructors must be preserved in such a way that a designer could either use them to model classical databases, or when used in a genuine temporal setting their meaning must be preserved at each instant of time.

These requirements are not so obvious when dealing with evolving objects. In particular, snapshot reducibility is hard to preserve when dealing with both generation and cross-time relationships where involved object may not coexist. The formalization carried out in this paper provides a data model able to respect these requirements also in presence of evolving objects.

4 The Temporal Conceptual Model \mathcal{ER}_{VT}

In this Section, the temporal EER model \mathcal{ER}_{VT} [3, 4] is briefly introduced. \mathcal{ER}_{VT} supports timestamping for classes, attributes, and relationships. \mathcal{ER}_{VT} is equipped with both a linear and a graphical syntax along with a model-theoretic semantics as a temporal extension of the EER semantics [9].

An \mathcal{ER}_{VT} schema is a tuple: $\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{ISA}, \text{DISJ}, \text{COVER}, \text{S}, \text{T}, \text{KEY})$, such that: \mathcal{L} is a finite alphabet partitioned into the sets: \mathcal{C} (*class* symbols), \mathcal{A} (*attribute* symbols), \mathcal{R} (*relationship* symbols), \mathcal{U} (*role* symbols), and \mathcal{D} (*domain* symbols). ATT is a function that maps a class symbol in \mathcal{C} to an \mathcal{A} -labeled tuple over \mathcal{D} , $\text{ATT}(E) = \langle A_1 : D_1, \dots, A_h : D_h \rangle$. REL is a function that maps a relationship symbol in \mathcal{R} to an \mathcal{U} -labeled tuple over \mathcal{C} , $\text{REL}(R) = \langle U_1 : C_1, \dots, U_k : C_k \rangle$, and k is the *arity* of R . CARD is a function $\mathcal{C} \times \mathcal{R} \times \mathcal{U} \mapsto \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ denoting cardinality constraints. We denote with $\text{CMIN}(C, R, U)$ and $\text{CMAX}(C, R, U)$ the first and second component of CARD . In Figure 2, $\text{CARD}(\text{TopManager}, \text{Manages}, \text{man}) = (1, 1)$. ISA is a binary relationship $\text{ISA} \subseteq (\mathcal{C} \times \mathcal{C}) \cup (\mathcal{R} \times \mathcal{R})$. ISA between relationships is restricted to relationships with the same arity. ISA is visualized with a directed arrow, e.g. `Manager ISA Employee` in Figure 2. DISJ , COVER are binary relations over $2^{\mathcal{C}} \times \mathcal{C}$, describing disjointness and covering partitions, respectively. DISJ is visualized with a circled “d” and COVER with a double directed arrow, e.g. `Department, InterestGroup` are both disjoint and they cover `OrganizationalUnit`. The set \mathcal{C} is partitioned into: a set \mathcal{C}^S of *Snapshot classes* (the **S**-marked classes in Figure 2), a set \mathcal{C}^M of *Mixed classes* (the *unmarked* classes

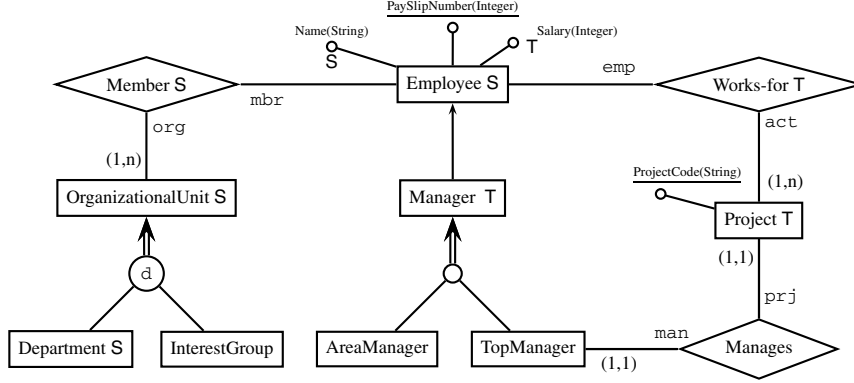


Fig. 2. The company \mathcal{ER}_{VT} diagram

in Figure 2), and a set \mathcal{C}^T of *Temporary classes* (the T-marked classes in Figure 2). A similar partition applies to the set \mathcal{R} . S, T are binary relations over $\mathcal{C} \times \mathcal{A}$ containing, respectively, the snapshot and temporary attributes of a class (see S, T marked attributes in Figure 2). KEY is a function that maps class symbols in \mathcal{C} to their key attributes, $\text{KEY}(E) = A$. Keys are visualized as underlined attributes.

The model-theoretic semantics associated with the \mathcal{ER}_{VT} modeling language adopts the *snapshot*¹ representation of abstract temporal databases and temporal conceptual models [10]. Following this paradigm, the flow of time $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, where \mathcal{T}_p is a set of time points (or chronons) and $<$ is a binary precedence relation on \mathcal{T}_p , is assumed to be isomorphic to either $\langle \mathbb{Z}, < \rangle$ or $\langle \mathbb{N}, < \rangle$. Thus, standard relational databases can be regarded as the result of mapping a temporal database from time points in \mathcal{T} to atemporal constructors, with the same interpretation of constants and the same domain.

Definition 1 (\mathcal{ER}_{VT} Semantics). Let Σ be an \mathcal{ER}_{VT} schema. A temporal database state for the schema Σ is a tuple $\mathcal{B} = \langle \mathcal{T}, \Delta^{\mathcal{B}} \cup \Delta_D^{\mathcal{B}}, \cdot^{\mathcal{B}(t)} \rangle$, such that: $\Delta^{\mathcal{B}}$ is a nonempty set disjoint from $\Delta_D^{\mathcal{B}}$; $\Delta_D^{\mathcal{B}} = \bigcup_{D_i \in \mathcal{D}} \Delta_{D_i}^{\mathcal{B}}$ is the set of basic domain values used in the schema Σ ; and $\cdot^{\mathcal{B}(t)}$ is a function that for each $t \in \mathcal{T}$ maps:

- every domain symbol D_i into a set $D_i^{\mathcal{B}(t)} = \Delta_{D_i}^{\mathcal{B}}$.
- Every class C to a set $C^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}}$.
- Every relationship R to a set $R^{\mathcal{B}(t)}$ of U -labeled tuples over $\Delta^{\mathcal{B}}$ —i.e., let R be an n -ary relationship connecting the classes C_1, \dots, C_n , $\text{REL}(R) = \langle U_1 : C_1, \dots, U_n : C_n \rangle$, then, $r \in R^{\mathcal{B}(t)} \rightarrow (r = \langle U_1 : o_1, \dots, U_n : o_n \rangle \wedge \forall i \in \{1, \dots, n\}. o_i \in C_i^{\mathcal{B}(t)})$. We adopt the convention: $\langle U_1 : o_1, \dots, U_n : o_n \rangle \equiv \langle o_1, \dots, o_n \rangle$, when U -labels are clear from the context.
- Every attribute A to a set $A^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}} \times \Delta_D^{\mathcal{B}}$.

\mathcal{B} is said a *legal temporal database state* if it satisfies all of the constraints expressed in the schema (see [4] for full details).

¹ The snapshot model represents the same class of temporal databases as the *timestamp* model [21, 22] defined by adding temporal attributes to a relation [10].

Given such a set-theoretic semantics we are able to rigorously define some relevant modeling notions such as satisfiability, subsumption and derivation of new constraints by means of logical implication.

Definition 2. Let Σ be a schema, $C \in \mathcal{C}$ a class, and $R \in \mathcal{R}$ a relationship. The following modeling notions can be defined:

1. $C(R)$ is satisfiable if there exists a legal temporal database state \mathcal{B} for Σ such that $C^{\mathcal{B}(t)} \neq \emptyset$ ($R^{\mathcal{B}(t)} \neq \emptyset$), for some $t \in \mathcal{T}$;
2. Σ is satisfiable if there exists a legal temporal database state \mathcal{B} for Σ that satisfies at least one class in Σ (\mathcal{B} is said a model for Σ);
3. $C_1(R_1)$ is subsumed by $C_2(R_2)$ in Σ if every legal temporal database state for Σ is also a legal temporal database state for C_1 ISA C_2 (R_1 ISA R_2);
4. A schema Σ' is logically implied by a schema Σ over the same signature if every legal temporal database state for Σ is also a legal temporal database state for Σ' .

In the following Subsection we will show how temporal database states, \mathcal{B} , support defining the semantics of timestamping.

4.1 Timestamping

\mathcal{ER}_{VT} is able to distinguish between *snapshot* constructors—i.e. constructors which bear no explicit specification of a given lifespan [20], which we convey by assuming a global lifespan (see Section 6.1) associated to each of their instances—*temporary* constructors—i.e. each of their instances has a limited lifespan—or *mixed* constructors—i.e. their instances can have either a global or a temporary existence. In the following, a class, relationship or attribute is called temporal if it is either temporary or mixed. The two temporal marks, **S** (snapshot) and **T** (temporary), introduced at the conceptual level, capture such temporal behavior. The semantics of timestamping can now be defined as follows (we illustrate timestamping just for classes; similar ideas are used in \mathcal{ER}_{VT} to associate timestamping to both relationships and attributes):

$$\begin{aligned} o \in C^{\mathcal{B}(t)} &\rightarrow \forall t' \in \mathcal{T}. o \in C^{\mathcal{B}(t')} && \text{Snapshot Class} \\ o \in C^{\mathcal{B}(t)} &\rightarrow \exists t' \neq t. o \notin C^{\mathcal{B}(t')} && \text{Temporary Class} \end{aligned}$$

The two cases are captured by the following \mathcal{DLR}_{US} axioms, respectively:

$$\begin{aligned} C &\sqsubseteq (\Box^+ C) \sqcap (\Box^- C) && \text{Snapshot Class} \\ C &\sqsubseteq (\Diamond^+ \neg C) \sqcup (\Diamond^- \neg C) && \text{Temporary Class} \end{aligned}$$

The distinction between snapshot, temporary and mixed constructors has been adopted in \mathcal{ER}_{VT} to avoid *overloading* the meaning of un-marked constructors. Indeed, the classical distinction between temporal (using a temporal mark) and atemporal (leaving the constructor un-marked) constructors may be ambiguous in the meaning of un-marked constructors. In this classical setting, un-marking is used to model both truly atemporal constructors (i.e., snapshot classes whose instances lifespan is always equal to the whole database lifespan), as well as legacy constructors (for *upward compatibility*) where the constructor is not marked as temporal because the original data model

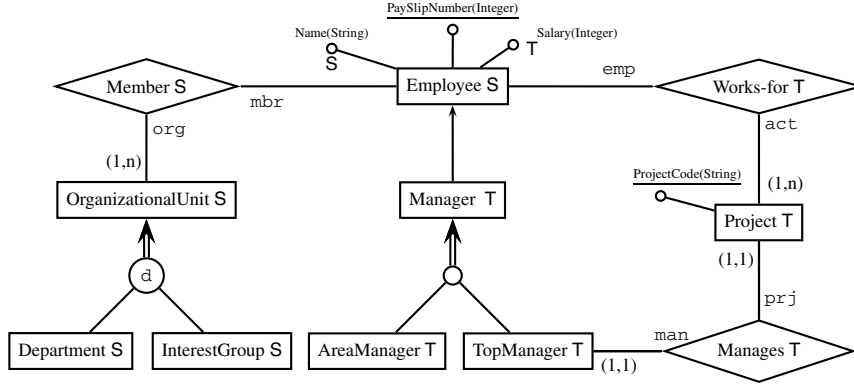


Fig. 3. The company diagram with deductions on timestamps

did not support the temporal dimension. The problem is that, due to the interaction between the various components of a temporal model, un-marked constructors can even purposely represent temporary constructors. As an example, think of an ISA involving a temporary entity (as superclass) and an un-marked entity (as a subclass). Since a designer cannot forecast all the possible interactions between the (temporal) constraints of a given conceptual schema, this ultimately means that in the classical approach *atemporality cannot be guaranteed* and this is true even for the upward compatibility.

\mathcal{ER}_{VT} explicitly introduces a snapshot mark to force both atemporality and upward compatibility. As logical implication is formally defined in \mathcal{ER}_{VT} (see Definition 2), missing specifications can be inferred and in particular a set of logical implications hold in the case of timestamping. For instance, in Figure 2, as *Manager* is temporary both *AreaManager* and *TopManager* are temporary, too. Because *OrganizationalUnit* is snapshot and partitioned into two sub-classes, *Department* which is snapshot and *InterestGroup*, the latter should be snapshot, too. As the temporary class *TopManager* participates in the relationships *Manages*, then the latter must be temporary, too (see [4] for an exhaustive list of deductions involving timestamps). The result of these deductions is given in Figure 3. Note that, when mapping \mathcal{ER}_{VT} into a relational schema both temporary and un-marked constructors are mapped into a relation with added timestamp attributes, while snapshot constructors do not need any additional time attribute (for full details on the \mathcal{ER}_{VT} relational mapping see [1]).

5 Evolution Constraints

Evolution constraints are intended to help in modeling the temporal behavior of an object. This section briefly recalls the basic concepts that have been proposed in the literature to deal with evolution, and their impact on the resulting conceptual language.

Status [25, 11] is a notion associated to temporal classes to describe the evolving status of membership of each object in the class. In a generic temporal setting, objects can be suspended and later resumed in their membership. Four different statuses can be specified, together with precise transitions between them:

- **Scheduled.** An object is scheduled if its existence within the class is known but its membership in the class will only become effective some time later. For example, a new project is approved but will not start until a later date. Each scheduled object will eventually become an active object.
- **Active.** The status of an object is active if the object is a full member of the class. For example, a currently ongoing project is an active member, at time now, of the Project class.
- **Suspended.** This status qualifies objects that exist as members of the class, but are to be seen as temporarily inactive members of the class. Being inactive means that the object cannot undergo some operations, e.g., it is not allowed to modify the values of its properties (see [11] for more details). For example, an employee taking a temporary leave of absence can be considered as a suspended employee. A suspended object was in the past an active one.
- **Disabled.** It is used to model expired objects in a class. A disabled object was in the past a member of the class. It can never again become a non-disabled member of that class (e.g., an expired project cannot be reactivated).

Transitions [16, 25] have been introduced to model the phenomenon called *object migration*. A transition records objects migrating from a *source* class to a *target* class. At the schema level, it expresses that the instances of the source class may *migrate* into the target class. Two types of transitions have been considered: *dynamic evolution*, when objects cease to be instances of the source class, and *dynamic extension*, otherwise. For example considering the company schema (Figure 3), if we want to record data about the promotion of area managers into top managers we can specify a dynamic evolution from the class `AreaManager` to the class `TopManager`. We can also record the fact that a mere employee becomes a manager by defining a dynamic extension from the class `Employee` to the class `Manager` (see Figure 5).

Generation relationships [25, 17, 24] express that (sets of) objects in a target class may be generated from (sets of) objects in a source class. While transitions involve object instances bearing the same oid, object instances linked by generation relationships necessarily bear different oids. Depending whether the source objects are preserved (as member of the source class) or disabled, we distinguish between a *production* and a *transformation*, respectively. Cardinality constraints can be added to specify the cardinality of sets involved in a generation. For example (see Figures 3,6), if we want to record the fact that (a group of) managers propose a new project a production relationship between `Manager` and `Project` can be introduced. Let us now assume that the structure of the departments of the company is dynamic, e.g., some departments may either merge or split and be replaced by others, and that it is useful to record these changes. One way would be to define a transformation relationship linking (a set of) existing departments to (a set of) new departments.

Cross-Time relationships [26, 23, 25] describe relationships between objects that do not coexist at the same time and possibly not at the time the relationship is asserted. There are many examples of these relationships (see Figure 7). Consider, for example, a relationship “biography” between an author and a famous person already dead, or the relationship “grandparent” that holds even if the grandparent passed away before the grandchild was born or the grandchild is not yet born. Still, considering the company

schema (Figure 3), the relationship *Works-for* can be changed to cross-time whenever one wants to assign an employee to a project before its official launching, or if some employee keeps on working on a project after its official closure.

6 Formalizing Evolving Objects

The proposed formalization is based on a model-theoretic semantics and a correspondent set of axioms expressed using the temporal description logic \mathcal{DLR}_{US} . This will give us both a formal characterization of the temporal conceptual modeling constructors, and the possibility to use the reasoning capabilities of \mathcal{DLR}_{US} to check satisfiability, subsumption and logical implications over temporal schemas. The model-theoretic semantics we illustrate here for the various evolution constraints is an extension of the one developed for the model \mathcal{ER}_{VT} , introduced in Section 4. The validity of the proposed formalization is justified by providing a set of logical implications which are in agreement with the derivations mentioned in the literature on temporal data modeling.

6.1 Status Classes

The evolution in the membership of an object to a temporal class is reflected in the changing values of the status of the object in the class. This evolution obeys some rules that give rise to a set of constraints. This Subsection formally capture these constraints.

Let C be a temporal (i.e., temporary or mixed) class. We capture status transition of membership in C by associating to C the following *status classes*: *Scheduled-C*, *Suspended-C*, *Disabled-C*. In particular, status classes are represented by the hierarchy of Figure 4 (where C may also be mixed) that classifies C instances according to their actual status. To preserve upward compatibility we do not explicitly introduce an active class, but assume by default that the name of the class itself denotes the set of active objects. i.e., $\text{Active-C} \equiv C$. We can assume that the status classes are created automatically by the system each time a class is declared temporal. Thus, designers and users are not forced neither to introduce nor to manipulate status classes. They only have to be aware of the different statuses in the lifecycle of an object. Note that, since membership of objects into snapshot classes is global, i.e. objects are always active, the notion of status classes does not apply to snapshot classes.

To capture the intended meaning of status classes, we define ad-hoc constraints and then prove that such constraints capture their evolving behavior as described in the literature [25, 11]. First of all, disjointness and ISA constraints between statuses of a class C can be described as illustrated in Figure 4, where *Top* is supposed to be snapshot and represents the universe of discourse (i.e., $\text{Top}^{\mathcal{B}(t)} \equiv \Delta^{\mathcal{B}}$). Other than hierarchical constraints, the intended semantics of status classes induces the following rules that are related to their temporal behavior:

(EXISTS) *Existence persists until Disabled.*

$$o \in \text{Exists-C}^{\mathcal{B}(t)} \rightarrow \forall t' > t. (o \in \text{Exists-C}^{\mathcal{B}(t')} \vee o \in \text{Disabled-C}^{\mathcal{B}(t')})$$

(DISAB1) *Disabled persists.*

$$o \in \text{Disabled-C}^{\mathcal{B}(t)} \rightarrow \forall t' > t. o \in \text{Disabled-C}^{\mathcal{B}(t')}$$

(DISAB2) *Disabled was Active in the past.*

$$o \in \text{Disabled-C}^{\mathcal{B}(t)} \rightarrow \exists t' < t. o \in \text{C}^{\mathcal{B}(t')}$$

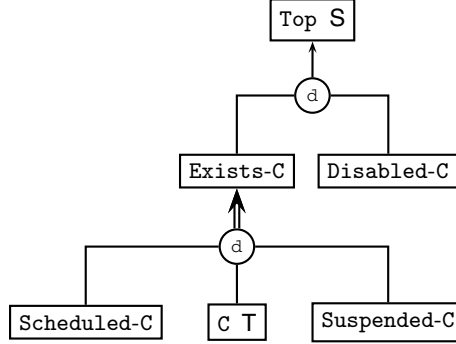


Fig. 4. Status classes

(SUSP) *Suspended was Active in the past.*

$$o \in \text{Suspended-C}^{\mathcal{B}(t)} \rightarrow \exists t' < t. o \in \mathcal{C}^{\mathcal{B}(t')}$$

(SCH1) *Scheduled will eventually become Active.*

$$o \in \text{Scheduled-C}^{\mathcal{B}(t)} \rightarrow \exists t' > t. o \in \mathcal{C}^{\mathcal{B}(t')}$$

(SCH2) *Scheduled can never follow Active.*

$$o \in \mathcal{C}^{\mathcal{B}(t)} \rightarrow \forall t' > t. o \notin \text{Scheduled-C}^{\mathcal{B}(t')}$$

\mathcal{DLR}_{US} axioms are able to fully capture the hierarchical constraints of Figure 4 (see [4] for more details). Moreover, the above semantic equations are captured by the following \mathcal{DLR}_{US} axioms:

$$\text{(EXISTS)} \quad \text{Exists-C} \sqsubseteq \Box^+(\text{Exists-C} \sqcup \text{Disabled-C})$$

$$\text{(DISAB1)} \quad \text{Disabled-C} \sqsubseteq \Box^+\text{Disabled-C}$$

$$\text{(DISAB2)} \quad \text{Disabled-C} \sqsubseteq \Diamond^-C$$

$$\text{(SUSP)} \quad \text{Suspended-C} \sqsubseteq \Diamond^-C$$

$$\text{(SCH1)} \quad \text{Scheduled-C} \sqsubseteq \Box^+C$$

$$\text{(SCH2)} \quad C \sqsubseteq \Box^+\neg\text{Scheduled-C}$$

As a consequence of the above formalization, scheduled and disabled status classes can be true only over a single interval, while active and suspended can hold at set of intervals (i.e., an object can move many times back and forth from active to suspended status and viceversa). In particular, the following set of new rules can be derived.

Proposition 1 (Status Classes: Logical Implications). *The following logical implications hold given the above formalization of status classes:*

(SCH3) *Scheduled persists until active:* $\text{Scheduled-C} \sqsubseteq \text{Scheduled-C} \cup C$.

Together with axiom (SCH2), we can conclude that Scheduled-C is true just on a single interval.

(SCH4) *Scheduled cannot evolve directly to Disabled:* $\text{Scheduled-C} \sqsubseteq \oplus \neg \text{Disabled-C}$.

(DISAB3) *Disabled was active but it will never become active anymore:*

$$\text{Disabled-C} \sqsubseteq \Diamond^-(C \sqcap \Box^+\neg C).$$

In the following we show the adequacy of the semantics associated to status classes to describe: *a)* the behavior of temporal classes involved in ISA relationships; *b)* the notions of *lifespan*, *birth* and *death* of an object; *c)* the object migration between classes; *d)* the relationships that involve objects existing at different times (both generation and cross-time relationships).

Isa vs. status. When an ISA relationship is specified between two temporal classes, say $B \text{ ISA } A$, then the following constraints must hold between the respective status classes:

- (ISA1) *Objects active in B must be active in A.* $B \sqsubseteq A$
- (ISA2) *Objects suspended in B must be either suspended or active in A.*
 $\text{Suspended-B} \sqsubseteq \text{Suspended-A} \sqcup A$
- (ISA3) *Objects disabled in B must be either disabled, suspended or active in A.*
 $\text{Disabled-B} \sqsubseteq \text{Disabled-A} \sqcup \text{Suspended-A} \sqcup A$
- (ISA4) *Objects scheduled in B cannot be disabled in A.*
 $\text{Scheduled-B} \sqsubseteq \neg \text{Disabled-A}$
- (ISA5) *Objects disabled in A, and active in B in the past, must be disabled in B.*
 $\text{Disabled-A} \sqcap \diamond^- B \sqsubseteq \text{Disabled-B}$

The formalization of status classes provided above is not sufficient to guarantee properties (ISA1-5)². We need to further assume that the system behaves under the *temporal ISA assumption*: Each time an ISA between two temporal classes holds ($B \text{ ISA } A$), then an ISA between the respective existence status classes ($\text{Exists-B} \text{ ISA } \text{Exists-A}$) is automatically added by the system. Now, we are able to prove that points (ISA1-5) above are entailed by the semantics associated to status classes under the temporal ISA assumption.

Proposition 2 (Status Classes Vs. ISA: Logical Implications). *Let A, B be two temporal classes such that $B \text{ ISA } A$, then properties (ISA1-5) are valid logical implications.*

- (ISA1) Obviously true since $B \text{ ISA } A$ holds, and both A, B are considered active.
- (ISA2) Let $o \in \text{Suspended-B}^{\mathcal{B}(t_0)}$, since $\text{Suspended-B} \text{ ISA } \text{Exists-B}$, and (by temporal ISA assumption) $\text{Exists-B} \text{ ISA } \text{Exists-A}$, then, $o \in \text{Exists-A}^{\mathcal{B}(t_0)}$. On the other hand, by (SUSP), $\exists t_1 < t_0. o \in B^{\mathcal{B}(t_1)}$, and then, $o \in A^{\mathcal{B}(t_1)}$. Then, by (SCH2), $o \notin \text{Scheduled-A}^{\mathcal{B}(t_0)}$. Thus, due to the disjoint covering constraint between active and suspended classes, either $o \in A^{\mathcal{B}(t_0)}$ or $o \in \text{Suspended-A}^{\mathcal{B}(t_0)}$.
- (ISA3) Let $o \in \text{Disabled-B}^{\mathcal{B}(t_0)}$, then, by (DISAB2), $\exists t_1 < t_0. o \in B^{\mathcal{B}(t_1)}$. By $B \text{ ISA } A$ and $A \text{ ISA } \text{Exists-A}$, then, $o \in \text{Exists-A}^{\mathcal{B}(t_1)}$. By (EXISTS) and the disjointness between existing and disabled classes, there are only two possibilities at point in time $t_0 > t_1$:
 1. $o \in \text{Exists-A}^{\mathcal{B}(t_0)}$, and thus, by (SCH2), $o \in A^{\mathcal{B}(t_0)}$ or $o \in \text{Suspended-A}^{\mathcal{B}(t_0)}$;
 - or
 2. $o \in \text{Disabled-A}^{\mathcal{B}(t_0)}$.
- (ISA4) Let $o \in \text{Scheduled-B}^{\mathcal{B}(t_0)}$, then, by (SCH1), $\exists t_1 > t_0. o \in B^{\mathcal{B}(t_1)}$, and by $B \text{ ISA } A$, $o \in A^{\mathcal{B}(t_1)}$. Thus, by (DISAB1) and the disjointness between active and disabled states, $o \notin \text{Disabled-A}^{\mathcal{B}(t_0)}$.

² We let the reader check that points 2 and 5 are not necessarily true.

(ISA5) Let $o \in \text{Disabled-A}^{\mathcal{B}(t_0)}$ and $o \in \mathcal{B}^{\mathcal{B}(t_1)}$ for some $t_1 < t_0$, then, $o \in \text{Exists-B}^{\mathcal{B}(t_1)}$. By (EXISTS) and the disjointness between existing and disabled classes, there are only two possibilities at time $t_0 > t_1$: either $o \in \text{Exists-B}^{\mathcal{B}(t_0)}$ or $o \in \text{Disabled-B}^{\mathcal{B}(t_0)}$. By absurd, let $o \in \text{Exists-B}^{\mathcal{B}(t_0)}$, then by temporal ISA assumption, $o \in \text{Exists-A}^{\mathcal{B}(t_0)}$, which contradicts the assumption that $o \in \text{Disabled-A}^{\mathcal{B}(t_0)}$.

Lifespan. Here we define the lifespan of objects belonging to a temporal class, together with other related notions. In particular, we define EXISTENCE_C , LIFESPAN_C , ACTIVE_C , BEGIN_C , BIRTH_C and DEATH_C as functions depending on the object membership to the status classes associated to a temporal class C .

The *existence time* of an object describes the temporal instants where the object is either a scheduled, active or suspended member of a given class. More formally, $\text{EXISTENCESPAN}_C : \Delta^{\mathcal{B}} \rightarrow 2^{\mathcal{T}}$, such that:

$$\text{EXISTENCESPAN}_C(o) = \{t \in \mathcal{T} \mid o \in \text{Exists-C}^{\mathcal{B}(t)}\}$$

The *lifespan* of an object describes the temporal instants where the object is an active or suspended member of a given class (thus, $\text{LIFESPAN}_C(o) \subseteq \text{EXISTENCESPAN}_C(o)$). More formally, $\text{LIFESPAN}_C : \Delta^{\mathcal{B}} \rightarrow 2^{\mathcal{T}}$, such that:

$$\text{LIFESPAN}_C(o) = \{t \in \mathcal{T} \mid o \in \mathcal{C}^{\mathcal{B}(t)} \cup \text{Suspended-C}^{\mathcal{B}(t)}\}$$

The *activespan* of an object describes the temporal instants where the object is an active member of a given class (thus, $\text{ACTIVESPAN}_C(o) \subseteq \text{LIFESPAN}_C(o)$). More formally, $\text{ACTIVESPAN}_C : \Delta^{\mathcal{B}} \rightarrow 2^{\mathcal{T}}$, such that:

$$\text{ACTIVESPAN}_C(o) = \{t \in \mathcal{T} \mid o \in \mathcal{C}^{\mathcal{B}(t)}\}$$

The functions BEGIN_C and DEATH_C associate to an object the first and the last appearance, respectively, of the object as a member of a given class, while BIRTH_C denotes the first appearance as an active object of that class. More formally, BEGIN_C , BIRTH_C , $\text{DEATH}_C : \Delta^{\mathcal{B}} \rightarrow \mathcal{T}$, such that:

$$\begin{aligned} \text{BEGIN}_C(o) &= \min(\text{EXISTENCESPAN}_C(o)) \\ \text{BIRTH}_C(o) &= \min(\text{ACTIVESPAN}_C(o)) \equiv \min(\text{LIFESPAN}_C(o)) \\ \text{DEATH}_C(o) &= \max(\text{LIFESPAN}_C(o)) \end{aligned}$$

We could still speak of existencespan, lifespan or activespan for snapshot classes, but in this case $\text{EXISTENCESPAN}_C(o) \equiv \text{LIFESPAN}_C(o) \equiv \text{ACTIVESPAN}_C(o) \equiv \mathcal{T}$.

6.2 Transition

Dynamic transitions between classes model the notion of object migration from a source to a target class. Two notions of dynamic transitions between classes are considered in the literature [25, 16]: *dynamic evolution*, when an object ceases to be an instance of a source class, and *dynamic extension*, when an object is still allowed to belong to the source. Concerning the graphical representation, as illustrated in Figure 5, we use a dashed arrow pointing to the target class and labeled with either DEX or DEV denoting dynamic extension and evolution, respectively.

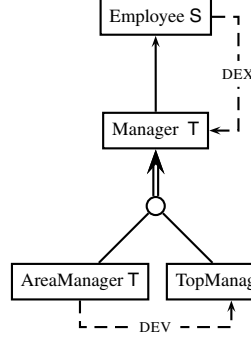


Fig. 5. Transitions employee-to-manager and area-to-top manager

In a temporal setting, objects can obviously change their membership class. Specifying a transition between two classes means that: *a.* We want to keep track of such migration; *b.* Not necessarily all the objects in the source or in the target participate in the migration; *c.* When the source class is a temporal class, migration involves only objects active or suspended. Thus, neither disabled nor scheduled objects can take part in a transition.

In the following, we present a formalization that satisfies the above requirements. Formalizing dynamic transitions as relationships would result in binary relationships linking the same object that migrates from the source to the target class. Rather than defining a relationship type with an equality constraint on the identity of the linked objects, we represent transitions by introducing a new class denoted by either DEX_{C_1, C_2} or DEV_{C_1, C_2} for dynamic extension and evolution, respectively. More formally, in case of a *dynamic extension* between classes C_1, C_2 the following semantic equation holds:

$$o \in DEX_{C_1, C_2}^{\mathcal{B}(t)} \rightarrow (o \in (\text{Suspended-}C_1^{\mathcal{B}(t)} \cup C_1^{\mathcal{B}(t)}) \wedge o \notin C_2^{\mathcal{B}(t)} \wedge o \in C_2^{\mathcal{B}(t+1)})$$

And the equivalent \mathcal{DLR}_{US} axiom is:

$$(DEX) \ DEX_{C_1, C_2} \sqsubseteq (\text{Suspended-}C_1 \sqcup C_1) \sqcap \neg C_2 \sqcap \oplus C_2$$

In case of a *dynamic evolution* between classes C_1, C_2 the source object cannot remain active in the source class. Thus, the following semantic equation holds:

$$o \in DEV_{C_1, C_2}^{\mathcal{B}(t)} \rightarrow (o \in (\text{Suspended-}C_1^{\mathcal{B}(t)} \cup C_1^{\mathcal{B}(t)}) \wedge o \notin C_2^{\mathcal{B}(t)} \wedge o \in C_2^{\mathcal{B}(t+1)} \wedge o \notin C_1^{\mathcal{B}(t+1)})$$

And the equivalent \mathcal{DLR}_{US} axiom is:

$$(DEV) \ DEV_{C_1, C_2} \sqsubseteq (\text{Suspended-}C_1 \sqcup C_1) \sqcap \neg C_2 \sqcap \oplus (C_2 \sqcap \neg C_1)$$

Please note that, in case C_1 is a snapshot class, then, $\text{Exists-}C_1 \equiv C_1$. Finally, we formalize the case where the source (C_1) and/or the target (C_2) totally participate in a dynamic extension (at schema level we add mandatory cardinality constraints on DEX/DEV links):

$$\begin{aligned}
 o \in C_1^{\mathcal{B}(t)} &\rightarrow \exists t' > t. o \in \text{DEX}_{C_1, C_2}^{\mathcal{B}(t')} && \text{Source Total Transition} \\
 o \in C_2^{\mathcal{B}(t)} &\rightarrow \exists t' < t. o \in \text{DEX}_{C_1, C_2}^{\mathcal{B}(t')} && \text{Target Total Transition}
 \end{aligned}$$

The above cases are captured by the following \mathcal{DLR}_{US} axioms, respectively:

$$\begin{aligned}
 (\text{STT}) \quad C_1 &\sqsubseteq \diamond^+ \text{DEX}_{C_1, C_2} && \text{Source Total Transition} \\
 (\text{TTT}) \quad C_2 &\sqsubseteq \diamond^- \text{DEX}_{C_1, C_2} && \text{Target Total Transition}
 \end{aligned}$$

In a similar way we deal with dynamic evolution constraints.

Proposition 3 (Transition: Logical Implications). *The following logical implications hold as a consequence of the transition semantics:*

1. *The classes DEX_{C_1, C_2} and DEV_{C_1, C_2} are temporary classes; actually, they hold at single time points.*
 $\text{DEX}_{C_1, C_2} \sqsubseteq \oplus \neg \text{DEX}_{C_1, C_2} \sqcap \ominus \neg \text{DEX}_{C_1, C_2}$ (similar for DEV_{C_1, C_2})
Indeed, let $o \in \text{DEX}_{C_1, C_2}^{\mathcal{B}(t)}$, then $o \notin C_2^{\mathcal{B}(t)}$ and $o \in C_2^{\mathcal{B}(t+1)}$, thus $o \notin \text{DEX}_{C_1, C_2}^{\mathcal{B}(t+1)}$ and $o \notin \text{DEX}_{C_1, C_2}^{\mathcal{B}(t-1)}$. Note that, the time t such that $o \in \text{DEX}_{C_1, C_2}^{\mathcal{B}(t)}$ records when the transition event happens. Similar considerations apply for DEV_{C_1, C_2} .
2. *Objects in the classes DEX_{C_1, C_2} and DEV_{C_1, C_2} cannot be disabled as C_2 .*
 $\text{DEX}_{C_1, C_2} \sqsubseteq \neg \text{Disabled-}C_2$ (similar for DEV_{C_1, C_2})
Indeed, since $\text{DEX}_{C_1, C_2} \sqsubseteq \oplus C_2$, i.e. objects in DEX_{C_1, C_2} are active in C_2 starting from the next point in time, then by property (DISAB3), $\text{DEX}_{C_1, C_2} \sqsubseteq \neg \text{Disabled-}C_2$. The same holds for DEV_{C_1, C_2} .
3. *The target class C_2 cannot be snapshot (it becomes temporary in case of TTT constraints).*
 $\text{DEX}_{C_1, C_2} \sqsubseteq \diamond^* [C_2 \sqcap (\diamond^+ \neg C_2 \sqcup \diamond^- \neg C_2)]$
Indeed, from (DEX), $\text{DEX}_{C_1, C_2} \sqsubseteq \neg C_2 \sqcap \oplus C_2$ (the same holds for DEV_{C_1, C_2}).
4. *As a consequence of dynamic evolution, the source class, C_1 , cannot be snapshot (and it becomes temporary in case of STT constraints).*
 $\text{DEV}_{C_1, C_2} \sqsubseteq \diamond^* [C_1 \sqcap (\diamond^+ \neg C_1 \sqcup \diamond^- \neg C_1)]$
Indeed, an object evolving from C_1 to C_2 ceases to be a member of C_1 .
5. *Dynamic evolution cannot involve a class and one of its sub-classes.*
 $C_2 \sqsubseteq C_1 \models \text{DEV}_{C_1, C_2} \sqsubseteq \perp$
Indeed, from (DEV), $\text{DEV}_{C_1, C_2} \sqsubseteq \oplus (C_2 \sqcap \neg C_1)$ which contradicts $C_2 \sqsubseteq C_1$.
6. *Dynamic extension between disjoint classes logically implies Dynamic evolution.*
 $\{\text{DEX}_{C_1, C_2}, C_1 \sqsubseteq \neg C_2\} \models \text{DEV}_{C_1, C_2}$

6.3 Generation Relationships

Generation relationships [25, 17] represent processes that lead to the emergence of *new instances* starting from a set of instances. Two distinct generation relationships have been introduced: *production*, when the source objects survive the generation process; *transformation*, when all the instances involved in the process are consumed. At the conceptual level we introduce two marks associated to a relationship: **GP** for production and **GT** for transformation relationships, and an arrow points to the target class (see Figure 6).

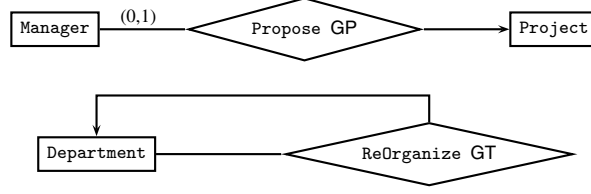


Fig. 6. Production and transformation generation relationships

We model generation as binary relationships connecting a source class to a target one: $\text{REL}(R) = \langle \text{source} : C_1, \text{target} : \text{Scheduled-}C_2 \rangle$. The semantics of *production relationships*, R , is described by the following equation:

$$\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow (o_1 \in C_1^{\mathcal{B}(t)} \wedge o_2 \in \text{Scheduled-}C_2^{\mathcal{B}(t)} \wedge o_2 \in C_2^{\mathcal{B}(t+1)})$$

Thus, objects active in the source class produce objects active in the target class at the next point in time. Notice that, the use of status classes allow us to preserve snapshot reducibility. Indeed, for each pair of objects, $\langle o_1, o_2 \rangle$, belonging to a generation relationships o_1 is active in the source while o_2 is scheduled in the target. The \mathcal{DLR}_{US} axiom capturing the production semantics is:

$$\text{(PROD)} \quad R \sqsubseteq \text{source} : C_1 \sqcap \text{target} : (\text{Scheduled-}C_2 \sqcap \oplus C_2)$$

The case of *transformation* is captured by the following semantic equation:

$$\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow (o_1 \in C_1^{\mathcal{B}(t)} \wedge o_1 \in \text{Disabled-}C_1^{\mathcal{B}(t+1)} \wedge o_2 \in \text{Scheduled-}C_2^{\mathcal{B}(t)} \wedge o_2 \in C_2^{\mathcal{B}(t+1)})$$

Thus, objects active in the source generate objects active in the target at the next point in time while the source objects cease to exist as member of the source. The \mathcal{DLR}_{US} axiom capturing the transformation semantics is:

$$\text{(TRANS)} \quad R \sqsubseteq \text{source} : (C_1 \sqcap \oplus \text{Disabled-}C_1) \sqcap \text{target} : (\text{Scheduled-}C_2 \sqcap \oplus C_2)$$

Proposition 4 (Generation: Logical Implications). *The following logical implications hold as a consequence of the generation semantics:*

1. *A generation relationship, R , is temporary; actually, it is instantaneous.*
 $R \sqsubseteq \square^+ \neg R \sqcap \square^- \neg R$
Indeed, let $\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)}$, then, since $o_2 \notin \text{Scheduled-}C_2^{\mathcal{B}(t+1)}$, then $\langle o_1, o_2 \rangle \notin R^{\mathcal{B}(t+1)}$. Since, $o_2 \notin C_2^{\mathcal{B}(t)}$, then $\langle o_1, o_2 \rangle \notin R^{\mathcal{B}(t-1)}$.
2. *The target class, C_2 , cannot be snapshot (it becomes temporary if total participation is specified).*
 $R \sqsubseteq \text{target} : \diamond^* [C_2 \sqcap (\diamond^+ \neg C_2 \sqcup \diamond^- \neg C_2)]$
Indeed, let $\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)}$, then, $o_2 \notin C_2^{\mathcal{B}(t)}$ and $o_2 \in C_2^{\mathcal{B}(t+1)}$.
3. *The target class, C_2 , cannot be disabled.*
 $R \sqsubseteq \text{target} : \neg \text{Disabled-}C_2$
Indeed, let $\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)}$, then, $o_2 \in C_2^{\mathcal{B}(t+1)}$. Thus $o_2 \notin \text{Disabled-}C_2^{\mathcal{B}(t)}$.

4. If R is a transformation relationship, then, C_1 cannot be snapshot (it becomes temporary if total participation is specified).

$$R \sqsubseteq \text{source} : \diamond^* [C_1 \sqcap (\diamond^+ \neg C_1 \sqcup \diamond^- \neg C_1)]$$

Indeed, C_1 will be disabled at the next point in time.

Note that, the Department class which is both the source and target of a transformation relationship (Figure 6) cannot longer be snapshot (as was in Figure 3) and it must be changed to temporary (as a consequence of this new timestamp, InterestGroup is a genuine mixed class).

6.4 Cross-Time Relationships

Cross-time relationships relate objects that are members of the participating classes at different times. The conceptual model MADS [25] allows for *synchronization* relationships to specify temporal constraints (Allen temporal relations) between the lifespan of linked objects. *Historical marks* are used in the ERT model [23] to express a relationship between objects not existing at the same time (both past and future historical marks are introduced).

This Section formalizes cross-time relationships with the aim of preserving the snapshot reducibility of the resulting model. Let us consider a concrete example. Let “biography” be a cross-time relationship linking the author of a biography with a famous person no more in existence. Snapshot reducibility says that if there is an instance (say, $\text{bio} = \langle \text{Tulard}, \text{Napoleon} \rangle$) of the Biography relationship at time t_0 (in particular, Tulard wrote a bio on Napoleon in 1984), then, the projection of Biography at time t_0 (1984 in our example) must contain the pair $\langle \text{Tulard}, \text{Napoleon} \rangle$. Now, while Tulard is a member of the class Author in 1984, we cannot say that Napoleon is member of the class Person in 1984. Our formalization of cross-time relationships proposes the use of status classes to preserve snapshot reducibility. The biography example can be solved by asserting that Napoleon is a member of the Disabled-Person class in 1984.

At the conceptual level, we mark with P,=,F (standing for Past, Now and Future, respectively) the links of cross-time relationships. Furthermore, we allow for the compound marks $\langle \text{P}, = \rangle$, $\langle \text{F}, = \rangle$ and $\langle \text{P}, =, \text{F} \rangle$, while just specifying = doesn’t add any constraint (see Figure 7). Assuming that R is a cross-time relationship between classes C_1, C_2 , then, the semantics of marking the C_1 link is:

$$\begin{array}{ll} \langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow o_1 \in \text{Disabled-}C_1^{\mathcal{B}(t)} & \text{Strictly Past } \langle \text{P} \rangle \\ \langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow o_1 \in (C_1 \sqcup \text{Disabled-}C_1)^{\mathcal{B}(t)} & \text{Past } \langle \text{P}, = \rangle \\ \langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow o_1 \in \text{Scheduled-}C_1^{\mathcal{B}(t)} & \text{Strictly Future } \langle \text{F} \rangle \\ \langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow o_1 \in (C_1 \sqcup \text{Scheduled-}C_1)^{\mathcal{B}(t)} & \text{Future } \langle \text{F}, = \rangle \\ \langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow o_1 \in (C_1 \sqcup \text{Scheduled-}C_1 \sqcup \text{Disabled-}C_1)^{\mathcal{B}(t)} & \text{Full-Cross } \langle \text{P}, =, \text{F} \rangle \end{array}$$

The corresponding \mathcal{DLR}_{US} axioms are:

$$\begin{array}{ll} R \sqsubseteq U_1 : \text{Disabled-}C_1 & \text{Strictly Past } \langle \text{P} \rangle \\ R \sqsubseteq U_1 : (C_1 \sqcup \text{Disabled-}C_1) & \text{Past } \langle \text{P}, = \rangle \\ R \sqsubseteq U_1 : \text{Scheduled-}C_1 \sqcap & \text{Strictly Future } \langle \text{F} \rangle \\ R \sqsubseteq U_1 : (C_1 \sqcup \text{Scheduled-}C_1) & \text{Future } \langle \text{F}, = \rangle \\ R \sqsubseteq U_1 : (C_1 \sqcup \text{Scheduled-}C_1 \sqcup \text{Disabled-}C_1) & \text{Full-Cross } \langle \text{P}, =, \text{F} \rangle \end{array}$$

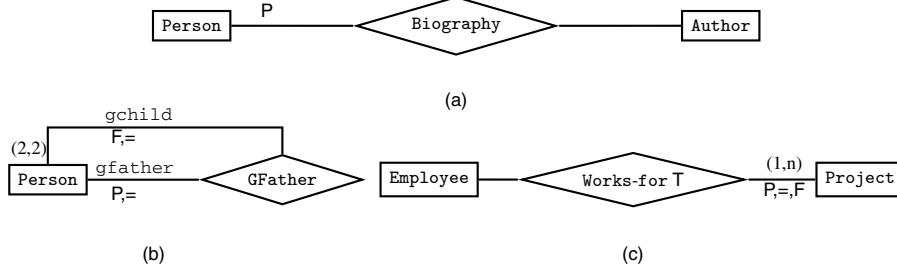


Fig. 7. Cross-Time Relationships

Proposition 5 (Cross-Time: Logical Implications). *The following logical implications hold as a consequence of the cross-time semantics (apart from point 1., we assume that C_1 (C_2) participates as either strict past or strict future):*

1. *If a relationship, R , is snapshot then historical marks reduce to the = mark (i.e., R is not a genuine cross-time relationships).
See next point.*
2. *A cross-time relationship, R , is temporary ($R \sqsubseteq \diamond^+ \neg R \sqcup \diamond^- \neg R$).
Let assume that C_1 participates as strict past. Thus, if $\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)}$, then $o_1 \in \text{Disabled-}C_1^{\mathcal{B}(t)}$ and, by (DISAB2), $\exists t_1 < t$ s.t. $o_1 \in C_1^{\mathcal{B}(t_1)}$. Then $\langle o_1, o_2 \rangle \notin R^{\mathcal{B}(t_1)}$.*
3. *C_1 (C_2) cannot be snapshot (is temporary if total participation is specified).
 $R \sqsubseteq U_1 : \diamond^* [C_1 \sqcap (\diamond^+ \neg C_1 \sqcup \diamond^- \neg C_1)]$
Let assume that C_1 participates as strict past. Thus, if $\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)}$, then, $o_1 \in \text{Disabled-}C_1^{\mathcal{B}(t)}$. Then, $o_1 \notin C_1^{\mathcal{B}(t)}$ while, by (DISAB2), $\exists t_1 < t$ s.t. $o_1 \in C_1^{\mathcal{B}(t_1)}$.*

7 Complexity of Reasoning on Temporal Models

As this paper shows, the temporal description logic \mathcal{DLR}_{US} is able to fully capture temporal schemas with both timestamping and evolution constraints. Reasoning over \mathcal{DLR}_{US} knowledge bases, i.e., checking satisfiability, subsumption and logical implications, turns out to be undecidable [5]. The main reason for this is the possibility to postulate that a binary relation does not vary in time. Note that, showing that temporal schemas can be mapped into \mathcal{DLR}_{US} axioms does not necessarily imply that reasoning over temporal schemas is an undecidable problem. Unfortunately, [2] shows that the undecidable Halting Problem can be encoded as the problem of class satisfiability w.r.t. a temporal schema with both timestamping and evolution constraints.

On the other hand, the fragment, \mathcal{DLR}_{US}^- , of \mathcal{DLR}_{US} deprived of the ability to talk about temporal persistence of n -ary relations, for $n \geq 2$, is decidable. Indeed, reasoning in \mathcal{DLR}_{US}^- is an EXPTIME-complete problem [5]. This result gives us an useful scenario where reasoning over temporal schemas becomes decidable. In particular, if we forbid timestamping for relationships (i.e., relationships are just unmarked) reasoning on temporal models with both concept timestamping and full evolution constraints can be reduced to reasoning over \mathcal{DLR}_{US}^- . The problem of reasoning in this setting is complete for EXPTIME since the EXPTIME-complete problem of reasoning with \mathcal{ALC} knowledge bases can be captured by such schemas [6].

It is an open problem whether reasoning is still decidable by regaining timestamping for relationships (and maintaining timestamping for classes) but dropping evolution constraints. We have a strong feeling that this represents a decidable scenario since it is possible to encode temporal schemas without evolution constraints by using a combination between the description logic \mathcal{DLR} and the epistemic modal logic $\mathbf{S5}$. Decidability results have been proved for the sub-logic \mathcal{ALC}_{S5} [13]. But, it is an open problem whether this result still holds for the more complex logic \mathcal{DLR}_{S5} .

8 Conclusions

In this paper we proposed a formalization of the various modeling constructors that support the design of temporal DBMS with particular attention to evolution constraints. The formalization, based on a model-theoretic semantics, has been developed with the aim to preserve three fundamental modeling requirements: Orthogonality, Upward Compatibility and Snapshot Reducibility. The introduction of status classes, which describe the evolution in the membership of an object to a temporal class, allowed us to maintain snapshot reducibility when characterizing both generations and cross-time relationships. The formal semantics clarified the meaning of the language's constructors but it also gave a rigorous definition to relevant modeling notions like: satisfiability of schemas, classes and relationships; subsumption for both classes and relationships; logical implication. Furthermore, for each constructor we presented its formalization together with the set of logical implications associated to such formalization.

Finally, we have been able to show how temporal schemas can be equivalently expressed using a subset of first-order temporal logic, i.e., \mathcal{DLR}_{US} , the description logic \mathcal{DLR} extended with the temporal operators *Since* and *Until*. Overall, we obtained a temporal conceptual model that preserves well established modeling requirements, equipped with a model-theoretic semantics where each constructor can be seen as a set of precise rules, and with the possibility to perform automated reasoning by mapping temporal schemas (without timestamping on relationships) into temporal description logic knowledge bases.

References

1. B. Ahmad. Modeling bi-temporal databases. Master's thesis, UMIST Department of Computation, UK, 2003.
2. A. Artale. Reasoning on temporal conceptual schemas with dynamic constraints. In *11th Int. Symposium on Temporal Representation and Reasoning (TIME04)*. IEEE Computer Society, 2004. Also in Proc. of the 2004 Int. Workshop on Description Logics (DL'04).
3. A. Artale and E. Franconi. Temporal ER modeling with description logics. In *Proc. of the Int. Conf. on Conceptual Modeling (ER'99)*. Springer-Verlag, November 1999.
4. A. Artale, E. Franconi, and F. Mandreoli. Description logics for modelling dynamic information. In Jan Chomicki, Ron van der Meyden, and Gunter Saake, editors, *Logics for Emerging Applications of Databases*. Lecture Notes in Computer Science, Springer-Verlag, 2003.
5. A. Artale, E. Franconi, F. Wolter, and M. Zakharyashev. A temporal description logic for reasoning about conceptual schemas and queries. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proceedings of the 8th Joint European Conference on Logics in Artificial Intelligence (JELIA-02)*, volume 2424 of *LNAI*, pages 98–110. Springer, 2002.

6. Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2):70–118, 2005.
7. D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.
8. D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer, 1998.
9. D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.
10. J. Chomicki and D. Toman. Temporal logic in information systems. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, chapter 1. Kluwer, 1998.
11. O. Etzion, A. Gal, and A. Segev. Extended update functionality in temporal databases. In O. Etzion, S. Jajodia, and S. Sripada, editors, *Temporal Databases - Research and Practice*, Lecture Notes in Computer Science, pages 56–95. Springer-Verlag, 1998.
12. M. Finger and P. McBrien. Temporal conceptual-level databases. In D. Gabbay, M. Reynolds, and M. Finger, editors, *Temporal Logics – Mathematical Foundations and Computational Aspects*, pages 409–435. Oxford University Press, 2000.
13. D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyashev. *Many-dimensional modal logics: theory and applications*. Studies in Logic. Elsevier, 2003.
14. H. Gregersen and J.S. Jensen. Conceptual modeling of time-varying information. Technical Report TimeCenter TR-35, Aalborg University, Denmark, 1998.
15. H. Gregersen and J.S. Jensen. Temporal Entity-Relationship models – a survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):464–497, 1999.
16. R. Gupta and G. Hall. Modeling transition. In *Proc. of ICDE'91*, pages 540–549, 1991.
17. R. Gupta and G. Hall. An abstraction mechanism for modeling generation. In *Proc. of ICDE'92*, pages 650–658, 1992.
18. I. Hodkinson, F. Wolter, and M. Zakharyashev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106:85–134, 2000.
19. I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
20. C. S. Jensen, J. Clifford, S. K. Gadia, P. Hayes, and S. Jajodia et al. The Consensus Glossary of Temporal Database Concepts. In O. Etzion, S. Jajodia, and S. Sripada, editors, *Temporal Databases - Research and Practice*, pages 367–405. Springer-Verlag, 1998.
21. C. S. Jensen and R. T. Snodgrass. Temporal data management. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):36–44, 1999.
22. C. S. Jensen, M. Soo, and R. T. Snodgrass. Unifying temporal data models via a conceptual model. *Information Systems*, 9(7):513–547, 1994.
23. P. McBrien, A.H. Seltveit, and B. Wangler. An Entity-Relationship model extended to describe historical information. In *Proc. of CISMOT'92*, pages 244–260, Bangalore, India, 1992.
24. C. Parent, S. Spaccapietra, and E. Zimanyi. The MurMur project: Modeling and querying multi-representation spatio-temporal databases. *Information Systems*, 2005.
25. S. Spaccapietra, C. Parent, and E. Zimanyi. Modeling time from a conceptual perspective. In *Int. Conf. on Information and Knowledge Management (CIKM98)*, 1998.
26. C. Theodoulidis, P. Loucopoulos, and B. Wangler. A conceptual modelling formalism for temporal database applications. *Information Systems*, 16(3):401–416, 1991.
27. Wikipedia. Wikipedia, the free encyclopedia. Temporal Databases. see http://en.wikipedia.org/wiki/Temporal_database.