Modeling the Evolution of Objects in Temporal Information Systems

A. Artale[1], C. Parent[2], S. Spaccapietra[3]

[1] Faculty of Computer Science, Free Univ. of Bolzano, I; `artale@inf.unibz.it`
[2] HEC/INFORGE, Université de Lausanne, CH; `christin@lbd.epfl.ch`
[3] Database Laboratory, Swiss Federal Institute of Technology, CH;
`Stefano.Spaccapietra@epfl.ch`

Abstract. This paper gives a formalization of the various modeling constructs that support the design of temporal DBMS. We conduct a deep investigation on evolution constraints, eventually devising a model-theoretic semantics for a full-fledged model with both timestamping and evolution constraints. Furthermore, we also show how to express temporal constraints using a subset of first-order temporal logic, i.e., the temporal description logic $\mathcal{DLR}_{\mathcal{US}}$.

1 Introduction

This paper aims at continuing the research efforts in the Conceptual Modeling community to model temporal information systems. An analysis of many proposals for temporal models (aiming in particular at helping designing temporal databases) and a summary of results achieved can be found in a good survey by Jensen and Snodgrass [17]. The main features of a temporal modeling language can be summarized as:

- **Timestamping.** The data model should obviously distinguish between temporal and atemporal modeling constructs. 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 objects and relationship instances, and time-varying values to attributes. Lifecycle information expresses when and how an object belongs to a class. Time-varying attributes store values together with when they hold (usually referring to valid time).
- **Evolution Constraints.** *Model-level* constraints rule the permissible evolution (change of membership status) of an object along its lifespan phases. For example, an object that is an active member of a class may become an inactive member of the same class. *Application-level* constraints 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.

The contribution of this paper is to give a formalization of the various temporal constructs with particular attention to evolution constraints. Indeed, while timestamping aspects have been extensively discussed [3, 4, 10, 12, 18, 21], 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, 20, 14, 13, 19, 21]. The proposed formalization relies on a model-theoretic semantics aiming at both formally clarifying the temporal constructs and to support reasoning over them. Concerning the reasoning aspects, we target a description logic approach, best suited for reasoning on conceptual models [8]. On the other hand, we do not address here well known issues related to the implementation of temporal specifications within a DBMS.

The formalization proposed here builds on previous efforts to formalize temporal conceptual models. In particular, [3, 4] define the $\mathcal{ER}_{VT}$ model, a temporal EER model based on a model-theoretic semantics. $\mathcal{ER}_{VT}$ is equipped with timestamping capabilities and both a linear and a graphical syntax. In this paper 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. Furthermore, we also show how to express temporal constraints using a subset of first-order temporal logic, i.e., the temporal description logic $\mathcal{DLR}_{\mathcal{US}}$ [5]. $\mathcal{DLR}_{\mathcal{US}}$ is based on the expressive and decidable description logic $\mathcal{DLR}$ that allows for the logical reconstruction and the extension of representational tools such as object-oriented and semantic data models, frame-based and web ontology languages [7–9]. In this way, temporal constraints can be expressed in a succinct way while reasoning techniques could be used to derive new constraints. Few remarks on the complexity of reasoning over temporal schemas. Full $\mathcal{DLR}_{\mathcal{US}}$ is undecidable [5]. The language $\mathcal{DLR}_{\mathcal{US}}^-$, obtained by eliminating temporal operators in front of relationships, is decidable in ExpSpace [5]—thus $\mathcal{DLR}_{\mathcal{US}}^-$ cannot express the temporal behavior of relationships. The above undecidability result holds true even when reasoning just on $\mathcal{ER}_{VT}$ with timestamping and evolution constraints [2]. On the other hand, reasoning just on timestamping can be done in 2-ExpTime [6].

The paper is organized as follows. The next two Sections recall the characteristics of the description logic and the temporal conceptual model on which we build our proposal. Section 4 discusses the evolution constraints we address. Section 5 shows the formal definition of our evolution framework.

2 The Temporal Description Logic

As a language for expressing temporal conceptual schemas we use the combination of the propositional temporal logic with *Since* and *Until* and the (non-temporal) description logic $\mathcal{DLR}$ [7]. The resulting $\mathcal{DLR}_{\mathcal{US}}$ [5] *temporal description logic* can be regarded as a rather expressive fragment of the first-order temporal logic $L^{\{\text{since, until}\}}$ (cf. [10, 15]).

The basic syntactical types of $\mathcal{DLR}_{\mathcal{US}}$ 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 define inductively (complex) class and relation expressions as is shown in the upper part of Figure 1, where the binary constructs $(\sqcap, \sqcup, \mathcal{U}, \mathcal{S})$ 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 $\mathcal{DLR}_{\mathcal{US}}$ coincides with $\mathcal{DLR}$. For both class and relation expressions all the Boolean constructs 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 $\exists^{\lessgtr k}[U_j]R$ 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 symbols U_i with the corresponding position numbers i.

The language of $\mathcal{DLR}_{\mathcal{US}}$ 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 $t \in \mathcal{T}$, 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 $(u, v) = \{w \in \mathcal{T} \mid u < w < v\}$ and the operators $\square^+$ (always in the future) and $\square^-$ (always in the past) are

$$C \to \top \ | \ CN \ | \ \neg C \ | \ C_1 \sqcap C_2 \ | \ C_1 \sqcup C_2 \ | \ \exists^{\lessgtr k}[U_j]R \ |$$
$$\diamond^+ C \ | \ \diamond^- C \ | \ \Box^+ C \ | \ \Box^- C \ | \ \oplus C \ | \ \ominus C \ | \ C_1 \, \mathcal{U} \, C_2 \ | \ C_1 \, \mathcal{S} \, C_2$$
$$R \to \top_n \ | \ RN \ | \ \neg R \ | \ R_1 \sqcap R_2 \ | \ R_1 \sqcup R_2 \ | \ U_i/n : C \ |$$
$$\diamond^+ R \ | \ \diamond^- R \ | \ \Box^+ R \ | \ \Box^- R \ | \ \oplus R \ | \ \ominus R \ | \ R_1 \, \mathcal{U} \, R_2 \ | \ R_1 \, \mathcal{S} \, R_2$$

$$\top^{\mathcal{I}(t)} = \Delta^{\mathcal{I}}$$
$$CN^{\mathcal{I}(t)} \subseteq \top^{\mathcal{I}(t)}$$
$$(\neg C)^{\mathcal{I}(t)} = \top^{\mathcal{I}(t)} \setminus C^{\mathcal{I}(t)}$$
$$(C_1 \sqcap C_2)^{\mathcal{I}(t)} = C_1^{\mathcal{I}(t)} \cap C_2^{\mathcal{I}(t)}$$
$$(\exists^{\lessgtr k}[U_j]R)^{\mathcal{I}(t)} = \{ \, d \in \top^{\mathcal{I}(t)} \mid \sharp\{\langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(t)} \mid d_j = d\} \lessgtr k \}$$
$$(C_1 \, \mathcal{U} \, C_2)^{\mathcal{I}(t)} = \{ \, d \in \top^{\mathcal{I}(t)} \mid \exists v > t.(d \in C_2^{\mathcal{I}(v)} \wedge \forall w \in (t, v).d \in C_1^{\mathcal{I}(w)}) \}$$
$$(C_1 \, \mathcal{S} \, C_2)^{\mathcal{I}(t)} = \{ \, d \in \top^{\mathcal{I}(t)} \mid \exists v < t.(d \in C_2^{\mathcal{I}(v)} \wedge \forall w \in (v, t).d \in C_1^{\mathcal{I}(w)}) \}$$

$$(\top_n)^{\mathcal{I}(t)} \subseteq (\Delta^{\mathcal{I}})^n$$
$$RN^{\mathcal{I}(t)} \subseteq (\top_n)^{\mathcal{I}(t)}$$
$$(\neg R)^{\mathcal{I}(t)} = (\top_n)^{\mathcal{I}(t)} \setminus R^{\mathcal{I}(t)}$$
$$(R_1 \sqcap R_2)^{\mathcal{I}(t)} = R_1^{\mathcal{I}(t)} \cap R_2^{\mathcal{I}(t)}$$
$$(U_i/n : C)^{\mathcal{I}(t)} = \{ \, \langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid d_i \in C^{\mathcal{I}(t)} \}$$
$$(R_1 \, \mathcal{U} \, R_2)^{\mathcal{I}(t)} = \{ \, \langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid$$
$$\exists v > t.(\langle d_1, \ldots, d_n \rangle \in R_2^{\mathcal{I}(v)} \wedge \forall w \in (t, v).\langle d_1, \ldots, d_n \rangle \in R_1^{\mathcal{I}(w)}) \}$$
$$(R_1 \, \mathcal{S} \, R_2)^{\mathcal{I}(t)} = \{ \, \langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid$$
$$\exists v < t.(\langle d_1, \ldots, d_n \rangle \in R_2^{\mathcal{I}(v)} \wedge \forall w \in (v, t).\langle d_1, \ldots, d_n \rangle \in R_1^{\mathcal{I}(w)}) \}$$
$$(\diamond^+ R)^{\mathcal{I}(t)} = \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v > t.\langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(v)}\}$$
$$(\oplus R)^{\mathcal{I}(t)} = \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(t+1)}\}$$
$$(\diamond^- R)^{\mathcal{I}(t)} = \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v < t.\langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(v)}\}$$
$$(\ominus R)^{\mathcal{I}(t)} = \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(t-1)}\}$$

Fig. 1. Syntax and semantics of $\mathcal{DLR}_{\mathcal{US}}$.

the duals of $\diamond^+$ (some time in the future) and $\diamond^-$ (some time in the past), respectively, i.e., $\Box^+ C \equiv \neg\diamond^+\neg C$ and $\Box^- C \equiv \neg\diamond^-\neg C$, for both classes and relations. For classes, the temporal operators $\diamond^+$, $\oplus$ (at the next moment), and their past counterparts can be defined via $\mathcal{U}$ and $\mathcal{S}$: $\diamond^+ C \equiv \top \, \mathcal{U} \, C$, $\oplus C \equiv \bot \, \mathcal{U} \, C$, etc. The operators $\diamond^*$ (at some moment) and its dual $\Box^*$ (at all moments) can be defined for both classes and relations as $\diamond^* C \equiv C \sqcup \diamond^+ C \sqcup \diamond^- C$ and $\Box^* C \equiv C \sqcap \Box^+ C \sqcap \Box^- C$, respectively.

A *knowledge base* is a finite set Σ of $\mathcal{DLR}_{\mathcal{US}}$ 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 $t \in \mathcal{T}$. 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. Finally, we say that Σ *implies* $C_1 \sqsubseteq C_2$, and write $\Sigma \models C_1 \sqsubseteq C_2$, if we have $\mathcal{I} \models C_1 \sqsubseteq C_2$ whenever $\mathcal{I} \models \Sigma$.

3 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 valid time for classes, attributes, and relationships. $\mathcal{ER}_{VT}$ is equipped with both
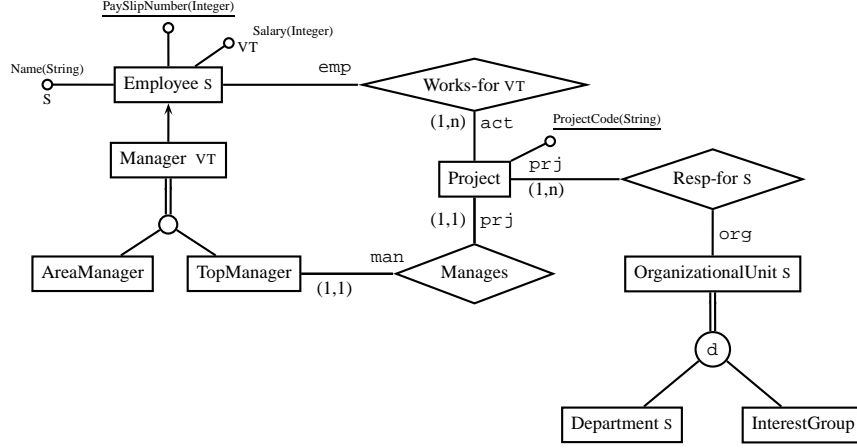
Fig. 2. An $\mathcal{ER}_{VT}$ diagram

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). $\mathcal{C}$ is further partitioned into: a set $\mathcal{C}^S$ of *snapshot classes* (the S-*marked* classes in Figure 2)[1], a set $\mathcal{C}^M$ of M*ixed classes* (the *unmarked* classes in Figure 2), and a set $\mathcal{C}^T$ of *temporary classes* (the VT-*marked* classes in Figure 2). A similar partition applies to the set $\mathcal{R}$. ATT is a function that maps a class symbol in $\mathcal{C}$ to an $\mathcal{A}$-labeled tuple over $\mathcal{D}$, $\text{ATT}(E) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$. REL is a function that maps a relationship symbol in $\mathcal{R}$ to an $\mathcal{U}$-labeled tuple over $\mathcal{E}$, $\text{REL}(R) = \langle U_1 : C_1, \ldots, U_k : C_k \rangle$, 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, $\text{CARD}(\texttt{TopManager}, \texttt{Manages}, \texttt{man}) = (1, 1)$. 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. $\texttt{Manager}$ ISA $\texttt{Employee}$ 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. $\texttt{Department}, \texttt{InterestGroup}$ are both disjoint and they cover $\texttt{OrganizationalUnit}$. 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*[2] 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

---

[1] We adopt an EER style where classes are in boxes and relationships inside diamonds, ISA are directed lines, generalized hierarchies could be disjoint (circle with a 'd' inside) or covering (double directed lines).

[2] The snapshot model represents the same class of temporal databases as the *timestamp* model [17, 18] defined by adding temporal attributes to a relation [10].

be isomorphic to either $\langle \mathbb{Z}, < \rangle$ or $\langle \mathbb{N}, < \rangle$. Thus, a temporal database can be regarded as a mapping from time points in $\mathcal{T}$ to standard relational databases, 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 $\mathcal{B} = (\mathcal{T}, \Delta^{\mathcal{B}} \cup \Delta_D^{\mathcal{B}}, \cdot^{\mathcal{B}(t)})$, 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. $\cdot^{\mathcal{B}(t)}$ is a function such 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 an n-ary relationship connecting the classes $C_1, \ldots, C_n$, $\text{REL}(R) = \langle U_1 : C_1, \ldots, U_n : C_n \rangle$, then, $r \in R^{\mathcal{B}(t)} \rightarrow (r = \langle U_1 : o_1, \ldots, U_n : o_n \rangle \wedge \forall i \in \{1, \ldots, n\}.o_i \in C_i^{\mathcal{B}(t)})$. We adopt the convention: $\langle U_1 : o_1, \ldots, U_n : o_n \rangle \equiv \langle o_1, \ldots, o_n \rangle$, when U-labels are clear from the context.*
- *Every attribute A to a set $A^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}} \times \Delta_D^{\mathcal{B}}{}^{\mathcal{B}}$.*

$\mathcal{B}$ is said a *legal temporal database state* if it satisfies all of the constraints expressed in the schema. In particular, in the following we will show how $\mathcal{B}$ gives a semantic account to timestamping.

3.1 Timestamping

We illustrate timestamping just for classes. Similar ideas are used in $\mathcal{ER}_{VT}$ to associate timestamping to both relationships and attributes.

$\mathcal{ER}_{VT}$ is able to distinguish between *snapshot* (see the consensus glossary [16] for the terminology used) constructs—i.e. each of their instances has a global lifespan (see Section 5.1)—*temporary* constructs—i.e. each of their instances have a limited lifespan—or *mixed* constructs—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. Two temporal marks, s (snapshot) and t (temporary), are introduced at the conceptual level to capture such temporal behavior. The semantics of timestamping is the following:

$$o \in C^{\mathcal{B}(t)} \rightarrow \forall t' \in \mathcal{T}.o \in C^{\mathcal{B}(t')} \quad \texttt{Snapshot Class}$$
$$o \in C^{\mathcal{B}(t)} \rightarrow \exists t' \neq t.o \notin C^{\mathcal{B}(t')} \quad \texttt{Temporary Class}$$

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

$$C \sqsubseteq (\Box^+ C) \sqcap (\Box^- C) \quad \texttt{Snapshot Class}$$
$$C \sqsubseteq (\Diamond^+ \neg C) \sqcup (\Diamond^- \neg C) \quad \texttt{Temporary Class}$$

Note that, the distinction between snapshot, temporary and mixed constructors has been adopted to avoid *overloading* the meaning of un-marked constructors. Indeed, those models that distinguish just between temporal (using a temporal mark) and atemporal (leaving the constructor un-marked) constructors are ambiguous in the meaning of un-marked constructors. From one side, the intended meaning of un-marked constructors is that they retain the atemporal semantics of standard ER models. Thus, they are used to model either legacy constructors (achieving *upward compatibility*) or atemporal portion of the database. On the other side, due to the interaction between the various component of a temporal model, un-marked constructors could be intended even as temporary

constructors. As an example, think to an ISA involving a temporary class (as superclass) and an un-marked class (as a subclass). Since a designer cannot forecast all the possible temporal interactions, this ultimately means that atemporality cannot be guaranteed and this is true even for the upward compatibility. $\mathcal{ER}_{VT}$ is more strict imposing a snapshot mark to force both atemporality and upward compatibility. This point of view is also reflected when mapping $\mathcal{ER}_{VT}$ into a relational schema where 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]).

4 Evolution Constraints

Evolution Constraints are used in order to model the temporal behaviors of an object. Here we mention the various evolution constructors appeared in the literature and their impact in the resulting conceptual language.

Status Classes [20, 11] are used to describe the status of membership of an object w.r.t. a temporal class. In a temporal setting, objects can be suspended and further resumed in their membership. Usually, four different status are specified together with precise transitions between them:

- Scheduled. An object is know in advance to belong to a class (e.g., an new approved project but still not officially started). Each scheduled object will eventually become an active one.
- Active. Describes an object that is actually member of the class.
- Suspended. Describes objects that still exist as member of the class but some operations are no more permitted. Usually it is not allowed to modify properties of suspended objects (e.g., an employee taking 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. In particular, a disabled object was in the past a member of the class but it cannot become again a member of that class (e.g., an expired project).

Transition constraints [13, 14, 20] have been introduced to model the phenomenon called *object migration*. A transition constraint involves a *source* and a *target* class: the instances of the source class may *migrate* into the target class and migrating objects must be recorded. 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, we could specify a dynamic evolution between the class of Undergraduate and that one of Postgraduate students, while a dynamic extension could model the transition between the class of Students and that one of Employees (thus, we allow for an Employee to migrate back and become again a Student).

Generation relationships [20] involve different instances—differently from the transition case: an instance (or set of instances) from a source class is (are) transformed in an instance (or set of instances) of the target class. Depending whether the source object is preserved (as member of the source class) or disabled, we distinguish between a *production* and a *transformation*, respectively. For example, a generation relationship, Generate, between Orange and Juice specifies that oranges are transformed into orange juice. Cardinality constraints can be added to specify the cardinality of sets involved in a generation.

Cross-Time relationships [21, 19, 20] involve objects that do not exist at the same time the relationship is asserted. There are many examples of these relationships, consider, for
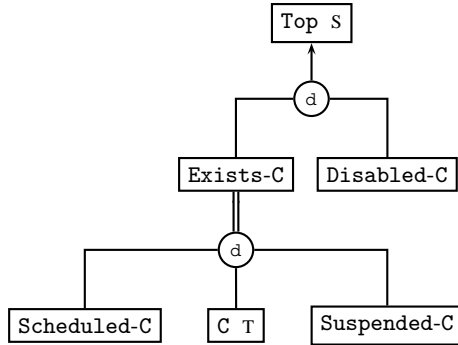
Fig. 3. Status classes.

example, a relationship "biography" between an author and a famous person already dead, or the relationships "grandparent" that could be asserted even when either the grandparent or the grandchild do not exist anymore.

5 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 $\mathcal{DLR}_{\mathcal{US}}$. This will give us both a formal characterization of the temporal conceptual modeling constructs, and the possibility to use the reasoning capabilities of $\mathcal{DLR}_{\mathcal{US}}$ to reason 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 $\mathcal{ER}_{VT}$ and introduced in Section 3.

5.1 Status Classes

Transitions between status classes model the evolution in the membership of an object to a temporal class. Let C be a temporal class, i.e., either a temporary or a mixed class (see Sect. 3.1), we capture status transition of membership in C by associating to C the following *status classes*: Scheduled-C, Suspended-C, Disabled-C. To preserve upward compatibility we do not explicitly introduce an active class but we assume by default that the name of the class qualify itself as the set of active objects— Active-C $\equiv$ C. In this way, we could assume that the status classes are created by the system each time a class is declared temporal. Thus, the user is not forced neither to introduce nor to manipulate status classes: designers could only be aware of active classes while status classes could be completely transparent to them. Note that, since membership of objects into snapshot classes is global, the notion of status classes does not apply to snapshot classes.

To capture the intended meaning of status classes we associate ad-hoc constraints and then prove that such constraints capture their evolving behavior as described in the literature [20, 11]. First of all, disjointness constraints can be described by the conceptual schema in Figure 3 where C is marked as a temporary class while Top is supposed to be snapshot[3]. Other than the disjointness constraints the semantic of status classes should reflect the following temporal behavior:

---

[3] A similar diagram holds when C is an unmarked, i.e., mixed, class.

(EXISTS) *Existence persists until Disabled.*
$$o \in \texttt{Exists-C}^{\mathcal{B}(t)} \to \forall t' > t.(o \in \texttt{Exists-C}^{\mathcal{B}(t')} \vee o \in \texttt{Disabled-C}^{\mathcal{B}(t')})$$
(DISAB1) *Disabled persists.*
$$o \in \texttt{Disabled-C}^{\mathcal{B}(t)} \to \forall t' > t.o \in \texttt{Disabled-C}^{\mathcal{B}(t')}$$
(DISAB2) *Disabled was Active in the past.*
$$o \in \texttt{Disabled-C}^{\mathcal{B}(t)} \to \exists t' < t.o \in \texttt{C}^{\mathcal{B}(t')}$$
(SUSP) *Suspended was Active in the past.*
$$o \in \texttt{Suspended-C}^{\mathcal{B}(t)} \to \exists t' < t.o \in \texttt{C}^{\mathcal{B}(t')}$$
(SCH1) *Scheduled will eventually become Active.*
$$o \in \texttt{Scheduled-C}^{\mathcal{B}(t)} \to \exists t' > t.o \in \texttt{C}^{\mathcal{B}(t')}$$
(SCH2) *Scheduled can never follow Active.*
$$o \in \texttt{C}^{\mathcal{B}(t)} \to \forall t' > t.o \notin \texttt{Scheduled-C}^{\mathcal{B}(t')}$$

The above semantics is captured by the following $\mathcal{DLR}_{\mathcal{US}}$ axioms:

(EXISTS) $\texttt{Exists-C} \sqsubseteq \Box^+(\texttt{Exists-C} \sqcup \texttt{Disabled-C})$
(DISAB1) $\texttt{Disabled-C} \sqsubseteq \Box^+\texttt{Disabled-C}$
(DISAB2) $\texttt{Disabled-C} \sqsubseteq \Diamond^-\texttt{C}$
(SUSP) $\texttt{Suspended-C} \sqsubseteq \Diamond^-\texttt{C}$
(SCH1) $\texttt{Scheduled-C} \sqsubseteq \Diamond^+\texttt{C}$
(SCH2) $\texttt{C} \sqsubseteq \Box^+\neg\texttt{Scheduled-C}$

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, as a logical consequence from the above axioms we have:

(SCH3) Scheduled persists until active: $\texttt{Scheduled-C} \sqsubseteq \texttt{Scheduled-C}\,\mathcal{U}\,\texttt{C}$. Together with axiom (SCH2), we can conclude that $\texttt{Scheduled-C}$ is true just on a single interval.
(SCH4) Scheduled cannot evolve directly to Disabled: $\texttt{Scheduled-C} \sqsubseteq \oplus \neg\texttt{Disbled-C}$.
(DISAB3) Disabled was active but it will never become active anymore:
$\texttt{Disabled-C} \sqsubseteq \Diamond^-(\texttt{C} \sqcap \Box^+\neg\texttt{C})$.

In the following we will show the adequacy of the semantics associated to status classes to describe: *a*) the notions of *lifespan, birth* and *death* of an object; *b*) the behavior of temporal classes involved in ISA relations; *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 relation is specified between two temporal classes, say B ISA A, then the following constraints hold between the status classes:

1. Objects active in B must be active in A;
2. Objects suspended in B must be either suspended or active in A;
3. Objects disabled in B must be either disabled, suspended or active in A;
4. Objects scheduled in B cannot be disabled in A.

Assuming that each time an isa between two temporal classes is asserted (B ISA A) then this is implicitly followed by an isa between the respective existing status classes (Exists-B ISA Exists-A), then the following proposition shows that points (1-4) above are entailed by the semantics associated to status classes.

Proposition 1. *Let A, B two temporal classes such that B ISA A, then properties (1-4) are true.*
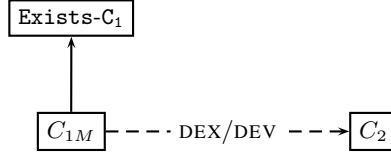
Fig. 4. Dynamic Transition

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, $\text{EXISTENCESPAN}_C : \Delta^{\mathcal{B}} \to 2^{\mathcal{T}}$, such that:

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

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, $\text{LIFESPAN}_C : \Delta^{\mathcal{B}} \to 2^{\mathcal{T}}$, such that:

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

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, $\text{ACTIVESPAN}_C : \Delta^{\mathcal{B}} \to 2^{\mathcal{T}}$, such that:

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

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, $\text{BEGIN}_C, \text{BIRTH}_C$, $\text{DEATH}_C : \Delta^{\mathcal{B}} \to \mathcal{T}$, such that:

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

Remark 1. We could still speak of existencespan, lifespan or activespan in case of snapshot classes but $\text{EXISTENCESPAN}_C(o) \equiv \text{LIFESPAN}_C(o) \equiv \text{ACTIVESPAN}_C(o) \equiv \mathcal{T}$.

5.2 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 [20, 14, 13]: *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. In a temporal setting objects could obviously change their membership class. Specifying a transition between two classes means that:

1. We want to keep track of such migration;
2. Not necessarily all the objects in the source participate in the migration;
3. When the source class is a temporal class, migration involves only objects "existing" in the class (i.e., scheduled, active and suspended objects).

To satisfy the above requirements, a transition between classes C_1, C_2 is actually a transition between a sub-class of the source and the target. More precisely, as illustrated in Figure 4, the (sub-)class C_{1M} keeps track of the objects (either scheduled, active or suspended) in the source class that migrate to the target class C_2. Note that, we disallow disabled objects to take part in a transition. Concerning the graphical representation, we use a dashed arrow pointing to the target class and labeled with either DEX or DEV denoting dynamic extension and evolution, respectively. Please note that, in case C_1 is a snapshot class, then Figure 4 is changed in such a way that C_{1M} ISA C_1 will hold (instead of C_{1M} ISA Exists-C$_1$). More formally, in case of a *dynamic extension* between classes C_1, C_2 the following semantic equation hold:

$$o \in C_{1M}^{\mathcal{B}(t)} \rightarrow (o \in \texttt{Exists-C}_1{}^{\mathcal{B}(t)} \wedge o \notin C_2^{\mathcal{B}(t)} \wedge o \in C_2^{\mathcal{B}(t+1)})$$

And the equivalent set of $\mathcal{DLR_{US}}$ axioms is:

$$C_{1M} \sqsubseteq \texttt{Exists-C}_1$$
$$C_{1M} \sqsubseteq \neg C_2 \sqcap \oplus C_2$$

In case of a *dynamic evolution* between classes C_1, C_2 the source object cannot belong to the source class till the migration is in place. Thus, the following semantic equation holds:

$$o \in C_{1M}^{\mathcal{B}(t)} \rightarrow (o \in \texttt{Exists-C}_1{}^{\mathcal{B}(t)} \wedge o \notin C_2^{\mathcal{B}(t)} \wedge o \in C_2^{\mathcal{B}(t+1)} \wedge$$
$$\forall t' \geq t + 1.(o \in C_2^{\mathcal{B}(t')} \rightarrow o \notin C_1^{\mathcal{B}(t')}))$$

And the equivalent set of $\mathcal{DLR_{US}}$ axioms is:

$$C_{1M} \sqsubseteq \texttt{Exists-C}_1$$
$$C_{1M} \sqsubseteq \neg C_2 \sqcap \oplus C_2$$
$$C_{1M} \sqsubseteq \square^+(C_2 \rightarrow \neg C_1)$$

An interesting set of logical consequences of the above proposed modeling of dynamic extension is:

1. The class C_{1M} is a temporary class.
2. Objects in the class C_{1M} cannot be disabled as C_2.
3. The target class C_2 cannot be snapshot.

On the other hand, a logical consequence of dynamic evolution (in addition to the ones stated above) is that the source class, C_1, cannot be snapshot (and it becomes temporary if all of its members are involved in the migration). Indeed, an object evolving from C_1 to C_2 ceases to be a member of C_1.

5.3 Generation Relationships

Generation relationships [20] 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
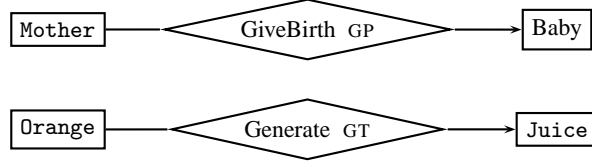
Fig. 5. Production and transformation generation relationships.

level we introduce two marks associated to a relationship: GP for production and GT for transformation relationships (see Figure 5). Furthermore, an arrow points to the target class.

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

$$\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow (o_1 \in C_1^{\mathcal{B}(t)} \wedge o_2 \in \texttt{Scheduled-C}_2{}^{\mathcal{B}(t)} \wedge o_2 \in C_2^{\mathcal{B}(t+1)})$$

Thus, an object active in the source produces an object active in the target at the next point in time. The $\mathcal{DLR}_{\mathcal{US}}$ axiom capturing the production semantics is:

$$R \sqsubseteq \texttt{source} : C_1 \sqcap \texttt{target} : (\texttt{Scheduled-C}_2 \sqcap \oplus C_2)$$

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

$$\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow (o_1 \in C_1^{\mathcal{B}(t)} \wedge o_1 \in \texttt{Disabled-C}_1{}^{\mathcal{B}(t+1)} \wedge$$
$$o_2 \in \texttt{Scheduled-C}_2{}^{\mathcal{B}(t)} \wedge o_2 \in C_2^{\mathcal{B}(t+1)})$$

Thus, an object active in the source is transformed in an object active in the target at the next point in time while ceasing to exist as member of the source. The $\mathcal{DLR}_{\mathcal{US}}$ axiom capturing the production semantics is:

$$R \sqsubseteq \texttt{source} : (C_1 \sqcap \oplus \texttt{Disabled-C}_1) \sqcap \texttt{target} : (\texttt{Scheduled-C}_2 \sqcap \oplus C_2)$$

Logical consequences of the above formalization are:

1. The target class, C_2, cannot be snapshot (is temporary in case of total participation).
2. A generation relationship, R, is temporary.
3. If R is a transformation relationship, then, both C_1 and C_2 cannot be snapshot.

5.4 Cross-Time Relationships

Cross-time relationships relate objects that are member of the participating classes at different times. The conceptual model MADS [20] 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 [19] to express a relationships 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. We explain this with a concrete example. Let
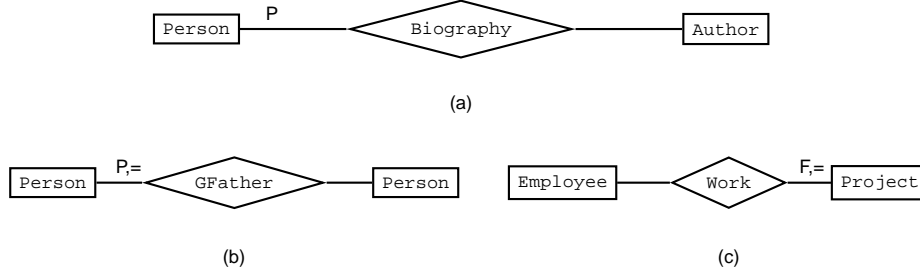
Fig. 6. Cross-Time Relationships

`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, $\texttt{bio} = \langle \texttt{Tulard}, \texttt{Napoleon} \rangle$) of the `Biography` relationship at time t_0 (in particular, Tulard wrote a bio on Napoleon on 1984), then, the snapshot of `Biography` at time t_0 (1984 in our example) must contain the pair $\langle \texttt{Tulard}, \texttt{Napoleon} \rangle$. 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 P,= and F,= while specifying just = doesn't add any constraint. Assuming that R is a cross-time relationship between classes C_1, C_2, then, the semantics of marking the C_1 link is:

$r = \langle e_1, e_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow e_1 \in \texttt{Disabled-C}_1^{\mathcal{B}(t)} \wedge e_2 \in C_2^{\mathcal{B}(t)}$ Strictly Past (P)

$r = \langle e_1, e_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow e_1 \in (C_1 \sqcup \texttt{Disabled-C}_1)^{\mathcal{B}(t)} \wedge e_2 \in C_2^{\mathcal{B}(t)}$ Past (P,=)

$r = \langle e_1, e_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow e_1 \in \texttt{Scheduled-C}_1^{\mathcal{B}(t)} \wedge e_2 \in C_2^{\mathcal{B}(t)}$ Strictly Future (F)

$r = \langle e_1, e_2 \rangle \in R^{\mathcal{B}(t)} \rightarrow e_1 \in (C_1 \sqcup \texttt{Scheduled-C}_1)^{\mathcal{B}(t)} \wedge e_2 \in C_2^{\mathcal{B}(t)}$ Future (F,=)

The corresponding $\mathcal{DLR}_{\mathcal{US}}$ formalization is:

$R \sqsubseteq U_1 : \texttt{Disabled-C}_1 \sqcap U_2 : \texttt{C}_2$ Strictly Past (P)
$R \sqsubseteq U_1 : (C_1 \sqcup \texttt{Disabled-C}_1) \sqcap U_2 : C_2$ Past (P,=)
$R \sqsubseteq U_1 : \texttt{Scheduled-C}_1 \sqcap U_2 : \texttt{C}_2$ Strictly Future (F)
$R \sqsubseteq U_1 : (C_1 \sqcup \texttt{Scheduled-C}_1) \sqcap U_2 : C_2$ Future (F,=)

The diagram (a) of Figure 6 shows the modeling of the `Biography` example. The diagram (b) shows how to use past marks to represent the `GrandFather` relationship assuming that the grandfather can be either alive or dead for the relationship to hold. Finally, diagram (c) shows the use of the future mark to model the fact that an employee can work on a project before the project officially starts. Note that marks can be added to both participating classes. For example, adding the mark F,= on the grandchild link allows for representing the case where grandparent holds even when the grandchild is not yet born.

Interesting logical consequences of the given formalization hold when strict constraints are specified (let assume that C_1 participates with a strict past or future mark):

1. Both C_1 and the cross-time relationship are temporary.
2. The lifespan of objects in C_1 is strictly before (strictly after for future marks) of the lifespan of linked objects in C_2.

References

1. Basel 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 International Symposium on Temporal Representation and Reasoning (TIME04)*. IEEE Computer Society, 2004. Also in Proc. of the 2004 International Workshop on Description Logics (DL'04).
3. A. Artale and E. Franconi. Temporal ER modeling with description logics. In *Proc. of the International Conference 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. Zakharyaschev. 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. Alessandro Artale, Carsten Lutz, and David Toman. A description logic of change. LTCS-Report 05-06, Technical University Dresden, 2002. see http://lat.inf.tu-dresden.de/research/reports.html.
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 functionality in temporal databases. In O. Etzion, S. Jajodia, and S. Sripada, editors, *Temporal Databases - Research and Practice*, Lecture Notes in Computer Science, pages 36–55. Springer-Verlag, 1998.
12. H. Gregersen and J.S. Jensen. Conceptual modeling of time-varying information. Technical Report TimeCenter TR-35, Aalborg University, Denmark, 1998.
13. R. Gupta and G. Hall. Modeling transition. In *Proc. of ICDE'91*, pages 540–549, 1991.
14. R. Gupta and G. Hall. An abstraction mechanism for modeling generation. In *Proc. of ICDE'92*, pages 650–658, 1992.
15. I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106:85–134, 2000.
16. 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.
17. C. S. Jensen and R. T. Snodgrass. Temporal data management. *IEEE Transactions on Knowledge and Data Engineering*, 111(1):36–44, 1999.
18. C. S. Jensen, M. Soo, and R. T. Snodgrass. Unifying temporal data models via a conceptual model. *Information Systems*, 9(7):513–547, 1994.
19. P. McBrien, A.H. Seltveit, and B. Wangler. An Entity-Relationship model extended to describe historical information. In *Proc. of CISMOD'92*, pages 244–260, Bangalore, India, 1992.
20. S. Spaccapietra, C. Parent, and E. Zimanyi. Modeling time from a conceptual perspective. In *Int. Conf. on Information and Knowledge Management (CIKM98)*, 1998.
21. C. Theodoulidis, P. Loucopoulos, and B. Wangler. A conceptual modelling formalism for temporal database applications. *Information Systems*, 16(3):401–416, 1991.