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; christine.parent@unil.ch
[3] Database Laboratory, Ecole Polytechnique Fédérale Lausanne, 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., $\mathcal{DLR}_{\mathcal{US}}$, the description logic $\mathcal{DLR}$ extended with the temporal operators *Since* and *Until*. The proposed formalization is meant both to clarify the meaning of the various temporal constructors appeared in the literature and to show the possibility to perform automated reasoning on temporal conceptual models.

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 [15]. 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. Complementary aspects of evolution are modeled through *generation relationships*, which describe the fact that objects in a class are generated by other objects in another (possibly the same) class. For example, in a cadastre database, splitting of a parcel translates into the fact that the original parcel generates two (or more) new parcels.

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 [2, 3, 8, 10, 16, 19], 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 [3, 18, 12, 11, 17, 19]. 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 adopt a description logic approach, best suited for reasoning on conceptual models [6]. 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. Namely, we rely on previous work to define the $\mathcal{ER}_{VT}$ model [3], 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 $\mathcal{DLR}_{\mathcal{US}}$ [4]. $\mathcal{DLR}_{\mathcal{US}}$ 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* ($\mathcal{S}$) and *Until* ($\mathcal{U}$) 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 [5–7]. In this paper, we use $\mathcal{DLR}_{\mathcal{US}}$ to capture the temporal constraints useful to design a temporal database in a succinct way while reasoning techniques[1] can be used to derive new constraints.

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 illustrates the modeling requirements that lead us in elaborating, in Section 6, the formal definition of our evolution framework. Section 7 concludes the paper.

2 The Temporal Description Logic

As a language for expressing temporal conceptual schemas we use the $\mathcal{DLR}_{\mathcal{US}}$ [4] *temporal description logic*, which combines the propositional temporal logic with *Since* and *Until* and the (non-temporal) description logic $\mathcal{DLR}$ [5]. $\mathcal{DLR}_{\mathcal{US}}$ can be regarded as a rather expressive fragment of the first-order temporal logic $L^{\{\textbf{since, until}\}}$ (cf. [8, 13]).

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 hereinafter define inductively (complex) class and relation expressions as is shown in the upper part of Fig. 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

[1] Even if full $\mathcal{DLR}_{\mathcal{US}}$ is undecidable interesting subsets of it are decidable [4].

$$
\begin{aligned}
C \rightarrow\ & \top \mid CN \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists^{\leqslant k}[U_j]R \mid \\
& \Diamond^+ C \mid \Diamond^- C \mid \Box^+ C \mid \Box^- C \mid \oplus C \mid \ominus C \mid C_1 \,\mathcal{U}\, C_2 \mid C_1 \,\mathcal{S}\, C_2 \\
R \rightarrow\ & \top_n \mid RN \mid \neg R \mid R_1 \sqcap R_2 \mid R_1 \sqcup R_2 \mid U_i/n : C \mid \\
& \Diamond^+ R \mid \Diamond^- R \mid \Box^+ R \mid \Box^- R \mid \oplus R \mid \ominus R \mid R_1 \,\mathcal{U}\, R_2 \mid R_1 \,\mathcal{S}\, R_2
\end{aligned}
$$

$$
\begin{aligned}
\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^{\leqslant 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\} \leqslant 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 \\
&\qquad \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 \\
&\qquad \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)}\}
\end{aligned}
$$

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

$U_i/n : C$ 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 $\exists^{\leqslant k}[U_j]R$ 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. To show the expressive power of $\mathcal{DLR}_{\mathcal{US}}$ we refer to the next Sections where $\mathcal{DLR}_{\mathcal{US}}$ is used to capture various forms of temporal constraints.

The model-theoretic semantics of $\mathcal{DLR}_{\mathcal{US}}$ 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 $\mathcal{DLR}_{\mathcal{US}}$ 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 $\Box^+$ (always in the future) and $\Box^-$ (always in the past) are 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

$\Diamond^+$, $\bigoplus$ (at the next moment), and their past counterparts can be defined via $\mathcal{U}$ and $\mathcal{S}$: $\Diamond^+ C \equiv \top \mathcal{U} C$, $\bigoplus C \equiv \bot \mathcal{U} C$, 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 $t \in \mathcal{T}$. Various *reasoning services* can be defined in $\mathcal{DLR}_{\mathcal{US}}$. 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. $\Sigma \not\models C \sqsubseteq \bot$.

3 The Temporal Conceptual Model $\mathcal{ER}_{VT}$

In this Section, the temporal EER model $\mathcal{ER}_{VT}$ [2, 3] is briefly introduced. $\mathcal{ER}_{VT}$ supports valid time 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 [7].

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 $\mathcal{D}$, $\text{ATT}(E) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$. REL is a function that maps a relationship symbol in $\mathcal{R}$ to an $\mathcal{U}$-labeled tuple over $\mathcal{C}$, $\text{REL}(R) = \langle U_1 : C_1, \ldots, U_k : C_k \rangle$, 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. `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)[2], 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 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.

[2] 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).

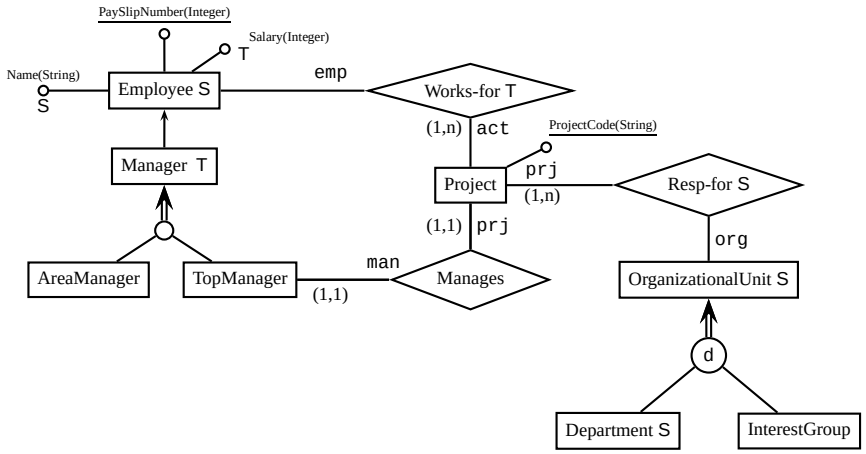

Fig. 2. An $\mathcal{ER}_{VT}$ diagram

The model-theoretic semantics associated with the $\mathcal{ER}_{VT}$ modeling language adopts the *snapshot*[3] representation of abstract temporal databases and temporal conceptual models [8]. 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 constructs, 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 $\mathcal{B} = (\mathcal{T}, \Delta^{\mathcal{B}} \cup \Delta^{\mathcal{B}}_D, \cdot^{\mathcal{B}(t)})$, such that: $\Delta^{\mathcal{B}}$ is a nonempty set disjoint from $\Delta^{\mathcal{B}}_D$; $\Delta^{\mathcal{B}}_D = \bigcup_{D_i \in \mathcal{D}} \Delta^{\mathcal{B}}_{D_i}$ 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 $D_i^{\mathcal{B}(t)} = \Delta^{\mathcal{B}}_{D_i}$.*
- *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 $C_1, \ldots, C_n$, $\text{REL}(R) = \langle U_1 : C_1, \ldots, U_n : C_n \rangle$, 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 $A^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}}_D$.*

$\mathcal{B}$ 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 $\mathcal{B}$ supports defining the semantics of 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.

[3] The snapshot model represents the same class of temporal databases as the *timestamp* model [15, 16] defined by adding temporal attributes to a relation [8].

$\mathcal{ER}_{VT}$ is able to distinguish between *snapshot* constructs—i.e. constructs which bear no explicit specification of a given lifespan [14], which we convey by assuming a global lifespan (see Section 6.1) associated to each of their instances—*temporary* constructs—i.e. each of their instances has 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. 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:

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

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

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

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, a mere 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 setting, un-marking is used to model both truly atemporal constructs (i.e., snapshot classes whose instances lifespan is always equal to the whole database lifespan), as well as legacy constructs (for *upward compatibility*) where the construct is not marked as temporal because the original data model 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 constructs. 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 *atemporality cannot be guaranteed* and this is true even for the upward compatibility. $\mathcal{ER}_{VT}$ is stricter in imposing a snapshot mark to force both atemporality and upward compatibility. Furthermore, $\mathcal{ER}_{VT}$ relies on a reasoning mechanism that in the above ISA example would acknowledge the designer of the change from un-marked to temporary; e.g. a temporary mark is deduced for both `AreaManger` and `TopManager` in Figure 2 (see [3] for an exaustive list of deductions involving timestamps). 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 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 [18, 9] is a concept 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. Normally, 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 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). 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 [11, 12, 18] 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, we could specify a dynamic evolution between the class of Undergraduate students and the class of Postgraduate students, while a dynamic extension could model the transition between the class Students and the class Employees (assuming an employee, formerly a student, may migrate back and become a student again).

Generation relationships [18] express that (sets of) objects in a target class may be generated from (sets of) objects in a source class. The same class may serve as source and target 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. For example, a transformation relationship, `Give`, 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 (e.g., no more than 5 oranges for 1 juice).

Cross-Time relationships [19, 17, 18] describe relationships between objects that do not exist at the same time and possibly not at the time the relationship is asserted. There are many examples of these relationships, consider, for example, a relationship "biography" between an author and a famous person already dead, or the relationships "grandparent" that holds even if the grandparent passed away before the grandchild was born, and could be asserted even when either the grandparent or the grandchild do not exist anymore.

5 Modeling Requirements

This Section illustrates the requirements that are frequently advocated in the literature on temporal data models. These requirements are not so obvious when dealing with evolving objects. The formalization carried out in this paper is mainly motivated by providing a data model able to respect these requirements also in presence of evolving objects.

– Orthogonality. Temporal constructs 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 constructs 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 [18] 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 are strictly related. Considered together, they allow to preserve the meaning of atemporal constructs. In particular, the meaning of classical constructs 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.

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 $\mathcal{DLR}_{\mathcal{US}}$. 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 $\mathcal{ER}_{VT}$, introduced in Section 3.

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 specifies these constraints.

Let C be a temporal (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 a fixed hierarchy (Figure 3) that classifies the 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., `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, a designer is not forced neither to introduce nor to manipulate status classes: (s)he can be aware only of active classes while status classes can be completely transparent to him/her.

Note that, since membership of objects into snapshot classes is global, the notion of status classes does not apply to snapshot classes.

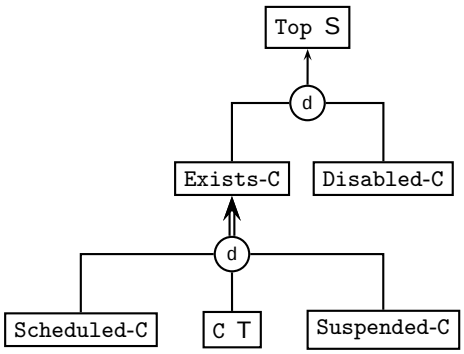

Fig. 3. Status classes.

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 [18, 9]. First of all, disjointness and ISA constraints between statuses can be described as illustrated in Figure 3, where C is marked as a temporary class while Top is supposed to be snapshot[4]. 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.*
$o \in \texttt{Exists-C}^{\mathcal{B}(t)} \rightarrow \forall t' > t.(o \in \texttt{Exists-C}^{\mathcal{B}(t')} \vee o \in \texttt{Disabled-C}^{\mathcal{B}(t')})$
(DISAB1**)** *Disabled persists.*
$o \in \texttt{Disabled-C}^{\mathcal{B}(t)} \rightarrow \forall t' > t.o \in \texttt{Disabled-C}^{\mathcal{B}(t')}$
(DISAB2**)** *Disabled was Active in the past.*
$o \in \texttt{Disabled-C}^{\mathcal{B}(t)} \rightarrow \exists t' < t.o \in \texttt{C}^{\mathcal{B}(t')}$
(SUSP**)** *Suspended was Active in the past.*
$o \in \texttt{Suspended-C}^{\mathcal{B}(t)} \rightarrow \exists t' < t.o \in \texttt{C}^{\mathcal{B}(t')}$
(SCH1**)** *Scheduled will eventually become Active.*
$o \in \texttt{Scheduled-C}^{\mathcal{B}(t)} \rightarrow \exists t' > t.o \in \texttt{C}^{\mathcal{B}(t')}$
(SCH2**)** *Scheduled can never follow Active.*
$o \in \texttt{C}^{\mathcal{B}(t)} \rightarrow \forall t' > t.o \notin \texttt{Scheduled-C}^{\mathcal{B}(t')}$

$\mathcal{DLR}_{\mathcal{US}}$ is able to fully capture the hierarchical constraints of Figure 3 (see [3] for more details). Moreover, the above semantic equations are 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:

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

(SCH3**)** Scheduled persists until active: $\texttt{Scheduled-C} \sqsubseteq \texttt{Scheduled-C}\,\mathcal{U}\,\texttt{C}$. Together with axiom (SCH2), we can conclude that Scheduled-C is true just on a single interval.
(SCH4**)** Scheduled cannot evolve directly to Disabled: $\texttt{Scheduled-C} \sqsubseteq \bigoplus \neg\texttt{Disbled-C}$.
(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 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 relationships; 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 ISA A, then the following constraints must hold between the respective 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;
5. Objects disabled in A and active in B in the past must be disabled in B.

The formalization of status classes provided above is not sufficient to guarantee properties (1-5)[5]. We need to further assume that the system behaves under the *temporal* ISA *assumption*: Each time an ISA between two temporal classes holds (B ISA A), then an ISA between the respective existence status classes (Exists-B ISA Exists-A) is automatically added by the system. Now, we are able to prove that points (1-5) above are entailed by the semantics associated to status classes under the temporal ISA assumption.

Proposition 1. *Let A, B be two temporal classes such that B ISA A, then properties (1-5) are true.*

Proof.

1. Obviously true since B ISA A holds, and both A, B are considered active.
2. Let $o \in \texttt{Suspended-B}^{\mathcal{B}(t_0)}$, since Suspended-B ISA Exists-B, and (by temporal ISA assumption) Exists-B ISA Exists-A, then, $o \in \texttt{Exists-A}^{\mathcal{B}(t_0)}$. 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), $o \notin \texttt{Scheduled-A}^{\mathcal{B}(t_0)}$. Thus, due to the disjoint covering constraint between active and suspended classes, either $o \in A^{\mathcal{B}(t_0)}$ or $o \in \texttt{Suspended-A}^{\mathcal{B}(t_0)}$.
3. Let $o \in \texttt{Disabled-B}^{\mathcal{B}(t_0)}$, then, by (DISAB2), $\exists t' < t_0 . o \in B^{\mathcal{B}(t')}$. By B ISA A and A ISA Exists-A, then, $o \in \texttt{Exists-A}^{\mathcal{B}(t')}$. By (EXISTS) and the disjointness between existing and disabled classes, there are only two possibilities at point in time $t_0 > t'$:
   (a) $o \in \texttt{Exists-A}^{\mathcal{B}(t_0)}$, and thus, by (SCH2), $o \in A^{\mathcal{B}(t_0)}$ or $o \in \texttt{Suspended-A}^{\mathcal{B}(t_0)}$; or
   (b) $o \in \texttt{Disabled-A}^{\mathcal{B}(t_0)}$.
4. Let $o \in \texttt{Scheduled-B}^{\mathcal{B}(t_0)}$, then, by (SCH1), $\exists t' > t_0 . o \in B^{\mathcal{B}(t')}$, and by B ISA A, $o \in A^{\mathcal{B}(t')}$. Thus, by (DISAB1) and the disjointness between active and disabled states, $o \notin \texttt{Disabled-A}^{\mathcal{B}(t_0)}$.

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

5. Let $o \in \texttt{Disabled-A}^{\mathcal{B}(t_0)}$ and $o \in B^{\mathcal{B}(t')}$ for some $t' < t_0$, then, $o \in \texttt{Exists-B}^{\mathcal{B}(t')}$. By (EXISTS) and the disjointness between existing and disabled classes, there are only two possibilities at point in time $t_0 > t'$: either $o \in \texttt{Exists-B}^{\mathcal{B}(t_0)}$ or $o \in \texttt{Disabled-B}^{\mathcal{B}(t_0)}$. By absurd, let $o \in \texttt{Exists-B}^{\mathcal{B}(t_0)}$, then by temporal ISA assumption, $o \in \texttt{Exists-A}^{\mathcal{B}(t_0)}$, which contradicts the assumption that $o \in \texttt{Disabled-A}^{\mathcal{B}(t_0)}$.

Please note that, as far as disjointness between classes is considered, this constraint just involves active classes. Thus no further constraints need to be specified.

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:

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

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, $\text{LIFESPAN}_C : \Delta^{\mathcal{B}} \rightarrow 2^{\mathcal{T}}$, 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, $\text{ACTIVESPAN}_C : \Delta^{\mathcal{B}} \rightarrow 2^{\mathcal{T}}$, 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, BEGIN_C, BIRTH_C, $\text{DEATH}_C : \Delta^{\mathcal{B}} \rightarrow \mathcal{T}$, such that:

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

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

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 [18, 12, 11]: *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 4, we use a dashed arrow pointing to the target class and labeled with either DEX or DEV denoting dynamic extension and evolution, respectively.

In a temporal setting, objects can obviously change their membership class. Specifying a transition between two classes means that:

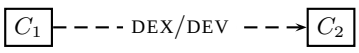

Fig. 4. Dynamic Transition

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). Thus, disabled objects cannot 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. Thus, a more natural choice seems to describe them as classes 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:

$$o \in \text{DEX}_{C_1,C_2}^{\mathcal{B}(t)} \rightarrow (o \in \texttt{Exists-C}_1{}^{\mathcal{B}(t)} \wedge o \in \texttt{Scheduled-C}_2{}^{\mathcal{B}(t)} \wedge o \in C_2^{\mathcal{B}(t+1)})$$

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

$$\text{DEX}_{C_1,C_2} \sqsubseteq \texttt{Exists-C}_1$$
$$\text{DEX}_{C_1,C_2} \sqsubseteq \texttt{Scheduled-C}_2 \sqcap \bigoplus C_2$$

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:

$$o \in \text{DEV}_{C_1,C_2}^{\mathcal{B}(t)} \rightarrow (o \in \texttt{Exists-C}_1{}^{\mathcal{B}(t)} \wedge o \in \texttt{Scheduled-C}_2{}^{\mathcal{B}(t)} \wedge o \in C_2^{\mathcal{B}(t+1)} \wedge \forall t' \geq t+1.(o \in C_2^{\mathcal{B}(t')} \rightarrow o \notin C_1^{\mathcal{B}(t')}))$$

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

$$\text{DEV}_{C_1,C_2} \sqsubseteq \texttt{Exists-C}_1$$
$$\text{DEV}_{C_1,C_2} \sqsubseteq \texttt{Scheduled-C}_2 \sqcap \bigoplus C_2$$
$$\text{DEV}_{C_1,C_2} \sqsubseteq \Box^+(C_2 \rightarrow \neg C_1)$$

Please note that, in case C_1 is a snapshot class, then, $\texttt{Exists-C}_1 \equiv C_1$. Finally, we formalize the case where the source (C_1) and/or the target (C_2) totally participate in a dynamic extension (at the conceptual level we add mandatory cardinality constraints):

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

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

$$C_1 \sqsubseteq \Diamond^+ \text{DEX}_{C_1,C_2} \quad \texttt{Source Total Transition}$$
$$C_2 \sqsubseteq \Diamond^- \text{DEX}_{C_1,C_2} \quad \texttt{Target Total Transition}$$

In a similar way we deal with dynamic evolution constraints.

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

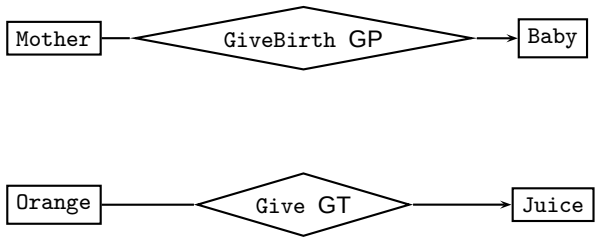

Fig. 5. Production and transformation generation relationships.

1. The classes DEX_{C_1,C_2} and DEV_{C_1,C_2} are temporary classes (actually, they are instantaneous).
   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, $o \notin \text{DEX}_{C_1,C_2}^{\mathcal{B}(t+1)}$. 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.
   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), $\text{DEX}_{C_1,C_2} \sqsubseteq \neg\texttt{Disabled-C}_2$. The same holds for DEV_{C_1,C_2}.
3. The target class C_2 cannot be snapshot (it becomes temporary if all of its members are involved in the migration).
   This is a direct consequence of the semantics of transitions where the migrating object cannot be a member of the target class before the transition happens.

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.

6.3 Generation Relationships

Generation relationships [18] 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 (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: $\text{REL}(R) = \langle \texttt{source} : C_1, \texttt{target} : \texttt{Scheduled-C}_2 \rangle$. The semantics of *production relationships*, R, is described by the following equation:

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

Thus, objects active in the source class produce objects active in the target class (possibly the same as the source 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 $\mathcal{DLR}_{\mathcal{US}}$ 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, 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 $\mathcal{DLR}_{\mathcal{US}}$ axiom capturing the transformation 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 (it becomes temporary if total participation is specified).
   Indeed, let $\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)}$, then, $o_2 \notin C_2^{\mathcal{B}(t)}$ and $o_2 \in C_2^{\mathcal{B}(t+1)}$.
2. A generation relationship, R, is temporary.
   Indeed, let $\langle o_1, o_2 \rangle \in R^{\mathcal{B}(t)}$, then, since $o_2 \notin \texttt{Scheduled-C}_2^{\mathcal{B}(t+1)}$, then, $\langle o_1, o_2 \rangle \notin R^{\mathcal{B}(t+1)}$.
3. If R is a transformation relationship, then, C_1 cannot be snapshot.
   Indeed, C_1 will be disabled at the next point in time.

6.4 Cross-Time Relationships

Cross-time relationships relate objects that are members of the participating classes at different times. The conceptual model MADS [18] 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 [17] 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 `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 just specifying = 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:

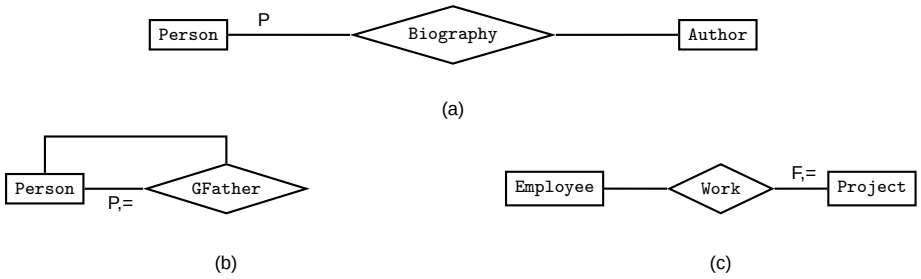

Fig. 6. Cross-Time Relationships

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

$R \sqsubseteq U_1 : \texttt{Disabled-C}_1 \sqcap \texttt{U}_2 : \texttt{C}_2$ Strictly Past (P)
$R \sqsubseteq U_1 : (C_1 \sqcup \texttt{Disabled-C}_1) \sqcap U_2 : C_2$ Past (P,=)
$R \sqsubseteq U_1 : \texttt{Scheduled-C}_1 \sqcap \texttt{U}_2 : \texttt{C}_2$ 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 assuming that a biography is written just on dead persons. 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.

7 Conclusions

In this paper we proposed a formalization of the various modeling constructs 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.

Starting from the model-theoretic semantics assigned to the temporal conceptual constructs, we have been able to show how temporal constraints can be equivalently expressed using a subset of first-order temporal logic, i.e., $\mathcal{DLR}_{\mathcal{US}}$, 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, and with the possibility to perform automated reasoning by mapping the conceptual model into a description logic theory.

References

1. B. Ahmad. Modeling bi-temporal databases. Master's thesis, UMIST Department of Computation, UK, 2003.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.
8. 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.
9. 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 56–95. Springer-Verlag, 1998.
10. H. Gregersen and J.S. Jensen. Conceptual modeling of time-varying information. Technical Report TimeCenter TR-35, Aalborg University, Denmark, 1998.
11. R. Gupta and G. Hall. Modeling transition. In *Proc. of ICDE'91*, pages 540–549, 1991.
12. R. Gupta and G. Hall. An abstraction mechanism for modeling generation. In *Proc. of ICDE'92*, pages 650–658, 1992.
13. I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106:85–134, 2000.
14. 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.
15. C. S. Jensen and R. T. Snodgrass. Temporal data management. *IEEE Transactions on Knowledge and Data Engineering*, 111(1):36–44, 1999.
16. C. S. Jensen, M. Soo, and R. T. Snodgrass. Unifying temporal data models via a conceptual model. *Information Systems*, 9(7):513–547, 1994.
17. 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.
18. S. Spaccapietra, C. Parent, and E. Zimanyi. Modeling time from a conceptual perspective. In *Int. Conf. on Information and Knowledge Management (CIKM98)*, 1998.
19. C. Theodoulidis, P. Loucopoulos, and B. Wangler. A conceptual modelling formalism for temporal database applications. *Information Systems*, 16(3):401–416, 1991.