Foundations of Temporal Conceptual Data Models

Alessandro Artale and Enrico Franconi

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
{artale,franconi}@inf.unibz.it

Abstract. This chapter considers the different temporal constructs appeared in the literature of temporal conceptual models (*timestamping* and *evolution constraints*), and it provides a coherent model-theoretic formalisation for them. It then introduces a correct and succinct encoding in a subset of first-order temporal logic, namely $\mathcal{DLR}_{\mathcal{US}}$ – the description logic $\mathcal{DLR}$ extended with the temporal operators *Since* and *Until*. At the end, results on the complexity of reasoning in temporal conceptual models are presented.

1 Introduction

Conceptual data models describe an application domain in a declarative and reusable way while constraining the use of the data by understanding what can be drawn from it. A number of conceptual modelling languages has emerged as de facto standards; in particular, we mention entity-relationship (ER) for the relational data model, UML and ODMG for the object-oriented data model, and RDF and OWL for the web ontology languages.

We consider here conceptual modelling languages able to represent dynamic and evolving information in the context of temporal databases [21, 26, 27, 32, 33, 38–40]. We provide in this chapter a mathematical foundation for them by summarising the various efforts appeared in the literature [4, 7, 23, 34, 35]. The main temporal modelling constructs we analyse can be distinguished in two main categories, *timestamping* and *evolution constraints*. To support *timestamping*, the data model should distinguish between temporal and atemporal modelling constructors; this is usually realised by a temporal marking of classes, relationships and attributes that translates into a timestamping mechanism in the corresponding database. A data model supports *evolution* constraints if it is able to keep track of how the domain elements evolve along time. In particular, *status classes* describe how elements of classes change their status from being a potential member till they cease forever to be member of the class; *transitions* deal with the fact that an object may migrate from one class to another one; while *generation constraints* describe processes that are responsible for the creation/disappearance of objects from classes.

The formalisation is based on a model-theoretic semantics that captures the meaning of both timestamping and evolution constraints. The semantics is obtained as a temporal extension of the model-theoretic semantics associated to

© Springer-Verlag Berlin Heidelberg 2009

conceptual models [14, 18]. The advantage of associating a set-theoretic semantics to a language is not only to clarify the meaning of the language constructors but also to give a semantic definition to relevant modelling notions. In particular, we are able to give a rigorous definition to the notions of: *schema satisfiability* – when a schema admits a non empty interpretation which guarantees that the constraints expressed by the schema are not contradictory; *class* and *relationships satisfiability* – when a class or a relation admits at least an interpretation in which it is not empty; *logical implication* when a new (temporal) constraint is necessarily true in a schema even if not explicitly mentioned; and finally the special case of logical implication involving *subsumption* between classes (resp. relationships) – when the interpretation of a class (resp. relationship) is necessarily a subset of the interpretation of another class (resp. relationship).

Building on the provided model-theoretic semantics we provide a correspondence between temporal conceptual models and logical theories expressed in a fragment of first order temporal logic, namely as a Description Logics (DLs) theory. DLs allow for the logical reconstruction and the extension of conceptual models (see [9, 14, 19]). The advantage of using a DL to formalise a conceptual data model lies basically on the fact that complete logical reasoning can be employed using an underlying DL inference engine to verify a conceptual specification, to infer implicit facts and stricter constraints, and to manifest any inconsistencies during the conceptual design phase. In addition, given the high complexity of the modelling task when complex data are involved, there is the demand for more sophisticated and expressive languages than for normal databases. Again, DL research is very active in providing more expressive languages for conceptual modelling (see [13, 14, 17, 18, 18, 19, 24, 31]).

In this context, we consider the temporal description logic $\mathcal{DLR}_{\mathcal{US}}$ [5], a combination of the expressive and decidable description logic $\mathcal{DLR}$ [17] (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 classes and relations. We use $\mathcal{DLR}_{\mathcal{US}}$ both to capture the temporal modelling constructors in a succinct way, and to use reasoning techniques to check satisfiability, subsumption and logical implication. The mapping towards DLs presented in this chapter builds on top of a mapping which has been proved correct in [3, 4] while complexity results and algorithmic techniques can be found in [1, 5, 11]. Even if full $\mathcal{DLR}_{\mathcal{US}}$ is undecidable we address interesting modelling scenarios where subsets of the full $\mathcal{DLR}_{\mathcal{US}}$ logic is needed and where reasoning becomes a decidable problem.

The chapter is organised as follows. Section 2 describes the temporal constructs that will be the subject of the formalisation. Section 3 shows the modelling requirements that lead us to elaborate the rigorous definition of the framework presented here. Section 4 introduces the model-theoretic semantics and the notions of satisfiability, subsumption and logical implication for temporal conceptual models. The two Sections 5, 6 are the core sections where we describe how timestamping and evolution constraints can be formalised. After presenting the $\mathcal{DLR}_{\mathcal{US}}$ logic in Section 7 we proceed with a $\mathcal{DLR}_{\mathcal{US}}$ encoding of the

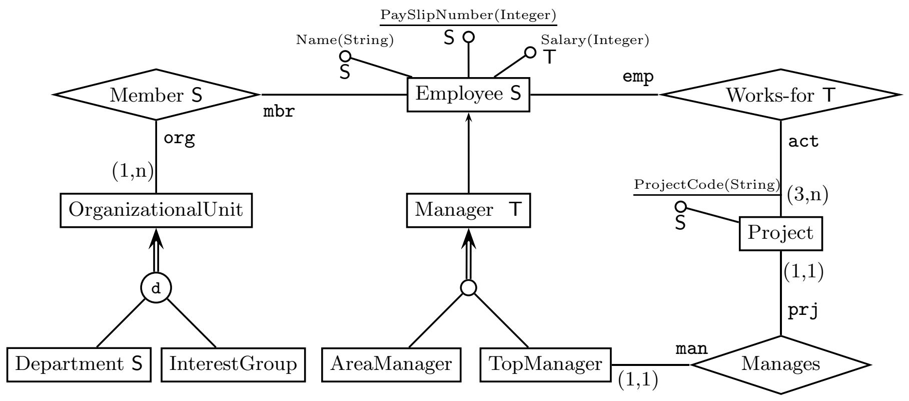

Fig. 1. The Company example

various temporal constructs in Section 8. Section 9 investigates the complexity of reasoning over temporal conceptual models and presents various scenarios where sound, complete and terminating procedures can be used. In Section 10 we state our final remarks.

2 Temporal Modelling Constructors

Temporal constructs are usually added to the classical constructs to capture the temporal behaviour of the different components of a conceptual schema. In this chapter we distinguish them in two generic classes: *Timestamping* and *Evolution* constructs.

Timestamping. It is concerned with the discrimination at the schema level between those elements of the model that change over time and others that are time invariant. Timestamping applies to classes, relationships and attributes. Data models should allow for both temporal and atemporal modelling constructors. Timestamping for attributes allows keeping how an attribute of a given object changes over time. For example (see Figure 1), the salary of an employee *emp-123* has value "*2.5K $*" for the period from 01/2004 to 12/2005, then "*3.0K $*" from 01/2006 to 12/2007, then "*3.2K $*" from 01/2008 to 12/2009.

Similarly, temporal periods can characterise an object or relationship instance as a whole rather than through its attributes. Membership in a class (relationship) can be characterised as limited in time or, vice versa, global—possibly modelling legacy classes (relationships). For example, the company schema (Figure 1) models the membership of objects in the `Employee` class as time-invariant while objects in the `Manager` class have a limited lifespan as member of that class. Timestamping is the basis for associating the notion of *lifecycle* [37] to the object/relationship instances as members of a given class/relationship (more details are given in Section 6.1). Section 5 shows how evolution constraints can be formalised.

Evolution Constraints. They control the mechanism that rules dynamic aspects, i.e., what are the permissible transitions from one state of the database to the next one [6, 7, 22, 38]. When applied to classes we talk about *Object Migration*, i.e., the evolution of an object from being member of a class to being member of another class [29]. For example, an object in the `Employee` class may migrate to become an object of the `Manager` class or an object of the `AreaManager` class can evolve into a `TopManager` class. When object migration combines with timestamping we talk about *Status Classes*. In this case we specify constraints on the membership of an object in a class by splitting it into periods according to a given classification criterion. For example, existence of a manager object in the `Manager` class can include periods where the object is an active member of the class (e.g., the manager is currently on payroll), periods where its membership is suspended (e.g., the manager is on temporary leave), and a period where its membership is disabled (e.g., the manager has left the company) [22]. The notion of status for classes allows also for a fine grained notion of lifecycle which can now depend on the membership to a particular status of a class.

Evolution-related knowledge may be conveyed also through relationships. *Generation relationships* [28] between objects of class `A` and objects of class `B` (possibly equal to `A`) describe the fact that objects in `B` are generated by objects in `A`. For example, in a company database, the splitting of a department translates into the fact that the original department generates two (or more) new departments. Clearly, if `A` and `B` are temporal classes, a generation relationship with source `A` and target `B` entails that the lifecycle of a `B` object cannot start before the lifecycle of the related `A` object. This particular temporal framework, where related objects do not coexist in time, is a form of, so called, *across-time relationships* [7, 38]. Section 6 shows how timestamping can be formalised.

In the conceptual modelling literature, different notion of 'time' have been considered. Notably, the most relevant distinction is between the so called *valid time*—which is the time when a property holds, i.e., it is true in the representation of the world—and *transaction time*—which records the history of database states rather than the world history, i.e., it is the time when a fact is *current* in the database and can be retrieved. In the following, we will consider both timestamping and evolution constructs as ranging over the valid time dimension.

3 Modelling Requirements

This Section briefly illustrates the requirements that are frequently advocated in the literature on temporal data models when dealing with temporal constraints [34, 38].

- Orthogonality. Temporal constructors should be specified separately and independently for classes, relationships, and attributes. Depending on application requirements, the temporal support must be decided by the designer.
- Upward Compatibility. This term denotes the capability of preserving the non-temporal semantics of conventional (legacy) conceptual schemas when embedded into temporal schemas.

– Snapshot Reducibility. Snapshots of the database described by a temporal schema are the same as the database described by the same schema, where all temporal constructors are eliminated and the schema is interpreted atemporally. Indeed, this property specifies that we should be able to fully rebuild a temporal database by starting from the single snapshots.

These requirements are not so obvious when dealing with evolving objects. The formalisation carried out in this chapter provides a data model able to respect these requirements also in presence of evolving objects. In particular, orthogonality affects mainly timestamping [37] and our formalisation satisfies this principle by introducing temporal marks that could be used to specify the temporal behaviour of classes, relationships, and attributes in an independent way (see Section 5). Upward compatibility and snapshot reducibility [34] are strictly related. Considered together, they allow to preserve the meaning of atemporal constructors. In particular, the meaning of classical constructors must be preserved in such a way that a designer could either use them to model classical databases, or when used in a genuine temporal setting their meaning must be preserved at each instant of time. We enforce upward compatibility by using global timestamps over legacy constructors (see Section 5). Snapshot reducibility is hard to preserve when dealing with generation relationships where involved object may not coexist. We enforce snapshot reducibility by a particular treatment of relationship typing (see Section 6.3).

4 A Formalisation of Temporal Data Models

To give a formal foundation to temporal conceptual models we briefly describe here how to associate a textual syntax to a generic EER/UML modelling language. Having a textual syntax at hand will facilitate the association of a model-theoretic semantics. In the next sections we will take advantage of such a model-theoretic temporal semantics to formally describe the temporal constructs we are interested in.

We consider a temporal conceptual model over a finite alphabet, $\mathcal{L}$, 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). We consider n-ary relationships where roles from the $\mathcal{U}$ alphabet are used to distinguish the different components of a relationship, i.e., an n-ary relationship, R, connecting the (not necessarily distinct) classes $C_1, \ldots, C_n$, is defined as $R = \langle U_1 : C_1, \ldots, U_n : C_n \rangle$. Standard EER/UML constructs can also be textually defined, like *Attributes* for both classes and relationships (we use the notation $\textsc{att}(C) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$ to denote all the attributes of a class C, and similarly for attributes of relationships); *Participation Constraints* denoting the cardinality in the participation of a class into a relationship; *Isa* for both classes and relationships (denoted as $C_1 \textsc{isa} C_2$ or $R_1 \textsc{isa} R_2$, respectively); *Disjointness* and *Covering* constraints over a class hierarchy. For a complete set of EER/UML constructs and their textual definition we refer to [3, 4, 19].

In Figure 1 we show our running example of an EER schema for a company database where classes and relationships are denoted by boxes and diamonds, respectively; directed arrows stand for isa; double arrows denote a covering constraint; a circled '`d`' denotes a disjoint hierarchy; participation constraints are indicated with numbers in round brackets; timestamps are denoted with S (snapshot) and T (temporary).

The model-theoretic semantics that gives a foundation to temporal modelling languages adopts the so called *snapshot*[1] representation of abstract temporal databases and temporal conceptual models [20]. Following the snapshot paradigm, relations of a temporal database are interpreted by a mapping function depending on a specific point in time. 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, a standard relational database can be regarded as the result of mapping a temporal database from a specific time point in $\mathcal{T}$ to an atemporal database, with the assumption that the interpretation of both constants and the domain are invariant over time.

Definition 1 (Temporal Schemas Semantics). *Let Σ be a temporal 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 of abstract objects 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 basic 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 $\mathcal{U}$-labeled tuples over $\Delta^{\mathcal{B}}$—i.e., let $R = \langle U_1 : C_1, \ldots, U_n : C_n \rangle$ be an n-ary relationship connecting the classes $C_1, \ldots, C_n$, then, $\forall t \in \mathcal{T}.\forall r \in R^{\mathcal{B}(t)} \rightarrow (r = \langle U_1 : o_1, \ldots, U_n : o_n \rangle \wedge \forall i \in \{1, \ldots, n\}.o_i \in C_i^{\mathcal{B}(t)})$. We adopt the convention: $\langle U_1 : o_1, \ldots, U_n : o_n \rangle \equiv \langle o_1, \ldots, o_n \rangle$, when $\mathcal{U}$-labels are clear from the context.*
- *Every attribute A to a set $A^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}}_D$, such that, for each $C \in \mathcal{C}$, if $\text{ATT}(C) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$, then, $\forall t \in \mathcal{T}.\forall o \in C^{\mathcal{B}(t)} \rightarrow (\forall i \in \{1, \ldots, h\}, \forall a_i.\langle o, a_i \rangle \in A_i^{\mathcal{B}(t)} \rightarrow a_i \in D_i^{\mathcal{B}(t)})$.*

$\mathcal{B}$ is said a legal temporal database state *if it satisfies all of the constraints expressed in the schema*[2].

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

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

[1] The snapshot model represents the same class of temporal databases as the so called *timestamp* model [33, 34] which adds a temporal attribute to each relation [20].

[2] We don't report here the semantics for temporal constraints since they will be discussed in details in the next Sections. As for the semantics of participation constraints, isa, disjointness and covering constraints we refer to [4].

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

5 Timestamping

A temporal model supports *timestamping* if it is able to distinguish between *snapshot* constructors—i.e., constructors with a global lifespan associated to each of their instances—*temporary* constructors—i.e., each of their instances has a limited lifespan—or *mixed* constructors—i.e., their instances can have either a global or a temporary existence. In the following, a class, relationship or attribute is called temporal if it is either temporary or mixed. The two temporal marks, S (snapshot) and T (temporary), introduced at the conceptual level (see Figure 1), together with unmarked constructors capture the temporal distinction between snapshot, temporary and mixed constructors. Notice that, the temporal behaviour of an attribute can be either *globally* forced, or *locally* defined when associated to single classes. Since the local constraint is more general we assume that attributes are locally temporally constrained. At the end of this section we also introduce two notions strictly related to timestamping: that one of a (temporal) key, and a variant of participation constraints called *lifespan participation constraints*. We now proceed with the semantics of timestamping that can be defined as follows (not quantified variables are assumed to be universally quantified):

$o \in C^{\mathcal{B}(t)} \to \forall t' \in \mathcal{T}. o \in C^{\mathcal{B}(t')}$	X`Snapshot Class`
$o \in C^{\mathcal{B}(t)} \to \exists t' \neq t. o \notin C^{\mathcal{B}(t')}$	`Temporary Class`
$r \in R^{\mathcal{B}(t)} \to \forall t' \in \mathcal{T}. r \in R^{\mathcal{B}(t')}$	`Snapshot Relationship`
$r \in R^{\mathcal{B}(t)} \to \exists t' \neq t. r \notin R^{\mathcal{B}(t')}$	`Temporary Relationship`
$(o \in C^{\mathcal{B}(t)} \wedge \langle o, a_i \rangle \in A_i^{\mathcal{B}(t)}) \to \forall t' \in \mathcal{T}. \langle o, a_i \rangle \in A_i^{\mathcal{B}(t')}$	`Snapshot Attribute`
$(o \in C^{\mathcal{B}(t)} \wedge \langle o, a_i \rangle \in A_i^{\mathcal{B}(t)}) \to \exists t' \neq t. \langle o, a_i \rangle \notin A_i^{\mathcal{B}(t')}$	`Temporary Attribute`

The following "classical" desirable features found in the literature of temporal conceptual modelling come as almost trivial logical implications form the above formalisation.

Proposition 1. (Timestamps: Logical Implications [4]) *In every temporal schema supporting timestamping, the following temporal properties hold:*

1. *Subclass of temporary classes are also temporary (similarly for relationships).*
2. *If exactly one of a whole set of snapshot subclasses partitioning*[3] *a snapshot superclass is temporary, then, the whole set of classes is unsatisfiable.*
3. *Participants of snapshot relationships are either snapshot or unmarked classes.*
4. *Participants of snapshot relationships are snapshot when they participate at least once in the relationship.*
5. *A relationship is temporary if one of the participating classes is temporary.*

On the other hand, nothing can be said about subclasses of snapshot or unmarked classes and classes participating to temporary or unmarked relationships: they can be snapshot, temporary, or unmarked classes. Since the domain of an attribute is not restricted to the classes they are attached to, the temporal behaviour of a class is independent of that of its attributes.

Example 1. Considering our running example of Figure 1, the following logical implications hold:

– Because `Manager` is a temporary class, then both `AreaManager` and `TopManager` are temporary classes; constraining either `AreaManager` or `TopManager` as snapshot classes would lead to a contradiction. On the other hand, even if `Employee` is a snapshot class, it is consistent to have `Manager`—a temporary class—as a subclass of `Employee`.
– Because `OrganizationalUnit` participates at least once in a snapshot relationship, then, it must be a snapshot class.
– Since `InterestGroup` participates in a partition of snapshot classes it must be a snapshot class, too.
– The fact that `Manages` must be a temporary relationship follows logically from our theory because the temporary class `TopManager` participates in the relationship.
– Since the temporal behaviour of classes is independent from that one of its attributes, the fact the `Salary` is a temporary attribute of the snapshot class `Employee` is admitted.

Key Constraints. As a byproduct of attribute timestamping we can define single-attribute keys (visualised in a schema as an underlined attribute, e.g., `PaySlipNumber` is a key for the class `Employee`) as a mandatory and single-valued snapshot attribute that uniquely identifies objects of the class. Assuming that A_{key} is a key for the class C, then the the following formalisation holds:

$$
\begin{array}{ll}
(o \in C^{\mathcal{B}(t)} \wedge \langle o, a_{key} \rangle \in A_{key}^{\mathcal{B}(t)}) \rightarrow \forall t' \in \mathcal{T}.\langle o, a_{key} \rangle \in A_{key}^{\mathcal{B}(t')} & \texttt{Snapshot Attribute} \\
o \in C^{\mathcal{B}(t)} \rightarrow \exists^{=1} a_{key} \in \Delta_D^{\mathcal{B}}.\langle o, a_{key} \rangle \in A_{key}^{\mathcal{B}(t)} & \texttt{Mandatory \&} \\
 & \quad\texttt{Single-Valued} \\
a_{key} \in \Delta_D^{\mathcal{B}} \rightarrow \exists^{\leq 1} o \in C^{\mathcal{B}(t)}.\langle o, a_{key} \rangle \in A_{key}^{\mathcal{B}(t)} & \texttt{Uniqueness}
\end{array}
$$

[3] The partition must be a disjoint covering.

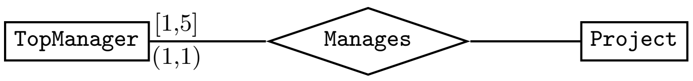

Fig. 2. Lifespan and "classical" participation constraints

Lifespan Participation Constraints. While classical participations constraints are evaluated at each point in time *lifespan participation constraints* (represented in a schema by a pair of values in square brackets) [26, 37, 39] are evaluated during the entire existence of the object. Notice that, since the set of instances of snapshot relationships does not change in time, there is no difference between "classical" and lifespan participation constraints for snapshot relationships. For example, if we want to state that a top manager should manage at most five different projects in his entire existence while still being constrained in managing exactly one project at a time, we can use a combination of the two participation constraints (see Figure 2). The model-theoretic semantics for lifespan participation is the following:

$o \in C^{\mathcal{B}(t)} \rightarrow k \leq \# \bigcup_{t' \in \mathcal{T}} \{ r \in R^{\mathcal{B}(t')} \mid r[U] = o \} \leq m$ `Lifespan Participation Constraint`

6 Evolution Constraints

Evolution constraints contribute in modelling the temporal dynamic of an object. In this section we propose a formalisation of the basic temporal concepts that are at the root of advanced conceptual temporal models: *status classes*, distinguished in scheduled, active, suspended and disabled; *transitions* of objects between different classes; *generation* relationships asserting evolution constraints on objects linked by temporal relationships. In this section we aim at presenting a formal characterisation of the temporal conceptual modelling constructors capturing the evolution of objects.

6.1 Status Classes

The *Status* [7, 22, 37] is a conceptual notion associated to temporal classes to rule the lifecycle of their objects. It records the evolving state of membership of each object in the class. Following [37], status modelling includes up to four different statuses, and the allowed transitions between them:

– Scheduled. An object is scheduled if the planning of its existence within the class has to be recorded while its membership in the class will only become effective (active) some time later. For example, if a new project is approved but will not start until a later date the given project can be created as a new object in the `Project` class, with status scheduled for the valid time interval starting at the date of the approval decision and ending at the expected launching date. Each scheduled object will eventually become an active object. Supporting a scheduled status avoids the introduction of a new time type, the decision time [22].

– Active. The status of an object is active if the object is a full member of the class (and therefore conforms to its type). For example, a currently ongoing project is an active member, at time now, of the `Project` class.
– Suspended. This status qualifies objects that exist as members of the class, but are to be seen as temporarily inactive members of the class [22]. An employee taking a temporary leave of absence is an example of what can be considered as a suspended employee. Only active objects can be suspended. A suspended object was in the past an active one.
– Disabled. This status is used to specify that the object's membership in the class has expired. While logically deleted, disabled objects are kept for some specific application purposes, e.g., statistical analyses. A disabled object was in the past an active member of the class (an object cannot be created in the disabled status). It can never again become a non-disabled member of that class (e.g., an expired project cannot be reactivated).

Let C be a temporal (i.e., temporary or mixed) class. We capture status transition of membership in C by associating to C the following *status classes*: `Scheduled-C`, `Suspended-C`, `Disabled-C`. In particular, status classes are constrained by the hierarchy of Figure 3 (where C may also be mixed) that classifies C instances according to their actual status. To preserve upward compatibility we do not explicitly introduce an active class, but assume by default that the name of the class itself denotes the set of active objects, i.e., `Active-C` $\equiv$ `C`. Note that, since membership of objects into snapshot classes is global, i.e., objects are always active, the notion of status classes does not apply to snapshot classes.

To capture the intended meaning of status classes, we define ad-hoc constraints and then show that such constraints capture the evolving behaviour of status classes as described in the literature [22, 37]. First of all, disjointness and isa constraints between statuses of a class C can be described as illustrated in

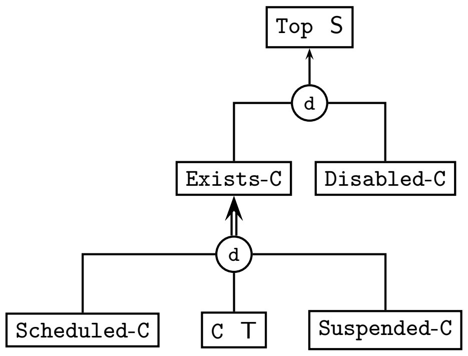

Fig. 3. Status classes

Figure 3, where `Top` is supposed to be a snapshot class which represents the universe of abstract objects (i.e., $\texttt{Top}^{\mathcal{B}(t)} \equiv \Delta^{\mathcal{B}}$). Other than hierarchical constraints, the intended semantics of status classes induces the following rules that are related to their temporal behaviour:

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

As a consequence of the above formalisation the following set of new rules can be derived.

Proposition 2 (Status Classes: Logical Implications [7]). *Given a temporal schema supporting status classes, then, the following logical implications hold:*

1. *Disabled classes will never become active anymore.*
2. *The scheduled status persists until the class become active.*
3. *A scheduled class cannot evolve directly into a disabled status.*

Temporal applications often use concepts that are derived from the notion of object statuses, e.g., the *lifespan* of a temporal object or its *birth* and *death* instants. Hereinafter we provide formal definitions for these concepts.

Lifespan and related notions. The lifespan of an object w.r.t. a class describes the temporal instants where the object can be considered a member of the class. With the introduction of status classes we can distinguish between the following notions: $\textsc{ExistenceSpan}_C$, $\textsc{LifeSpan}_C$, $\textsc{ActiveSpan}_C$, $\textsc{Begin}_C$, $\textsc{Birth}_C$ and $\textsc{Death}_C$. They are functions which depend on the object membership to the status classes associated to a temporal class C.

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

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

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

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

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

$$\textsc{ActiveSpan}_C(o) = \{t \in \mathcal{T} \mid o \in \mathtt{C}^{\mathcal{B}(t)}\}$$

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

$$\begin{aligned}
&\textsc{Begin}_C(o) = \mathtt{min}(\textsc{ExistenceSpan}_C(o))\\
&\textsc{Birth}_C(o) = \mathtt{min}(\textsc{ActiveSpan}_C(o)) \equiv \mathtt{min}(\textsc{LifeSpan}_C(o))\\
&\textsc{Death}_C(o) = \mathtt{max}(\textsc{LifeSpan}_C(o))
\end{aligned}$$

We could still speak of existencespan, lifespan or activespan for snapshot classes, but in this case they all collapse to the full time line, $\mathcal{T}$. Furthermore, $\textsc{Begin}_C(o)$ = $\textsc{Birth}_C(o) = -\infty$, and $\textsc{Death}_C(o) = +\infty$ either when C is a snapshot class or in cases of instances existing since ever and/or living forever.

6.2 Transition

Transition constraints [29, 37] 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 to become instances of the target class, and *dynamic extension*, when the creation of the target instance does not force the removal of the source instance. For example, considering the company schema (Figure 1), if we want to record data about the promotion of area managers into top managers we can specify a dynamic evolution from the class `AreaManager` to the class `TopManager`. We can also record the fact that a mere employee becomes a manager by defining a dynamic extension from the class `Employee` to the class `Manager` (see Figure 4). Regarding 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.

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

In the following, we present a formalisation that satisfies the above requirements. We represent transitions by introducing a new class denoted by either $\textsc{dex}_{C_1,C_2}$ or $\textsc{dev}_{C_1,C_2}$ for dynamic extension and evolution, respectively. More

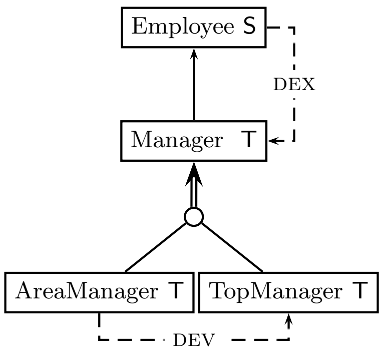

Fig. 4. Transitions employee-to-manager and area-to-top manager

formally, in case of a *dynamic extension* between classes C_1, C_2 the following semantic equation holds:

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

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

$$o \in \text{DEV}^{\mathcal{B}(t)}_{C_1,C_2} \rightarrow (o \in (\texttt{Suspended-C}_1{}^{\mathcal{B}(t)} \cup \texttt{C}_1{}^{\mathcal{B}(t)}) \wedge o \notin \texttt{C}_2{}^{\mathcal{B}(t)} \wedge \\ o \in C_2^{\mathcal{B}(t+1)} \wedge o \notin C_1^{\mathcal{B}(t+1)})$$

Finally, we formalise the case where the source (C_1) and/or the target (C_2) totally participate in a dynamic extension/evolution (at schema level we add mandatory cardinality constraints on DEX/DEV links):

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

An interesting set of consequences of the above proposed modelling of dynamic transitions are shown in the following proposition.

Proposition 3 (Transition: Logical Implications [7]). *Given a schema supporting transitions for objects, then the following logical implications hold:*

1. *The classes* DEX_{C_1,C_2} *and* DEV_{C_1,C_2} *are temporary classes; actually, they hold at single time points.*
2. *Objects in the classes* DEX_{C_1,C_2} *and* DEV_{C_1,C_2} *cannot be disabled as* C_2.
3. *The target class* C_2 *cannot be snapshot (it becomes temporary in case of both* `Source Total Transition` *and* `Target Total Evolution` *constraints).*
4. *As a consequence of dynamic evolution, the source class,* C_1*, cannot be snapshot (and it becomes temporary in case of* `Source Total Evolution` *constraints).*

5. *Dynamic evolution cannot be specified between a class and one of its sub-classes.*
6. *Dynamic extension between disjoint classes logically implies Dynamic evolution.*

6.3 Generation Relationships

Generation relationships [7, 28, 36, 37] represent processes that lead to the emergence of *new objects* starting from a set of existing objects. In their most generic form, a generation relationship can have a collection of objects as source and a collection of objects as target. For example (see Figure 5), assuming an organisation remodels its departments, it may be that an existing department is split into two new departments, while two existing departments are merged into a single new department and three existing departments are reorganised as two new departments. Cardinality constraints can be added to specify the cardinality of sets involved in a generation. For example, if we want to record the fact that a group of managers proposes at most one new project at a time a generation relationship from `Manager` to `Project` can be defined with the cardinality "*at most one*" on the manager side.

Depending whether the source objects are preserved (as member of the source class) or disabled by the generation process, we distinguish between *production* and *transformation* relationships, respectively. Managers creating projects is an example of the former, while departmental reorganisation is an example of the latter. At the conceptual level we introduce two marks for generation relationships: GP for production and GT for transformation relationships, and an arrow pointing to the target class (see Figure 5).

We model generation as binary relationships connecting a source class to a target one, with the target being in its scheduled status: $\textsc{rel}(R) = \langle \texttt{source} : C_1, \texttt{target} : \texttt{Scheduled-C}_2\rangle$. The semantics of *production relationships*, R, is described by the following equation:

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

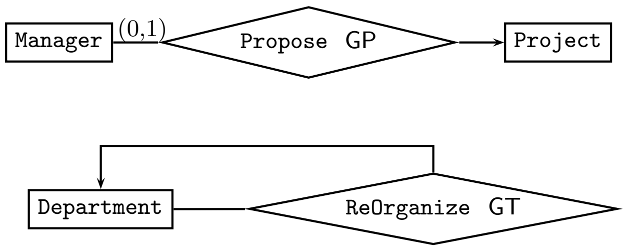

Fig. 5. Production and transformation generation relationships

Thus, objects active in the source class produce objects active in the target class at the next point in time. A production relationship is a case of across-time relationships [7]—i.e., relationships connecting objects which are active in the connected classes at different points in time—where the use of status classes allows 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 case of *transformation* is captured by the following semantic equation:

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

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. As for production relationships, transformations are special cases of across-time relationships.

Proposition 4 (Generation: Logical Implications [7]). *The following logical implications hold as a consequence of the generation semantics:*

1. *A generation relationship, R, is temporary; actually, it is instantaneous.*
2. *The target class, C_2, cannot be snapshot (C_2 must be temporary if it participates at least once).*
3. *Objects participating as target cannot be disabled.*
4. *If R is a transformation relationship, then, C_1 cannot be snapshot (C_1 must be temporary if it participates at least once).*

Note that the `Department` class that is both the source and target of a transformation relationship (Figure 5) can no longer be snapshot (as it was in Example 1) and must be changed to temporary. Furthermore, as a consequence of this new timestamp for the `Department` class, `InterestGroup` is now a genuine mixed class.

Starting with the next section we provide a correspondence between temporal conceptual schemas and theories expressed in temporal description logics.

7 The Temporal Description Logic

As the description logic $\mathcal{DLR}$ has been used to reason over conceptual models [14, 17, 18] in this chapter we use a temporal extension of $\mathcal{DLR}$ to capture temporal conceptual models. The temporal description logic $\mathcal{DLR}_{\mathcal{US}}$ [5, 25] combines the propositional temporal logic with *Since* and *Until* and the (non-temporal) description logic $\mathcal{DLR}$ [13, 17]. $\mathcal{DLR}_{\mathcal{US}}$ can be regarded as a rather expressive fragment of the first-order temporal logic $L^{\{\textbf{since, until}\}}$ (cf. [20, 30]).

The basic syntactical types of $\mathcal{DLR}_{\mathcal{US}}$ are *concepts* (unary predicates) and n-ary *relations* of arity ≥ 2. Starting from a set of *atomic concepts* (denoted

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

$$
\begin{aligned}
\top^{\mathcal{I}(t)} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}(t)} &= \emptyset \\
CN^{\mathcal{I}(t)} &\subseteq \top^{\mathcal{I}(t)} \\
(\neg C)^{\mathcal{I}(t)} &= \top^{\mathcal{I}(t)} \setminus C^{\mathcal{I}(t)} \\
(C_1 \sqcap C_2)^{\mathcal{I}(t)} &= C_1^{\mathcal{I}(t)} \cap C_2^{\mathcal{I}(t)} \\
(\exists^{\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 \\
&\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. 6. Syntax and semantics of $\mathcal{DLR}_{\mathcal{US}}$

by CN), a set of *atomic relations* (denoted by RN), and a set of *role symbols* (denoted by U) we hereinafter define inductively (complex) concepts and relation expressions as is shown in the upper part of Figure 6, where the binary constructors ($\sqcap, \sqcup, \mathcal{U}, \mathcal{S}$) are applied to relations of the same arity, i, j, k, n are natural numbers, $i \leq n$, and j does not exceed the arity of R.

The non-temporal fragment of $\mathcal{DLR}_{\mathcal{US}}$ coincides with $\mathcal{DLR}$. For both concept and relation expressions all the Boolean constructors are available. The selection expression $U_i/n : C$ denotes an n-ary relation whose argument named U_i ($i \leq n$) is of type C; if it is clear from the context, we omit n and write ($U_i : C$). The projection expression $\exists^{\lessgtr k}[U_j]R$ is a generalisation with cardinalities of the projection operator over the argument named U_j of the relation R; the plain classical projection is $\exists^{\geq 1}[U_j]R$ (we will use $\exists[U_j]R$ as a shortcut). It is also possible to use the pure argument position version of the language 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 $t \in \mathcal{T}$ (in the following the notation $t \in \mathcal{T}$ is used as a shortcut for $t \in \mathcal{T}_p$), every concepts 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 concept and relation expressions is defined in the lower part of Figure 6, where $(u, v) = \{w \in \mathcal{T} \mid u < w < v\}$. For concepts, the temporal operators $\diamond^+$ (some time in the future), $\oplus$ (at the next moment), and their past counterparts can be defined via $\mathcal{U}$ and $\mathcal{S}$: $\diamond^+ C \equiv \top \mathcal{U} C$, $\oplus\, C \equiv \bot \mathcal{U} C$, etc. 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 concepts and relations. The operators $\diamond^*$ (at some moment) and its dual $\Box^*$ (at all moments) can be defined for both concepts 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*, $\mathcal{K}$, is a finite set 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. The notation $C_1 \doteq C_2$ ($R_1 \doteq R_2$) is a shortcut for $C_1 \sqsubseteq C_2$, $C_2 \sqsubseteq C_1$ ($R_1 \sqsubseteq R_2$, $R_2 \sqsubseteq R_1$). 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, $\mathcal{K}$, is *satisfiable* if there is an interpretation that satisfies all the axioms in $\mathcal{K}$ (in symbols, $\mathcal{I} \models \mathcal{K}$). A knowledge base, $\mathcal{K}$, *logically implies* an axiom, $C_1 \sqsubseteq C_2$ ($R_1 \sqsubseteq R_2$), and write $\mathcal{K} \models C_1 \sqsubseteq C_2$ ($\Sigma \models R_1 \sqsubseteq R_2$), if we have $\mathcal{I} \models C_1 \sqsubseteq C_2$ ($\mathcal{I} \models R_1 \sqsubseteq R_2$) whenever $\mathcal{I} \models \mathcal{K}$. In this latter case, the concept C_1 (relation R_1) is said to be *subsumed* by the concepts C_2 (relation R_2) in the knowledge base $\mathcal{K}$. A concepts C is satisfiable, given a knowledge base $\mathcal{K}$, if there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}(t)} \neq \emptyset$ for some $t \in \mathcal{T}$, i.e., $\mathcal{K} \not\models C \sqsubseteq \bot$. A relation R is satisfiable, given a knowledge base $\mathcal{K}$, if there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $R^{\mathcal{I}(t)} \neq \emptyset$ for some $t \in \mathcal{T}$, i.e., $\mathcal{K} \not\models R \sqsubseteq \bot$. Finally, knowledge base satisfiability, concepts subsumption and relation satisfiability can be reduced to concepts satisfiability in the following way: $\mathcal{K} \not\models \top \sqsubseteq \bot$, $\mathcal{K} \models C_1 \sqcap \neg C_2 \sqsubseteq \bot$, $\mathcal{K} \not\models \exists^{\geq 1}[U_j]R \sqsubseteq \bot$ for some $j \leq n$ where n is the arity of R, respectively.

While $\mathcal{DLR}$ knowledge bases are fully able to capture atemporal EER/UML schemas [14, 17, 18]—i.e., given an EER schema there is an equi-satisfiable $\mathcal{DLR}$ knowledge base—in the following sections we use $\mathcal{DLR}_{\mathcal{US}}$ knowledge bases to capture temporal EER/UML schemas with both timestamping and evolution constraints.

8 Encoding Temporal Schemas in Description Logics

We start by briefly summarising how knowledge bases in the description logic $\mathcal{DLR}$ can capture conceptual schemas. The correspondence we report here is based on a mapping introduced by [14, 18, 19] for atemporal EER models.

Informally, the encoding works as follows. Class and relationship symbols in a conceptual diagram are mapped into $\mathcal{DLR}$ concept names and relation names (with the same arity of the original relationship), respectively. Domain symbols are mapped into additional concept names, pairwise disjoint. Attributes of classes are mapped to binary relation names in $\mathcal{DLR}$ with number restrictions stating the cardinality of the attribute to distinguish between single- and multi-valued attributes. *Isa* links between classes or between relationships are mapped using $\mathcal{DLR}$ axioms. Generalised hierarchies with disjointness and covering constraints can be captured using Boolean connectives. Cardinality constraints are mapped using number restriction in $\mathcal{DLR}$.

Let us consider the class diagram depicted in Figure 1 and representing (a portion of) a company database. According to the diagram, all managers are employees and are partitioned into area managers and top managers. This information can be represented by means of the following $\mathcal{DLR}$ axioms:

$$\begin{aligned}
\texttt{Manager} &\sqsubseteq \texttt{Employee}\\
\texttt{AreaManager} &\sqsubseteq \texttt{Manager}\\
\texttt{TopManager} &\sqsubseteq \texttt{Manager}\\
\texttt{AreaManager} &\sqsubseteq \neg\texttt{TopManager}\\
\texttt{Manager} &\sqsubseteq \texttt{AreaManager} \sqcup \texttt{TopManager}
\end{aligned}$$

Binary relation names of the form $A \sqsubseteq \texttt{From}:\top \sqcap \texttt{To}:\top$ capture attributes. Each employee has three functional attributes (by default, we assume that attributes are single-valued and mandatory), Salary, PaySlipNumber, with integer values, and Name, with string values; here we show only the first:

$$\texttt{Employee} \sqsubseteq \exists^{=1}[\texttt{from}]\texttt{Salary} \sqcap \exists^{=1}[\texttt{from}](\texttt{Salary} \sqcap \texttt{to}/2 : \texttt{Integer})$$

The binary relationship Works-for has Employees as domain, while the range is restricted to Projects:

$$\texttt{Works-for} \sqsubseteq \texttt{emp}/2 : \texttt{Employee} \sqcap \texttt{act}/2 : \texttt{Project}$$

Each top manager manages exactly one project, while a project must involve at least three employees:

$$\begin{aligned}
\texttt{TopManager} &\sqsubseteq \exists^{=1}[\texttt{man}]\texttt{Manages}\\
\texttt{Project} &\sqsubseteq \exists^{\geq 3}[\texttt{act}]\texttt{Works-For}
\end{aligned}$$

Temporal properties expressed in the diagram are mapped using temporal operators in $\mathcal{DLR}_{\mathcal{US}}$. In the following we will show how to extend the above translation in order to capture both timestamping and evolution constraints.

Encoding Timestamping

Timestamping is the ability to distinguish between *snapshot* constructors—i.e., constructors with a global lifespan associated to each of their instances—*temporary* constructors—i.e., each of their instances has a limited lifespan—or *mixed* constructors—i.e., their instances can have either a global or a temporary existence. Timestamps for both classes and relationships are captured by the following $\mathcal{DLR}_{\mathcal{US}}$ axioms (remember that $\Box^*$ is the "at all time" operator while $\Diamond^*$ is the "at some time" operator, see Section 7):

(SnapC)	$C \sqsubseteq \Box^* C$	`Snapshot Class`
(TempC)	$C \sqsubseteq \Diamond^* \neg C$	`Temporary Class`
(SnapT)	$R \sqsubseteq \Box^* R$	`Snapshot Relationship`
(TempR)	$R \sqsubseteq \Diamond^* \neg R$	`Temporary Relationship`

Considering timestamping for attributes we first recall that attributes are captured in $\mathcal{DLR}$ as binary relations. Thus, the following $\mathcal{DLR}_{\mathcal{US}}$ axioms hold[4]:

(SnapA)	$C \sqsubseteq \neg\exists[\mathtt{From}](A \sqcap \Diamond^* \neg A)$	`Snapshot Attribute`
(TempA)	$C \sqsubseteq \neg\exists[\mathtt{From}](\Box^* A)$	`Temporary Attribute`

Key Constraints. We now show the $\mathcal{DLR}_{\mathcal{US}}$ axioms that capture the notion of single-attribute keys (see the case of a pay slip number that uniquely identifies an employee in Figure 1). We need three axioms: the first to specify that a key is a *snapshot* attribute, the second to characterise a key as *mandatory* and *single-valued*, and the last axiom to specify *uniqueness*. Assuming that A_{key} is a key for the class C, then its semantics is captured by the following $\mathcal{DLR}_{\mathcal{US}}$ axioms:

(Key1)	$C \sqsubseteq \neg\exists[\mathtt{From}](A_{key} \sqcap \Diamond^* \neg A_{key})$	`Snapshot Attribute`
(Key2)	$C \sqsubseteq \exists^{=1}[\mathtt{from}]A_{key}$	`Mandatory & Single-Valued`
(Key3)	$\top \sqsubseteq \exists^{\leq 1}[\mathtt{to}](A_{key} \sqcap \mathtt{from} : C)$	`Uniqueness`

Lifespan Participation Constraints. Lifespan participation constraints (see Figure 2) are formalised in $\mathcal{DLR}_{\mathcal{US}}$ using a combination of number restrictions and temporal operators for relations:

(LPC) $C \sqsubseteq \exists^{\geq \mathtt{k}}[\mathtt{U}]\Diamond^* \mathtt{R} \sqcap \exists^{\leq \mathtt{m}}[\mathtt{U}]\Diamond^* \mathtt{R}$ `Lifespan Participation Constraint`

The standard logical implications due to timestamping and showed in Proposition 1 can be rephrased in terms of $\mathcal{DLR}_{\mathcal{US}}$ logical implications.

Proposition 5 (Timestamps: Logical Implications [4])). *In every temporal schema supporting timestamping, the following temporal properties hold:*

1. *Subclass of temporary classes are also temporary (similarly for relationships).* $\{C_1 \sqsubseteq C, C \sqsubseteq \Diamond^* \neg C\} \models C_1 \sqsubseteq \Diamond^* \neg C_1$
2. *If exactly one of a whole set of snapshot subclasses partitioning a snapshot superclass is temporary, then, the whole set of classes is unsatisfiable (we consider a three class partition).* $\{C_0 \doteq C_1 \sqcup C_2, C_1 \sqsubseteq \neg C_2, C_0 \sqsubseteq \Box^* C_0, C_1 \sqsubseteq \Box^* C_1, C_2 \sqsubseteq \Diamond^* \neg C_2\} \models C_i \sqsubseteq \bot, \quad i = 0, 1, 2$

[4] The axioms consider a local temporal behaviour for attributes. To associate a global behaviour to an attribute we consider it as a binary relationship and apply the axioms for timestamping relationships.

3. *Participants of snapshot relationships are either snapshot or unmarked classes.*
   $\{R \sqsubseteq \Box^* R, R \sqsubseteq U_i : C_i, C_i \sqsubseteq \Diamond^* \neg C_i\} \models R \sqsubseteq \bot$
4. *Participants of snapshot relationships are snapshot when they participate at least once in the relationship.*
   $\{R \sqsubseteq \Box^* R, R \sqsubseteq U_i : C_i, C_i \sqsubseteq \exists [U_i] R\} \models C_i \sqsubseteq \Box^* C_i$
5. *A relationship is temporary if one of the participating classes is temporary.*
   $\{R \sqsubseteq U_i : C_i, C_i \sqsubseteq \Diamond^* \neg C_i\} \models R \sqsubseteq \Diamond^* \neg R$

Encoding Status Classes

Status classes record the evolving state of membership of each object in the class. We distinguish four status: scheduled, active, suspended and disabled. $\mathcal{DLR}_{\mathcal{US}}$ axioms are able to fully capture the hierarchical constraints of Figure 3. Moreover, the semantic equations formalising status classes are captured by the following set of $\mathcal{DLR}_{\mathcal{US}}$ 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)	$\texttt{C} \sqsubseteq \Box^+ \neg \texttt{Scheduled-C}$

We denote with Σ_{st} the above set of axioms together with the $\mathcal{DLR}_{\mathcal{US}}$ axioms that capture the hierarchy of Figure 3. We can now rephrase the logical implications involving status classes showed in Proposition 2 as $\mathcal{DLR}_{\mathcal{US}}$ logical implications.

Proposition 6 (Status Classes: Logical Implications [7]). *Given the set of $\mathcal{DLR}_{\mathcal{US}}$ axioms Σ_{st} that capture status classes, the following logical implications hold:*

1. *Disabled will never become active anymore.*
   $\Sigma_{st} \models \texttt{Disabled-C} \sqsubseteq \Box^+ \neg \texttt{C}$
2. *Scheduled persists until active.*
   $\Sigma_{st} \models \texttt{Scheduled-C} \sqsubseteq \texttt{Scheduled-C} \mathcal{U} \texttt{C}$
3. *Scheduled cannot evolve directly to Disabled.*
   $\Sigma_{st} \models \texttt{Scheduled-C} \sqsubseteq \oplus \neg \texttt{Disabled-C}$

Encoding Transition

Transition constraints model the so called object migration. They are distinguished in *dynamic evolution*—when objects cease to be instances of the source class to become instances of the target class—and *dynamic extension*—when the creation of the target instance does not force the removal of the source instance. We represent transitions by introducing a new class denoted by either $\textsc{dex}_{C_1,C_2}$ or $\textsc{dev}_{C_1,C_2}$ for dynamic extension and evolution, respectively. The $\mathcal{DLR}_{\mathcal{US}}$ axioms capturing these temporal constraints are:

(Dex) $\textsc{dex}_{C_1,C_2} \sqsubseteq (\texttt{Suspended-C}_1 \sqcup \texttt{C}_1) \sqcap \neg \texttt{C}_2 \sqcap \oplus\, C_2$
(Dev) $\textsc{dev}_{C_1,C_2} \sqsubseteq (\texttt{Suspended-C}_1 \sqcup \texttt{C}_1) \sqcap \neg \texttt{C}_2 \sqcap \oplus\, (C_2 \sqcap \neg C_1)$

The $\mathcal{DLR}_{\mathcal{US}}$ axioms capturing the cases where the source (C_1) and/or the target (C_2) totally participate in a dynamic extension/evolution are:

(Stt)	$C_1 \sqsubseteq \Diamond^{+}\textsc{dex}_{C_1,C_2}$	`Source Total Transition`
(Ttt)	$C_2 \sqsubseteq \Diamond^{-}\textsc{dex}_{C_1,C_2}$	`Target Total Transition`
(Ste)	$C_1 \sqsubseteq \Diamond^{+}\textsc{dev}_{C_1,C_2}$	`Source Total Evolution`
(Tte)	$C_2 \sqsubseteq \Diamond^{-}\textsc{dev}_{C_1,C_2}$	`Target Total Evolution`

We can now rephrase the logical implications involving transition constraints showed in Proposition 3 as $\mathcal{DLR}_{\mathcal{US}}$ logical implications.

Proposition 7 (Transition: Logical Implications [7]). *Let $\Sigma_{tr} = \{$(Dev), (Dex)$\}$, then the following logical implications hold:*

1. *The classes $\textsc{dex}_{C_1,C_2}$ and $\textsc{dev}_{C_1,C_2}$ are temporary classes; actually, they hold at single time points.*
$\Sigma_{st} \cup \Sigma_{tr} \models \textsc{dex}_{C_1,C_2} \sqsubseteq \oplus\, \neg\textsc{dex}_{C_1,C_2} \sqcap \ominus\, \neg\textsc{dex}_{C_1,C_2}$
$\Sigma_{st} \cup \Sigma_{tr} \models \textsc{dev}_{C_1,C_2} \sqsubseteq \oplus\, \neg\textsc{dev}_{C_1,C_2} \sqcap \ominus\, \neg\textsc{dev}_{C_1,C_2}$
2. *Objects in the classes $\textsc{dex}_{C_1,C_2}$ and $\textsc{dev}_{C_1,C_2}$ cannot be disabled as C_2.*
$\Sigma_{st} \cup \Sigma_{tr} \models \textsc{dex}_{C_1,C_2} \sqsubseteq \neg\texttt{Disabled-C}_2$
$\Sigma_{st} \cup \Sigma_{tr} \models \textsc{dev}_{C_1,C_2} \sqsubseteq \neg\texttt{Disabled-C}_2$
3. *The target class C_2 cannot be snapshot (it becomes temporary in case of both* (Ttt) *and* (Tte) *constraints).*
$\Sigma_{st} \cup \Sigma_{tr} \models \textsc{dex}_{C_1,C_2} \sqsubseteq \Diamond^{*}[C_2 \sqcap (\Diamond^{+}\neg C_2 \sqcup \Diamond^{-}\neg C_2)]$
4. *As a consequence of dynamic evolution, the source class, C_1, cannot be snapshot (and it becomes temporary in case of* (Ste) *constraints).*
$\Sigma_{st} \cup \Sigma_{tr} \models \textsc{dev}_{C_1,C_2} \sqsubseteq \Diamond^{*}[C_1 \sqcap (\Diamond^{+}\neg C_1 \sqcup \Diamond^{-}\neg C_1)]$
5. *Dynamic evolution cannot be specified between a class and one of its subclasses.*
$\Sigma_{st} \cup \Sigma_{tr} \cup \{C_2 \sqsubseteq C_1\} \models \textsc{dev}_{C_1,C_2} \sqsubseteq \bot$
6. *Dynamic extension between disjoint classes logically implies Dynamic evolution.*
$\Sigma_{st} \cup \Sigma_{tr} \cup \{C_1 \sqsubseteq \neg C_2\} \models \textsc{dex}_{C_1,C_2} \sqsubseteq \textsc{dev}_{C_1,C_2}$

Encoding Generation Relationships

Generation relationships lead to the emergence of new objects starting from a set of existing objects. Depending whether the source objects are preserved (as member of the source class) or disabled, we distinguish between *production* and *transformation* relationships, respectively. The $\mathcal{DLR}_{\mathcal{US}}$ axioms capturing the production and transformation semantics are:

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

We can now rephrase the logical implications involving generation relationships showed in Proposition 4 as $\mathcal{DLR}_{\mathcal{US}}$ logical implications.

Proposition 8 (Generation: Logical Implications [7]). *The following logical implications hold as a consequence of the $\mathcal{DLR}_{\mathcal{US}}$ axioms capturing generation relationships:*

1. *A generation relationship, R, is temporary; actually, it is instantaneous.*
   $\Sigma_{st} \cup \{(\textsc{Prod})\} \models R \sqsubseteq \Box^{+}\neg R \sqcap \Box^{-}\neg R$
2. *The target class, C_2, cannot be snapshot (C_2 must be temporary if it participates at least once).*
   $\Sigma_{st} \cup \{(\textsc{Prod})\} \models R \sqsubseteq \texttt{target}\!:\!\Diamond^{*}[C_2 \sqcap (\Diamond^{+}\neg C_2 \sqcup \Diamond^{-}\neg C_2)]$
3. *Objects participating as target cannot be disabled.*
   $\Sigma_{st} \cup \{(\textsc{Prod})\} \models R \sqsubseteq \texttt{target}\!:\!\neg\texttt{Disabled-C}_2$
4. *If R is a transformation relationship, then, C_1 cannot be snapshot (C_1 must be temporary if it participates at least once).*
   $\Sigma_{st} \cup \{(\textsc{Trans})\} \models R \sqsubseteq \texttt{source}\!:\!\Diamond^{*}[C_1 \sqcap (\Diamond^{+}\neg C_1 \sqcup \Diamond^{-}\neg C_1)]$

8.1 Correctness of the Encoding

To prove that reasoning on temporal schemas can be done by reasoning on their $\mathcal{DLR}_{\mathcal{US}}$ translation, we need to prove the correctness of the encoding. That temporal schemas with timestamping and transition constraints can be encoded as description logic theories has been proven correct in [4, 5] by establishing a precise correspondence between legal database states of temporal schemas and models of the corresponding description logic theories. This result can be easily extended to the full set of temporal constraints presented here.

Theorem 1 (Correctness of the encoding). *Let Σ be a temporal schema. Then, Σ admits a legal database state if and only if the corresponding $\mathcal{DLR}_{\mathcal{US}}$ knowledge base encoding the schema has a model.*

This characterisation allows us to support the reasoning on temporal conceptual models, as in Definition 2, by using the reasoning services of $\mathcal{DLR}_{\mathcal{US}}$. On the other hand, since reasoning with $\mathcal{DLR}_{\mathcal{US}}$ theories is undecidable, in the following section we present interesting scenarios where reasoning become decidable together with their respective complexity results.

9 Complexity of Reasoning on Temporal Models

As this chapter shows, the temporal description logic $\mathcal{DLR}_{\mathcal{US}}$ is able to fully capture temporal schemas with both timestamping and evolution constraints. On the other hand, reasoning over $\mathcal{DLR}_{\mathcal{US}}$ knowledge bases, i.e., checking satisfiability, subsumption and logical implications, turns out to be undecidable [5]. The main reason for this is the possibility to couple the evolution of concepts with the possibility to postulate that a binary relation does not vary in time (i.e., global relations). Note that, showing that temporal schemas can be mapped into $\mathcal{DLR}_{\mathcal{US}}$ axioms does not necessarily imply that reasoning over temporal schemas is an undecidable problem. Unfortunately, [1] shows that the undecidable Halting

Problem can be encoded as the problem of class satisfiability w.r.t. a temporal schema with, among the others, the following constructs: disjoint and covering constraints, sub-relationships, timestamping on both classes and relationships, and evolution constraints.

On the other hand, the fragment, $\mathcal{DLR}^{-}_{\mathcal{US}}$, of $\mathcal{DLR}_{\mathcal{US}}$ deprived of the ability to talk about temporal persistence of n-ary relations, for $n \geq 2$, is decidable. Indeed, reasoning in $\mathcal{DLR}^{-}_{\mathcal{US}}$ is an EXPTIME-complete problem [5]. This result gives us an useful scenario where reasoning over temporal schemas becomes decidable. In particular, if we forbid timestamping for relationships (i.e., relationships are just unmarked and interpreted as mixed constructors) reasoning on temporal models with just class timestamping but full evolution constraints can be reduced to reasoning over $\mathcal{DLR}^{-}_{\mathcal{US}}$. The problem of reasoning in this setting is complete for EXPTIME since the EXPTIME-complete problem of reasoning with $\mathcal{ALC}$ knowledge bases can be captured by such schemas [14].

We maintain decidability also by allowing full timestamping (i.e., timestamping for relationships, attributes and classes) but dropping evolution constraints. This is the basic temporal conceptual modelling scenario where temporal marks allow to distinguish between temporary and global constructs (this scenario also allows for both temporal keys and lifespan participation constraints). This scenario is decidable since it is possible to encode temporal schemas without evolution constraints by using a combination between the description logic $\mathcal{DLR}$ and the epistemic modal logic $\mathsf{S5}$ (see [12] for the exact mapping). Reasoning over $\mathcal{DLR}_{\mathcal{S}5}$ has been proved to be decidable and 2-EXPTIME-complete [11] by extending a previous result on the logic $\mathcal{ALC}_{\mathcal{S}5}$ [25].

Other interesting scenarios currently under investigation are the cases where the temporal expressivity is maintained in its full capability (i.e., both full timestamping and evolution constraints) but some of the constructs used at the conceptual level are dropped. In particular, by dropping isa between relationships and/or partitioning constraints we could regain decidability in the full temporal scenario. In this case we can use description logics from the *DL-Lite* family [2, 15, 16] to capture these weaker forms of conceptual schemas (see [8] for an exhaustive description of the data models that can be captured inside *DL-Lite*). A demoralisation of *DL-Lite* has been proposed in [10] where reasoning is showed to be EXPSPACE-complete. As a future work we plan to study the mapping of the various temporal constructs presented here in the temporal extension of *DL-Lite* and to investigate a tight complexity bound for the resulting temporal data modelling language.

10 Conclusions

This chapter summarises the various proposals appeared in the literature about temporal conceptual data models within a formal framework. We presented a model-theoretic semantics for different temporal constructs grouped along two generic categories, i.e., timestamping and evolution constraints. The given formal semantics clarifies the meaning of the modelling constructors and also gives a rigorous definition to relevant design support tasks such as satisfiability of schemas, classes and relationships; subsumption for both classes and relationships; general logical implication. Furthermore, for each constructor we have shown how desirable properties can be derived as logical implications from the proposed formalisation.

We have been able to show how temporal schemas can be equivalently expressed using a subset of first-order temporal logic, i.e., $\mathcal{DLR}_{\mathcal{US}}$, the description logic $\mathcal{DLR}$ extended with the temporal operators *Since* and *Until.* While $\mathcal{DLR}_{\mathcal{US}}$ is an undecidable language, several decidable sub-languages can be used to reason over temporal schemas. Since these sub-languages usually do not mix timestamping with evolution constraints, we are currently investigating new scenarios where, by weakening the atemporal expressiveness of the conceptual model, we regain decidability of the full temporal setting. We started to work on these encouraging scenarios by using *DL-Lite* as the atemporal DL and extending it to capture time varying domains.

Acknowledgements

We would like to thank our colleagues Roman Kontchakov, Carsten Lutz, Federica Mandreoli, Christine Parent, Vladislav Ryzhikov, Stefano Spaccapietra, David Toman, Frank Wolter, and Michael Zakharyaschev with whom we carried out most of the work presented in this chapter.

References

[1] Artale, A.: Reasoning on temporal conceptual schemas with dynamic constraints. In: 11th Int. Symposium on Temporal Representation and Reasoning (TIME 2004). IEEE Computer Society, Los Alamitos (2004); also in Proc. of the 2004 Int. Workshop on Description Logics (DL 2004)

[2] Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: DL-Lite in the light of first-order logic. In: Proc. of AAAI 2007, pp. 361–366 (2007)

[3] Artale, A., Franconi, E.: Temporal ER modeling with description logics. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, pp. 81–95. Springer, Heidelberg (1999)

[4] Artale, A., Franconi, E., Mandreoli, F.: Description logics for modelling dynamic information. In: Chomicki, J., van der Meyden, R., Saake, G. (eds.) Logics for Emerging Applications of Databases. LNCS, Springer, Heidelberg (2003)

[5] Artale, A., Franconi, E., Wolter, F., Zakharyaschev, M.: A temporal description logic for reasoning over conceptual schemas and queries. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS(LNAI), vol. 2424, pp. 98–110. Springer, Heidelberg (2002)

[6] Artale, A., Parent, C., Spaccapietra, S.: Modeling the evolution of objects in temporal information systems. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 22–42. Springer, Heidelberg (2006)

[7] Artale, A., Parent, C., Spaccapietra, S.: Evolving objects in temporal information systems. Annals of Mathematics and Artificial Intelligence 50(1-2), 5–38 (2007)

[8] Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: Reasoning over extended ER models. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 277–292. Springer, Heidelberg (2007)

[9] Artale, A., Cesarini, F., Soda, G.: Describing database objects in a concept language environment. IEEE Trans. on Knowledge and Data Engineering 8(2), 345–351 (1996)

[10] Artale, A., Kontchakov, R., Lutz, C., Wolter, F., Zakharyaschev, M.: Temporalising tractable description logics. In: 14th International Symposium on Temporal Representation and Reasoning (TIME 2007). IEEE Computer Society Press, Los Alamitos (2007)

[11] Artale, A., Lutz, C., Toman, D.: A description logic of change. In: Int. Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India (January 2007)

[12] Artale, A., Toman, D.: Decidable reasoning over timestamped conceptual models. In: Proc. of the 21st Int. Workshop on Description Logics (DL 2008), Dresden, Germany (May 2008)

[13] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2002)

[14] Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence 168(1-2), 70–118 (2005)

[15] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005), pp. 602–607 (2005)

[16] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 260–270 (2006)

[17] Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS 1998), pp. 149–158 (1998)

[18] Calvanese, D., Lenzerini, M., Nardi, D.: Description logics for conceptual data modeling. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems. Kluwer, Dordrecht (1998)

[19] Calvanese, D., Lenzerini, M., Nardi, D.: Unifying class-based representation formalisms. J. of Artificial Intelligence Research 11, 199–240 (1999)

[20] Chomicki, J., Toman, D.: Temporal logic in information systems. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems, ch. 1. Kluwer, Dordrecht (1998)

[21] Combi, C., Degani, S., Jensen, C.S.: Capturing temporal constraints in temporal ER models. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231. Springer, Heidelberg (2008)

[22] Etzion, O., Gal, A., Segev, A.: Extended update functionality in temporal databases. In: Etzion, O., Jajodia, S., Sripada, S. (eds.) Dagstuhl Seminar 1997. LNCS, vol. 1399, pp. 56–95. Springer, Heidelberg (1998)

[23] Finger, M., McBrien, P.: Temporal conceptual-level databases. In: Gabbay, D., Reynolds, M., Finger, M. (eds.) Temporal Logics – Mathematical Foundations and Computational Aspects, pp. 409–435. Oxford University Press, Oxford (2000)

[24] Franconi, E., Sattler, U.: A data warehouse conceptual data model for multidimensional aggregation. In: Proc. of the Workshop on Design and Management of Data Warehouses (DMDW 1999) (1999)

[25] Gabbay, D., Kurucz, A., Wolter, F., Zakharyaschev, M.: Many-dimensional modal logics: theory and applications. Studies in Logic. Elsevier, Amsterdam (2003)

[26] Gregersen, H., Jensen, J.S.: Conceptual modeling of time-varying information. Technical Report TimeCenter TR-35, Aalborg University, Denmark (1998)

[27] Gregersen, H., Jensen, J.S.: Temporal Entity-Relationship models – a survey. IEEE Transactions on Knowledge and Data Engineering 11(3), 464–497 (1999)

[28] Gupta, R., Hall, G.: An abstraction mechanism for modeling generation. In: Proc. of ICDE 1992, pp. 650–658 (1992)

[29] Hall, G., Gupta, R.: Modeling transition. In: Proc. of ICDE 1991, pp. 540–549 (1991)

[30] Hodkinson, I., Wolter, F., Zakharyaschev, M.: Decidable fragments of first-order temporal logics. Annals of Pure and Applied Logic 106, 85–134 (2000)

[31] Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics 1(1), 7–26 (2003)

[32] Jensen, C.S., Clifford, J., Gadia, S.K., Hayes, P., Jajodia, S., et al.: The Consensus Glossary of Temporal Database Concepts. In: Etzion, O., Jajodia, S., Sripada, S. (eds.) Temporal Databases - Research and Practice, pp. 367–405. Springer, Heidelberg (1998)

[33] Jensen, C.S., Snodgrass, R.T.: Temporal data management. IEEE Transactions on Knowledge and Data Engineering 111(1), 36–44 (1999)

[34] Jensen, C.S., Soo, M., Snodgrass, R.T.: Unifying temporal data models via a conceptual model. Information Systems 9(7), 513–547 (1994)

[35] McBrien, P., Seltveit, A.H., Wangler, B.: An Entity-Relationship model extended to describe historical information. In: Proc. of CISMOD 1992, Bangalore, India, pp. 244–260 (1992)

[36] Parent, C., Spaccapietra, S., Zimanyi, E.: The MurMur project: Modeling and querying multi-representation spatio-temporal databases. Information Systems 31(8), 733–769 (2006)

[37] Spaccapietra, S., Parent, C., Zimanyi, E.: Modeling time from a conceptual perspective. In: Int. Conf. on Information and Knowledge Management (CIKM 1998) (1998)

[38] Spaccapietra, S., Parent, C., Zimanyi, E.: Conceptual Modeling for Traditional and Spatio-Temporal Applications—The MADS Approach. Springer, Heidelberg (2006)

[39] Tauzovich, B.: Towards temporal extensions to the entity-relationship model. In: Proc. of the Int. Conf. on Conceptual Modeling (ER 1991). Springer, Heidelberg (1991)

[40] Theodoulidis, C., Loucopoulos, P., Wangler, B.: A conceptual modelling formalism for temporal database applications. Information Systems 16(3), 401–416 (1991)