Description Logics for Modelling Dynamic Information

Alessandro Artale[1], Enrico Franconi[2], and Federica Mandreoli[3]

[1] Dept. of Computation, UMIST, Manchester, UK
email: artale@co.umist.ac.uk
[2] Faculty of Computer Science, Free Univ. of Bozen-Bolzano, Italy
email: franconi@inf.unibz.it
[3] DII, Univ. of Modena and Reggio Emilia, Italy
email: mandreoli.federica@unimo.it

Abstract. In the first part of this Chapter we will introduce a general temporally enhanced conceptual data model able to represent time varying data, in the spirit of a temporally enhanced Entity-Relationship data model. In the second part, we will introduce an object-oriented conceptual data model enriched with schema change operators, which are able to represent the explicit temporal evolution of the schema while maintaining a consistent view on the (static) instantiated data. We will introduce a provably correct encoding of both conceptual data models and their inference problems in Description Logics. In this way, we study the properties of both the temporal conceptual data model and the object-oriented data model with schema change facilities.

1 Introduction

In recent years, data and knowledge base applications have progressively converged towards integrated technologies that try to overcome the limits of each single discipline. Research in Knowledge Representation (KR) and Computational Logic (CL) originally concentrated around formalisms that are typically tuned to deal with relatively small knowledge bases, but provide powerful deduction services, and the language to structure information is highly expressive. In contrast, Database (DB) research mainly dealt with efficient storage and retrieval with powerful query languages, and with sharing and displaying large amounts of (multimedia) documents. However, data representations were relatively simple and flat, and reasoning over the structure and the content of the documents played only a minor role.

This distinction between the requirements in Knowledge Representation and Databases is vanishing rapidly. On the one hand, to be useful in realistic applications, a modern Knowledge Representation system must be able to handle large data sets, and to provide expressive query languages. This suggests that techniques developed in the DB area could be useful for knowledge bases. On the other hand, the information stored on the web, in digital libraries, and in data warehouses is now very complex and with deep semantic

structures, thus requiring more intelligent modelling languages and methodologies, and reasoning services on those complex representations to support design, management, retrieval, and integration. Therefore, a great call for an integrated view of Knowledge Representation and Databases is emerging.

Description Logics (DL) are a very promising research area in KR and CL with applications in DBs. The main effort of the research in DL is in providing both theories and systems for expressing structured knowledge and for accessing and reasoning with it in a principled way. Recently, basic progress has been made by establishing the theoretical foundations for the effective use of DL in DBs [7]. DL offer promising formalisms for solving several problems concerning conceptual data modelling (see, e.g., [12]), intelligent information access and query processing (see, e.g., [8,34,9,24]), and information integration (see, e.g., [14,30,35,25]). This Chapter will focus on conceptual modelling issues only.

Conceptual modelling deals with the question on how to describe in a declarative and reusable way the domain information of an application, its relevant vocabulary, and how to constrain the use the data, by understanding what can be drawn from it. Recently, a number of conceptual modelling languages has emerged as de-facto standard, in particular we mention Entity-Relationship (ER) for the relational data model, UML and ODMG for the object oriented data model, and RDF, DAML+OIL and OWL for the web ontology languages. DL can be considered as an unifying formalism, since they allow the logical reconstruction and the extension of all the above representational tools (see, e.g., [12,13]). 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 is involved, there is the demand of more sophisticated and expressive languages than for normal databases. Again, DL research is very active in providing more expressive languages for conceptual modelling (see, e.g., [12,13,21,23]). In the case of conceptual modelling of *dynamic information*, which is the main topic of this Chapter, we should distinguish two cases. In the first one, the dynamics involves the data to be modelled – this is the case of *temporal conceptual modelling* for *temporal DBs* – and in the second one the dynamics involves the change of the conceptual schema modelling a static domain – this is the case of *conceptual schema evolution*. This Chapter summarises the research trends in the application of DL for conceptual modelling in the context of both temporal DBs and schema evolution, and it is structured as follows.

After an introductory Section, the Chapter is organised in three parts. In the first part, the description logic $\mathcal{ALCQI}$ and the its temporal extension

$\mathcal{ALCQI}_{\mathcal{US}}$ is introduced. Results on the complexity of the most important reasoning tasks are summarised.

The second part is on temporal conceptual modelling in the relational data model. Temporal conceptual modelling constructs for the valid time representation as appeared in the literature on the Entity-Relationship data model (see, e.g., [40,26]) are considered. A systematic characterisation (first introduced in [3]) of the constructs introduced in the majority of temporal conceptual modelling systems is provided.

The third part is about schema evolution in the object-oriented data model. A data model supports *schema evolution* if it allows for explicit changes in the schema without the loss of extant data. A complete formal characterisation of the semantics of schema changes in a classical object-oriented schema evolution framework (first introduced in [22]) will be provided.

The organisation of both part two and part three of this Chapter is similar. We first define the syntax and the semantics of the considered data model: in part two it is a temporal Entity-Relationship data model— that is an Entity-Relationship data model extended with constructs to express temporal properties; in part three it is an object-oriented data model extended with constructs to express the evolution of the schema. Then, we define the semantics of the reasoning tasks that are relevant for the data model at hand. Finally, we introduce in both cases a provably correct encoding of the data model and the reasoning problems in a suitable description logic: the DL is $\mathcal{ALCQI}_{\mathcal{US}}$ for the temporal conceptual data model, and $\mathcal{ALCQI}$ for the object-oriented data model with schema change facilities. The encoding in DL has the following advantages: an homogeneous framework can be used to encode the dynamic aspects in both the relational and the object-oriented data models (as first introduced in [12] for the non dynamic aspects); upper bounds to the computational complexity of the original reasoning problems can be stated; algorithms and practical reasoners from the DL field can be used to solve the original problems. In particular, we show that reasoning in both temporal Entity-Relationship diagrams (without snapshot relations, temporal keys, or temporal cardinalities) and in evolving object-oriented schemas is in EXPTIME, while we refer to the results in [2] to show that reasoning in the full temporal Entity-Relationship data model is undecidable.

2 Conceptual Modelling of Dynamic Information

In this Section we first review the common features of temporally extended conceptual data models developed to abstract the temporal aspects of information. Without loss of generality, we will specifically refer to a general temporal ER model, and we will consider it as capturing the relevant common features of the models introduced by the systems TimeER [27], ERT [42], MADS [40]. These models cover the full spectrum of temporal constructs

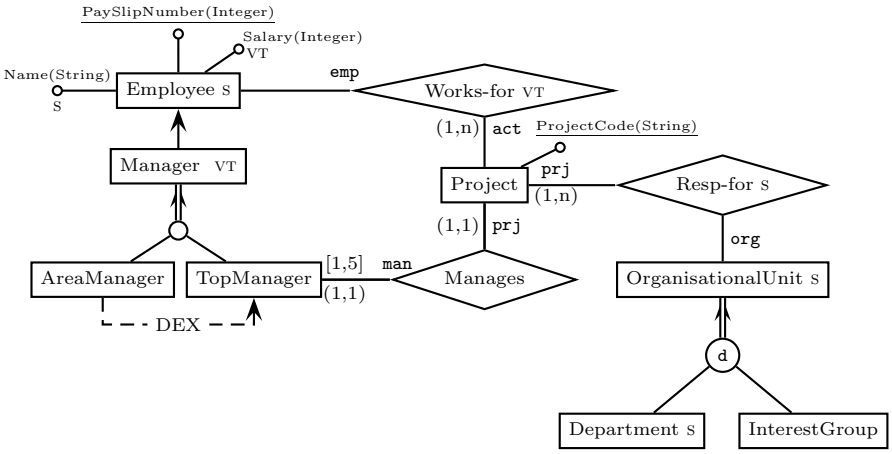

Fig. 1. A temporal ER diagram.

(cf. [26] for an extensive overview of temporally extended ER models). We refer to ER models because they are the most mature field in temporal conceptual modelling. As far as the notation for temporal constructs is concerned, we use a unifying notation that will capture the commonalities between the different models.

As a main characteristics, a data model for time-varying information should support—for all the types of the model—the notion of *valid time*—which is the time when a property holds, i.e., it is true in the representation of the world. Possibly, the data model should support also *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.

Temporal support is achieved by giving a temporal meaning to each standard non-temporal conceptual construct, and then adding new temporal constructs. The ability to add temporal constructs on top of a temporally implicit model has the advantage of preserving the non-temporal semantics of conventional (legacy) conceptual schemas when embedded into temporal schemas—what is called *upward compatibility*. The possibility of capturing the meaning of both legacy and temporal schemas is crucial in modelling data warehouses or federated databases, where sources may be collections of both temporal and legacy databases. *Orthogonality* [40] is another desirable principle: temporal constructs should be specified separately and independently for entities, relationships and attributes. Depending on the application requirements, the temporal support must be decided by the designer. Furthermore, *snapshot reducibility* [39] of a schema says that snapshots of the database described by a temporal schema are the same as the database described by the same

Fig. 2. Generation relationships.

schema where all the temporal constructs are eliminated and the schema is interpreted atemporally.

Temporal marks are usually added to capture the temporal behaviour of the different components of a conceptual schema. In particular, entities, relationships and attributes can be either S-*marked* in what case they are considered *snapshot*[1] constructs (i.e., each of their instances has a global lifetime, as in the case they belong to a legacy diagram), VT-*marked* and they are considered *temporary* constructs (i.e., each of their instances has a temporary existence), or *un-marked*, i.e. without any temporal mark, in what case they have temporally unconstrained instances (i.e., their instances can have either a global or a temporary existence). Participation constraints are distinguished between *snapshot participation constraints*—true at each point in time and represented by a pair of values in parentheses—and *lifespan participation constraints*—evaluated during the entire existence of the entity and represented by a pair of values in square brackets [41,27,40]. Fig. 1 gives a diagram showing the various temporal constructs, and will form a running example trough the Chapter.

Dynamic relationships between entities [40] can be either *transition* or *generation* relationships. In a *transition* relationship the instances of an entity may eventually become instances of another entity. The instances of the source entity are said to *migrate* into the target entity and the phenomenon is called *object migration*. In the temporal conceptual modelling literature, two types of transitions have been considered [28,29,40]: *dynamic evolution* when objects cease to be instances of the source entity, and *dynamic extension* otherwise. In general, type constraints enforce that both the source and the target entity belong to the same generalisation hierarchy. Fig. 1 shows an example of dynamic extension between the entities `AreaManager` and a `TopManager`—represented with an arrow from the source to the target entity with the label DEX. *Generation* relationships involve different instances—differently from the transition case: an instance (or set of instances) from a source entity is (are) transformed in an instance (or set of instances) of the target entity. Fig. 2 shows a generation relationship `Generate` between `Orange` and `Juice` marked with the label G and an arrow pointing to the target entity. The lifespan cardinality says that 5 oranges are needed to produce a single juice, and each orange is, sooner or later, consumed in the transformation process.

[1] See the consensus glossary [31] for the terminology used.

We have considered so far a temporally enhanced conceptual data model able to represent time varying data. Now we want to introduce a conceptual data model enriched with schema change operators, which are able to represent the explicit evolution of the schema while maintaining a consistent view on the (static) instantiated data.

The problems of schema evolution becomes relevant in the context of long-lived database applications, where stored data is considered worth surviving changes in the database schema [37]. One of the fundamental issues concerning the introduction of schema change operators in a data model is the *semantics of change*, which refers to the effects of the change on the schema itself, and, in particular the checking and maintenance of schema consistency after changes. In the literature, the problem have been widely studied in relational and object-oriented database papers. In the relational field [38,17], the problem is solved by specifying a precise semantics of the schema changes, for example via algebraic operations on catalogue and base tables. However, the related consistency problems have not been considered so far. The correct use of schema changes is completely under the database designer/administrator's responsibility, without automatic system aid and control. In the object-oriented field, two main approaches were followed to ensure consistency in pursuing the "semantics of change" problem. The first approach is based on the adoption of *invariants* and *rules*, and has been used, for instance, in the ORION [6] and O_2 [20] systems. The second approach, which was proposed in [36], is based on the introduction of *axioms*. In the former approach, the invariants define the consistency of a schema, and definite rules must be followed to maintain the invariants satisfied after each schema change. Invariants and rules are strictly dependent on the underlying object model, as they refer to specific model elements. In the latter approach, a sound and complete set of axioms (provided with an inference mechanism) formalises the *dynamic schema evolution*, which is the actual management of schema changes in a system in operation. The approach is general enough to capture the behaviour of several different systems and, thus, is useful for their comparison in a unified framework. The compliance of the available primitive schema changes with the axioms automatically ensures schema consistency, without need for explicit checking, as incorrect schema versions cannot actually be generated.

In the context of dynamic schema evolution, we deal with the "semantics of change" problem, and try to give a general answer to the problem of deciding the consistency and the consequences of any given sequence of elementary schema changes. To this end, we will introduce a formal approach for the specification and management of evolving database schemas [22].

$$
\begin{array}{lll}
C, D \rightarrow & A \mid & \text{(primitive concept)} \\
& \top \mid & \text{(top)} \\
& \bot \mid & \text{(bottom)} \\
& \neg C \mid & \text{(complement)} \\
& C \sqcap D \mid & \text{(conjunction)} \\
& C \sqcup D \mid & \text{(disjunction)} \\
& \forall R.\, C \mid & \text{(univ. quantifier)} \\
& \exists R.\, C \mid & \text{(exist. quantifier)} \\
& (\geqslant n\ R.\ C) \mid & \text{(min cardinality)} \\
& (\leqslant n\ R.\ C) & \text{(max cardinality)} \\
\\
R, S \rightarrow & P \mid & \text{(primitive role)} \\
& R^{-} & \text{(inverse role)}
\end{array}
$$

Fig. 3. Syntax rules for the $\mathcal{ALCQI}$ Description Logic

3 Description Logics

In this Section we give a brief introduction to the $\mathcal{ALCQI}$ description logic, which will serve as the basic representation language for the non-temporal information. With respect to the formal apparatus, we will strictly follow the concept language formalism presented in [5]. In this perspective, Description Logics are considered as a *structured* fragment of predicate logic.

The basic types of $\mathcal{ALCQI}$ are *concepts* and *roles*. A concept is a description gathering the common properties among a collection of individuals; from a logical point of view it is a unary predicate ranging over the domain of individuals. Inter-relationships between these individuals are represented by means of roles, which are interpreted as binary relations over the domain of individuals. According to the syntax rules of Figure 3, $\mathcal{ALCQI}$ *concepts* (denoted by the letters C and D) are built out of *primitive concepts* (denoted by the letter A) and *roles* (denoted by the letter R, S); roles are built out of *primitive roles* (denoted by the letter P). In the following, the shortcut $(\geqslant n\ R)$ is used instead of $(\geqslant n\ R.\ \top)$, and similarly for $(\leqslant n\ R)$.

Let us now consider the formal semantics of $\mathcal{ALCQI}$. We define the *meaning* of concepts as sets of individuals – as for unary predicates – and the meaning of roles as sets of pairs of individuals – as for binary predicates. Formally, an *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a set $\Delta^{\mathcal{I}}$ of individuals (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) mapping every concept to a subset of $\Delta^{\mathcal{I}}$, every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and such that the equations in Figure 4 are satisfied.

For example, we can consider the concept of *happy father*, defined using the primitive concepts `Man, Doctor, Rich, Famous` and the roles `CHILD, FRIEND`. The concept HAPPY-FATHER can be expressed in $\mathcal{ALCQI}$ with the

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.\, C)^{\mathcal{I}} &= \{i \in \Delta^{\mathcal{I}} \mid \forall j.\ R^{\mathcal{I}}(i,j) \Rightarrow C^{\mathcal{I}}(j)\} \\
(\exists R.\, C)^{\mathcal{I}} &= \{i \in \Delta^{\mathcal{I}} \mid \exists j.\ R^{\mathcal{I}}(i,j) \wedge C^{\mathcal{I}}(j)\} \\
(\geqslant n\ R.\ C)^{\mathcal{I}} &= \{i \in \Delta^{\mathcal{I}} \mid\ \sharp\{j \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(i,j) \wedge C^{\mathcal{I}}(j)\} \geq n\} \\
(\leqslant n\ R.\ C)^{\mathcal{I}} &= \{i \in \Delta^{\mathcal{I}} \mid\ \sharp\{j \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(i,j) \wedge C^{\mathcal{I}}(j)\} \leq n\}
\end{aligned}
$$

$$
(R^{-})^{\mathcal{I}} = \{(i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(j,i)\}
$$

Fig. 4. The semantics of $\mathcal{ALCQI}$

expression:
$\texttt{Man} \sqcap (\exists \texttt{CHILD}.\, \top) \sqcap \forall \texttt{CHILD}.\, (\texttt{Doctor} \sqcap \exists \texttt{FRIEND}.\, (\texttt{Rich} \sqcup \texttt{Famous}))$
i.e., those men having some child and all of whose children are doctors having some friend who is rich or famous.

A *knowledge base*, in this context, is a finite set Σ of *terminological axioms*; it can also be called a *terminology* or TBox. For a primitive concept A, and concepts C, D, terminological axioms are of the form $A \doteq C$, $A \sqsubseteq C$ and $C \sqsubseteq D$. An interpretation $\mathcal{I}$ satisfies $C \sqsubseteq D$ if and only if the interpretation of C is included in the interpretation of D, i.e., $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. The axiom $C \sqsubseteq D$ is a generalisation of the axioms $A \doteq C$, $A \sqsubseteq C$; for example, an axiom $A \doteq C$ – where A is a primitive concept – can be reduced to the pair of axioms $(A \sqsubseteq C)$ and $(C \sqsubseteq A)$. An interpretation $\mathcal{I}$ is a *model* of a knowledge base Σ iff every terminological axiom of Σ is satisfied by $\mathcal{I}$. If Σ has a model, then it is *satisfiable*; thus, checking for KB satisfiability is deciding whether there is at least one model for the knowledge base. Σ *logically implies* an axiom $C \sqsubseteq D$ (written $\Sigma \models C \sqsubseteq D$) if $C \sqsubseteq D$ is satisfied by every model of Σ. We also say that the concept C is *subsumed* by a concept D in the knowledge base Σ. For example, the concept $\texttt{Person} \sqcap (\exists \texttt{CHILD}.\, \texttt{Person})$ denoting the class PARENT – i.e., a person having at least a child which is a person – subsumes the concept
$\texttt{Man} \sqcap (\exists \texttt{CHILD}.\, \top) \sqcap \forall \texttt{CHILD}.\, (\texttt{Doctor} \sqcap \exists \texttt{FRIEND}.\, (\texttt{Rich} \sqcup \texttt{Famous}))$
denoting the class of HAPPY-FATHERS, with respect to the following knowledge base Σ:
$\texttt{Doctor} \doteq \texttt{Person} \sqcap \exists \texttt{DEGREE}.\, \texttt{Phd}$,
$\texttt{Man} \doteq \texttt{Person} \sqcap \exists \texttt{sex}.\, \texttt{Male} \sqcap (\leqslant 1\ \texttt{sex}.\ \top)$,
i.e., every happy father is also a person having at least one child, given the background knowledge that men are male persons, and that doctors are persons. A concept C is satisfiable, given a knowledge base Σ, if there exists a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$. For example, given the above knowledge

base Σ, the concept $(\exists \texttt{CHILD}.\,\texttt{Man}) \sqcap (\forall \texttt{CHILD}.\,(\exists \texttt{sex}.\,\neg \texttt{Male}))$ is unsatisfiable. In fact, an individual whose children are not male cannot have a child being a man.

Both concept and knowledge base satisfiability problems are reducible to logical implication. Indeed, a concept C is satisfiable in Σ iff $\Sigma \not\models C \doteq \bot$; a knowledge base Σ is *satisfiable* iff $\Sigma \not\models \top \sqsubseteq \bot$. On the other hand, logical implication can be reduced to a satisfiability problem since $\Sigma \models C \sqsubseteq D$ iff $(C \sqcap \neg D)$ is unsatisfiable in Σ. Reasoning in $\mathcal{ALCQI}$ (i.e., deciding knowledge base and concept satisfiability, deciding concept subsumption and logical implication) is decidable, and it is an EXPTIME-complete problem [18].

3.1 Adding Tense Logic

The tense-logical extension of the non-temporal description logic $\mathcal{ALCQI}$ is able to describe the time-varying properties of objects. Let $\mathcal{ALCQI}_{\mathcal{US}}$ be the extension of $\mathcal{ALCQI}$ with the temporal operators $\mathcal{U}$ (Until) and $\mathcal{S}$ (Since). We add the following rules to the syntax presented in figure 3:

$$C \rightarrow \Diamond^{+}C \mid \Diamond^{-}C \mid \Diamond^{*}C \mid \Box^{+}C \mid \Box^{-}C \mid \Box^{*}C \mid \oplus C \mid \ominus C \mid C\,\mathcal{U}\,D \mid C\,\mathcal{S}\,D$$
$$R \rightarrow \Diamond^{+}R \mid \Diamond^{-}R \mid \Diamond^{*}R \mid \Box^{+}R \mid \Box^{-}R \mid \Box^{*}R \mid \oplus R \mid \ominus R \mid R\,\mathcal{U}\,S \mid R\,\mathcal{S}\,S$$

The tense operators for concepts $\Diamond^{+}$ (sometime in the future), $\Diamond^{-}$ (sometime in the past), $\Box^{+}$ (always in the future), $\Box^{-}$ (always in the past), $\oplus$ (at the next point in time), and $\ominus$ (at the previous point in time) are derived operators from the basic $\mathcal{S}$ (since) and $\mathcal{U}$ (until) operators: $\Diamond^{+}, \Diamond^{-}, \oplus$ are defined as $\Diamond^{+}C \doteq \top\mathcal{U}C$, $\Diamond^{-}C \doteq \top\mathcal{S}C$, $\oplus C \doteq \bot\mathcal{U}C$, while $\Box^{+}, \Box^{-}, \ominus$ are their duals. The tense operators $\Diamond^{*}$ (at some moment) and its dual $\Box^{*}$ (at all moments) can be defined as $\Diamond^{*}C \doteq C \sqcup \Diamond^{+}C \sqcup \Diamond^{-}C$ and $\Box^{*}C \doteq C \sqcap \Box^{+}C \sqcap \Box^{-}C$, respectively. We call $\mathcal{ALCQI}^{-}_{\mathcal{US}}$ the tense-logical extension of $\mathcal{ALCQI}$ only at the level of concepts – i.e., no temporal operators are allowed in role expressions.

$\mathcal{ALCQI}_{\mathcal{US}}$ semantics naturally extends with time the non-temporal semantics presented in the previous Section. A temporal structure $\mathcal{T} = (\mathcal{T}_p, <)$ is assumed, where $\mathcal{T}_p$ is a set of time points and $<$ is a strict linear order on $\mathcal{T}_p$; $\mathcal{T}$ is assumed to be isomorphic to $(\mathbb{Z}, <)$. The assumption of the flow of time being isomorphic to the integer numbers $\mathbb{Z}$ is necessary if we want to model abstract temporal databases that extend both in the past and in the future; the results in this Chapter are based on this assumption. An $\mathcal{ALCQI}_{\mathcal{US}}$ *temporal interpretation* over $\mathcal{T}$ is a pair $\mathcal{M} \doteq \langle \mathcal{T}, \mathcal{I} \rangle$, where $\mathcal{I}$ is a function associating to each $t \in \mathcal{T}_p$ a standard non-temporal $\mathcal{ALCQI}$ interpretation, $\mathcal{I}(t) \doteq \langle \Delta^{\mathcal{I}}, R_0^{\mathcal{I}(t)}, \ldots, C_0^{\mathcal{I}(t)}, \ldots \rangle$, such that it satisfies the equations in figure 4 plus the equations in figure 5.
An interpretation $\mathcal{M}$ over a temporal structure $\mathcal{T} = (\mathcal{T}_p, <)$ satisfies a terminological axiom $C \sqsubseteq D$ if $C^{\mathcal{I}(t)} \subseteq D^{\mathcal{I}(t)}$ for every $t \in \mathcal{T}_p$. A knowledge base Σ

$$
\begin{aligned}
(C\,\mathcal{U}\,D)^{\mathcal{I}(t)} &= \{\ i \in \Delta^{\mathcal{I}} \mid \exists v > t.\ (i \in D^{\mathcal{I}(v)} \wedge \forall w(t < w < v).\ i \in C^{\mathcal{I}(w)})\} \\
(C\,\mathcal{S}\,D)^{\mathcal{I}(t)} &= \{\ i \in \Delta^{\mathcal{I}} \mid \exists v < t.\ (i \in D^{\mathcal{I}(v)} \wedge \forall w(v < w < t).\ i \in C^{\mathcal{I}(w)})\} \\
(R\,\mathcal{U}\,S)^{\mathcal{I}(t)} &= \{\ (i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \\
&\qquad \exists v > t.\ ((i,j) \in S^{\mathcal{I}(v)} \wedge \forall w(t < w < v).\ (i,j) \in R^{\mathcal{I}(w)})\} \\
(R\,\mathcal{S}\,S)^{\mathcal{I}(t)} &= \{\ (i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \\
&\qquad \exists v < t.\ ((i,j) \in S^{\mathcal{I}(v)} \wedge \forall w(v < w < t).\ (i,j) \in R^{\mathcal{I}(w)})\} \\
(\Diamond^{+} R)^{\mathcal{I}(t)} &= \{(i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists v > t.\ (i,j) \in R^{\mathcal{I}(v)}\} \\
(\oplus\ R)^{\mathcal{I}(t)} &= \{(i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (i,j) \in R^{\mathcal{I}(t+1)}\} \\
(\Diamond^{-} R)^{\mathcal{I}(t)} &= \{(i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists v < t.\ (i,j) \in R^{\mathcal{I}(v)}\} \\
(\ominus\ R)^{\mathcal{I}(t)} &= \{(i,j) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (i,j) \in R^{\mathcal{I}(t-1)}\}
\end{aligned}
$$

Fig. 5. The semantics of the temporal extension of $\mathcal{ALCQI}$

is *satisfiable* in the temporal structure $\mathcal{T}$ if there is a temporal interpretation $\mathcal{M}$ over $\mathcal{T}$ which satisfies every axiom in Σ; in this case $\mathcal{M}$ is called a *model* over $\mathcal{T}$ of Σ. Σ *logically implies* an axiom $C \sqsubseteq D$ in the temporal structure $\mathcal{T}$ (written $\Sigma \models C \sqsubseteq D$) if $C \sqsubseteq D$ is satisfied by every model over $\mathcal{T}$ of Σ. In this latter case, the concept C is said to be *subsumed* by the concept D in the knowledge base Σ and the temporal structure $\mathcal{T}$. A concept C is satisfiable, given a knowledge base Σ, if there exists a model $\mathcal{M} \doteq \langle \mathcal{T}, \mathcal{I} \rangle$ of Σ such that $C^{\mathcal{I}(t)} \neq \emptyset$ for some $t \in \mathcal{T}$, i.e. $\Sigma \not\models C \doteq \bot$.

As an example of a concept using temporal operators, consider the definition of a mortal. The class of mortals denotes all the individuals which are currently living beings and live in some place, and will maintain this essence until they will stop to be living beings forever:

$$
\begin{aligned}
\texttt{Mortal} \doteq\ & \texttt{LivingBeing} \sqcap \forall \texttt{LIVES-IN}.\texttt{Place} \sqcap \\
& (\texttt{LivingBeing}\ \mathcal{U}\ \Box^{+} \neg \texttt{LivingBeing})
\end{aligned}
$$

The tense-logical extensions of $\mathcal{ALCQI}$ have been thoroughly studied in [43,4]. It has been proved that reasoning in $\mathcal{ALCQI}_{\mathcal{US}}$ (i.e., deciding knowledge base and concept satisfiability, deciding concept subsumption, and deciding logical implication) is undecidable, while it is EXPTIME-complete in its fragment $\mathcal{ALCQI}^{-}_{\mathcal{US}}$. Even pure $\mathcal{ALC}$ with just the $\Box^{+}$ operator at the level of roles is enough to get undecidability. Reasoning in $\mathcal{ALCQI}^{-}_{\mathcal{US}}$ with *complex terminological axioms* – obtained by arbitrarily combining terminological axioms with boolean and temporal operators, i.e., if φ and ψ are $\mathcal{ALCQI}^{-}_{\mathcal{US}}$ axioms, then so are $\neg\varphi, \varphi \wedge \psi, \varphi\,\mathcal{U}\,\psi, \varphi\,\mathcal{S}\,\psi$ – becomes EXPSPACE-complete.

4 $\mathcal{ER}_{VT}$: A Formal Temporal Conceptual Model

In this Section a temporal ER model ($\mathcal{ER}_{VT}$) is introduced in agreement with the general principles devised in Section 2. $\mathcal{ER}_{VT}$ supports valid time for entities, attributes and relationships in the line of TimeER [27] and ERT [42], while supporting dynamic relationships as presented in MADS [40]. The motivation behind the development of $\mathcal{ER}_{VT}$ is twofold: (1) the need for a formally specified temporal ER language with a model theoretic semantics associated to each (temporal) construct; (2) the possibility to perform automatic deductions on $\mathcal{ER}_{VT}$ schemas as the result of mapping $\mathcal{ER}_{VT}$ schemas to knowledge bases expressed in the temporal description logic $\mathcal{ALCQI}_{\mathcal{US}}$.

The proposed formalisation is based on a linear syntax and an associated model-theoretic semantics as presented in [13] for the non-temporal EER model[2], which is here extended to take into account both the time dimension and the temporal constructs. This will give both a formal characterisation of the most important temporal conceptual modelling constructs (for the valid time representation) appeared in the literature, and the formal background to develop the correspondence to the temporal description logic $\mathcal{ALCQI}_{\mathcal{US}}$ in order to reason over temporal schemas.

Presenting the $\mathcal{ER}_{VT}$ syntax we adopt the following notation: given two sets X, Y a *X-labelled* tuple over Y is a function from X to Y; the labelled tuple T that maps the set $\{x_1, \ldots, x_n\} \subseteq X$ to the set $\{y_1, \ldots, y_n\} \subseteq Y$ is denoted by $\langle x_1 : y_1, \ldots, x_n : y_n \rangle$, while $T[x_i] = y_i$.

Definition 1 ($\mathcal{ER}_{VT}$ Syntax). An $\mathcal{ER}_{VT}$ schema is a tuple:
$\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{CARD}^L, \text{ISA}, \text{DISJ}, \text{COVER}, \text{S}, \text{T}, \text{KEY}, \text{DEX}, \text{DEV}, \text{GEN})$, where:

- $\mathcal{L}$ a finite alphabet partitioned into
  - a set $\mathcal{E}$ of *entity* symbols,
  - a set $\mathcal{A}$ of *attribute* symbols,
  - a set $\mathcal{R}$ of *relationship* symbols,
  - a set $\mathcal{U}$ of *role* symbols,
  - and a set $\mathcal{D}$ of *domain* symbols.

  We will call the tuple $(\mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{U}, \mathcal{D})$ the *signature* of the schema Σ.
- $\mathcal{E}$ is partitioned into
  - a set $\mathcal{E}^S$ of *snapshot entities* (the S-*marked* entities, in the graphical representation of Fig. 1);
  - a set $\mathcal{E}^I$ of I*mplicitly temporal entities* (the *un-marked* entities);
  - and a set $\mathcal{E}^T$ of *temporary entities* (the VT-*marked* entities).
- A similar partition applies to the set $\mathcal{R}$.

[2] EER is the standard entity-relationship data model, enriched with ISA links (solid directed lines), generalised hierarchies with disjoint (circle with a 'd' inside) and covering (double directed lines) constraints, and full cardinality constraints [19].

- ATT is a function that maps an entity symbol in $\mathcal{E}$ to an $\mathcal{A}$-labelled 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 $\mathcal{U}$-labelled tuple over $\mathcal{E}$,
$$\text{REL}(R) = \langle U_1 : E_1, \ldots, U_k : E_k \rangle,$$
and k is the *arity* of R.
- CARD, and CARD^L are functions
$$\mathcal{E} \times \mathcal{R} \times \mathcal{U} \mapsto \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$$
denoting snapshot and lifespan cardinality constraints, respectively.
They are such that if $\text{REL}(R) = \langle U_1 : E_1, \ldots, U_k : E_k \rangle$ then both $\text{CARD}(E, R, U)$ and $\text{CARD}^L(E, R, U)$ are defined only if $U = U_i$ and $E = E_i$, for some $i \in \{1, \ldots, k\}$. We denote with $\text{CMIN}(E, R, U)$ and $\text{CMAX}(E, R, U)$ ($\text{CMIN}^L(E, R, U)$ and $\text{CMAX}^L(E, R, U)$) the first and second component of CARD (CARD^L). If not stated otherwise, CMIN and CMIN^L are assumed to be 0 while CMAX and CMAX^L are assumed to be ∞.
- ISA is a binary relationship
$$\text{ISA} \subseteq (\mathcal{E} \times \mathcal{E}) \cup (\mathcal{R} \times \mathcal{R}).$$
ISA between relationships is restricted to relationships with the same arity[3].
- DISJ, COVER are binary relations over $2^{\mathcal{E}} \times \mathcal{E}$, describing disjointness and covering partitions, respectively.
- S, T are binary relations over $\mathcal{E} \times \mathcal{A}$ containing, respectively, the snapshot and temporary attributes of an entity. Furthermore, if $\langle E, A \rangle \in$ S, T then A is between the attributes in $\text{ATT}(E)$.
- KEY is a function that maps entity symbols in $\mathcal{E}$ to their key attributes, $\text{KEY}(E) = A$. Furthermore, if $\text{KEY}(E) = A$ then A is between the attributes in $\text{ATT}(E)$.
- Both DEX and DEV are binary relations over $\mathcal{E} \times \mathcal{E}$ describing the evolution of entities.
- GEN is a unary relation over $\mathcal{R}$ describing generation relationships. If $R \in$ GEN then R is a binary relationship such that
$\text{REL}(R) = \langle \texttt{source} : E_s, \texttt{target} : E_t \rangle$.

The model-theoretic semantics associated to the $\mathcal{ER}_{VT}$ modelling language adopts the *snapshot* representation of abstract temporal databases and temporal conceptual models (see e.g. [16]). Following this paradigm, given a

[3] For ISA relations we use the notation $E_1 \,\text{ISA}\, E_2$ instead of $\langle E_1, E_2 \rangle \in$ ISA. Similarly for DISJ,COVER,DEV,DEX.

flow of time $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$ – where $\mathcal{T}_p$ is a set of time points (or chronons) isomorphic to $\langle \mathbb{Z}, < \rangle$, and $<$ a binary precedence relation on $\mathcal{T}_p$ – a temporal database can be regarded as a mapping from time points in $\mathcal{T}$ to standard relational databases, with the same interpretation of constants and the same domain along time. In alternative to the snapshot model, the *timestamp* model is defined by adding temporal attributes to a relation. The timestamp model is used in the bitemporal conceptual data model (*BCDM*) [33,32] where each tuple in a bitemporal relation is associated with a bitemporal timestamp value recording both the valid and the transaction time of the tuple. The snapshot model is in correspondence with the timestamp model. Indeed, both snapshot databases and the projection over valid time of timestamp databases represent the same class of temporal databases [16]. The following model theoretic semantics for $\mathcal{ER}_{VT}$ is in accordance with a snapshot representation of valid time temporal databases. Examples will follow immediately after the definition.

Definition 2 ($\mathcal{ER}_{VT}$ Semantics). Let Σ be an $\mathcal{ER}_{VT}$ schema, and $BD = \bigcup_{D_i \in \mathcal{D}} BD_i$ be a set of *basic domains* such that $BD_i \cap BD_j = \emptyset$ for $i \neq j$. $\mathcal{B} = (\mathcal{T}, \Delta^{\mathcal{B}} \cup \Delta^{\mathcal{B}}_D, \cdot^{\mathcal{B}(t)})$ is a *Temporal Database State* for the schema Σ where:

- $\Delta^{\mathcal{B}}$ is a non-empty 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 such that $\Delta^{\mathcal{B}}_{D_i} \subseteq BD_i$ – we call $\Delta^{\mathcal{B}}_{D_i}$ *active domain*.
- $\cdot^{\mathcal{B}(t)}$ is a function that for each $t \in \mathcal{T}$ maps:
  - Every domain symbol $D_i \in \mathcal{D}$ to the corresponding active domain $D_i^{\mathcal{B}(t)} = \Delta^{\mathcal{B}}_{D_i}$ – then $D_i^{\mathcal{B}(t)}$ does not depend from the time t of evaluation.
  - Every entity $E \in \mathcal{E}$ to a set $E^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}}$.
  - Every relationship $R \in \mathcal{R}$ to a set $R^{\mathcal{B}(t)}$ of $\mathcal{U}$-labelled tuples over $\Delta^{\mathcal{B}}$.
  - Every attribute $A \in \mathcal{A}$ to a set $A^{\mathcal{B}(t)} \subseteq \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}}_D$.

$\mathcal{B}$ is a *legal temporal database state* if it satisfies all the integrity constraints expressed in the schema:

(c1) For each $E_1, E_2 \in \mathcal{E}$, if E_1 ISA E_2 then, $E_1^{\mathcal{B}(t)} \subseteq E_2^{\mathcal{B}(t)}$.

(c2) For each $R_1, R_2 \in \mathcal{R}$, if R_1 ISA R_2, then, $R_1^{\mathcal{B}(t)} \subseteq R_2^{\mathcal{B}(t)}$.

(c3) For each $E \in \mathcal{E}$, if $\text{ATT}(E) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$, then, $e \in E^{\mathcal{B}(t)} \rightarrow (\forall i \in \{1, \ldots, h\}, \exists! a_i.\ \langle e, a_i \rangle \in A_i^{\mathcal{B}(t)} \wedge \forall a_i.\ \langle e, a_i \rangle \in A_i^{\mathcal{B}(t)} \rightarrow a_i \in \Delta^{\mathcal{B}}_{D_i})$. We consider, for simplicity, only single-valued attributes.

(c4) For each $R \in \mathcal{R}$, if $\text{REL}(R) = \langle U_1 : E_1, \ldots, U_k : E_k \rangle$, then, $r \in R^{\mathcal{B}(t)} \rightarrow (r = \langle U_1 : e_1, \ldots, U_k : e_k \rangle \wedge \forall i \in \{1, \ldots, k\}.\ e_i \in E_i^{\mathcal{B}(t)})$. In the following we adopt the convention: $\langle U_1 : e_1, \ldots, U_k : e_k \rangle \equiv \langle e_1, \ldots, e_k \rangle$, and $r[U_i] \equiv r[i]$ to denote the U_i/i-component of r – i.e., the *naming* and the *positional* notation for tuples are two equivalent notations.

(c5) For each cardinality constraint $\text{CARD}(E, R, U)$, then, $e \in E^{\mathcal{B}(t)} \rightarrow \text{CMIN}(E, R, U) \leq \#\{r \in R^{\mathcal{B}(t)} \mid r[U] = e\} \leq \text{CMAX}(E, R, U)$.

(c6) For each lifespan cardinality constraint $\text{CARD}^L(E, R, U)$, then, $e \in E^{\mathcal{B}(t)} \rightarrow \text{CMIN}^L(E, R, U) \leq \#\bigcup_{t' \in \mathcal{T}} \{r \in R^{\mathcal{B}(t')} \mid r[U] = e\} \leq \text{CMAX}^L(E, R, U)$.

(c7) For each snapshot entity $E \in \mathcal{E}^S$, then, $e \in E^{\mathcal{B}(t)} \rightarrow \forall t' \in \mathcal{T}.\ e \in E^{\mathcal{B}(t')}$.

(c8) For each temporary entity $E \in \mathcal{E}^T$, then, $e \in E^{\mathcal{B}(t)} \rightarrow \exists t' \neq t.\ e \notin E^{\mathcal{B}(t')}$.

(c9) For each snapshot relationship $R \in \mathcal{R}^S$, then, $r \in R^{\mathcal{B}(t)} \rightarrow \forall t' \in \mathcal{T}.\ r \in R^{\mathcal{B}(t')}$.

(c10) For each temporary relationship $R \in \mathcal{R}^T$, then, $r \in R^{\mathcal{B}(t)} \rightarrow \exists t' \neq t.\ r \notin R^{\mathcal{B}(t')}$.

(c11) For each entity $E \in \mathcal{E}$, if $\text{ATT}(E) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$, and $\langle E, A_i \rangle \in \text{S}$ then, $(e \in E^{\mathcal{B}(t)} \wedge \langle e, a_i \rangle \in A_i^{\mathcal{B}(t)}) \rightarrow \forall t' \in \mathcal{T}.\ \langle e, a_i \rangle \in A_i^{\mathcal{B}(t')}$.

(c12) For each entity $E \in \mathcal{E}$, if $\text{ATT}(E) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$, and $\langle E, A_i \rangle \in \text{T}$ then, $(e \in E^{\mathcal{B}(t)} \wedge \langle e, a_i \rangle \in A_i^{\mathcal{B}(t)}) \rightarrow \exists t' \neq t.\ \langle e, a_i \rangle \notin A_i^{\mathcal{B}(t')}$.

(c13) For $E, E_1, \ldots, E_n \in \mathcal{E}$, if $\{E_1, \ldots, E_n\}$ DISJ E then, $\forall i \in \{1, \ldots, n\}.\ E_i$ ISA $E \wedge \forall j \in \{1, \ldots, n\},\ j \neq i.\ E_i^{\mathcal{B}(t)} \cap E_j^{\mathcal{B}(t)} = \emptyset$.

(c14) For $E, E_1, \ldots, E_n \in \mathcal{E}$, if $\{E_1, \ldots, E_n\}$ COVER E then, $\forall i \in \{1, \ldots, n\}.\ E_i$ ISA $E \wedge E^{\mathcal{B}(t)} = \bigcup_{i=1}^{n} E_i^{\mathcal{B}(t)}$.

(c15) For each $E \in \mathcal{E}, A \in \mathcal{A}$ such that $\text{KEY}(E) = A$, then the same semantic equation of c11 is true – i.e., a key is a snapshot attribute – and $\forall a \in \Delta_D^{\mathcal{B}}.\ \#\{e \in E^{\mathcal{B}(t)} \mid \langle e, a \rangle \in A^{\mathcal{B}(t)}\} \leq 1$.

(c16) For each $E_s, E_t \in \mathcal{E}$, then:
1. If E_s DEX E_t, then, $e \in E_t^{\mathcal{B}(t)} \rightarrow \exists t_1 < t.\ e \in E_s^{\mathcal{B}(t_1)}$;
2. If E_s DEV E_t, then, $e \in E_t^{\mathcal{B}(t)} \rightarrow (e \notin E_s^{\mathcal{B}(t)} \wedge \exists t_1 < t.\ e \in E_s^{\mathcal{B}(t_1)})$.

(c17) For each $R \in$ GEN with $\text{REL}(R) = \langle \texttt{source} : E_s, \texttt{target} : E_t \rangle$, then, $\langle e_s, e_t \rangle \in R^{\mathcal{B}(t)} \rightarrow (e_s \in E_s^{\mathcal{B}(t)} \wedge \forall t_1 > t.\ e_s \notin E_s^{\mathcal{B}(t_1)}) \wedge (e_t \in E_t^{\mathcal{B}(t)} \wedge \forall t_2 < t.\ e_t \notin E_t^{\mathcal{B}(t_2)})$.

Example 1. The various components of a schema are now illustrated with respect to the schema of Fig. 1. We start by showing the alphabet of the example schema. The sets of snapshot entities and relationships are:

$\mathcal{E}^S = \{\texttt{Employee}, \texttt{Department}, \texttt{OrganisationalUnit}\}$,
$\mathcal{R}^S = \{\texttt{Resp-for}\}$

The sets of temporary entities and relationships are:

$\mathcal{E}^T = \{\texttt{Manager}\}, \qquad \mathcal{R}^T = \{\texttt{Works-for}\}$

The set of un-marked entities and relationships are:

$\mathcal{E}^I = \{\texttt{AreaManager}, \texttt{TopManager}, \texttt{OrganisationalUnit}, \texttt{InterestGroup}\}$
$\mathcal{R}^I = \{\texttt{Manages}\}$

The meaning of temporary entities and relationships is such that instances of temporary entities and relationships always have a limited lifespan. On

the other hand, the set of instances of snapshot entities and relationships never change along time. For un-marked entities and relationships no temporal constraint holds – i.e., the set of instances of un-marked entities and relationships can contain either instances with a limited lifespan or instances with unlimited lifespan (both in the past and in the future). It has to be noted that, in order to capture the semantics of a *legacy* diagram each of its elements should be necessarily snapshot marked. This pre-processing step is necessary to enforce the upward compatibility in $\mathcal{ER}_{VT}$ when *legacy* diagrams are included in a temporal model.

The function ATT describes the attributes of an entity, together with their types, e.g.:

$$\text{ATT}(\texttt{Employee}) = \langle \texttt{PaySlipNumber} : \texttt{Integer}, \texttt{Salary} : \texttt{Integer}, \texttt{Name} : \texttt{String} \rangle$$

The function REL associates a name or, alternatively, a position number – in both cases called *role* – to each argument of a relationship, and for each role an entity describing its type, e.g.:

$$\text{REL}(\texttt{Manages}) = \langle \texttt{man} : \texttt{TopManager}, \texttt{prj} : \texttt{Project} \rangle$$

describes `Manages` as a binary relationships where a `TopManager` manages a `Project`.

$\mathcal{ER}_{VT}$ distinguishes between *snapshot participation constraints* (CARD, true at each point in time) and *lifespan participation constraints* (CARD^L, evaluated during the entire existence of the entity), e.g.:

$$\text{CARD}(\texttt{TopManager}, \texttt{Manages}, \texttt{man}) = \langle 1, 1 \rangle$$
$$\text{CARD}^L(\texttt{TopManager}, \texttt{Manages}, \texttt{man}) = \langle 1, 5 \rangle$$

state that a `TopManager` should manage at most 5 different projects in his entire existence while still being constrained in managing exactly one project at a time. Notice that, since for snapshot relationships the set of instances does not change in time, there is no difference between snapshot and lifespan participation constraints.

ISA, COVER and DISJ are used for representing generalised hierarchies. ISA models the sub-entity relationship, e.g., `Manager` ISA `Employee` says that manager is a sub-entity of employee. COVER models the fact that a set of sub-entities may have common instances but each instance of the super-entity belongs to at least one of those sub-entities, e.g.:

$$\{\texttt{AreaManager}, \texttt{TopManager}\} \text{ COVER } \texttt{Manager}$$

DISJ models disjoint hierarchies, e.g.:

$$\{\texttt{Department}, \texttt{InterestGroup}\} \text{ DISJ } \texttt{OrganisationalUnit}$$

says that department is disjoint from interest group and both are sub-entities of organisational units. By using both relations we can model disjoint and covering hierarchies, e.g.:

$\{\texttt{Department}, \texttt{InterestGroup}\}$ DISJ $\texttt{OrganisationalUnit}$
$\{\texttt{Department}, \texttt{InterestGroup}\}$ COVER $\texttt{OrganisationalUnit}$.

At different points in time, an entity may have different values for the same attribute. The relations S, T model snapshot and temporary attributes, respectively, e.g.:

$\langle \texttt{Employee}, \texttt{Name} \rangle \in \text{S}, \qquad \langle \texttt{Employee}, \texttt{Salary} \rangle \in \text{T}$

state that whenever an employee has a name this value will globally persist over time. On the other hand, a salary will (hopefully) change over time. In $\mathcal{ER}_{VT}$ the timestamp for an entity is independent from the timestamp of its attributes respecting the orthogonality principle. Indeed, it is consistent to have both snapshot attributes of a temporary entity, and temporary attributes of a snapshot entity. In the former case, the $\mathcal{ER}_{VT}$ semantics says that during the (limited) lifespan of an entity the value of a snapshot attribute never changes. In the latter one, the meaning is that each instance always belongs to the snapshot entity, but the value of the temporary attribute changes during its existence. In our running example, where `Employee` is a snapshot entity, `Salary` is modelled as a temporary attribute – i.e., the salary of an employee will change – while its `Name` persists over time and is represented as a snapshot attribute. Furthermore, the temporal behaviour of an attribute is only specified locally to an entity, i.e., the same attribute associated to two different entities can have a different temporal behaviour – e.g., the phone number of a department is not supposed to change over time and should be modelled as a snapshot attribute, while the phone number of an employee can change and should be modelled as a temporary attribute.

The function KEY captures keys for entities, e.g., the fact that the pay slip number is a key for an employee is captured by:

$\text{KEY}(\texttt{Employee}) = \texttt{PaySlipNumber}$

$\mathcal{ER}_{VT}$ models keys as snapshot attributes that uniquely identifies instances of an entity.

$\mathcal{ER}_{VT}$ can model *transition* relationships between entities whose instances may migrate from one entity to the other; this is called *object migration* from a source to a target entity. Both *dynamic evolution* – when an object ceases to be an instance of a source entity – and *dynamic extension* – when an object continues to belong to the source – can be represented in $\mathcal{ER}_{VT}$. That a top manager was an area manager sometime in the past is captured by the dynamic extension:

$\texttt{AreaManager}$ DEX $\texttt{TopManager}$

$\mathcal{ER}_{VT}$ is also able to capture *generation* relationships that generate new instances in the target entity starting from instances that are in the source entity. That oranges are transformed into orange juice – as in the diagram of Fig. 2 – is modelled by:

$\text{REL}(\texttt{Generate}) = \langle \texttt{source} : \texttt{Orange}, \texttt{target} : \texttt{Juice} \rangle$
$\texttt{Generate} \in \text{GEN}$.

Lifespan cardinality constraints associated to generation relationships are expressed trough CARD^L.

4.1 Reasoning Problems

Reasoning tasks over a temporal conceptual model include verifying whether an entity, relationship or a schema are *satisfiable*, or checking whether a new schema property is *logically implied* by a given schema. The model theoretic semantics associated to $\mathcal{ER}_{VT}$ allow us to formally define these reasoning tasks. This Chapter not only presents the first complete formalisation of a temporal extended ER data model, but it introduces for the first time a formal characterisation of all the various inferences on diagrams expressed in such a data model.

Definition 3 (Reasoning in $\mathcal{ER}_{VT}$). Let Σ an $\mathcal{ER}_{VT}$ schema, $E \in \mathcal{E}$ an entity and $R \in \mathcal{R}$ a relationship. The following are the reasoning tasks over Σ:

1. E (R) is *satisfiable* if there exists a legal temporal database state $\mathcal{B}$ for Σ such that $E^{\mathcal{B}(t)} \neq \emptyset$ ($R^{\mathcal{B}(t)} \neq \emptyset$), for some $t \in \mathcal{T}$;
2. E (R) is *liveness satisfiable* if there exists a legal temporal database state $\mathcal{B}$ for Σ such that $\forall t \in \mathcal{T}.\ \exists t' > t.\ E^{\mathcal{B}(t')} \neq \emptyset$ ($R^{\mathcal{B}(t')} \neq \emptyset$), i.e., E (R) is satisfiable infinitely often;
3. E (R) is *globally satisfiable* if there exists a legal temporal database state $\mathcal{B}$ for Σ such that $E^{\mathcal{B}(t)} \neq \emptyset$ ($R^{\mathcal{B}(t)} \neq \emptyset$), for all $t \in \mathcal{T}$;
4. Σ is *satisfiable* if there exists a legal temporal database state $\mathcal{B}$ for Σ[4];
5. A schema Σ' is *logically implied* by a schema Σ over the same signature if every legal temporal database state for Σ is also a legal temporal database state for Σ'.

Please note that the above definition of logical implication accounts for checking properties of a schema whenever they are expressible in the $\mathcal{ER}_{VT}$ schema language (see Definition 1). In particular, checking whether an entity E is satisfiable can be reduced to logical implication. Indeed, by choosing $\Sigma' = \{E \text{ ISA } A, E \text{ ISA } B, \{A, B\} \text{ DISJ } C\}$, with A, B, C arbitrary entities, then E is satisfiable iff $\Sigma \not\models \Sigma'$. Furthermore, given two entities (relationships) E_1, E_2 (R_1, R_2), checking for sub-entity (sub-relationship) can be reduced to logical implication by choosing $\Sigma' = \{E_1 \text{ ISA } E_2\}$ ($\Sigma' = \{R_1 \text{ ISA } R_2\}$).

[4] For a schema to be satisfiable the constraints expressed by the schema should hold true at *all* points in time—as from the definition of legal temporal database state. Therefore, there is no distinction between satisfiability, liveness satisfiability or global satisfiability for schemas.

The following 'classical' desirable features found in the literature in temporal conceptual modelling come as almost trivial logical implications in our framework.

Proposition 1. *In every $\mathcal{ER}_{VT}$ schema the following temporal properties hold:*

1. *Sub-entities of temporary entities are also temporary.*
2. *Sub-entities of snapshot entities, and super-entities of temporary or un-marked entities can be either snapshot, temporary or un-marked entities.*
3. *A schema is inconsistent if exactly one of a whole set of snapshot partitioning sub-entities is temporary.*
4. *Participants of snapshot relations are either snapshot or un-marked entities. They are snapshot when they participate at least once in the relationship.*
5. *Participants of temporary or un-marked relations can be either snapshot, temporary or un-marked entities.*
6. *A relationship is temporary if one of the participating entities is temporary.*
7. *The temporal behaviour for an entity is independent from that of its attributes.*

Points 1 *and* 2 *are true also for relationships.*

Example 2. From the $\mathcal{ER}_{VT}$ diagram in Fig. 1 the following logical implications hold:

1. Since `Manager` is a temporary entity then both sub-entities `AreaManager` and `TopManager` are temporary entities, constraining either `AreaManager` or `TopManager` as snapshot entities would lead to a contradiction.
2. Even if `Employee` is a snapshot entity it is consistent to have `Manager` – a temporary entity – as a sub-entity of `Employee`.
3. The fact that `InterestGroup` is a snapshot entity follows logically from our theory.
4. Since `Project` participates at least once in the snapshot relationship `Resp-For` it must be a snapshot entity.
5. The un-marked relationship `Manages` is consistent even if the snapshot entity `Project` participates in the relationship.
6. The fact that `Manages` is a temporary relationship follows logically from our theory since the temporary entity `TopManager` participates in the relationship.

4.2 Encoding in Description Logics

In this Section we show how the temporal description logic $\mathcal{ALCQI}_{\mathcal{US}}$ is able to capture temporal conceptual schemas expressed in $\mathcal{ER}_{VT}$. This characterisation allows us to support the reasoning on temporal conceptual models

as in Definition 3 by using the reasoning services of $\mathcal{ALCQI}_{\mathcal{US}}$. The correspondence is based on a mapping function Φ – extending the one introduced by [12,13] for non-temporal ER models – from $\mathcal{ER}_{VT}$ schemas to $\mathcal{ALCQI}_{\mathcal{US}}$ knowledge bases.

Informally, the encoding works as follows. Both entity and relationship symbols in the $\mathcal{ER}_{VT}$ diagram are mapped into $\mathcal{ALCQI}_{\mathcal{US}}$ concept names (i.e., relationships are reified). Domain symbols are mapped into additional concept names, pairwise disjoint. Both attributes of entities and roles of relationships are mapped to role names in $\mathcal{ALCQI}_{\mathcal{US}}$ with number restrictions stating the single-valuedness[5]. ISA links between entities or between relationships are mapped using terminological axioms. Generalised hierarchies with disjointness and covering constraints can be captured using the Boolean connectives. Cardinality constraints are mapped using the number restriction quantifiers in $\mathcal{ALCQI}_{\mathcal{US}}$. Temporal properties in $\mathcal{ER}_{VT}$ are mapped using the temporal operators in $\mathcal{ALCQI}_{\mathcal{US}}$.

Definition 4 (Mapping $\mathcal{ER}_{VT}$ into $\mathcal{ALCQI}_{\mathcal{US}}$). Let $\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}, \text{CARD}^L, \text{ISA}, \text{DISJ}, \text{COVER,S,T}, \text{KEY}, \text{DEX}, \text{DEV}, \text{GEN})$ be an $\mathcal{ER}_{VT}$ schema. The $\mathcal{ALCQI}_{\mathcal{US}}$ knowledge base $\Phi(\Sigma) = (CN, RN, \Gamma)$ is defined as follows. The set CN of concept names is such that:

- For each domain symbol $D \in \mathcal{D}$ then $\Phi(D) \in CN$;
- For each entity symbol $E \in \mathcal{E}$ then $\Phi(E) \in CN$.
- For each relationship $R \in \mathcal{R}$ then $\Phi(R) \in CN$;

The set RN of atomic roles is such that:

- For each attribute $A \in \mathcal{A}$ then $\Phi(A) \in RN$.
- For each role symbol $U \in \mathcal{U}$ then $\Phi(U) \in RN$;

Φ is functional over $\mathcal{D} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{U} \cup \mathcal{A}$. The set Γ contains the following $\mathcal{ALCQI}_{\mathcal{US}}$ axioms – i.e., *temporal integrity constraints.*

`(ax1)` For each $D_i \in \mathcal{D}$:
$\Phi(D_i) \sqsubseteq (\Box^+\Phi(D_i)) \sqcap (\Box^-\Phi(D_i))$ – i.e., $\Phi(D_i) \doteq \Box^*\Phi(D_i)$.
`(ax2)` For each relationship $R \in \mathcal{R}$ such that $\text{REL}(R) = \langle U_1 : E_1, \ldots, U_k : E_k \rangle$:
$\Phi(R) \sqsubseteq \forall\Phi(U_1).\Phi(E_1) \sqcap \ldots \sqcap \forall\Phi(U_k).\Phi(E_k) \sqcap$
$(=1\ \Phi(U_1)) \sqcap \ldots \sqcap (=1\ \Phi(U_k))$
`(ax3)` For each entity $E \in \mathcal{E}$ such that $\text{ATT}(E) = \langle A_1 : D_1, \ldots, A_h : D_h \rangle$:
$\Phi(E) \sqsubseteq \forall\Phi(A_1).\Phi(D_1) \sqcap \ldots \sqcap \forall\Phi(A_h).\Phi(D_h) \sqcap$
$(=1\ \Phi(A_1)) \sqcap \ldots \sqcap (=1\ \Phi(A_h))$
`(ax4)` For each role symbol $U_i \in \mathcal{U}$ between $R \in \mathcal{R}$ and $E \in \mathcal{E}$:
a. If $m = \text{CMIN}(E, R, U_i) \neq 0$: $\Phi(E) \sqsubseteq (\geqslant m\ (\Phi(U_i))^-.\ \Phi(R))$;
b. If $n = \text{CMAX}(E, R, U_i) \neq \infty$: $\Phi(E) \sqsubseteq (\leqslant n\ (\Phi(U_i))^-.\ \Phi(R))$.

[5] Multi-valued attributes can be easily captured by eliminating such cardinality constraint.

(ax5) For each role symbol $U_i \in \mathcal{U}$ between $R \in \mathcal{R}$ and $E \in \mathcal{E}$:
a. If $m = \text{CMIN}^L(E, R, U_i) \neq 0$: $\Phi(E) \sqsubseteq (\geqslant m\ (\Phi(U_i))^-.\ \Diamond^*\Phi(R))$;
b. If $n = \text{CMAX}^L(E, R, U_i) \neq \infty$: $\Phi(E) \sqsubseteq (\leqslant n\ (\Phi(U_i))^-.\ \Diamond^*\Phi(R))$.
(ax6) For each pair of entities (relationships) $E_1, E_2 \in \mathcal{E}$ ($R_1, R_2 \in \mathcal{R}$) such that E_1 ISA E_2 (R_1 ISA R_2):
a. $\Phi(E_1) \sqsubseteq \Phi(E_2)$;
b. $\Phi(R_1) \sqsubseteq \Phi(R_2)$.
(ax7) For each snapshot entity $E \in \mathcal{E}^S$:
$\Phi(E) \sqsubseteq (\Box^+\Phi(E)) \sqcap (\Box^-\Phi(E))$ – i.e., $\Phi(E) \doteq \Box^*\Phi(E)$.
(ax8) For each snapshot relationship $R \in \mathcal{R}^S$:
$\Phi(R) \sqsubseteq (\Box^+\Phi(R)) \sqcap (\Box^-\Phi(R))$ – i.e., $\Phi(R) \doteq \Box^*\Phi(R)$;
$\Phi(R) \sqsubseteq (=1\ \Box^*\Phi(U_1)) \sqcap \ldots \sqcap (=1\ \Box^*\Phi(U_k))$.
(ax9) For each snapshot attribute A_i with $\langle E, A_i\rangle \in$ S:
$\Phi(E) \sqsubseteq (=1\ \Box^*\Phi(A_i))$.
(ax10) For each temporary entity $E \in \mathcal{E}^T$:
$\Phi(E) \sqsubseteq (\Diamond^+\neg\Phi(E)) \sqcup (\Diamond^-\neg\Phi(E))$.
(ax11) For each temporary relationship $R \in \mathcal{R}^T$:
$\Phi(R) \sqsubseteq (\Diamond^+\neg\Phi(R)) \sqcup (\Diamond^-\neg\Phi(R)) \sqcup \neg((=1\ \Box^*\Phi(U_1)) \sqcap \ldots \sqcap (=1\ \Box^*\Phi(U_k)))$.
(ax12) For each temporary attribute A_i with $\langle E, A_i\rangle \in$ T:
$\Phi(E)\rangle \sqsubseteq \neg(=1\ \Box^*\Phi(A_i))$.
(ax13) For each pair of symbols X_1, X_2 such that one of the following is true:
1. $X_1 \in \mathcal{D}$, $X_2 \in \mathcal{E} \cup \mathcal{D}$, $X_1 \neq X_2$;
2. $X_1 \in \mathcal{R}$, $X_2 \in \mathcal{E} \cup \mathcal{R}$, X_1, X_2 with different arity.

$\Phi(X_1) \sqsubseteq \neg\Phi(X_2)$.
(ax14) For $E, E_1, \ldots, E_n \in \mathcal{E}$, if $\{E_1, \ldots, E_n\}$ DISJ E:
$E_1 \sqsubseteq E \sqcap \neg E_2 \sqcap \ldots \sqcap \neg E_n$
$E_2 \sqsubseteq E \sqcap \neg E_3 \sqcap \ldots \sqcap \neg E_n$
$\ldots$
$E_n \sqsubseteq E$.
(ax15) For $E, E_1, \ldots, E_n \in \mathcal{E}$, if $\{E_1, \ldots, E_n\}$ COVER E:
$E_1 \sqsubseteq E$
$\ldots$
$E_n \sqsubseteq E$
$E \sqsubseteq E_1 \sqcup \ldots \sqcup E_n$.
(ax16) For each key attribute A with KEY$(E) = A$:
$\Phi(E) \sqsubseteq (=1\ \Box^*\Phi(A))$
$\top \sqsubseteq (\leqslant 1\ (\Phi(A))^-.\ \Phi(E))$.
(ax17) For each $E_s, E_t \in \mathcal{E}$ if E_s DEX E_t then:
$E_t \sqsubseteq \Diamond^- E_s$.
(ax18) For each $E_s, E_t \in \mathcal{E}$ if E_s DEV E_t then:
$E_t \sqsubseteq \neg E_s \sqcap \Diamond^- E_s$.
(ax19) For each $R \in$ GEN, then:
$R \sqsubseteq \forall \texttt{source}.\, (E_s \sqcap \Box^+\neg E_s) \sqcap \forall \texttt{target}.\, (E_t \sqcap \Box^-\neg E_t)$.

To prove that reasoning on $\mathcal{ER}_{VT}$ schemas can be done by reasoning on their $\mathcal{ALCQI}_{\mathcal{US}}$ translation we need to prove the correctness of the encoding. The encoding that makes use of the $\mathcal{DLR}_{\mathcal{US}}$ description logic – a DL with n-ary relations – has been proven correct in [2] by establishing a precise correspondence between legal database states of $\mathcal{ER}_{VT}$ schemas and models of the corresponding $\mathcal{DLR}_{\mathcal{US}}$ theories. The same result holds true for the $\mathcal{ALCQI}_{\mathcal{US}}$ mapping.

Proposition 2 (Correctness of the encoding). *Let the schema Σ be an $\mathcal{ER}_{VT}$ schema. Then, Σ admits a legal database state if and only if the corresponding $\mathcal{ALCQI}_{\mathcal{US}}$ knowledge base $\Phi(\Sigma)$ has a model.*

The following theorem that reduces reasoning over $\mathcal{ER}_{VT}$ schemas to reasoning over $\mathcal{ALCQI}_{\mathcal{US}}$ knowledge bases is a direct consequence of the proposition 2.

Theorem 1 (Reasoning over $\mathcal{ER}_{VT}$ schemas). *Let the schema Σ be an $\mathcal{ER}_{VT}$ schema, $E \in \mathcal{E}$ an entity, $R \in \mathcal{R}$ a relationship, and $\Phi(\Sigma)$ the $\mathcal{ALCQI}_{\mathcal{US}}$ knowledge base corresponding to Σ. The following holds:*

1. *E or R is satisfiable iff, respectively*
   $\Phi(\Sigma) \not\models \Phi(E) \sqsubseteq \bot$;
   $\Phi(\Sigma) \not\models \Phi(R) \sqsubseteq \bot$;
2. *Σ is satisfiable iff $\Phi(\Sigma)$ is satisfiable, i.e., $\Phi(\Sigma) \not\models \top \sqsubseteq \bot$;*
3. *A schema Σ' is logically implied by a schema Σ over the same signature iff $\Phi(\Sigma) \models \bigwedge \Phi(\Sigma')$; where $\bigwedge \Phi(\Sigma')$ stands for the conjunction of all axioms in $\Phi(\Sigma')$.*

The above theorem reduces both satisfiability and logical implication in $\mathcal{ER}_{VT}$ to logical implication in $\mathcal{ALCQI}_{\mathcal{US}}$. On the other hand, neither global nor liveness satisfiability can be captured by $\mathcal{ALCQI}_{\mathcal{US}}$ axioms as presented in this Chapter. Indeeed, a more expressive axiom language is required to capture these reasoning tasks, i.e., a language where complex axioms can be formed using full boolean and tempoaral operators (see [2,4]).

Important complexity results can be proved if we consider the fragment of $\mathcal{ER}_{VT}$ that can be encoded in $\mathcal{ALCQI}^{-}_{\mathcal{US}}$, namely the fragment without snapshot relations and attributes, temporal keys, and lifespan cardinalities. In [4] an EXPTIME upper bound was proved for $\mathcal{ALCQI}^{-}_{\mathcal{US}}$: therefore, we can conclude that the same complexity upper bounds holds for $\mathcal{ER}_{VT}$. Moreover, in [10] an EXPTIME complexity lower bound was proved for reasoning in non-temporal UML/ER diagrams, which are expressible in the non-temporal part of $\mathcal{ER}_{VT}$; therefore, we can conclude that reasoning in this fragment of $\mathcal{ER}_{VT}$ is then EXPTIME-complete. We conjecture in [2] that the fragment of $\mathcal{ER}_{VT}$ including the standard non-temporal EER data model, and temporally enhanced with the capability to express temporal behaviour for both entities and relationships, and dynamic constraints is still undecidable.

Example 3. We now show how the schema of the running example of Fig. 1 is translated. Let Σ^{ex} be the schema associated to the diagram of Fig. 1 as illustrated in Example 1, and $\Phi(\Sigma^{ex}) = (CN^{ex}, RN^{ex}, \Gamma^{ex})$ be its $\mathcal{ALCQI}_{\mathcal{US}}$ translation. The alphabet of $\Phi(\Sigma^{ex})$ is such that:

$$\begin{aligned} CN^{ex} = \{&\texttt{Employee}, \texttt{Manager}, \texttt{AreaManager}, \texttt{TopManager}, \texttt{Department}, \\ &\texttt{OrganisationalUnit}, \texttt{InterestGroup}, \texttt{Project}, \texttt{Integer}, \texttt{String}\} \\ RN^{ex} = \{&\texttt{Manages}, \texttt{Works-for}, \texttt{Resp-for}, \texttt{PaySlipNumber}, \texttt{Salary}, \texttt{Name}, \\ &\texttt{ProjectCode}, \texttt{emp}, \texttt{act}, \texttt{prj}, \texttt{man}, \texttt{org}\} \end{aligned}$$

$\mathcal{ALCQI}$ – the non temporal fragment of $\mathcal{ALCQI}_{\mathcal{US}}$ – is enough to translate entities, relationships, attributes, generalised hierarchies and snapshot cardinality constraints. Entities together with their attributes in Σ^{ex} give rise to the following terminological axioms:

$$\begin{aligned} \texttt{Employee} \sqsubseteq\ & \forall \texttt{PaySlipNumber}.\, \texttt{Integer} \sqcap (=1\ \texttt{PaySlipNumber}) \sqcap \\ & \forall \texttt{Salary}.\, \texttt{Integer} \sqcap (=1\ \texttt{Salary}) \sqcap \\ & \forall \texttt{Name}.\, \texttt{String} \sqcap (=1\ \texttt{Name}) \\ \texttt{Project} \sqsubseteq\ & \forall \texttt{ProjectCode}.\, \texttt{String} \sqcap (=1\ \texttt{ProjectCode}). \end{aligned}$$

Relationships in Σ^{ex} give rise to the following axioms:

$$\begin{aligned} \texttt{Works-for} \sqsubseteq\ & \forall \texttt{emp}.\, \texttt{Employee} \sqcap (=1\ \texttt{emp}) \sqcap \forall \texttt{act}.\, \texttt{Project} \sqcap (=1\ \texttt{act}) \\ \texttt{Manages} \sqsubseteq\ & \forall \texttt{man}.\, \texttt{TopManager} \sqcap (=1\ \texttt{man}) \sqcap \forall \texttt{prj}.\, \texttt{Project} \sqcap (=1\ \texttt{prj}) \\ \texttt{Resp-for} \sqsubseteq\ & \forall \texttt{prj}.\, \texttt{Project} \sqcap (=1\ \texttt{prj}) \sqcap \\ & \forall \texttt{org}.\, \texttt{OrganisationalUnit} \sqcap (=1\ \texttt{org}) \end{aligned}$$

Notice that for the mapping to be correct the appropriate disjointness axioms (ax13) have to be considered.

Generalised hierarchies in Σ^{ex} give rise to the following axioms:

$$\begin{aligned} \texttt{Manager} &\sqsubseteq \texttt{Employee} \sqcap (\texttt{AreaManager} \sqcup \texttt{TopManager}) \\ \texttt{AreaManager} &\sqsubseteq \texttt{Manager} \\ \texttt{TopManager} &\sqsubseteq \texttt{Manager} \\ \texttt{OrganisationalUnit} &\sqsubseteq \texttt{Department} \sqcup \texttt{InterestGroup} \\ \texttt{Department} &\sqsubseteq \texttt{OrganisationalUnit} \sqcap \neg \texttt{InterestGroup} \\ \texttt{InterestGroup} &\sqsubseteq \texttt{OrganisationalUnit} \end{aligned}$$

Snapshot cardinality constraints in Σ^{ex} give rise to the following axioms:

$$\begin{aligned} \texttt{Project} \sqsubseteq\ & (\geqslant 1\ \texttt{act}^{-}.\, \texttt{Works-for}) \sqcap (\geqslant 1\ \texttt{prj}^{-}.\, \texttt{Resp-for}) \sqcap \\ & (=1\ \texttt{prj}^{-}.\, \texttt{Manages}) \\ \texttt{TopManager} \sqsubseteq\ & (=1\ \texttt{man}^{-}.\, \texttt{Manages}) \end{aligned}$$

Let's illustrate now the mapping of the temporal constructs. $\mathcal{ALCQI}_{\mathcal{US}}$ axioms can enforce either that entities (relationships) cannot last forever, or that their extension never changes in time. Snapshot entities and relationships in Σ^{ex} give rise to the following axioms:

$$\begin{aligned} \texttt{Employee} &\sqsubseteq (\Box^{+}\texttt{Employee}) \sqcap (\Box^{-}\texttt{Employee}) \\ \texttt{Department} &\sqsubseteq (\Box^{+}\texttt{Department}) \sqcap (\Box^{-}\texttt{Department}) \\ \texttt{Resp-for} &\sqsubseteq (\Box^{+}\texttt{Resp-for}) \sqcap (\Box^{-}\texttt{Resp-for}) \\ \texttt{Resp-for} &\sqsubseteq (=1\ \Box^{*}\texttt{prj}) \sqcap (=1\ \Box^{*}\texttt{org}) \end{aligned}$$

Temporary entities and relationships in Σ^{ex} gives rise to the following axioms:

$$\begin{aligned}
\texttt{Manager} &\sqsubseteq (\Diamond^{+}\neg\texttt{Manager}) \sqcup (\Diamond^{-}\neg\texttt{Manager}) \\
\texttt{Works-for} &\sqsubseteq (\Diamond^{+}\neg\texttt{Works-for}) \sqcup (\Diamond^{-}\neg\texttt{Works-for}) \sqcup \\
&\quad \neg((=1\ \Box^{*}\texttt{emp}) \sqcap (=1\ \Box^{*}\texttt{act}))
\end{aligned}$$

Let's now show how the temporal properties of attributes in Σ^{ex} are translated. The following axioms:

$$\begin{aligned}
&\texttt{Employee} \sqsubseteq (=1\ \Box^{*}\texttt{Name}) \\
&\texttt{Employee} \sqsubseteq \neg(=1\ \Box^{*}\texttt{Salary})
\end{aligned}$$

state that the name of an employee does not change – it is a snapshot attribute – while its salary is a temporary attribute.

Key constraints in Σ^{ex} give rise to the following axioms:

$$\begin{aligned}
\texttt{Employee} &\sqsubseteq (=1\ \Box^{*}\texttt{PaySlipNumber}) \\
\top &\sqsubseteq (\leqslant 1\ \texttt{PaySlipNumber}^{-}.\ \texttt{Employee}) \\
\texttt{Project} &\sqsubseteq (=1\ \Box^{*}\texttt{ProjectCode}) \\
\top &\sqsubseteq (\leqslant 1\ \texttt{ProjectCode}^{-}.\ \texttt{Project})
\end{aligned}$$

Notice that in our approach only single-attribute keys are captured. The case of full key dependencies in the a-temporal $\mathcal{ALCQI}$ language as been recently solved [11] and its temporal extension has to be investigated yet.

Lifespan cardinality constraints in Σ^{ex} give rise to the following axiom:

$$\texttt{TopManager} \sqsubseteq (\geqslant 1\ \texttt{man}^{-}.\ \Diamond^{*}\texttt{Manages}) \sqcap (\leqslant 5\ \texttt{man}^{-}.\ \Diamond^{*}\texttt{Manages})$$

i.e., a top manager should manage at least 1 and at most 5 different projects in his entire existence as a top manager.

We conclude by showing how $\mathcal{ALCQI}_{\mathcal{US}}$ captures dynamic relationships between the entities of the schema. The dynamic extension between being an area manager and a top manager gives rise to the following axiom:

$$\texttt{TopManager} \sqsubseteq \Diamond^{-}\texttt{AreaManager}$$

The generation relationships as in Fig. 2 gives rise to the following axiom:

$$\begin{aligned}
\texttt{Generate} \sqsubseteq\ & \forall\texttt{source}.\,(\texttt{Orange} \sqcap \Box^{+}\neg\texttt{Orange}) \sqcap \\
& \forall\texttt{target}.\,(\texttt{Juice} \sqcap \Box^{-}\neg\texttt{Juice})
\end{aligned}$$

Furthermore, to fully capture Fig. 2 the lifespan cardinality constraints need to be mapped:

$$\begin{aligned}
&\texttt{Orange} \sqsubseteq (=1\ \texttt{source}^{-}.\ \Diamond^{*}\texttt{Generate}) \\
&\texttt{Juice} \sqsubseteq (=5\ \texttt{target}^{-}.\ \Diamond^{*}\texttt{Generate})
\end{aligned}$$

5 An Object-Oriented Data Model for Evolving Schemas

As anticipated in the introduction, the conceptual data model for evolving schemas we propose is based on a formal framework consisting of:

- an extended object-oriented model which supports a standard notion of evolving schemas (equipped with all the usually adopted schema change operators) for which a semantics is provided;
- a collection of interesting reasoning tasks, in order to support the design and the management of an evolving schema;
- an encoding, which has been proved correct, in the description logic $\mathcal{ALCQI}$, which can then be used to solve the tasks defined for the schema evolution.

The object-oriented model we propose allows for the representation of multiple schema versions. It is based on an expressive version of the "snapshot" – i.e., single-schema – object-oriented model introduced by [1] and further extended and elaborated in its relationships with Description Logics by [12,13]. The language embodies the features of the static parts of UML and ODMG and, therefore, it does not take into account those aspects related to the definition of methods.

The definition of an evolving schema $\mathcal{S}$ is based on a set of class and attribute names ($\mathcal{C}_\mathcal{S}$ and $\mathcal{A}_\mathcal{S}$ respectively) and includes a partially ordered set of schema versions. The initial schema version of $\mathcal{S}$ contains a set of class definitions having one of the following forms:

Class C is-a $C_1,\ldots,C_h$ disjoint $C_{h+1},\ldots,C_k$ not-is-a $C_{k+1},\ldots,C_n$ type-is T.
View-class C type-is T.

A class definition introduces just necessary conditions regarding the type of the class – this is the standard case in object-oriented data models – while views are defined by means of both necessary and sufficient conditions. The symbol T denotes a type expression built according to the following syntax:

$$
\begin{array}{lll}
T \rightarrow C \ \mid & & \\
\quad \text{Union } T_1,\ldots,T_k \text{ End} \ \mid & & \text{(union type)} \\
\quad \text{Set-of [m,n] } T \ \mid & & \text{(set type)} \\
\quad \text{Record } A_1{:}T_1,\ldots,A_k{:}T_k \text{ End} . & & \text{(record type)}
\end{array}
$$

where $C \in \mathcal{C}_\mathcal{S}$, $A_i \in \mathcal{A}_\mathcal{S}$, and [m,n] denotes an optional cardinality constraint.

A schema version in $\mathcal{S}$ is defined by the application of a sequence of schema changes to a previous schema version. The schema change taxonomy is built by combining the model elements which are subject to change with

the elementary modifications, add, drop and change, they undergo. We introduce only a basic set of *elementary schema change operators* including the standard ones found in the literature[6], see, e.g., [6]:

Add-attribute C, A, T
Drop-attribute C, A
Change-attr-name C, A, A'
Change-attr-type C, A, T'
Add-class C, T
Drop-class C
Change-class-name C, C'
Change-class-type C, T'
Add-is-a C, C'
Drop-is-a C, C'

We are now ready to define the syntax of our object-oriented data model for evolving schemas:

Definition 5 (Evolving Schema). An evolving object-oriented schema is a tuple $\mathcal{S} = (\mathcal{C}_\mathcal{S}, \mathcal{A}_\mathcal{S}, \mathcal{SV}_0, \mathcal{M}_\mathcal{S})$, where:

- $\mathcal{C}_\mathcal{S}$ is a finite set of class names;
- $\mathcal{A}_\mathcal{S}$ is a finite set of attribute names;
- $\mathcal{SV}_0$ is the initial schema version, which includes class and view definitions for some $C \in \mathcal{C}_\mathcal{S}$;
- $\mathcal{M}_\mathcal{S}$ is a set of modifications $\mathcal{M}_{ij}$, where i, j denote a pair of version coordinates, inducing a tree partial order $\leq_\mathcal{S}$ over a finite and discrete set of schema versions $\mathcal{SV}$ with minimal element $\mathcal{SV}_0$ (i.e., every non-initial schema version has exactly one immediate predecessor in the partial order). Each modification is a finite sequence of elementary schema changes.

Notice that we omit the definition of a schema version coordinate mechanism and simply reference distinct schema versions by means of different subscripts. Any kind of versioning dimension usually considered in the literature could actually be employed – such as transaction time, valid time and symbolic labels – provided that a suitable mapping between version coordinates and index values is defined.

According to the above definition, $\mathcal{SV}_0$ precedes every other schema version, and a schema version $\mathcal{SV}_j$ represents the outcome of the application of $\mathcal{M}_{ij}$ to $\mathcal{SV}_i$. $\mathcal{S}$ is called *elementary* if every $\mathcal{M}_{ij}$ in $\mathcal{M}_\mathcal{S}$ contains only one elementary modification; in the following we will consider only elementary evolving schemas.

Definition 6. Let $\mathcal{S}$ be schema, we introduce the following notation:

[6] It is not difficult to consider the complete set of operators with respect to the constructs of the data model.

- The linear sequence of modifications $\mathcal{M}_{\mathcal{S}_{i\downarrow j}}$ which leads to the schema version $\mathcal{SV}_j$ starting from $\mathcal{SV}_i$, is the smallest sequence satisfying the following inductive definition:
  - $\mathcal{M}_{\mathcal{S}_{i\downarrow i}} = \emptyset$,
  - $\mathcal{M}_{\mathcal{S}_{i\downarrow j}} = \mathcal{M}_{\mathcal{S}_{i\downarrow k}}, \mathcal{M}_{kj}$, $j > i$, and $\mathcal{M}_{kj} \in \mathcal{M}_{\mathcal{S}}$.
- The j-th schema version $\mathcal{SV}_j$ is constituted by the initial set of class definitions $\mathcal{SV}_0$ and the sequence of schema modifications $\mathcal{M}_{\mathcal{S}_{0\downarrow j}}$ (in the following $\mathcal{M}_{\mathcal{S}_{\downarrow j}}$). Notice that, given Definition 5, $\mathcal{M}_{\mathcal{S}_{\downarrow j}}$ is uniquely defined for each schema version $\mathcal{SV}_j$.

Let us now introduce the meaning of an evolving object-oriented schema $\mathcal{S}$. Informally, the *semantics* is given by assigning to each schema version (a) a set of initial legal states – i.e., legal instances of the initial schema version – conforming to the class definitions in the initial schema version, and (b) the sequence of schema changes further constraining the legal instances of the schema version itself. For example, it may happen that, given an arbitrary legal state for an initial schema, different sequences of schema changes applied to the schema lead to the same final legal state: a notion of equivalence among schema modifications could be grounded on this semantic observation.

The semantics is based on the dualism between objects and values: a finite set $\mathcal{O}^{\mathcal{I}}$ of object identifiers is defined and a set of values $\mathcal{V}_{\mathcal{O}^{\mathcal{I}}}$ is constructed by induction as the smallest set including the union of $\mathcal{O}^{\mathcal{I}}$ with all possible "sets" of values and with all possible "records" of values over the given alphabet of attributes. A unique value is associated to each object; sets of objects form classes, while sets of values form types.

We first introduce the notion of *version instance* $\mathcal{I}$ for a schema version in $\mathcal{S}$. Formally, $\mathcal{I}$ is a tuple $(\mathcal{O}^{\mathcal{I}}, \rho^{\mathcal{I}}, \pi^{\mathcal{I}}, \cdot^{\mathcal{I}})$, including a non-empty finite set of objects $\mathcal{O}^{\mathcal{I}}$, a function $\rho^{\mathcal{I}} : \mathcal{O}^{\mathcal{I}} \mapsto \mathcal{V}_{\mathcal{O}^{\mathcal{I}}}$ giving a value to object identifiers, a total function $\pi^{\mathcal{I}} : \mathcal{C}_{\mathcal{S}} \mapsto 2^{\mathcal{O}^{\mathcal{I}}}$, giving the set of object identifiers in the extension of each class $C \in C_{\mathcal{S}}$, and the *interpretation* function $\cdot^{\mathcal{I}} : T \mapsto 2^{\mathcal{V}_{\mathcal{O}^{\mathcal{I}}}}$ mapping type expressions to sets of values. Although the set $\mathcal{V}_{\mathcal{O}^{\mathcal{I}}}$ is infinite, we consider the finite set $\mathcal{V}_{\mathcal{I}}$ of *active values*, which is the subset of $\mathcal{V}_{\mathcal{O}^{\mathcal{I}}}$ formed by the union of $\mathcal{O}^{\mathcal{I}}$ and the set of values assigned by $\rho^{\mathcal{I}}$ [12].

Schema versions are defined in an incremental fashion. At semantic level this means that each schema version $\mathcal{SV}_j$ adds a new constraint with respect to its immediate predecessor. Therefore, in order to be considered legal for a schema version $\mathcal{SV}_j$, a version instance $\mathcal{I}$ should satisfy the constraints imposed by $\mathcal{SV}_0$ and by the sequence of schema modifications $\mathcal{M}_{\mathcal{S}_{\downarrow j}}$. The semantics of schema modifications, shown in Tab. 6, is given by associating to each schema modification a constraint on the extension of the involved class **C**. For instance, the add-attribute semantics states that the objects instances of the class **C** should contain a value associated with the added attribute **A**; or the change-class-name semantics states that the extension of the class with the new name **C**' should coincide with that of the old name **C**.

Add-attribute **C**, **A**, **T**	$\pi^{\mathcal{I}}(\mathbf{C}) \subseteq \{o \in \mathcal{O}^{\mathcal{I}} \mid \exists v.\ \rho^{\mathcal{I}}(o) = [\![\ldots, \mathbf{A} : v, \ldots]\!] \wedge v \in \mathbf{T}^{\mathcal{I}}\}$
Drop-attribute **C**, **A**	$\pi^{\mathcal{I}}(\mathbf{C}) \subseteq \{o \in \mathcal{O}^{\mathcal{I}} \mid \exists v.\ \rho^{\mathcal{I}}(o) \neq [\![\ldots, \mathbf{A} : v, \ldots]\!]\}$
Change-attr-name **C**, **A**, **A**′	$\pi^{\mathcal{I}}(\mathbf{C}) \subseteq \{o \in \mathcal{O}^{\mathcal{I}} \mid \exists v.\ \rho^{\mathcal{I}}(o) = [\![\ldots, \mathbf{A} : v, \ldots, \mathbf{A}' : v, \ldots]\!]\}$
Change-attr-type **C**, **A**, **T**	$\pi^{\mathcal{I}}(\mathbf{C}) \subseteq \{o \in \mathcal{O}^{\mathcal{I}} \mid \exists v.\ \rho^{\mathcal{I}}(o) = [\![\ldots, \mathbf{A} : v, \ldots]\!] \wedge v \in \mathbf{T}^{\mathcal{I}}\}$
Add-class **C**	$\pi^{\mathcal{I}}(\mathbf{C}) \subseteq \mathcal{O}^{\mathcal{I}}$
Drop-class **C**	$\pi^{\mathcal{I}}(\mathbf{C}) = \emptyset$
Change-class-name **C**, **C**′	$\pi^{\mathcal{I}}(\mathbf{C}) = \pi^{\mathcal{I}}(\mathbf{C}')$
Change-class-type **C**, **T**	$\pi^{\mathcal{I}}(\mathbf{C}) \subseteq \{o \in \mathcal{O}^{\mathcal{I}} \mid \rho^{\mathcal{I}}(o) \in \mathbf{T}^{\mathcal{I}}\}$
Add-is-a **C**,**C**′	$\pi^{\mathcal{I}}(\mathbf{C}) \subseteq \pi^{\mathcal{I}}(\mathbf{C}')$
Drop-is-a **C**, **C**′	$\pi^{\mathcal{I}}(\mathbf{C}) \not\subseteq \pi^{\mathcal{I}}(\mathbf{C}')$

Fig. 6. Semantics of elementary schema changes

The constraint associated to the drop of an attribute enforces the instances of the class not to contain a value associated to the dropped attribute.

Definition 7 (Semantics). Let $\mathcal{S}$ be a schema and $\mathcal{SV}_j \in \mathcal{SV}$ one of its schema versions. A version instance $\mathcal{I} = (\mathcal{O}^{\mathcal{I}}, \rho^{\mathcal{I}}, \pi^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is said to be legal for a schema version $\mathcal{SV}_j$ if

- $\cdot^{\mathcal{I}}$ satisfies the following:

$$
\begin{aligned}
C^{\mathcal{I}} &= \pi^{\mathcal{I}}(C) \\
(\underline{\text{Union}} T_1, \ldots, T_k\ \underline{\text{End}})^{\mathcal{I}} &= T_1^{\mathcal{I}} \cup \ldots \cup T_k^{\mathcal{I}} \\
(\underline{\text{Set-of}}[m,n]T)^{\mathcal{I}} &= \{\{\!|\ v_1, \ldots, v_k\ |\!\} \mid \\
&\quad \text{for some } k \text{ such that } m \leq k \leq n, \\
&\quad \text{there exists } v_1, \ldots, v_k \text{ such that} \\
&\quad v_i \in T^{\mathcal{I}} \text{ for } i \in \{1, \ldots, k\}\} \\
(\underline{\text{Record}} A_1 : T_1, \ldots, A_k : T_k\ \underline{\text{End}})^{\mathcal{I}} &= \{[\![A_1 : v_1, \ldots, A_k : v_k, \ldots, A_h : v_h]\!] \mid \\
&\quad \text{for some } h \geq k, \\
&\quad \text{there exists } v_1, \ldots, v_h \text{ such that} \\
&\quad v_i \in T_i^{\mathcal{I}} \text{ for } i \in \{1, \ldots, k\}, \\
&\quad v_i \in \mathcal{V}_{\mathcal{O}^{\mathcal{I}}} \text{ for } i \in \{k+1, \ldots, h\}\}
\end{aligned}
$$

- for each class definition in the initial version $\mathcal{SV}_0$
  $\underline{\text{Class}}\ C\ \underline{\text{is-a}}\ C_1, \ldots, C_h\ \underline{\text{disjoint}}\ C_{h+1}, \ldots, C_k\ \underline{\text{not-is-a}}\ C_{k+1}, \ldots, C_n\ ,$
  $\underline{\text{type-is}}\ T$
  it holds that:
  $C^{\mathcal{I}} \subseteq C_i^{\mathcal{I}}$ for each $i \in \{1, \ldots, h\}$,
  $C^{\mathcal{I}} \cap C_i^{\mathcal{I}} = \emptyset$ for each $i \in \{h+1, \ldots, k\}$,
  $C^{\mathcal{I}} \not\subseteq C_i^{\mathcal{I}}$ for each $i \in \{k+1, \ldots, n\}$,
  $\{\rho^{\mathcal{I}}(o) \mid o \in \pi^{\mathcal{I}}(C)\} \subseteq T^{\mathcal{I}}$;
- for each view definition in the initial version $\mathcal{SV}_0$
  $\underline{\text{View-class}}\ C\ \underline{\text{type-is}}\ T$
  it holds that:
  $\{\rho^{\mathcal{I}}(o) \mid o \in \pi^{\mathcal{I}}(C)\} = T^{\mathcal{I}}$,
  $\{o \mid \rho^{\mathcal{I}}(o) \in T^{\mathcal{I}}\} = \pi^{\mathcal{I}}(C)$;
- for each schema modification in $\mathcal{M}_{S_{1j}}$, the functions $\pi^{\mathcal{I}}$ and $\rho^{\mathcal{I}}$ satisfy the equations of the corresponding schema change type at the right hand side of Tab. 6.

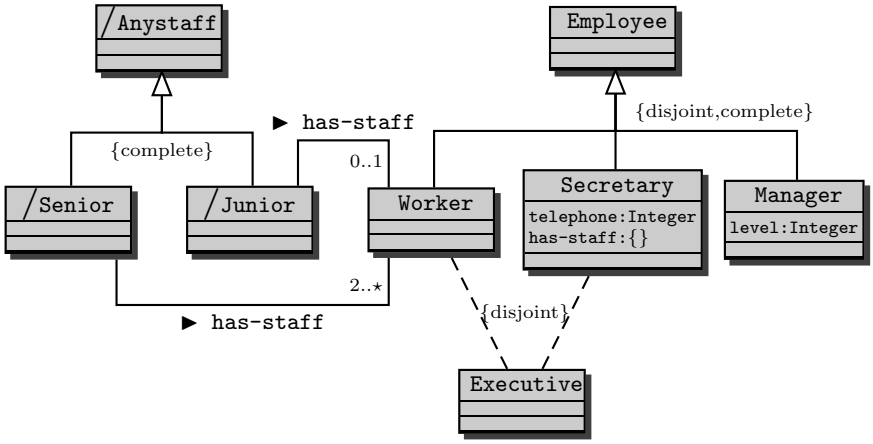

Fig. 7. The Employee initial schema version in UML-like notation.

Given a schema $\mathcal{S}$, with $\bar{\mathcal{I}}_j$ we denote the set of all possible legal version instances of the schema version $\mathcal{SV}_j$.

From the above definition it turns out that an open semantics for records has been adopted (called *-interpretation in [1]) in order to give the right semantics to inheritance. Please note also that we consider the finite set of object identifiers $\mathcal{O}^{\mathcal{I}}$ as a constant domain which is independent from each version instance.

Example 4. Let us consider an evolving schema $\mathcal{S}$ describing the employees of a company. The schema includes an initial schema version $\mathcal{SV}_0$ defined as follows:

Class `Employee` type-is Union `Manager`,`Secretary`,`Worker` End;
Class `Manager` is-a `Employee` disjoint `Secretary`, `Worker` type-is
  Record `level`: `Integer` End;
Class `Secretary` is-a `Employee` disjoint `Worker` type-is
  Record `telephone`: `Integer`, `has-staff`: Set-of [0,0] `Worker` End;
Class `Worker` is-a `Employee`;
View-class `Senior` type-is Record `has-staff`: Set-of [2,n] `Worker` End;
View-class `Junior` type-is Record `has-staff`: Set-of [0,1] `Worker` End;
Class `Executive` disjoint `Secretary`, `Worker`;
View-class `Anystaff` type-is Union `Senior`, `Junior` End;

Figure 7 shows the UML-like representation induced by the initial schema $\mathcal{SV}_0$; note that classes with names prefixed by a slash represent the views. The evolving schema $\mathcal{S}$ includes a set of schema modifications $\mathcal{M}_{\mathcal{S}}$ defined as follows:

($\mathcal{M}_{01}$) Add-is-a `Secretary`, `Manager`;
($\mathcal{M}_{12}$) Drop-is-a `Secretary`, `Anystaff`;
($\mathcal{M}_{03}$) Drop-is-a `Secretary`, `Employee`;
($\mathcal{M}_{04}$) Add-is-a `Executive`, `Employee`;
($\mathcal{M}_{45}$) Add-attribute `Manager`, `IdNum`, `Number`;
($\mathcal{M}_{56}$) Change-attr-type `Manager`, `IdNum`, `Integer`;
($\mathcal{M}_{67}$) Change-attr-type `Executive`, `IdNum`, `String`;
($\mathcal{M}_{68}$) Drop-class `Worker`;

5.1 Reasoning Problems

According to the semantic definitions given in the previous section, several reasoning problems can be introduced, in order to support the design and the management of an evolving schema.

Definition 8 (Reasoning). Given an evolving schema $\mathcal{S}$ we introduce the following reasoning problems:

a. Local/global Schema Inconsistency: a schema version $\mathcal{SV}_i$ of $\mathcal{S}$ is locally inconsistent if it does not admit any legal version instance, i.e., $\bar{\mathcal{I}}_i = \emptyset$; an evolving schema $\mathcal{S}$ is globally inconsistent if each any schema version $\mathcal{SV}_i$ does not admit any legal version instance.
b. Local/global Class Inconsistency: a class C is locally inconsistent in the version $\mathcal{SV}_i$ if for every legal version instance $\mathcal{I} \in \bar{\mathcal{I}}_i$ its extension is empty, i.e., $\pi^{\mathcal{I}}(C) = \emptyset$; a class C is globally inconsistent if it is locally inconsistent for all $\mathcal{SV}_i \in \mathcal{SV}$, i.e.,
$\forall \mathcal{SV}_i \in \mathcal{SV}.\ \forall \mathcal{I} \in \bar{\mathcal{I}}_i.\ \pi^{\mathcal{I}}(C) = \emptyset$.
c. Local/global Disjoint Classes: two classes C, D are locally disjoint in the version $\mathcal{SV}_i$ if for every legal instance $\mathcal{I} \in \bar{\mathcal{I}}_i$ their extensions are disjoint, i.e., $\pi^{\mathcal{I}}(C) \cap \pi^{\mathcal{I}}(D) = \emptyset$; two classes C, D are globally disjoint if they are locally disjoint for all $\mathcal{SV}_i \in \mathcal{SV}$, i.e., $\forall \mathcal{SV}_i \in \mathcal{SV}.\ \forall \mathcal{I} \in \bar{\mathcal{I}}_i.\ \pi^{\mathcal{I}}(C) \cap \pi^{\mathcal{I}}(D) = \emptyset$.
d. Local/global Class Subsumption: a class D locally subsumes (i.e., it is a superclass of) a class C in the version $\mathcal{SV}_i$ if for every legal instance $\mathcal{I} \in \bar{\mathcal{I}}_i$ the extension of C is included in the extension of D, i.e., $\pi^{\mathcal{I}}(C) \subseteq \pi^{\mathcal{I}}(D)$; a class D globally subsumes a class C if D locally subsumes C for all $\mathcal{SV}_i \in \mathcal{SV}$, i.e., $\forall \mathcal{SV}_i \in \mathcal{SV}.\ \forall \mathcal{I} \in \bar{\mathcal{I}}_i.\ \pi^{\mathcal{I}}(C) \subseteq \pi^{\mathcal{I}}(D)$.
e. Local/global Class Equivalence: two classes C, D are locally / globally equivalent if C locally / globally subsumes D and vice-versa.

Please note that the classical *subtyping* problem – i.e., finding the explicit representation of the partial order induced on a set of type expressions by the containment between their extensions – is a special case of class subsumption, if we restrict our attention to view definitions. In fact, in order to check whether a type T_1 is locally a subtype of a type T_2 in the version $\mathcal{SV}_i$, defined

as $\forall \mathcal{I} \in \bar{\mathcal{I}}_i$. $T_1^{\mathcal{I}} \subseteq T_2^{\mathcal{I}}$, it is enough to check whether the view (View-class C_1 type-is T_1) is locally subsumed in the version $\mathcal{SV}_i$ by the view (View-class C_2 type-is T_2). The global subtyping problem is reduced in a similar way to a global subsumption problem.

We will try to explain the application of the reasoning problems through an example.

Example 5. Let us analyse the effect of each schema change $\mathcal{M}_{ij}$ by considering the schema version $\mathcal{SV}_j$ it produces.

In the initial schema version it can be deduced that the `Junior` and `Senior` classes are *globally disjoint classes*, and that the class `Secretary` is a *global subclass* of both the `Anystaff` and `Junior` classes. The `Junior` and `Senior` view classes are complementary with respect to the record type with set values for the `has-staff` attribute. Moreover, any object in the `Anystaff` class is a record with any set value for `has-staff`. The class `Secretary` becomes automatically a subclass of both the `Junior` and `Anystaff` classes, since it has an empty valued set for the `has-staff` attribute.

`Secretary` is inconsistent in $\mathcal{SV}_1$ since `Secretary` and `Manager` are disjoint: its extension is included in the `Manager` extension only if it is empty (for each version instance $\mathcal{I} \in \bar{\mathcal{I}}_1$, $\texttt{Secretary}^{\mathcal{I}} = \emptyset$). Therefore, `Secretary` is *locally inconsistent*, as it is inconsistent in $\mathcal{SV}_1$ but not in $\mathcal{SV}_0$.

The whole schema version $\mathcal{SV}_2$ is *locally inconsistent* because the (inconsistent) `Secretary` class is necessarily a subclass of any class, including `Anystaff`. In this case, no legal version instance can exist satisfying both an isa and a not-isa constraint. $\mathcal{SV}_3$ is locally inconsistent for a similar reason: in this case there was an attempt to drop an explicitly existing isa link.

In $\mathcal{SV}_4$, it can be derived that `Executive` is locally subsumed by `Manager`, since it is a subclass of `Employee` disjoint from `Secretary` and `Worker` (`Manager`, `Secretary` and `Worker` form a partition of `Employee`).

The schema version $\mathcal{SV}_5$ exemplifies a case of attribute inheritance. The attribute `IdNum` which has been added to the `Manager` class is inherited by the `Executive` class together with the `level` attribute. This means that *every* legal instance of $\mathcal{SV}_5$ should be such that every object in the `Executive` class has a record value with an attribute `IdNum` of type `Number` and an attribute `level` of type `Integer`, i.e., for all $\mathcal{I} \in \bar{\mathcal{I}}_5$ $\pi^{\mathcal{I}}(\texttt{Executive}) \subseteq \{o \mid \rho^{\mathcal{I}}(o) \in [\![\texttt{IdNum} : \texttt{Number}, \texttt{level} : \texttt{Integer}]\!]^{\mathcal{I}}\}$. Of course, there is no restriction on the way classes are related via subsumption, and multiple inheritance is allowed even if it may sometimes cause an inconsistency – which can be easily spotted in this formalism.

The Change-attr-type elementary schema change allows for the modification of the type of an attribute with the proviso that the new type is not incompatible with the old one, like in $\mathcal{M}_{56}$. In fact, the semantics of elementary schema changes as defined in Tab. 6 is based on the assumption that we maintain one single data pool (i.e. $\rho^{\mathcal{I}}$) where each object is associated with one value and where version instances can be thought as views over the

data pool which should coexist with the starting data. Indeed, the sequence of schema versions can be seen as an increasing set of constraints; every elementary schema change introduces new constraints over a vocabulary possibly augmented by new classes or attributes in the new version. Therefore, if an object changes its value, then its object identifier should change, too. Notice that, for this reason, $\mathcal{M}_{67}$ leads to an inconsistent `Executive` class $\mathcal{SV}_7$ if `Number` and `String` are defined to be non-empty disjoint classes.

The deletion of the class `Worker` in $\mathcal{SV}_8$ leads to a locally inconsistent version of the class `Senior` but not `Junior` because of the different cardinality constraints. In fact, given that the class `Worker` is necessarily empty, then no object can exist in the class `Senior`, since it is required for them to have at least two workers as fillers of the set-valued attribute `has-staff`. On the other hand, in $\mathcal{SV}_8$ objects in the `Junior` can exist, with the proviso that they have no fillers of the attribute `has-staff`. Notice the elegance of this approach, differently from a classical object model where the class hierarchy is explicitly based on a DAG, and the deletion of a non-isolated class would require a restructuring of the DAG itself (e.g. to get rid of dangling edges).

5.2 Encoding in Description Logics

In this section we establish a relationship between the proposed model for evolving schemas and the $\mathcal{ALCQI}$ description logic. To this end, we provide an encoding from an evolving schema into an $\mathcal{ALCQI}$ knowledge base Σ, such that the reasoning problems mentioned in the previous section can be reduced to corresponding description logics reasoning problems, for which extensive theories and well founded and efficient implemented systems exist. The encoding is grounded on the fact that there is a correspondence between the models of the knowledge base and the legal instances of the evolving schema. The reason for using a non-temporal DL lies on the incremental nature of the constraints representing an evolving schema: the non-temporal DL can represent the initial schema and the constraints associated to the schema changes.

In [12], the encoding of a snapshot object-oriented schema in an $\mathcal{ALCQI}$ knowledge base is based on the *reification* of type expressions – i.e., explicit individuals exist to denote values of complex types. We introduce the concept AbstractClass to represent the classes, the concepts RecType, SetType to represent types, the role value to model the association between classes and types, and the role member to specify the type of the elements of a set. In particular, a record is represented as an individual connected by means of (functional) roles – corresponding to attributes – to the fillers of its attributes.

Our encoding is based on the mapping function ψ which translates a schema version in a knowledge base. With respect to the encoding of snapshot schemas [12], ψ extends it with set cardinalities and views.

Definition 9 (Mapping into $\mathcal{ALCQI}$).
Given a schema $\mathcal{S} = (\mathcal{C}_\mathcal{S}, \mathcal{A}_\mathcal{S}, \mathcal{SV}_0, \mathcal{M}_\mathcal{S})$ and $\mathcal{SV}_j \in \mathcal{SV}$, the $\mathcal{ALCQI}$ knowl-

edge base $\psi(\mathcal{SV}_j)$ corresponding to the schema version $\mathcal{SV}_j$ is based on the following symbols:

- Concepts: $\{\mathsf{AbstractClass}, \mathsf{RecType}, \mathsf{SetType}\} \cup \mathcal{C}_\mathcal{S}$,
- Roles: $\{\mathsf{member}\} \cup \mathcal{A}_\mathcal{S}$;

with ψ defined as follows:

$$\begin{aligned}
\psi(C) &= C \\
\psi(\underline{\text{Union}}\ T_1,\ldots,T_k\ \underline{\text{End}}) &= \psi(T_1) \sqcup \ldots \sqcup \psi(T_k) \\
\psi(\underline{\text{Set-of}}\ [\text{m,n}]\ T) &= \mathsf{SetType} \sqcap \forall \mathsf{member}.\, \psi(T) \sqcap \\
&\quad (\geqslant m\ \mathsf{member}.\ \top) \sqcap (\leqslant n\ \mathsf{member}.\ \top) \\
\psi(\underline{\text{Record}}\ A_1{:}T_1,\ldots,A_k{:}T_k\ \underline{\text{End}}) &= \mathsf{RecType} \sqcap \exists A_1.\, \psi(T_1) \sqcap \ldots \sqcap \exists A_k.\, \psi(T_k)
\end{aligned}$$

and it is composed by the following axioms:

- Axioms on basic types:
  $\mathsf{AbstractClass} \sqsubseteq \exists \mathsf{value}.\, \top \sqcap (\leqslant 1\ \mathsf{value}.\ \top)$
  $\mathsf{RecType} \sqsubseteq \forall \mathsf{value}.\, \bot$
  $\mathsf{SetType} \sqsubseteq \forall \mathsf{value}.\, \bot \sqcap \neg \mathsf{RecType}$
- For each attribute $A \in \mathcal{A}_\mathcal{S}$:
  $\exists A.\, \top \sqsubseteq (\leqslant 1\ A.\ \top)$
- For each class definition in $\mathcal{SV}_0$
  $\underline{\text{Class}}\ C\ \underline{\text{is-a}}\ C_1,\ldots,C_h\ \underline{\text{disjoint}}\ C_{h+1},\ldots,C_k\ \underline{\text{not-is-a}}\ C_{k+1},\ldots,C_n\ \underline{\text{type-is}}\ T$:
  $\psi(C) \sqsubseteq \mathsf{AbstractClass} \sqcap \psi(C_1) \sqcap \ldots \sqcap \psi(C_h) \sqcap \forall \mathsf{value}.\, \psi(T)$
  $\psi(C) \sqcap \psi(C_j) = \emptyset$ for each $j \in \{h+1, \ldots, k\}$
  $\psi(C) \not\sqsubseteq \psi(C_i)$ for each $i \in \{1, \ldots, h\}$
- For each view definition in $\mathcal{SV}_0$
  $\underline{\text{View-class}}\ C\ \underline{\text{type-is}}\ T$:
  $\psi(C) \equiv \forall \mathsf{value}.\, \psi(T)$
- For each schema modification in $\mathcal{M}_{\mathcal{S}_{\downarrow j}}$, a corresponding axiom from Tab. 8.

The knowledge base encoding an evolving schema consists of the union of all the knowledge bases of its schema versions. In the following, we will adopt subscripts in order to distinguish knowledge bases of different schema versions: given a schema version $\mathcal{SV}_j$, the corresponding knowledge base Σ_j is $\psi(\mathcal{SV}_j)$ where each concept is subscripted with j.

Definition 10. The $\mathcal{ALCQI}$ knowledge base $\Sigma = \psi(\mathcal{S})$ corresponding to the object-oriented evolving schema $\mathcal{S} = (\mathcal{C}_\mathcal{S}, \mathcal{A}_\mathcal{S}, \mathcal{SV}_0, \mathcal{M}_\mathcal{S})$ consists of the knowledge bases $\Sigma_j = \psi(\mathcal{SV}_j)$, for each $\mathcal{SV}_j \in \mathcal{SV}$.

Based on the results of [13], now we prove that the encoding is correct, in the sense that there is a correspondence between the models of the knowledge base and the legal instances of the evolving schema.

Add-attribute $\mathbf{C}$, $\mathbf{A}$, $\mathbf{T}$	$\psi(\mathbf{C}) \sqsubseteq \forall \mathsf{value}.\,(\mathsf{RecType} \sqcap \exists \mathbf{A}.\,\psi(\mathbf{T}))$
Drop-attribute $\mathbf{C}$, $\mathbf{A}$	$\psi(\mathbf{C}) \not\sqsubseteq \forall \mathsf{value}.\,(\mathsf{RecType} \sqcap \exists \mathbf{A}.\,\top)$
Change-attr-name $\mathbf{C}$, $\mathbf{A}$, $\mathbf{A}'$	$\psi(\mathbf{C}) \sqsubseteq \forall \mathsf{value}.\,(\mathsf{RecType} \sqcap \exists \mathbf{A}.\,\top \sqcap \exists \mathbf{A}'.\,\top)$
Change-attr-type $\mathbf{C}$, $\mathbf{A}$, $\mathbf{T}'$	$\psi(\mathbf{C}) \sqsubseteq \forall \mathsf{value}.\,(\mathsf{RecType} \sqcap \exists \mathbf{A}.\,\psi(\mathbf{T}'))$
Add-class $\mathbf{C}$, $\mathbf{T}$	$\psi(\mathbf{C}) \sqsubseteq \mathsf{AbstractClass}$
Drop-class $\mathbf{C}$	$\psi(\mathbf{C}) \equiv \bot$
Change-class-name $\mathbf{C}$, $\mathbf{C}'$	$\psi(\mathbf{C}) \equiv \psi(\mathbf{C}')$
Change-class-type $\mathbf{C}$, $\mathbf{T}'$	$\psi(\mathbf{C}) \sqsubseteq \forall \mathsf{value}.\,\psi(\mathbf{T}')$
Add-is-a $\mathbf{C}$, $\mathbf{C}'$	$\psi(\mathbf{C}) \sqsubseteq \psi(\mathbf{C}')$
Drop-is-a $\mathbf{C}$, $\mathbf{C}'$	$\psi(\mathbf{C}) \not\sqsubseteq \psi(\mathbf{C}')$

Fig. 8. The axioms induced by the schema changes.

Lemma 1. *For each object-oriented evolving schema $\mathcal{S}$ of depth m and each schema version $\mathcal{SV}_j \in \mathcal{SV}$, the following mappings exist:*

1. *$\alpha_{\mathcal{SV}}$ from version instances into finite interpretations and $\alpha_{\mathcal{V}}$ from active values of version instances into domain elements of the finite interpretation such that:*
   For each legal version instance $\mathcal{I} \in \bar{\mathcal{I}}_j$ of $\mathcal{SV}_j$, $\alpha_{\mathcal{SV}}(\mathcal{I})$ is a finite model of $\psi(\mathcal{SV}_j)$, and for each type expression T of $\mathcal{S}$ and each $v \in \mathcal{V}_{\mathcal{I}}$, $v \in T^{\mathcal{I}}$ iff $\alpha_{\mathcal{V}}(v) \in (\psi(T))^{\alpha_{\mathcal{SV}}(\mathcal{I})}$.
2. *$\beta_{\mathcal{SV}}$ from finite interpretations of $\psi(\mathcal{SV}_j)$ into version instances, and $\beta_{\mathcal{V}}$ from domain elements of the m-unfolded versions of finite interpretations into active values of version instances, such that:*
   For each finite model $\mathcal{I}$ of $\psi(\mathcal{SV}_j)$, $\beta_{\mathcal{SV}}(\mathcal{I})$ is a legal instance of $\mathcal{SV}_j$, and for each concept $\psi(T)$, which is the translation of a type expression T of $\mathcal{S}$ with respect to the schema version $\mathcal{SV}_j$, and each $d \in \Delta^{\mathcal{I}_{|m}}$, $d \in \psi(T)^{\mathcal{I}_{|m}}$ if and only if $\beta_{\mathcal{V}}(d) \in T^{\beta_{\mathcal{SV}}(\mathcal{I})}$.

We can now prove that the set of legal instances of the evolving schema $\mathcal{S}$ corresponds to the set of possible finite models of $\psi(\mathcal{S})$.

Proposition 3 (Correctness of the encoding). *For each object-oriented evolving schema $\mathcal{S}$ of depth m, the following mappings exist:*

1. *$\alpha_{\mathcal{S}}$ from instances of a schema into finite interpretations of $\psi(\mathcal{S})$ such that for each legal instance $\mathcal{I}$ of $\mathcal{S}$, $\alpha_{\mathcal{S}}(\mathcal{I})$ is a finite model of $\psi(\mathcal{S})$.*
2. *$\beta_{\mathcal{S}}$ from finite interpretations of $\psi(\mathcal{S})$ into instances of $\mathcal{S}$ such that for each finite model $\mathcal{I}$ of $\psi(\mathcal{S})$, $\beta_{\mathcal{S}}(\mathcal{I})$ is a legal instance of $\mathcal{S}$.*

The semantic correspondence is exploited to devise a correspondence between reasoning problems at the level of evolving schemas and reasoning problems at the level of the description logic [22].

Theorem 2 (Reasoning with evolving schemas). *Given an evolving schema $\mathcal{S}$, the reasoning problems defined in the previous section are all decidable in EXPTIME with a PSPACE lower bound and EXPTIME complete with full booleans in type expressions. The reasoning problems can be reduced to corresponding satisfiability problems in the $\mathcal{ALCQI}$ Description Logic.*

Example 6. In this Section we provide the complete encoding in $\mathcal{ALCQI}$ of the example introduced in Section 5.1. The following are the axioms induced by $\mathcal{SV}_0$:

$$\texttt{Employee} \sqsubseteq \mathsf{AbstractClass} \sqcap \forall \mathsf{value}.\,(\texttt{Manager} \sqcup \texttt{Secretary} \sqcup \texttt{Worker})$$
$$\texttt{Manager} \sqsubseteq \mathsf{AbstractClass} \sqcap \texttt{Employee} \sqcap \forall \mathsf{value}.(\mathsf{RecType} \sqcap \exists \texttt{level}.\texttt{Integer})$$
$$\texttt{Manager} \sqcap \texttt{Secretary} = \emptyset$$
$$\texttt{Manager} \sqcap \texttt{Worker} = \emptyset$$
$$\texttt{Secretary} \sqsubseteq \mathsf{AbstractClass} \sqcap \texttt{Employee} \sqcap \forall \mathsf{value}.(\mathsf{RecType} \sqcap \exists \texttt{telephone}.\texttt{Integer} \sqcap \exists \texttt{has\_staff}.(\mathsf{SetType} \sqcap \forall \mathsf{member}.\,\texttt{Worker} \sqcap (\geqslant 0\ \mathsf{member}.\ \top) \sqcap (\leqslant 0\ \mathsf{member}.\ \top)))$$
$$\texttt{Secretary} \sqcap \texttt{Worker} = \emptyset$$
$$\texttt{Worker} \sqsubseteq \texttt{Employee}$$
$$\texttt{Senior} \equiv \mathsf{AbstractClass} \sqcap \forall \mathsf{value}.(\mathsf{RecType} \sqcap \exists \texttt{has\_staff}.(\mathsf{SetType} \sqcap \forall \mathsf{member}.\,\texttt{Worker} \sqcap (\geqslant 2\ \mathsf{member}.\ \top) \sqcap (\leqslant n\ \mathsf{member}.\ \top)))$$
$$\texttt{Junior} \equiv \mathsf{AbstractClass} \sqcap \forall \mathsf{value}.(\mathsf{RecType} \sqcap \exists \texttt{has\_staff}.(\mathsf{SetType} \sqcap \forall \mathsf{member}.\,\texttt{Worker} \sqcap (\geqslant 0\ \mathsf{member}.\ \top) \sqcap (\leqslant 1\ \mathsf{member}.\ \top)))$$
$$\texttt{Executive} \sqsubseteq \mathsf{AbstractClass}$$
$$\texttt{Executive} \sqcap \texttt{Secretary} = \emptyset$$
$$\texttt{Executive} \sqcap \texttt{Worker} = \emptyset$$
$$\texttt{Anystaff} \equiv \mathsf{AbstractClass} \sqcap \forall \mathsf{value}.\,(\texttt{Senior} \sqcup \texttt{Junior})$$

whereas the following are the axioms induced by the applied schema modifications:

$$(\mathcal{M}_{01}) \quad \texttt{Secretary} \sqsubseteq \texttt{Manager}$$
$$(\mathcal{M}_{12}) \quad \texttt{Secretary} \not\sqsubseteq \texttt{Anystaff}$$
$$(\mathcal{M}_{03}) \quad \texttt{Secretary} \not\sqsubseteq \texttt{Employee}$$
$$(\mathcal{M}_{04}) \quad \texttt{Executive} \sqsubseteq \texttt{Employee}$$
$$(\mathcal{M}_{45}) \quad \texttt{Manager} \sqsubseteq \forall \mathsf{value}.\,(\mathsf{RecType} \sqcap \exists \texttt{IdNum}.\,\texttt{Number})$$
$$(\mathcal{M}_{56}) \quad \texttt{Manager} \sqsubseteq \forall \mathsf{value}.\,(\mathsf{RecType} \sqcap \exists \texttt{IdNum}.\,\texttt{Integer})$$
$$(\mathcal{M}_{67}) \quad \texttt{Executive} \sqsubseteq \forall \mathsf{value}.\,(\mathsf{RecType} \sqcap \exists \texttt{IdNum}.\,\texttt{String})$$
$$(\mathcal{M}_{68}) \quad \texttt{Employee} \equiv \bot$$

The knowledge base Σ corresponding to $\psi(\mathcal{S})$ is made up of the knowledge bases $\Sigma_0, \ldots, \Sigma_8$ corresponding to the schema versions $\mathcal{SV}_0, \ldots, \mathcal{SV}_8$. In particular the axioms having concepts and roles subscripted with 0 define $\Sigma_0 = \psi(\mathcal{SV}_0)$ whereas $\Sigma_i = \psi(\mathcal{SV}_i)$ with $i \geq 1$ is constituted by the axioms

induced by $\mathcal{SV}_0$ and of a subset of the axioms corresponding to the sequence of schema modifications $\mathcal{M}_{\mathcal{S}_{\downarrow i}}$ where each concept and role is subscripted with i.

6 Conclusions

We have shown in this Chapter how Description Logics are useful to encode not only conceptual data models of static information, but also conceptual data models of dynamic information. This includes both the case of dynamic data and the case of dynamic schemas. The encoding in Description Logics was useful to prove computational properties of the data models, and to allow for an implementation of the reasoning by exploiting existing inference engines. Moreover, this Chapter presents the first systematic formalisation of the constructs provided in the majority of temporally enhanced EER conceptual modelling systems.

Acknowledgements

The work presented in this Chapter refers to previous work done by the authors together with Fabio Grandi, Frank Wolter, and Michael Zakharyaschev; we wish to thank them all. The second author wishes to thank also the Department of Computer Science at the University of Manchester (UK) where most of the work presented here was carried out. The work presented in this Chapter has been partially funded by EPSRC grants GR/R45369/01, GR/R04348/01, GR/R09428/01.

References

1. S. Abiteboul and P. Kanellakis. Object identity as a query language primitive. *Journal of the ACM*, 45(5):798–842, 1998. A first version appeared in SIGMOD'89.
2. A. Artale, D. Calvanese, and E. Franconi. On the formalisation of the temporal entity-relationship model. Technical report, Free University of Bolzano, Faculty of Computer Science, 2003.
3. A. Artale and E. Franconi. Temporal ER modeling with description logics. In *Proc. of the International Conference on Conceptual Modeling (ER'99)*. Springer-Verlag, November 1999.
4. A. Artale, E. Franconi, F. Wolter, and M. Zakharyaschev. A temporal description logic for reasoning over conceptual schemas and queries. In *Proc. of the 8th European Conference on Logics in Artificial Intelligence (JELIA-2002)*, 2002.
5. F. Baader, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002.

6. J. Banerjee, W. Kim, H.-J. Kim, and H. F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. In *Proc. of the ACM-SIGMOD Annual Conference*, pages 311–322, San Francisco, CA, May 1987.
7. A. Borgida. Description logics in data management. *TKDE*, 7(5):671–682, 1995.
8. Alexander Borgida and Ronald J. Brachman. Loading data into description reasoners. In *Proc. of 1993 ACM SIGMOD International Conference on Management of Data*, pages 217–226, 1993.
9. Paolo Bresciani, Michele Nori, and Nicola Pedot. A knowledge based paradigm for querying databases. In *Proc. of DEXA-00*, pages 794–804, 2000.
10. Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. A formal framework for reasoning on UML class diagrams. In *Proc. of the 13th Int. Sym. on Methodologies for Intelligent Systems (ISMIS 2002)*, pages 503–513, 2002.
11. D. Calvanese, G. De Giacomo, and M. Lenzerini. Keys for free in description logics. In *Proc. of the 2000 International Workshop on Description Logics (DL'2000)*, 2000.
12. D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In Chomicki and Saake [15].
13. D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.
14. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.
15. J. Chomicki and G. Saake, editors. *Logics for Databases and Information Systems*. Kluwer, 1998.
16. J. Chomicki and D. Toman. Temporal logic in information systems. In Chomicki and Saake [15], chapter 1.
17. C. De Castro, F. Grandi, and M. R. Scalas. Schema Versioning for Multitemporal Relational Databases. *Information Systems*, 22(5):249–290, 1997.
18. F. Donini. Complexity of reasoning. In Baader et al. [5].
19. R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Benjamin/Cummings, 2nd edition, 1994.
20. F. Ferrandina, T. Meyer, R. Zicari, G. Ferran, and J. Madec. Schema and Database Evolution in the O_2 Object Database System. In *Proc. of the 21st Int'l Conf. on Very Large Databases (VLDB)*, pages 170–181, Zurich, Switzerland, September 1995.
21. E. Franconi, F. Baader, U. Sattler, and P. Vassiliadis. Multidimensional data models and aggregation. In M. Jarke, M. Lenzerini, Y. Vassilious, and P. Vassiliadis, editors, *Fundamentals of Data Warehousing*, chapter 5, pages 87–106. Springer-Verlag, 1999.
22. E. Franconi, F. Grandi, and F. Mandreoli. A semantic approach for schema evolution and versioning in object-oriented databases. In *Proc. of the 1st International Conf. on Computational Logic (CL'2000), DOOD stream*. Springer-Verlag, July 2000.
23. E. Franconi and U. Sattler. A data warehouse conceptual data model for multidimensional aggregation. In *Proceedings of the Workshop on Design and Management of Data Warehouses (DMDW'99)*, 1999.

24. Enrico Franconi. Knowledge representation meets digital libraries. In *Proc. of the 1st DELOS (Network of Excellence on Digital Libraries) workshop on "Information Seeking, Searching and Querying in Digital Libraries"*, 2000.
25. Francois Goasdoue, Veronique Lattes, and Marie-Christine Rousset. The use of CARIN language and algorithms for information integration: the picsel system. *International Journal on Cooperative Information Systems*, 2000.
26. H Gregersen and J. S. Jensen. Temporal Entity-Relationship models - a survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):464–497, 1999.
27. H. Gregersen and J.S. Jensen. Conceptual modeling of time-varying information. Technical Report TimeCenter TR-35, Aalborg University, Denmark, 1998.
28. R. Gupta and G. Hall. Modeling transition. In *Proc. of ICDE'91*, pages 540–549, 1991.
29. R. Gupta and G. Hall. An abstraction mechanism for modeling generation. In *Proc. of ICDE'92*, pages 650–658, 1992.
30. Mathias Jarke, V. Quix, D. Calvanese, Maurizio Lenzerini, Enrico Franconi, S. Ligoudistiano, P. Vassiliadis, and Yannis Vassiliou. Concept based design of data warehouses: The DWQ demonstrators. In *2000 ACM SIGMOD International Conference on Management of Data*, May 2000.
31. 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.
32. C. S. Jensen and R. T. Snodgrass. Temporal data management. *IEEE Transactions on Knowledge and Data Engineering*, 111(1):36–44, 1999.
33. C. S. Jensen, M. Soo, and R. T. Snodgrass. Unifying temporal data models via a conceptual model. *Information Systems*, 9(7):513–547, 1994.
34. Alon Y. Levy and Marie-Christine Rousset. Combining horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.
35. Eduardo Mena, Arantza Illarramendi, Vipul Kashyap, and Amit P. Sheth. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.
36. R. J. Peters and M. T. Özsu. An Axiomatic Model of Dynamic Schema Evolution in Objectbase Systems. *ACM Transactions On Database Systems*, 22(1):75–114, 1997.
37. J. F. Roddick. A Survey of Schema Versioning Issues for Database Systems. *Information and Software Technology*, 37(7):383–393, 1996.
38. J. F. Roddick and R. T. Snodgrass. Schema Versioning. In *The TSQL2 Temporal Query Language*, pages 427–449. Kluwer, 1995.
39. R. T. Snodgrass. The temporal query language TQUEL. *ACM Transaction on Database Systems*, 12(2):247–298, 1987.
40. S. Spaccapietra, C. Parent, and E. Zimanyi. Modeling time from a conceptual perspective. In *Int. Conf. on Information and Knowledge Management (CIKM98)*, 1998.
41. B. Tauzovich. Towards temporal extensions to the entity-relationship model. In *Proc. of the* 10^{th} *International Conference on the Entity-Relationship Approach (ER'91)*, pages 163–179, 1991.

42. C. Theodoulidis, P. Loucopoulos, and B. Wangler. A conceptual modelling formalism for temporal database applications. *Information Systems*, 16(3):401–416, 1991.
43. F. Wolter and M. Zakharyaschev. Modal description logics: Modalizing roles. *Fundamenta Informaticae*, 39(4):411–438, 1999.