Chapter 1
The Return of the Entity-Relationship Model: Ontological Query Answering

Andrea Calì, Georg Gottlob and Andreas Pieris

Abstract The Entity-Relationship (ER) model is a fundamental tool for database design, recently extended and employed in knowledge representation and reasoning due to its expressiveness and comprehensibility. We present an extension of the ER model, called ER^+, which is particularly suitable for ontology modeling, as well as being flexible and comprehensible. Our model comprises is-a constraints among entities and relationships, plus functional and mandatory participation constraints. In particular, it allows for arbitrary permutations of the roles in is-a among relationships. We argue that ER-based languages can be profitably used in ontology-enhanced database systems, where queries are evaluated against the union of a database instance and an ontology, which constitute a logical theory. In such systems, the instance has usually large size, therefore ensuring tractable complexity of query answering w.r.t. the instance size is crucial. A key notion that ensures tractability in ER^+ schemata is *separability*, i.e., the absence of interaction between the functional participation constraints and the other constructs. We provide a precise syntactic characterization of separable ER^+ schemata by means of a necessary and sufficient condition. We present a complete complexity analysis of the conjunctive query answering problem under separable ER^+ schemata, and also under several sublanguages of ER^+. We show that the addition of so-called *negative constraints* does not increase the complexity of query answering. With such constraints, our model properly generalizes the most widely-adopted tractable ontology languages, including those in the well-known *DL-Lite* family.

Andrea Calì
Department of Computer Science and Information Systems, Birkbeck University of London, e-mail: andrea@dcs.bbk.ac.uk

Georg Gottlob
Department of Computer Science, University of Oxford, e-mail: georg.gottlob@cs.ox.ac.uk

Andreas Pieris
Department of Computer Science, University of Oxford, e-mail: andreas.pieris@cs.ox.ac.uk

1.1 Introduction

1.1.1 Motivation

The Entity-Relationship (ER) model is a fundamental formalism for conceptual modeling in database design; it was introduced by Chen in his milestone paper [3], and it is now widely used, being flexible and easily understood by practitioners. With the rise of the Semantic Web, conceptual modeling formalisms have gained importance again as *ontology formalisms*, in the Semantic Web parlance. Ontologies and conceptual models are aimed at representing, rather than the structure of data, the domain of interest, that is, the fragment of the real world that is being represented by the data and the schema. A prominent formalism for modeling ontologies are *description logics (DLs)* [4], which are decidable fragments of first-order logic, particularly suitable for ontological modeling and querying. In particular, DL ontologies are sets of assertions describing sets of objects and (usually binary) relations among such sets, exactly in the same fashion as the ER model.

Recently, research on DLs has been focusing on the problem of *answering queries* under ontologies, that is, given a query q, an instance B, and an ontology $\mathscr{K}$, answering q under B and $\mathscr{K}$ amounts to compute the answers that are *logically entailed* from B by using the assertions of $\mathscr{K}$. In this context, where data size is usually large, a central issue the *data complexity* of query answering, i.e., the computational complexity with respect to the data set B only, while the ontology $\mathscr{K}$ and the query q are fixed.

For this reason the tractable *DL-Lite* family [5, 6], for instance, has become a prominent family of DL languages. More precisely, DL-Lite languages are *first-order rewritable*, i.e., for every conjunctive query q, for every initial instance B, and for every DL-Lite knowledge base $\mathscr{K}$, it is possible to construct a first-order query $q_{\mathscr{K}}$ (which, in particular, is a union of conjunctive queries), such that directly evaluating $q_{\mathscr{K}}$ over B (without resorting to the knowledge base $\mathscr{K}$ any more) returns the correct answers to q under B and $\mathscr{K}$. The first-order rewritability implies two desirable properties: *(i)* the data complexity of conjunctive query answering is very low, namely, in the highly tractable class AC_0 (which is the complexity class of recognizing words in languages defined by constant-depth Boolean circuits with (unlimited fan-in) AND and OR gates), and *(ii)* query answering can be performed efficiently by translating the rewritten query $q_{\mathscr{K}}$ into SQL, thus taking advantage of the optimization capabilities of the underlying relational DBMS.

In this chapter we consider the problem of conjunctive query answering under an extended version of the ER model, which we call ER^+. Our model is obtained from the original ER model by adding is-a relations among entities and relationships, functional and mandatory participation constraints on the participation of entities to relationships, and functional and mandatory constraints on attributes of entities and relationships. Figure 1.1 shows an ER^+ schema, where the reader will recognize the familiar graphical notation of Chen's ER model. The schema describes members of a university department working in research groups. Ph.D. students and professors

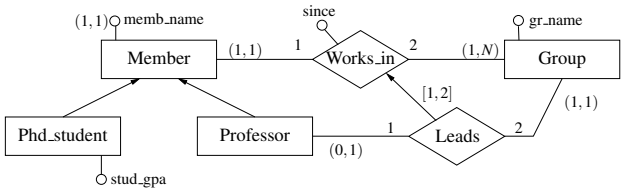

Fig. 1.1 An ER^+ schema.

are both members. For instance, the constraint $(1,1)$ (min. 1, max. 1) on the participation of *Member* in *Works_in* imposes that every instance of *Member* participates at least once (mandatory participation) and at most once (functional participation) in *Works_in*. The constraint $(0,1)$ (min. 0, max. 1) on the participation of *Professor* in *Leads* imposes that every instance of *Professor* participates at most once (functional participation) in *Leads*. The constraint $(1,N)$ (min. 1, max. arbitrary) on the participation of *Group* in *Works_in* imposes that every instance of *Group* participates at least once (mandatory participation) in *Works_in*. The constraint $(1,1)$ on the attribute *memb_name* states that such an attribute is mandatory (at least one value) and functional (at most one value). The is-a constraint among the relationships states that each professor works in the same group that (s)he leads, considering the components in the same order; components of relationships are ordered and marked by the integers 1 and 2. The notation $[1,2]$ indicates the permutation of the components involved in the containment.

In our setting we admit arbitrary *permutations* of objects in an is-a between two relationships. For example, we can assert that each instance $\langle a,b,c\rangle$ of a ternary relationship R_1 is also an instance of another ternary relationship R_2, but the three objects (instances of the participating entities) appear in R_2 in the order $\langle c,a,b\rangle$; this would be indicated by the annotation $[3,1,2]$ in the corresponding diagram. The addition of the permutation feature increases the complexity of query answering. The permutation is a common feature in DL-Lite and other Semantic Web languages, where it is usually in the form of *inverse roles*, where roles are binary relations among *concepts* (sets). With this feature we can express, for instance, the reflexive property of a relation; for example (in first-order logic) $\forall X \forall Y\, brother(X,Y) \rightarrow brother(Y,X)$.

Similarly to what is found in the literature on DL-Lite, we are interested in query answering under ontologies, and in particular in the data complexity of (the decision version of) such problem in the presence of ER^+ ontologies. We identify and study sub-languages of ER^+ which are tractable in data complexity, and we also provide a complete complexity study (data and combined complexity) of query answering under such sub-languages.

Language	Arity	Perm.	Combined	Data
non-conflicting ER^+	any	yes	PSPACE-complete UB: [8]	AC_0 UB: [9]
non-conflicting ER^+_{i}	any	no	NP-complete LB: [10]	AC_0 UB: [9]
non-conflicting $\mathsf{ER}^+_{\mathscr{B}}$	bounded	yes	NP-complete UB: [8] LB: [10]	AC_0 UB: [9]

Table 1.1 Summary of complexity results.

1.1.2 Summary of the Chapter

We first introduce, in Section 1.3, the ER^+ model, and provide a formal semantics for it in terms of a relational representation; that is, each ER^+ schema is encoded in a relational schema, plus a set of relational constraints – belonging to the well-known classes of *inclusion dependencies* and *key dependencies*. A set of inclusion and key dependencies in such a form that it can represent an ER^+ schema is called a set of *conceptual dependencies (CDs)*. We study different variants of the ER^+ model in terms of the corresponding (sub)classes of CDs. Our central algorithmic tool is the *chase* [7, 8], widely adopted in the literature in the context of query containment and answering under relational constraints.

It turns out that ER^+ is *not* first-order rewritable. The chief reason for this is the interaction between inclusion dependencies and key dependencies in a set of CDs, which mirrors the interaction between functional participation constraints and the other kinds of constraints in the corresponding ER^+ schema. The central (semantic) notion here is then *separability*, which guarantees a decoupling between inclusion and key dependencies in a set of CDs: under a set Σ of separable CDs, answering a query q on an instance D is equivalent to answering q on D under the inclusion dependencies of Σ only. We provide a *syntactic* condition, in Section 1.4, which implies separability of CDs; we do this by defining *non-conflicting CDs* (and, consequently, non-conflicting ER^+ schemata). We show that the aforementioned condition is not only sufficient, but also necessary; this way, we do precisely characterize the class of separable ER^+ schemata.

We provide, in Section 1.5, a complexity analysis of the query answering problem for the following three cases: *(i)* non-conflicting ER^+schemata, *(ii)* non-conflicting ER^+schemata with *identity permutations*, i.e., schemata where every is-a among relationships has the identity permutation associated to it (non-conflicting ER^+_{i}), and *(iii)* non-conflicting ER^+schemata with the arity of relationships bounded by a constant (non-conflicting $\mathsf{ER}^+_{\mathscr{B}}$). Observe that ER^+ properly generalizes both ER^+_{i} and $\mathsf{ER}^+_{\mathscr{B}}$. The complexity results shown in this chapter are summarized in Table 1.1. When results in the table are already known from the literature, this is indicated (UB and LB stand for upper and lower bound, respectively). Notice that *all* non-conflicting variants of ER^+ enjoy first-order rewritability, which in turn guarantees AC_0 data complexity.

We then turn our attention, in Section 1.6, to *negative constraints*, which allows us to express, for instance, non-participation of an entity to a relationship, but also more general assertions. We show that the addition of negative constraints does not increase the complexity of query answering.

Last but not least, in Section 1.7, we mention that the tractable ER^+ languages we present here have most DL-Lite languages as special cases. This confirms the usefulness of ER^+ in ontology modeling and ontology-based query answering.

1.1.3 Related work.

Our model is based on Chen's ER model [3]. The extended model considered in [16] includes the generalization and full aggregation constructs, while the one in [15] allows for arbitrary data types, optional and multivalued attributes, a concept for specialization and generalization, and complex structured entity types. [17] considers relational schemata with referential integrity constraints that can be associated with an extended ER schema.

The paper [18] gives a formal first-order semantics to the ER model. Reasoning tasks under theories expressed in different variants of the ER model, whose constructs are given a formal semantics, are studied in [19].

Query answering under ER ontologies is tackled in [20], where the proposed ER language is strictly less expressive than the ones presented in this chapter; the problem is studied in the context of data integration systems, and a a first-order rewriting technique is proposed, later extended in [9]. The notion of separability was first introduced in [11]. Sufficient conditions for separability appear, for instance, in [26, 27].

The formalism of [21], which is also studied with respect to query answering, is more expressive than ER^+, and it does not offer the same tractability of query answering.

Other works [8, 22] consider query containment, which is tightly related to the query answering problem in the case of incomplete information. [22] considers query containment in a formalism similar to ER^+, but with less expressive negative constraints, and eventually incomparable to ER^+; the combined complexity is higher than the one under non-conflicting ER^+, and data complexity is not studied.

Our work is tightly related to the Semantic Web and, in general, to web data management. The World Wide Web Consortium (W3C) defines several standards, including the *Resource Description Framework (RDF)* for the data layer, the *Web Ontology Language (OWL)*, based on Description Logics (DLs), for the ontology layer, and the *Rule Interchange Format (RIF)*, currently being defined as a standard, for the rule layer. Several ontology languages have been proposed, many of which are more expressive than ER^+, while being less efficient in terms of query answering. The ER^+ model is intended to be a viable alternative to the most prominent *tractable* OWL sublanguages. Our ER^+ variants offer the same, desirable tractability properties (first-order rewritability, in particular), while in addition avoiding the

description logic syntax, which could turn out to be slightly awkward to database practitioners. Due to the popularity of the ER model in the industry, and to its intuitive graphical representation, the ER^+ formalism is likely to be adopted as ontology language, especially when employed in visual design tools. More specifically, the languages of the DL-Lite family [5, 6], as already explained, enjoy the desirable property of tractable conjunctive query answering, in particular in AC_0 in data complexity, again ensured by their first-order rewritability. Both our formalisms non-conflicting ER^+ and non-conflicting $\mathsf{ER}^+_{\mathscr{B}}$, with the addition of negative constraints, which do not increase the complexity of query answering, properly generalize the languages DL-Lite$_{\mathscr{F}}$ and DL-Lite$_{\mathscr{R}}$; at the same time, the data complexity of query answering is the same as that of the DL-Lite languages.

The notion of *chase* [7, 8, 11] of a database against a set of inclusion dependencies is crucial in ontological query answering; such notion has been extended to expressive rules called *tuple-generating dependencies (TGDs)* [23, 24]. In particular, [23] considers the *data exchange* problem, where a target database is materialized starting from a source database and a mapping expressed as a set of TGDs; in this setting the chase procedure needs to terminate [24, 25]. However, this is not appropriate for ontological databases. The first work to tackle the problem of a non-terminating chase was [8].

Recent works [14, 26, 27, 28] deal with so-called *tuple-generating dependencies*, rules that constitute the *Datalog*$^{\pm}$ family of languages, some of which are first-order rewritable. Datalog$^{\pm}$ languages (enriched with non-conflicting key dependencies as in [2] or [26]) are not capable of capturing non-conflicting ER^+ schemata. For an overview of some Datalog$^{\pm}$ languages we refer the reader to [29].

Finally, the work [30] deals with general (non-separable) ER^+ schemata: a technique for query rewriting into recursive Datalog is presented, and complexity bounds are significantly higher than those shown here.

1.2 Preliminaries

In this section we recall some basics on databases, queries, dependencies, and the chase procedure.

1.2.1 General

We define the following pairwise disjoint (infinite) sets of symbols: *(i)* a set Γ of *constants* (constitute the "normal" domain of a database), *(ii)* a set Γ_N of *labeled nulls* (used as placeholders for unknown values, and thus can be also seen as variables), and *(iii)* a set Γ_V of *variables* (used in queries and dependencies). Different constants represent different objects (*unique name assumption*), while different nulls may represent the same object. A lexicographic order is defined on $\Gamma \cup \Gamma_N$,

such that every value in Γ_N follows all those in Γ. We denote by $\mathbf{X}$ sequences (or sets) of variables and constants $X_1,\ldots,X_k$, where $k>0$; it will be clear from the context when we regard $\mathbf{X}$ as a sequence of symbols, and when as a set of symbols. For an integer $n>1$, we denote by $[n]$ the set $\{1,\ldots,n\}$.

A *relational schema* $\mathscr{R}$ (or simply *schema*) is a set of *relational symbols* (or *predicates*), each with its associated arity. We write r/n to denote that the predicate r has arity n and we refer to the integer n by $arity(r)$. A *position* $r[i]$, in a schema $\mathscr{R}$, is identified by a predicate $r\in\mathscr{R}$ and its i-th argument (or attribute). A *term* t is either a constant or a null or a variable. An *atomic formula* (or simply *atom*) has the form $r(t_1,\ldots,t_n)$, where r/n is a relation, and $t_1,\ldots,t_n$ are terms. For an atom a, we denote as $dom(a)$ and $pred(a)$ the set of its terms and its predicate, respectively. These notations naturally extends to sets and conjunctions of atoms. Conjunctions of atoms will be often identified with the sets of their atoms.

A *substitution* from a set $S_1\subset\Gamma\cup\Gamma_N\cup\Gamma_V$ to a set $S_2\subset\Gamma\cup\Gamma_N\cup\Gamma_V$ is a function $h:S_1\to S_2$ defined as follows: *(i)* $\varnothing$ is a substitution, and *(ii)* if h is a substitution, then $h\cup\{X\to Y\}$ is a substitution, where $X\in S_1$ and $Y\in S_2$. A *homomorphism* from a set of atoms A_1 to a set of atoms A_2 is a substitution $h:dom(A_1)\to dom(A_2)$ such that: *(i)* if $t\in\Gamma$, then $h(t)=t$, and *(ii)* if $r(t_1,\ldots,t_n)\in A_1$, then $h(r(t_1,\ldots,t_n))=r(h(t_1),\ldots,h(t_n))\in A_2$. If there exists a homomorphism from A_1 to A_2 and vice-versa, then A_1 and A_2 are *homomorphically equivalent*. The notion of homomorphism naturally extends to conjunctions of atoms.

1.2.2 Databases and Queries

A *relational instance* (or simply *instance*) I for a schema $\mathscr{R}$ is a (possibly infinite) set of atoms of the form $r(\mathbf{t})$, where $r/n\in\mathscr{R}$ and $\mathbf{t}\in(\Gamma\cup\Gamma_N)^n$. We denote as $r(I)$ the set $\{\mathbf{t}\mid r(\mathbf{t})\in I\}$. A database D for $\mathscr{R}$ is a finite instance for $\mathscr{R}$ such that $dom(D)\subset\Gamma$. For simplicity, we assume source databases which contain only ground atoms, i.e., atoms whose arguments are constants of Γ. However, it is not difficult to show that the results of this paper hold even if we allow incomplete source databases which may include labeled nulls.

A *conjunctive query (CQ)* q of arity n over a schema $\mathscr{R}$, written as q/n, is an assertion the form $p(\mathbf{X})\leftarrow\varphi(\mathbf{X},\mathbf{Y})$, where $\varphi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms over $\mathscr{R}$, and p is an n-ary predicate not occurring in $\mathscr{R}$. $\varphi(\mathbf{X},\mathbf{Y})$ is called the *body* of q, denoted as $body(q)$. The *answer* to a CQ q/n of the above form over an instance I, denoted as $q(I)$, is the set of all n-tuples $\mathbf{t}\in\Gamma^n$ for which there exists a homomorphism $h:\mathbf{X}\cup\mathbf{Y}\to\Gamma\cup\Gamma_N$ such that $h(\varphi(\mathbf{X},\mathbf{Y}))\subseteq I$ and $h(\mathbf{X})=\mathbf{t}$.

1.2.3 Dependencies.

A *tuple-generating dependency (TGD)* σ over $\mathscr{R}$ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \psi(\mathbf{X},\mathbf{Z})$, where $\varphi(\mathbf{X},\mathbf{Y})$ and $\psi(\mathbf{X},\mathbf{Z})$ are conjunctions of atoms over $\mathscr{R}$, called the *body* and the *head* of σ, denoted as $body(\sigma)$ and $head(\sigma)$, respectively. Henceforth, for notational convenience, we will omit the universal quantifiers in front of TGDs. Such σ is satisfied by an instance I for $\mathscr{R}$ if whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X},\mathbf{Y})) \subseteq I$, then there exists an extension h' of h, i.e., $h' \supseteq h$, such that $h'(\psi(\mathbf{X},\mathbf{Z})) \subseteq I$.

A *key dependency (KD)* κ over $\mathscr{R}$ is an assertion of the form $key(r) = \mathbf{A}$, where $r \in \mathscr{R}$ and $\mathbf{A}$ is a set of attributes of r. Such κ is satisfied by an instance I for $\mathscr{R}$ if for each pair of distinct tuples $\mathbf{t}_1, \mathbf{t}_2 \in r(I)$, $\mathbf{t}_1[\mathbf{A}] \neq \mathbf{t}_2[\mathbf{A}]$, where $\mathbf{t}[\mathbf{A}]$ is the projection of tuple $\mathbf{t}$ over $\mathbf{A}$.

A *negative constraint (NC)* ν over $\mathscr{R}$ is a first-order formula $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \bot$, where $\varphi(\mathbf{X})$ is a conjunction of atoms over $\mathscr{R}$, called the *body* of ν and denoted as $body(\nu)$, and $\bot$ is the truth constant *false*. For brevity, we will omit the universal quantifiers in front of NCs, and assume such a quantification implicitly. Such ν is satisfied by an instance I if there is *no* homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$.

Given an instance I and a dependency δ, we write $I \models \delta$ (resp., $I \not\models \delta$) if I satisfies (resp., violates) δ. Given a set of dependencies Σ, we say that I satisfies Σ, written as $I \models \Sigma$, if for each $\delta \in \Sigma$ it holds that $I \models \delta$. Conversely, we say that I violates Σ, written as $I \not\models \Sigma$, if there exists $\delta \in \Sigma$ such that $I \not\models \delta$.

1.2.4 Query Answering under Dependencies

Let us now introduce the problem of query answering under dependencies. Given a database D and a set of dependencies Σ, the answers we consider are those that are true in *all* models of D with respect to Σ, i.e., all instances that contain D and satisfy the dependencies of Σ. Formally, the *models* of D with respect to Σ, denoted as $mods(D,\Sigma)$, is the set of all instances I such that $I \models D \cup \Sigma$. The *answer* to a CQ q/n with respect to D and Σ is the set of n-tuples $ans(q,D,\Sigma) = \{\mathbf{t} \mid \mathbf{t} \in q(I), \text{ for each } I \in mods(D,\Sigma)\}$.

The *decision problem* associated to conjunctive query answering under dependencies, dubbed *CQ-Ans*, is defined as follows: given a CQ q/n over a schema $\mathscr{R}$, a database D for $\mathscr{R}$, a set Σ of dependencies over $\mathscr{R}$, and an n-tuple $\mathbf{t} \in \Gamma^n$, decide whether $\mathbf{t} \in ans(q,D,\Sigma)$. Following Vardi's taxonomy [31], the *data complexity* of the above problem is the complexity calculated taking only the database as input, while the query and the set of dependencies are considered fixed. The *combined complexity* is the complexity calculated considering as input, together with the database, also the query and the set of dependencies.

For the moment we put NCs aside and deal only with TGDs and KDs; we shall return to consider also NCs in Section 1.6.

1.2.5 The Chase Procedure

The *chase procedure* (or simply *chase*) is a fundamental algorithmic tool introduced for checking implication of dependencies [7], and later for checking query containment [8]. The chase is a process of repairing a database with respect to a set of dependencies so that the resulting instance satisfies the dependencies. We shall use the term chase interchangeably for both the procedure and its result. The chase works on an instance through the so-called TGD and KD *chase rules*.

TGD Chase Rule. Consider an instance I for a schema $\mathscr{R}$, and a TGD $\sigma = \varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \psi(\mathbf{X},\mathbf{Z})$ over $\mathscr{R}$. If σ is *applicable* to I, i.e., there exists a homomorphism h such that $h(\varphi(\mathbf{X},\mathbf{Y})) \subseteq I$, then: *(i)* define $h' \supseteq h$ such that $h'(Z_i) = z_i$ for each $Z_i \in \mathbf{Z}$, where $z_i \in \Gamma_N$ is a "fresh" labeled null not introduced before, and following lexicographically all those introduced so far, and *(ii)* add to I the set of atoms in $h'(\psi(\mathbf{X},\mathbf{Z}))$, if not already in I.

KD Chase Rule. Consider an instance I for a schema $\mathscr{R}$, and a KD κ of the form $key(r) = \mathbf{A}$ over $\mathscr{R}$. If κ is *applicable* to I, i.e., there are two (distinct) tuples $\mathbf{t}_1, \mathbf{t}_2 \in r(I)$ such that $\mathbf{t}_1[\mathbf{A}] = \mathbf{t}_2[\mathbf{A}]$, then *(i)* if there exists $i \notin \mathbf{A}$ such that both $\mathbf{t}_1[i]$ and $\mathbf{t}_2[i]$ are constants of Γ, then there is a *hard violation* of κ and the chase *fails*, otherwise *(ii)* for each $i \notin \mathbf{A}$, either replace each occurrence of $\mathbf{t}_1[i]$ with $\mathbf{t}_2[i]$, if the former follows lexicographically the latter, or vice-versa otherwise.

Given a database D for a schema $\mathscr{R}$ and set $\Sigma = \Sigma_T \cup \Sigma_K$ over $\mathscr{R}$, where Σ_T are TGDs and Σ_K are KDs, the chase algorithm for D with respect to Σ consists of an exhaustive application of the chase rules, which leads to a (possibly infinite) instance, denoted as $chase(D,\Sigma)$. In particular, $chase(D,\Sigma)$ is the instance constructed by applying *(i)* the TGD chase rule once, and *(ii)* the KD chase rule as long as it is applicable (i.e., until a fixpoint is reached). We assume that the chase algorithm is *fair*, i.e., each TGD that must be applied during the construction of $chase(D,\Sigma)$ is eventually applied. We denote as $chase^{[k]}(D,\Sigma)$ the *initial segment* of the chase of D with respect to Σ obtained by applying k times the (TGD or KD) chase rule during the construction of the chase.

Example 1.1. Let $\mathscr{R} = \{r,s\}$. Consider the set Σ of TGDs and KDs over $\mathscr{R}$ constituted by the TGDs $\sigma_1 = r(X,Y) \rightarrow \exists Z\, r(Z,X), s(Z)$ and $\sigma_2 = r(X,Y) \rightarrow r(Y,X)$, and the KD κ of the form $key(r) = \{2\}$. Let $D = \{r(a,b)\}$. During the construction of $chase(D,\Sigma)$ we first apply σ_1, and we add the atoms $r(z_1,a), s(z_1)$, where z_1 is a "fresh" null of Γ_N. Moreover, σ_2 is applicable and we add the atom $r(b,a)$. Now, the KD κ is applicable and we replace each occurrence of z_1 with the constant b; thus, we get the atom $s(b)$. ∎

The fairness assumption allows us to show that the chase of D with respect to Σ is a *universal model* of D with respect to Σ, i.e., for each $I \in mods(D,\Sigma)$, there exists a homomorphism from $chase(D,\Sigma)$ to I (see, e.g., [23, 24]). Thus, the answer to a CQ q with respect to D and Σ, if the chase does not fail, can be obtained by evaluating q over $chase(D,\Sigma)$, and discarding tuples containing at least one null [23]. If the chase fails, then $mods(D,\Sigma) = \varnothing$ and $ans(q,D,\Sigma) = \Gamma^n$.

Theorem 1.1. *Consider a CQ q/n over a schema $\mathscr{R}$, a database D for $\mathscr{R}$, a set Σ of TGDs and KDs over $\mathscr{R}$, and an n-tuple $\mathbf{t} \in \Gamma^n$. It holds that $\mathbf{t} \in ans(q, D, \Sigma)$ if and only if $\mathbf{t} \in q(chase(D, \Sigma))$ or $chase(D, \Sigma)$ fails.*

1.3 The Conceptual Model ER^+

In this section we present the conceptual model we shall deal with in this paper, and we give its semantics in terms of relational schemata with dependencies. This model, which is called ER^+, incorporates the basic features of the Entity-Relationship model [3] and OO models, including subset (or is-a) constraints on both entities and relationships. It is an extension of the one presented in [20], and we use a notation analogous to that of [20].

1.3.1 Syntax of ER^+

The schemata expressed in the ER^+ model are called *ER^+ schemata*. An ER^+ schema consists of a collection of entity, relationship, and attribute definitions over an alphabet of symbols partitioned into a set of entity symbols *Ent*, a set of relationship symbols *Rel*, and a set of attribute symbols *Att*.

An *entity definition* has the form

entity E
  isa: $E_1, \ldots, E_k$
  participates($\geqslant 1$): $R_1 : c_1, \ldots, R_\ell : c_\ell$
  participates($\leqslant 1$): $R'_1 : c'_1, \ldots, R'_m : c'_m$,

where *(i)* $E \in Ent$ is the entity to be defined, *(ii)* the isa clause specifies the set of entities to which E is related via is-a, i.e., the set of entities that are supersets of E, *(iii)* the participates($\geqslant 1$) clause specifies that an instance of the entity E must necessarily participate in relationship $R_i \in Rel$ as the c_i-th component, and *(iv)* the participates($\leqslant 1$) clause specifies that an instance of the entity E cannot participate in relationship $R'_i \in Rel$ as the c'_i-th component more than once. The isa, participates($\geqslant 1$) and participates($\leqslant 1$) clauses are optional.

A *relationship definition* has the form

relationship R among $E_1, \ldots, E_n$
  isa: $R_1[j_{1_1}, \ldots, j_{1_n}], \ldots, R_\ell[j_{\ell_1}, \ldots, j_{\ell_n}]$,

where *(i)* $R \in Rel$ is the relationship to be defined, *(ii)* the entities of *Ent* listed in the among clause are those among which the relationship R is defined (n is the *arity* of R), i.e., the i-th component of R is an instance of the entity E_i, and *(iii)* the isa clause specifies the set of relationships to which R is related via is-a, i.e., the set of relationships that are supersets of R; for each relationship R_i, we specify in square

brackets how the components of R are related to those of R_i, by giving a permutation $[j_{i_1}, \ldots, j_{i_n}]$ of the set $\{1, \ldots, n\}$. Henceforth, for brevity, given an integer $n > 0$ we will denote by $[n]$ the set $\{1, \ldots, n\}$. The isa clause is optional.

An *attribute definition* has the form

attribute A of X
qualification,

where (*i*) $A \in Att$ is the attribute to be defined, (*ii*) $X \in (Ent \cup Rel)$ is either the entity or the relationship to which the attribute is associated, and (*iii*) *qualification* consists of none, one, or both of the keywords mandatory and functional, specifying respectively that each instance of X needs to have at least one value for attribute A, and that each instance of X has a unique value for A. If both mandatory and functional are missing, the attribute is assumed by default to be optional and multi-valued, respectively.

We assume, without loss of generality, that each attribute is associated to a unique entity or relationship, i.e., different entities and relationships have disjoint sets of attributes. Also, for simplicity, we assume that attributes have atomic values.

Example 1.2. Consider the ER$^+$ schema $\mathscr{C}$ defined as follows:

entity *Member*
  participates($\geqslant 1$): *Works_in* : 1
  participates($\leqslant 1$): *Works_in* : 1
entity *Phd_student*
  isa: *Member*
entity *Professor*
  isa: *Member*
  participates($\leqslant 1$): *Leads* : 1
entity *Group*
  participates($\geqslant 1$): *Works_in* : 2, *Leads* : 2
  participates($\leqslant 1$): *Leads* : 2
relationship *Works_in* among *Member*, *Group*
relationship *Leads* among *Professor*, *Group*
  isa: *Works_in*[1,2]
attribute *memb_name* of *Member*
  mandatory, *functional*
attribute *stud_gpa* of *Phd_student*
attribute *gr_name* of *Group*
attribute *since* of *Works_in*.

It is easy to verify that $\mathscr{C}$ is actually the schema depicted in Figure 1.1. ∎

ER^+ Construct	Relational Constraint
attribute A for an entity E	$a(X,Y) \rightarrow e(X)$
attribute A for a relationship R	$a(X_1,\ldots,X_n,Y) \rightarrow r(X_1,\ldots,X_n)$
rel. R with entity E as i-th component	$r(X_1,\ldots,X_n) \rightarrow e(X_i)$
mandatory attribute A of entity E	$e(X) \rightarrow \exists Y\, a(X,Y)$
mandatory attribute A of relationship R	$r(X_1,\ldots,X_n) \rightarrow \exists Y\, a(X_1,\ldots,X_n,Y)$
functional attribute A of an entity	$key(a) = \{1\}$ (a has arity 2)
functional attribute A of a relationship	$key(a) = \{1,\ldots,n\}$ (a has arity $n+1$)
is-a between entities E_1 and E_2	$e_1(X) \rightarrow e_2(X)$
is-a between relationships R_1 and R_2 where components $1,\ldots,n$ of R_1 correspond to components $i_1,\ldots,i_n$ of R_2	$r_1(X_1,\ldots,X_n) \rightarrow r_2(X_{i_1},\ldots,X_{i_n})$
mandatory part. of E in R (i-th comp.)	$e(X_i) \rightarrow \exists \mathbf{X} r(X_1,\ldots,X_n)$
functional part. of E in R (i-th comp.)	$key(r) = \{i\}$

Table 1.2 Derivation of relational constraints from an ER^+ schema.

1.3.2 Semantics of ER^+

The semantics of an ER^+ schema $\mathscr{C}$ is defined by associating a relational schema $\mathscr{R}$ to it, and then specifying when a database for $\mathscr{R}$ satisfies all the constraints imposed by the constructs of $\mathscr{C}$. We first define the relational schema that represents the so-called *concepts*, i.e., entities, relationships and attributes, of an ER^+ schema $\mathscr{C}$ as follows: *(i)* each entity E in $\mathscr{C}$ has an associated predicate $e/1$; intuitively, $e(c)$ asserts that c is an instance of entity E, *(ii)* each attribute A of an entity E in $\mathscr{C}$ has an associated predicate $a/2$; intuitively, $a(c,d)$ asserts that d is the value of attribute A (of some entity E) associated to c, where c is an instance of E, *(iii)* each relationship R of arity n in $\mathscr{C}$ has an associated predicate r/n; intuitively, $r(c_1,\ldots,c_n)$ asserts that $\langle c_1,\ldots,c_n\rangle$ is an instance of relationship R (among entities $E_1,\ldots,E_n$), where $c_1,\ldots,c_n$ are instances of $E_1,\ldots,E_n$, respectively, and *(iv)* each attribute A of a relationship R of arity n in $\mathscr{C}$ has an associated predicate $a/(n+1)$; intuitively, $a(c_1,\ldots,c_n,d)$ asserts that d is the value of attribute A (of some relationship R of arity n) associated to the instance $\langle c_1,\ldots,c_n\rangle$ of R.

Example 1.3. Consider the ER^+ schema $\mathscr{C}$ given in Example 1.2. The relational schema $\mathscr{R}$ associated to $\mathscr{C}$ consists of the predicates *member*/1, *phd_student*/1, *professor*/1, *group*/1, *works_in*/2, *leads*/2, *memb_name*/2, *stud_gpa*/2, *gr_name*/2, *since*/3. Note that queries over an ER^+ schema are queries over the relational schema associated to it. The CQ $p(B) \leftarrow phd\_student(A), memb\_name(A,B), works\_in(A,C), since(A,C,2006), gr\_name(C,db)$ asks for the names of the students who work in the "db" group since 2006. ∎

Once we have defined the relational schema $\mathscr{R}$ for an ER^+ schema $\mathscr{C}$, we give the semantics of each construct of $\mathscr{C}$. We do that by using the dependencies introduced in Subsection 1.2.3, as shown in Table 1.2 (the relationships have arity n).

Definition 1.1. Consider an ER^+ schema $\mathscr{C}$, and let $\mathscr{R}$ be the relational schema associated to it. The set of TGDs and KDs over $\mathscr{R}$ obtained from $\mathscr{C}$ as described above is called the set of *conceptual dependencies (CDs)* associated to $\mathscr{C}$.

Example 1.4. Consider the ER^+ schema $\mathscr{C}$ given in Example 1.2. The relational schema $\mathscr{R}$ associated to $\mathscr{C}$ is given in Example 1.3. The set Σ of CDs over $\mathscr{R}$ associated to $\mathscr{C}$ is the following:

$$
\begin{array}{ll}
\mathit{member}(X) \rightarrow \exists Y\, \mathit{works\_in}(X,Y) & \mathit{leads}(X,Y) \rightarrow \mathit{professor}(X) \\
\mathit{key}(\mathit{works\_in}) = \{1\} & \mathit{leads}(X,Y) \rightarrow \mathit{group}(Y) \\
\mathit{phd\_student}(X) \rightarrow \mathit{member}(X) & \mathit{leads}(X,Y) \rightarrow \mathit{works\_in}(X,Y) \\
\mathit{professor}(X) \rightarrow \mathit{member}(X) & \mathit{memb\_name}(X,Y) \rightarrow \mathit{member}(X) \\
\mathit{key}(\mathit{leads}) = \{1\} & \mathit{member}(X) \rightarrow \exists Y\, \mathit{memb\_name}(X,Y) \\
\mathit{group}(X) \rightarrow \exists Y\, \mathit{works\_in}(Y,X) & \mathit{key}(\mathit{memb\_name}) = \{1\}, \\
\mathit{group}(X) \rightarrow \exists Y\, \mathit{leads}(Y,X) & \mathit{stud\_gpa}(X,Y) \rightarrow \mathit{phd\_student}(X) \\
\mathit{key}(\mathit{leads}) = \{2\} & \mathit{gr\_name}(X,Y) \rightarrow \mathit{group}(X) \\
\mathit{works\_in}(X,Y) \rightarrow \mathit{member}(X), & \mathit{since}(X,Y,Z) \rightarrow \mathit{works\_in}(X,Y) \\
\mathit{works\_in}(X,Y) \rightarrow \mathit{group}(Y) &
\end{array}
$$

In general, a set of CDs associated to an ER^+ schema consists of *key* and *inclusion dependencies*, where the latter are a special case of TGDs [32].

Notice that it is possible to characterize the form of relational dependencies resulting from the translation of ER^+ schemata into relational schemata. This characterization appears in [30]. Relational dependencies (key and inclusion dependencies) which are derived from ER^+ schemata are said to be in *conceptual dependency form* (abbreviated as *CD-form*). It is also possible to show that a set of key and inclusion dependencies is associated to some ER^+ schema if and only if it is in CD-form.

1.3.3 Relevant sublanguages

Two subclasses of ER^+ schemata that are of special interest are schemata where only the identity permutation can be used in is-a constraints between relationships, and where only relationships of bounded arity are allowed.

Definition 1.2. Consider a set Σ of CDs over a schema $\mathscr{R}$. Σ is a set of *identity permutation CDs (IPCDs)* if for each TGD of Σ of the form $r_1(X_1,\ldots,X_n) \rightarrow r_2(X_{i_1},\ldots,X_{i_n})$, where $\{r_1,r_2\}$ encode relationships in the associated ER^+ schema, it holds that $[i_1,\ldots,i_n] = \mathsf{id}_n$, where id_n is the identity permutation $[1,\ldots,n]$ of the set $[n]$. Σ is a set of *bounded arity CDs (BACDs)* if all the predicates encoding relationships in the associated ER^+ schema are of bounded arity.

Being able to encode ER^+ schemata into relational ones with dependencies, in our study we will exploit techniques used in the case of relational schemata with

dependencies. Henceforth, when using the term TGD (resp., KD), unless stated otherwise, we shall refer to TGDs (resp., KDs) that are part of a set of CDs (the results of this paper do not hold in general).

1.4 Separable Conceptual Dependencies

In this section we introduce the novel class of *non-conflicting CDs*. This class ensures that the TGDs and the KDs do not interact, so that answers to queries over an ER^+ schema can be computed by considering the TGDs only, and ignoring the KDs, once it is known that the initial data are consistent with respect to the schema. This semantic property is known as *separability* [11, 26].

Definition 1.3. Consider a set $\Sigma = \Sigma_T \cup \Sigma_K$ of CDs over a schema $\mathscr{R}$, where Σ_T are TGDs and Σ_K are KDs. Σ is *separable* if for every database D for $\mathscr{R}$, either $chase(D,\Sigma)$ fails or, for every CQ q over $\mathscr{R}$, $ans(q,D,\Sigma) = ans(q,D,\Sigma_T)$.

1.4.1 Definition of Non-Conflicting CDs

Before syntactically defining non-conflicting CDs, we need some preliminary technical definitions.

Consider a set $\Sigma = \Sigma_T \cup \Sigma_K$ of CDs over a schema $\mathscr{R}$, where Σ_T and Σ_K are TGDs and KDs, respectively. The *CD-graph* of Σ is a multigraph $\langle V, E, \lambda \rangle$, where V is the node set, E is the edge set, and λ is a labeling function $E \to \Sigma_T$. The node set V is the set of positions of $(\mathsf{f}_e(\mathscr{R}) \cup \mathsf{f}_r(\mathscr{R}))$. For each TGD $\sigma \in \Sigma_T$ such that $\{pred(body(\sigma)), pred(head(\sigma))\} \subseteq V$, and for each universally quantified variable X that occurs in $body(\sigma)$ at position π and in $head(\sigma)$ at position π', there exist: *(i)* an edge e from π to π' with $\lambda(e) = \sigma$, and *(ii)* for each existentially quantified variable Y in $head(\sigma)$ at position π'', a *special* edge e' from π to π'' with $\lambda(e') = \sigma$. A node $v = p[i] \in V$ is called *e-node* (resp., *r-node*) if $p \in \mathsf{f}_e(\mathscr{R})$ (resp., $p \in \mathsf{f}_r(\mathscr{R})$). Moreover, if v is an r-node and the KD $key(p) = \{i\}$ occurs in Σ_K, then v is called *k-node*.

Let G be the CD-graph of Σ. Consider an edge $u \frown v$ in G which is labeled by the TGD $r_1(X_1,\ldots,X_n) \to r_2(X_{j_1},\ldots,X_{j_n})$, where $\{r_1, r_2\} \subseteq \mathsf{f}_r(\mathscr{R})$ and $[j_1,\ldots,j_n]$ is a permutation of the set $[n]$. Intuitively, the permutation $[j_1,\ldots,j_n]$ indicates that the j_i-th component of the relationship R_1 is the i-th component of the relationship R_2. This fact can be represented by the bijective function $f_{u \frown v} : [n] \to [n]$, where for each $i \in [n]$, $f_{u \frown v}(j_i) = i$. Now, consider a cycle $C = v_1 \frown v_2 \frown \ldots \frown v_m \frown v_1$ of only r-nodes in G. The permutation associated to C, denoted as $\pi_G(C)$, is defined as the permutation $g([1,\ldots,n]) = [g(1),\ldots,g(n)]$, where $g = f_{v_m \frown v_1} \circ \ldots \circ f_{v_2 \frown v_3} \circ f_{v_1 \frown v_2}$.

We are now ready to give the formal definition of non-conflicting CDs, which is based on the notion of the CD-graph.

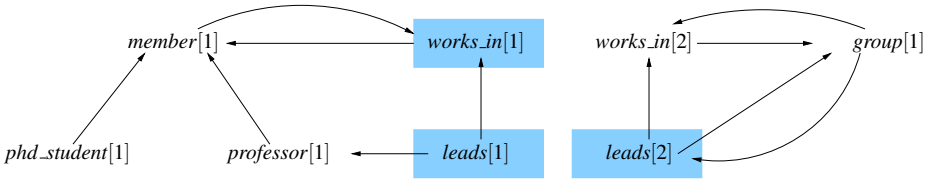

Fig. 1.2 CD-graph for Example 1.5; k-nodes are shaded and special edges are represented using broken lines.

Definition 1.4. Consider a set Σ of CDs over a schema $\mathscr{R}$, and let G be the CD-graph of Σ. Σ is *non-conflicting* if for each path $v_1^\frown v_2^\frown \ldots^\frown v_m$, for $m \geqslant 2$, in G such that v_1 is an e-node, $v_2, \ldots, v_{m-1}$ are r-nodes, and v_m is a k-node, the following two conditions are satisfied: *(i)* for each cycle C of only r-nodes in G going through v_m, $\pi_G(C) = \mathsf{id}_n$, where n is the arity of the predicate of v_m, and *(ii)* if $m \geqslant 3$ and the edge $v_1^\frown v_2$ is non-special, then there exists a path of only r-nodes from v_m to v_2.

Observe that if only the identity permutation is allowed on is-a among relationships, then the first condition stated above is satisfied trivially. Thus, given a set of IPCDs, to decide whether is non-conflicting it suffices to check only the validity of the second condition.

Example 1.5. Consider the ER^+ schema $\mathscr{C}$ given in Example 1.2; for simplicity we ignore the attributes. The set Σ of CDs associated to $\mathscr{C}$ is given in Example 1.4. The CD-graph of Σ is depicted in Figure 1.2 (edge labels are omitted for brevity). It is immediate to see that Σ is non-conflicting. ∎

In what follows we give a more involved example of non-conflicting CDs, where a cycle of only r-nodes going through a k-node occurs in the underlying CD-graph, and also arbitrary permutations (other than the identity one) are used in is-a constraints among relationships.

Example 1.6. Consider the ER^+ schema $\mathscr{C}$ depicted in Figure 1.3; the formal definition of $\mathscr{C}$ is omitted. The set Σ of CDs over $\mathscr{R} = \{e_1, \ldots, e_9, r_1, r_2, r_3\}$ associated to $\mathscr{C}$ is the following:

$$
\begin{array}{rlrl}
key(r_1) &= \{1\}, & r_2(X,Y,Z) &\rightarrow e_5(Y), \\
e_9(X) &\rightarrow \exists Y \exists Z\, r_3(Y,Z,X), & r_2(X,Y,Z) &\rightarrow e_6(Z), \\
r_1(X,Y,Z) &\rightarrow e_1(X), & r_2(X,Y,Z) &\rightarrow r_1(Z,Y,X), \\
r_1(X,Y,Z) &\rightarrow e_2(Y), & r_3(X,Y,Z) &\rightarrow e_7(X), \\
r_1(X,Y,Z) &\rightarrow e_3(Z), & r_3(X,Y,Z) &\rightarrow e_8(Y), \\
r_1(X,Y,Z) &\rightarrow r_3(Y,Z,X), & r_3(X,Y,Z) &\rightarrow e_9(Z), \\
r_2(X,Y,Z) &\rightarrow e_4(X), & r_3(X,Y,Z) &\rightarrow r_2(Y,X,Z).
\end{array}
$$

Observe that the path $e_9[1]^\frown r_3[3]^\frown r_2[3]^\frown r_1[1]$, where the edge $e_9[1]^\frown r_3[3]$ is non-special, occurs in the CD-graph G of Σ. Moreover, the edge $r_1[1]^\frown r_3[3]$ occurs in G, and thus the cycle $C = r_1[1]^\frown r_3[3]^\frown r_2[3]^\frown r_1[1]$ occurs in G; in fact, C is the only cycle of r-nodes in G going through the k-node $r_1[1]$. Clearly,

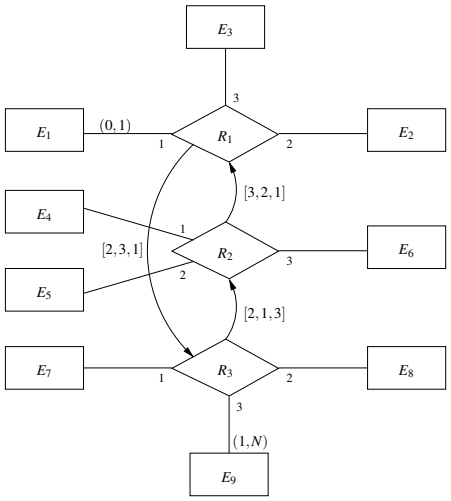

Fig. 1.3 ER^+ schema for Example 1.6.

$$f_{e_1} = \begin{cases} 3 & \text{if } x = 1, \\ 1 & \text{if } x = 2, \\ 2 & \text{if } x = 3, \end{cases} \qquad f_{e_2} = \begin{cases} 2 & \text{if } x = 1, \\ 1 & \text{if } x = 2, \\ 3 & \text{if } x = 3, \end{cases} \qquad f_{e_3} = \begin{cases} 3 & \text{if } x = 1, \\ 2 & \text{if } x = 2, \\ 1 & \text{if } x = 3, \end{cases}$$

where $e_1 = r_1[1]^\frown r_3[3]$, $e_2 = r_3[3]^\frown r_2[3]$ and $e_3 = r_2[3]^\frown r_1[1]$. By defining g to be the composition $f_{e_3} \circ f_{e_2} \circ f_{e_1}$, we get that the permutation associated to C is $\pi_G(C) = [g(1), g(2), g(3)] = [1,2,3] = \mathsf{id}_3$. Hence, the first condition of the Definition 1.4 is satisfied. Due to the edge $r_1[1]^\frown r_3[3]$, the second condition is also satisfied. Thus, Σ is non-conflicting. ∎

The following example shows that during the construction of the chase of a database with respect to a set of non-conflicting CDs, a KD may be violated which implies that the KD chase rule must be applied.

Example 1.7. Consider the set Σ of non-conflicting CDs given in Example 1.4. Let $D = \{professor(p), leads(p,g)\}$. During the construction of $chase(D,\Sigma)$, we add the atoms $member(p)$, $works\_in(p,g)$ and $works\_in(p,z_1)$, where $z_1 \in \Gamma_N$. The KD $key(works\_in) = \{1\}$ of Σ implies that p cannot participates more than once in *Works_in* as the first component. Thus, we deduce that $z_1 = g$. We must therefore replace each occurrence of z_1 with g in the part of the $chase(D,\Sigma)$ constructed so far. ∎

1.4.2 Separable CDs Vs. Non-Conflicting CDs

In this subsection we show that the non-conflicting condition is a sufficient and necessary condition for separability. We first establish that every set of non-conflicting CDs is separable. To this aim we need an auxiliary technical lemma.

Lemma 1.1. *Consider a database D for a schema $\mathscr{R}$, and a set $\Sigma = \Sigma_T \cup \Sigma_K$ of non-conflicting CDs over $\mathscr{R}$, where Σ_T are TGDs and Σ_K are KDs. If $chase(D,\Sigma)$ does not fail, then there exists a homomorphism h that maps $chase(D,\Sigma)$ to $chase(D,\Sigma_T)$.*

Proof. We proceed by induction on the number of applications of the (TGD or KD) chase rules in the construction of $chase(D,\Sigma)$. We need to show that, for each $k \geqslant 0$, there exists a homomorphism h_k that maps $chase^{[k]}(D,\Sigma)$ to a set of atoms of $chase(D,\Sigma_T)$.

BASE STEP. Clearly, for $k = 0$, $chase^{[0]}(D,\Sigma) = D \subseteq chase(D,\Sigma_T)$. Therefore, the claim holds trivially with h_0 be the identity homomorphism on $dom(D)$.

INDUCTIVE STEP. By induction hypothesis there exists a homomorphism h_{k-1} such that $h_{k-1}(chase^{[k-1]}(D,\Sigma)) \subseteq chase(D,\Sigma_T)$. We proceed by considering the two cases where the applied rule is either the TGD or the KD chase rule.

Suppose that we apply the TGD chase rule because the TGD $\sigma = r(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\, s(\mathbf{X},\mathbf{Z})$ of Σ_T is violated. This implies that there exists a homomorphism μ that maps $r(\mathbf{X},\mathbf{Y})$ to a set of atoms of $chase^{[k-1]}(D,\Sigma)$, and the atom $a = \mu'(s(\mathbf{X},\mathbf{Z}))$ is obtained, where μ' is an extension of μ as in the TGD chase rule. Clearly, $chase^{[k]}(D,\Sigma) = chase^{[k-1]}(D,\Sigma) \cup \{a\}$. Observe that the homomorphism $\lambda = h_{k-1} \circ \mu$ maps $r(\mathbf{X},\mathbf{Y})$ to a set of atoms of $chase(D,\Sigma_T)$. Since $chase(D,\Sigma_T) \models \Sigma_T$ it follows that there exists $\lambda' \supseteq \lambda$ such that $\lambda'(s(\mathbf{X},\mathbf{Z})) \in chase(D,\Sigma_T)$. Denoting $\mathbf{Z} = Z_1,\ldots,Z_m$, for $m \geqslant 0$, we define the substitution $h_k = h_{k-1} \cup \{\mu'(Z_i) \rightarrow \lambda'(Z_i)\}_{i\in[m]}$. Since none of the $\mu'(Z_i)$ occurs in h_{k-1}, it follows that h_k is a well-defined substitution. Clearly, $h_k(a) = h_k(\mu'(s(\mathbf{X},\mathbf{Z}))) = s(h_{k-1}(\mu(\mathbf{X})), h_k(\mu'(\mathbf{Z}))) = s(\lambda(\mathbf{X}),\lambda'(\mathbf{Z})) = \lambda'(s(\mathbf{X},\mathbf{Z})) \in chase(D,\Sigma_T)$. Thus, $h_k(chase^{[k]}(D,\Sigma)) \subseteq chase(D,\Sigma_T)$, as needed.

Now, suppose that we apply the KD chase rule because the KD $\kappa \in \Sigma_K$ is violated. We proceed by case analysis on the type of κ.

Assume first that κ is of the form $key(a) = \{1\}$, where $a \in \mathsf{f}_{ae}(\mathscr{R})$. This implies that in $chase^{[k-1]}(D,\Sigma)$ there exist two atoms $a = a(c_1,c_2)$ and $b = a(c_1,c_2')$. If $\{c_1,c_2,c_2'\} \subset \Gamma$, then $chase(D,\Sigma)$ fails and the claim holds trivially. The case where $\{c_1,c_2,c_2'\} \subset \Gamma_N$, and also the case where $c_1 \in \Gamma$ and $\{c_2,c_2'\} \subset \Gamma_N$ are excluded since, by definition of CDs, a and b must be the same atom which is a contradiction. Now, consider the case where $\{c_1,c_2\} \subset \Gamma$ and $c_2' \in \Gamma_N$ (the case where $\{c_1,c_2'\} \subset \Gamma$ and $c_2 \in \Gamma_N$ is symmetric). Observe that, by definition of CDs, c_2' does not occur elsewhere in $chase^{[k-1]}(D,\Sigma)$. This implies that by applying κ we just eliminate atom b, and thus $chase^{[k]}(D,\Sigma) \subset chase^{[k-1]}(D,\Sigma)$. Consequently, $h_k = h_{k-1}$.

Assume now that κ is of the form $key(a) = [n]$, where $a \in \mathsf{f}_{ar}(\mathscr{R})$. This implies that in $chase^{[k-1]}(D,\Sigma)$ there exist two atoms $a = a(c_1,\ldots,c_n,c_{n+1})$ and $b = a(c_1,\ldots,c_n,c_{n+1}')$. If $\{c_1,\ldots,c_{n+1},c_{n+1}'\} \subset \Gamma$, then $chase(D,\Sigma)$ fails

and the claim holds trivially. The cases where $\{c_1,\ldots,c_{n+1},c'_{n+1}\} \subset \Gamma_N$, or $\{c_1,\ldots,c_n\} \subset \Gamma$ and $\{c_{n+1},c'_{n+1}\} \subset \Gamma_N$, or for some $i \in [n]$, $c_i \in \Gamma$ and $\{c_1,\ldots,c_{i-1},c_{i+1},\ldots,c_{n+1},c'_{n+1}\} \subset \Gamma_N$, are excluded since, by definition of CDs, a and b must be the same atom which is a contradiction. Now, consider the case where $\{c_1,\ldots,c_{n+1}\} \subset \Gamma$ and $c'_{n+1} \in \Gamma_N$ (the case where $\{c_1,\ldots,c_n,c'_{n+1}\} \subset \Gamma$ and $c_{n+1} \in \Gamma_N$ is symmetric). By definition of CDs, c'_{n+1} does not occur elsewhere in $chase^{[k-1]}(D,\Sigma)$. By applying κ we just eliminate atom b, and hence $chase^{[k]}(D,\Sigma) \subset chase^{[k-1]}(D,\Sigma)$. Therefore, $h_k = h_{k-1}$.

We continue to consider the case where κ is of the form $key(r) = \{1\}$ (without loss of generality we assume the first attribute to form the key). This implies that in $chase^{[k-1]}(D,\Sigma)$ we have the atoms $a = r(c_1,c_2,\ldots,c_n)$ and $b = a(c_1,c'_2,\ldots,c'_n)$. If $\{c_1,\ldots,c_n,c'_2,\ldots,c'_n\} \subset \Gamma$, then $chase(D,\Sigma)$ fails and the claim holds trivially. The case where $\{c_1,\ldots,c_{i-1},c_{i+1},\ldots,c_n,c'_2,\ldots,c'_{j-1},c'_{j+1},\ldots,c'_n\} \subset \Gamma_N$ and $\{c_i,c'_j\} \subset \Gamma$, for some $i,j \in \{2,\ldots,n\}$, is excluded since, by definition of CDs, a and b must be the same atom which is a contradiction. We define $\mu = \{c'_i \to c_i\}_{2\leqslant i\leqslant n}$. During the application of the violated KD we apply μ to $chase^{[k-1]}(D,\Sigma)$ and the atom b is eliminated (the case where $\mu = \{c_i \to c'_i\}_{2\leqslant i\leqslant n}$ and a is eliminated is symmetric). Consider an arbitrary atom $c \in chase^{[k-1]}(D,\Sigma)$, where $dom(c) \cap \{c'_2,\ldots,c'_n\}$ is not empty. We are going to show that $h_{k-1}(\mu(c)) \in chase(D,\Sigma_T)$. Suppose that the atom b is obtained during the chase due to the application of the TGD $\sigma \in \Sigma_T$. We proceed by case analysis on the type of σ.

Let $\sigma = a(X_1,\ldots,X_n,Y) \to r(X_1,\ldots,X_n)$, where $a \in \mathsf{f}_{ar}(\mathscr{R})$. Observe that, by definition of CDs, the atoms generated during the chase due to the application of σ are necessarily of the form $r(d_1,\ldots,d_n)$, where $\{d_1,\ldots,d_n\} \subset \Gamma$. Therefore, this case is not possible since $\{c'_2,\ldots,c'_n\} \subset \Gamma_N$.

Assume now that $\sigma = e(X) \to \exists X_2 \ldots \exists X_n\, r(X,X_2,\ldots,X_n)$, where $e \in \mathsf{f}_e(\mathscr{R})$. Clearly, the labeled nulls $c'_2,\ldots,c'_n$ are created during the application of σ, i.e., b is the atom at which $c'_2,\ldots,c'_n$ are invented during the construction of the chase. Therefore, the atom c is obtained starting from b. Since, by induction hypothesis, $h_{k-1}(\mu(b)) \in chase(D,\Sigma_T)$, it follows that $h_{k-1}(\mu(c)) \in chase(D,\Sigma_T)$, as needed.

Suppose now that $\sigma = r'(X_1,\ldots,X_n) \to r(X_{i_1},\ldots,X_{i_n})$, where $r' \in \mathsf{f}_r(\mathscr{R})$ and $[i_1,\ldots,i_n]$ is a permutation of $[n]$. Assume first that c is such that $pred(c) \in \mathsf{f}_r(\mathscr{R})$ and $dom(c) = \{c_1,c'_2,\ldots,c'_n\}$. By definition of non-conflicting CDs, the only way to obtain b during the chase is due to a path P of the form $e[1]^\frown s_1[j_1]^\frown \ldots ^\frown s_m[j_m]^\frown r'[j_{m+1}]^\frown r[1]$, where $m \geqslant 0$, $\{j_1,\ldots,j_{m+1}\} \subseteq [n]$, $e \in \mathsf{f}_e(\mathscr{R})$ and $\{s_1,\ldots,s_m\} \subset \mathsf{f}_r(\mathscr{R})$, in the CD-graph G of Σ. Note that $e[1]$ is an e-node, and all the other nodes in P are r-nodes. Moreover, due to the existence of κ in Σ_K, $r[1]$ is a k-node. In case that c is obtained starting from b, then $h_{k-1}(\mu(c)) \in chase(D,\Sigma_T)$ since, by induction hypothesis, $h_{k-1}(\mu(b)) \in chase(D,\Sigma_T)$. The interesting case is when c is generated before b due to the existence of the path P. By definition of non-conflicting CDs, in G we have a path P' of only r-nodes from $r[1]$ to $s_1[j_1]$. Furthermore, the permutation $\pi_G(C)$, where C is the cycle $r[1]^\frown \ldots ^\frown s_1[j_1]^\frown \ldots ^\frown s_m[j_m]^\frown r'[j_{m+1}]^\frown r[1]$ in G,

is equal to id_n. Since, by induction hypothesis, $h_{k-1}(\mu(b)) \in chase(D,\Sigma_T)$, we get that $h_{k-1}(\mu(c)) \in chase(D,\Sigma_T)$. Finally, suppose that c is such that $pred(c) \in \mathsf{f}_e(\mathscr{R}) \cup \mathsf{f}_{ae}(\mathscr{R}) \cup \mathsf{f}_{ar}(\mathscr{R})$. It is clear that c is obtained starting from some atom $d \in chase^{[k-1]}(D,\Sigma)$ such that $pred(d) \in \mathsf{f}_r(\mathscr{R})$ and $dom(d) = \{c_1, c'_2, \ldots, c'_n\}$. Since $h_{k-1}(\mu(d)) \in chase(D,\Sigma_T)$, then $h_{k-1}(\mu(c)) \in chase(D,\Sigma_T)$, as needed.

The desired homomorphism from $chase(D,\Sigma)$ to $chase(D,\Sigma_T)$ is eventually $h = \bigcup_{k=0}^{\infty} h_k$. □

By using Lemma 1.1, it is easy to establish separability of non-conflicting CDs.

Theorem 1.2. *Consider a set Σ of CDs over a schema $\mathscr{R}$. If Σ is non-conflicting, then it is separable.*

Proof. Let $\Sigma = \Sigma_T \cup \Sigma_K$, where Σ_T are TGDs and Σ_K are KDs, and let D be a database for $\mathscr{R}$. Suppose that $chase(D,\Sigma)$ does not fail (otherwise the claim holds trivially). By Lemma 1.1, we get that there exists a homomorphism that maps $chase(D,\Sigma)$ to $chase(D,\Sigma_T)$. Moreover, since $chase(D,\Sigma_T)$ is a universal model of D with respect to Σ_T, and $chase(D,\Sigma) \in mods(D,\Sigma_T)$, there exists a homomorphism that maps $chase(D,\Sigma_T)$ to $chase(D,\Sigma)$. Therefore, $chase(D,\Sigma)$ and $chase(D,\Sigma_T)$ are homomorphically equivalent, which implies that for each CQ q over $\mathscr{R}$, $q(chase(D,\Sigma) = q(chase(D,\Sigma_T))$. The claim follows by Theorem 1.1. □

Interestingly, for CDs the property of being non-conflicting, it is not only sufficient for separability, but it is also necessary. This way we precisely characterize the class of separable CDs by means of a graph-based syntactic condition.

Theorem 1.3. *Consider a set Σ of CDs over a schema $\mathscr{R}$. If Σ is not non-conflicting, then it is not separable.*

Proof. Let $\Sigma = \Sigma_T \cup \Sigma_K$, where Σ_T are TGDs and Σ_K are KDs. We prove this result by exhibiting a database D such that $chase(D,\Sigma)$ does not fail, and show that there exists a Boolean CQ q such that $\langle\rangle \in ans(q,D,\Sigma)$ but $\langle\rangle \notin ans(q,D,\Sigma_T)$. Let G be the CD-graph for $\mathscr{R}$ and Σ. We proceed by considering the following two cases.

Suppose that the first condition of the Definition 1.4 is violated. Thus, there exists a path $v_1^\frown v_2^\frown \ldots^\frown v_m$, for $m \geqslant 2$, in G as in the Definition 1.4 for which the following holds. Let $v_1 = e_1[1]$ and $v_i = r_i[j_i]$, for each $i \in \{2,\ldots,m\}$. Assume that r_m is a predicate of arity n. There exists a cycle C of only r-nodes in G going through $r_m[j_m]$ such that $\pi_G(C) \neq \mathsf{id}_n$. Consider the database $D = \{e_1(c)\}$. The arc $e_1[1]^\frown r_2[j_2]$ is necessarily labeled by the TGD

$$e(X) \;\rightarrow\; \exists X_1 \ldots \exists X_{j_2-1} \exists X_{j_2+1} \ldots \exists X_n \, r_2(X_1,\ldots,X_{j_2-1},X,X_{j_2+1},X_n).$$

Thus, eventually the atom $r_m(z_1,\ldots,z_{j_m-1},c,z_{j_m+1},\ldots,z_n)$ is obtained during the chase, where, for each $i \in \{1,\ldots,j_m-1,j_m+1,\ldots,n\}$, $z_i \in \Gamma_N$. Since the cycle C exists in G, $r_m(w_1,\ldots,w_{j_m-1},c,w_{j_m+1},\ldots,w_n)$ is generated, where

$\{w_1,\ldots,w_{j_m-1},w_{j_m+1},\ldots,w_n\} = \{z_1,\ldots,z_{j_m-1},z_{j_m+1},\ldots,z_n\}$. Clearly, after the application of the KD $key(r_m) = \{j_m\}$, we get an atom a such that at least two attributes of a have the same null. Note that it is not possible to get such an atom during the construction of $chase(D,\Sigma_T)$. Therefore, we can construct an atomic Boolean CQ q, i.e., with just one atom in its body, for which there exists a homomorphism h such that $h(body(q)) = a$, and thus $h(body(q)) \in chase(D,\Sigma)$, but there is no a homomorphism h' such that $h'(body(q)) \in chase(D,\Sigma_T)$. Hence, $\langle\rangle \in ans(q,D,\Sigma)$ but $\langle\rangle \notin ans(q,D,\Sigma_T)$. Obviously, since in $dom(D)$ we have just one constant of Γ, the chase does not fail.

Now, suppose that the second condition of the Definition 1.4 is violated. This implies that there exists a path $v_1^\frown v_2^\frown \ldots^\frown v_m$, for $m \geqslant 3$, in G as in the Definition 1.4, but there is no path of only r-nodes from v_m to v_2. Assume that $v_1 = e_1[1]$ and $v_i = r_i[j_i]$, for each $i \in \{2,\ldots,m\}$. Consider the database $D = \{e_1(c), r_m(c,\ldots,c)\}$. The arc $e[1]^\frown r_2[j_2]$ is necessarily labeled by the TGD

$$e(X) \;\rightarrow\; \exists X_1 \ldots \exists X_{j_2-1} \exists X_{j_2+1} \ldots \exists X_n \, r_2(X_1,\ldots,X_{j_2-1},X,X_{j_2+1},X_n).$$

Hence, $r_2(z_1,\ldots,z_{j_2-1},c,z_{j_2+1},\ldots,z_n)$ is obtained during the chase, where, for each $i \in \{1,\ldots,j_2-1,j_2+1,\ldots,n\}$, $z_i \in \Gamma_N$. Eventually, due to the path $v_2^\frown \ldots^\frown v_m$, the atom $r_m(w_1,\ldots,w_{j_m-1},c,w_{j_m+1},\ldots,w_n)$ is generated, where $\{w_1,\ldots,w_{j_m-1},w_{j_m+1},\ldots,w_n\} = \{z_1,\ldots,z_{j_2-1},z_{j_2+1},\ldots,z_n\}$. Since v_m is a k-node, we replace w_k with c, for each $k \in \{1,\ldots,j_m-1,j_m+1,\ldots,n\}$. Thus, we get (among others) the atom $r_2(c,\ldots,c)$. Instead, in $chase(D,\Sigma_T)$ the atom $r_2(z_1,\ldots,z_{j_2-1},c,z_{j_2+1},\ldots,z_n)$ remains in place, and there is no way to obtain the atom $r_2(c,\ldots,c)$ due to the absence of a path of only r-nodes from v_m to v_2 in G. Now, let us define the atomic Boolean CQ $q = p \leftarrow r_2(c,\ldots,c)$. It is easy to see that $\langle\rangle \in ans(q,D,\Sigma)$ but $\langle\rangle \notin ans(q,D,\Sigma_T)$. Finally, since we have just one constant of Γ in $dom(D)$, there is no chase failure. □

It is straightforward to see that Theorem 1.2 holds for IPCDs and BACDs, since these two classes are special cases of CDs. Moreover, for both IPCDs and BACDs the property of being non-conflicting it is necessary for separability; the proof of this result is similar to that of Theorem 1.3. The following corrollary follows immediately.

Corollary 1.1. *Consider a set Σ of IPCDs or BACDs over a schema $\mathscr{R}$. Σ is separable if and only if is non-conflicting.*

1.5 Query Answering under Non-Conflicting CDs

Let us now investigate the data and combined complexity of the (decision) problem of conjunctive query answering under non-conflicting CDs.

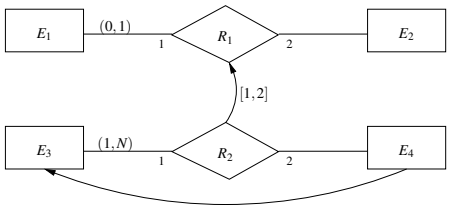

Fig. 1.4 ER^+ schema for the proof of Theorem 1.5.

1.5.1 Data Complexity

As we shall see, once we know that the chase does not fail, the problem under consideration is in the highly tractable class AC_0 in data complexity. Before we proceed further, let us recall the semantic notion of *first-order rewritability* introduced in the context of description logics [5].

A class $\mathfrak{C}$ of dependencies is *first-order rewritable*, henceforth abbreviated as *FO-rewritable*, if for every CQ q/n, and for every set Σ of dependencies in $\mathfrak{C}$, it is possible to construct a (finite) first-order query q_Σ such that, for every database D and an n-tuple $\mathbf{t} \in \Gamma^n$, $\mathbf{t} \in ans(q, D, \Sigma)$ if and only if $\mathbf{t} \in q_\Sigma(D)$. It is well-known that evaluating first-order queries is in the highly tractable class AC_0 in data complexity [33]. Recall that AC_0 is the complexity class of recognizing words in languages defined by constant-depth Boolean circuits with an (unlimited fan-in) AND and OR gates (see, e.g., [34]).

Theorem 1.4. *Consider a CQ q/n over a schema $\mathscr{R}$, a database D for $\mathscr{R}$, a set Σ of non-conflicting CDs over $\mathscr{R}$, and an n-tuple $\mathbf{t} \in \Gamma^n$. If $chase(D, \Sigma)$ does not fail, then deciding whether $\mathbf{t} \in ans(q, D, \Sigma)$ is in AC_0 in data complexity.*

Proof. Let $\Sigma = \Sigma_T \cup \Sigma_K$, where Σ_T are TGDs and Σ_K are KDs. Theorem 1.2 implies that $ans(q, D, \Sigma)$ and $ans(q, D, \Sigma_T)$ coincide. Thus, the problem whether $\mathbf{t} \in ans(q, D, \Sigma)$ is equivalent to the problem whether $\mathbf{t} \in ans(q, D, \Sigma_T)$. Recall that Σ_T is a set of inclusion dependencies. It is well-known that the class of inclusion dependencies is FO-rewritable [11], and the claim follows. □

Since IPCDs and BACDs are special cases of CDs, it is obvious that the above result holds also for non-conflicting IPCDs and non-conflicting BACDs. Notice that Theorem 1.4 does not give the exact upper bound for the data complexity of the problem under consideration. This is because we assume that the chase does not fail. The data complexity of the problem whether the chase fails will be studied in Subsection 1.5.3.

Interestingly, general CDs are not FO-rewritable; the proof of this result is similar to a non-FO-rewritability proof given in [35].

Theorem 1.5. *The class of CDs is not FO-rewritable.*

Proof. This result can be established by exhibiting a Boolean CQ q over a schema $\mathscr{R}$, a database D for $\mathscr{R}$, and a set Σ of CDs over $\mathscr{R}$ for which there is no a (finite) first-order query q_Σ such that $\langle\rangle \in ans(q, D, \Sigma)$ if and only if $\langle\rangle \in q_\Sigma(D)$. □

1.5.2 Combined Complexity

We now focus on the combined complexity of the CQ answering problem under non-conflicting CDs. First we show that, providing that the chase does not fail, the problem is in PSPACE in general, and is in NP in the case of IPCDs and BACDs.

Theorem 1.6. *Consider a CQ q/n over a schema $\mathscr{R}$, a database D for $\mathscr{R}$, a set Σ of non-conflicting CDs (resp., IPCDs or BACDs) over $\mathscr{R}$, and an n-tuple $\mathbf{t} \in \Gamma^n$. If $chase(D, \Sigma)$ does not fail, then the problem whether $\mathbf{t} \in ans(q, D, \Sigma)$ is in* PSPACE *(resp.,* NP*) in combined complexity.*

Proof. Let $\Sigma = \Sigma_T \cup \Sigma_K$, where Σ_T are TGDs and Σ_K are KDs. Since $chase(D, \Sigma)$ does not fail, Theorem 1.2 implies that our problem is equivalent to the problem whether $\mathbf{t} \in ans(q, D, \Sigma_T)$. Recall that Σ_T is a set of inclusion dependencies. Moreover, observe that if Σ is a set of BACDs, then the maximum arity over all predicates of $\mathscr{R}$ is bounded by a constant. The claim for non-conflicting CDs and non-conflicting BACDs follows from the fact that CQ answering under inclusion dependencies is in PSPACE in general, and in NP in the case of bounded arity [8].

The NP upper bound in the case of non-conflicting IPCDs does not follow immediately from the fact that CQ answering under inclusion dependencies, in the case of bounded arity, is in NP since we have to deal with predicates of unbounded arity. It is possible to show that under non-conflicting IPCDs, for query answering purposes, it suffices to consider an initial finite part of the chase whose depth is polynomial with respect to q and $\mathscr{R}$. This fact allows us to exhibit a non-deterministic PTIME algorithm which decides whether $\mathbf{t} \in chase(D, \Sigma_T)$. For the formal proof we refer the reader to [36]. □

As for Theorem 1.4, it is important to say that Theorem 1.6 does not provide the exact upper bound for the combined complexity of the problem under consideration, since we assume that the chase does not fail. The combined complexity of the problem whether the chase fails will be studied in the next subsection.

Let us now establish the desired lower bounds for CQ answering under non-conflicting CDs (resp., BACDs, IPCDs). A useful decision problem that we are going to employ in the proof of the following result is the *finite function generation (FFG)* problem introduced and studied by Kozen in [37]. Consider a finite set of functions $F \cup \{f\}$ from a set S to itself. The question is whether we can obtain f by composing functions of F. This problem is PSPACE-hard, even in the case of bijective functions.

Theorem 1.7. *CQ-Ans under non-conflicting CDs is* PSPACE*-hard in combined complexity.*

Proof. The proof is by reduction of FFG. Consider a set $F \cup \{f\}$ of bijective functions from a set S to itself. Suppose that $F = \{f_1, \ldots, f_m\}$, for $m \geqslant 1$. Moreover, without loss of generality, assume that $S = [n]$, for some integer $n \geqslant 2$.

Let q be the Boolean CQ $p \leftarrow r(c_{f(1)}, \ldots, c_{f(n)})$, $D = \{r(c_1, \ldots, c_n)\}$, where $\langle c_1, \ldots, c_n \rangle \in \Gamma^n$, and Σ be the set of CDs associated to the schema

entity E
relationship R among $\underbrace{E, \ldots, E}_{n}$
isa: $R[f_1(1), \ldots, f_1(n)], \ldots, R[f_m(1), \ldots, f_m(n)]$.

Clearly, Σ is a set of non-conflicting CDs since in the CD-graph of Σ there is no k-node. It is straightforward to see that $\langle\rangle \in ans(q, D, \Sigma)$ if and only if f can be obtained by composing functions of F. This completes the proof. □

We conclude this subsection by showing that in the case of non-conflicting IPCDs and non-conflicting BACDs CQ answering is NP-hard.

Theorem 1.8. *CQ-Ans under non-conflicting IPCDs and non-conflicting BACDs is* NP*-hard in combined complexity.*

Proof. It is well-known that the problem of CQ answering under no dependencies, even for a relational schema with fixed arity, is NP-hard in combined complexity [10]. The desired lower bounds can be easily established by providing a reduction of the above problem. □

1.5.3 Chase Failure

In this subsection we show, by exploiting a technique proposed in [26], that the problem whether the chase fails is tantamount to the CQ answering problem (providing that the chase does not fail). In what follows we are going to use the notion of union of CQs. A *union of conjunctive queries* Q of arity n is a set of CQs, where each $q \in Q$ has the same arity n and uses the same predicate symbol in the head. The answer to Q over a database D is defined as the set $Q(D) = \{\mathbf{t} \mid \text{there exists } q \in Q \text{ such that } \mathbf{t} \in q(D)\}$.

Lemma 1.2. *Consider a database D for $\mathscr{R}$, and a set $\Sigma = \Sigma_T \cup \Sigma_K$ over $\mathscr{R}$, where Σ_T are TGDs and Σ_K are KDs. If both $\mathscr{R}$ and Σ are arbitrary (resp., fixed), then we can construct in* PTIME *(resp., in* AC_0*) a database D' and a union of Boolean CQs Q such that $chase(D, \Sigma)$ fails if and only if $\langle\rangle \in Q(chase(D', \Sigma_T))$.*

Proof. Let D' be the database obtained from D by adding a fact $neq(c_1,c_2)$, for each pair of distinct constants c_1 and c_2 that appear in $dom(D)$, where neq is an auxiliary predicate symbol not occurring in $\mathscr{R}$. The union of Boolean CQs Q is constructed as follows: for every KD $key(r) = [m] \in \Sigma_K$, where $r/n \in \mathscr{R}$ (without loss of generality we assume the first $m < n$ attributes to form the key), and for each $j \in \{m+1,\ldots,n\}$, we add to Q the Boolean CQ $p \leftarrow r(\mathbf{X},Y_{m+1},\ldots,Y_n), r(\mathbf{X},Z_{m+1},\ldots,Z_n), neq(Y_j,Z_j)$, where $\mathbf{X} = X_1,\ldots,X_m$.

In the case where both $\mathscr{R}$ and Σ are arbitrary, then D' and Q can be constructed in polynomial time. In particular, the number of atoms that we add in D is less than $(|dom(D)|)^2$, while the number of Boolean CQs that we construct is $|\Sigma_K|$. Now, in the case where both $\mathscr{R}$ and Σ are fixed, the required atoms can be obtained by evaluating the first-order query

$$const(X_i) \leftarrow \bigvee_{r/n \in \mathscr{R}} \bigvee_{i \in [n]} \exists X_1 \ldots \exists X_{i-1} \exists X_{i+1} \ldots \exists X_n \; r(X_1,\ldots,X_n),$$
$$neq(X,Y) \leftarrow const(X) \wedge const(Y) \wedge X \neq Y,$$

over the database D, which is feasible in AC_0 in data complexity [33]. The above first-order query depends only on the schema $\mathscr{R}$, and thus can be constructed in constant time. Obviously Q can be constructed in constant time. It is not difficult to show that $chase(D,\Sigma)$ fails if and only if $\langle\rangle \in Q(chase(D',\Sigma_T))$. □

Notice that in the proof of the above result the fact that the TGDs and the KDs are part of a set of CDs is not needed. This implies that the above lemma holds for arbitrary TGDs and KDs. By exploiting Lemma 1.2 and Theorems 1.4, 1.6, 1.7 and 1.8, it is not difficult to establish the exact data and combined complexity of CQ answering.

Corollary 1.2. *CQ-Ans under non-conflicting CDs is in* AC_0 *in data complexity, and* PSPACE*-complete in combined complexity. Also, CQ-Ans under non-conflicting IPCDs and BACDs is* NP*-complete in combined complexity.*

1.6 Adding Negative Constraints

Negative constraints (NCs) of the form $\varphi(\mathbf{X}) \rightarrow \bot$ (see Subsection 1.2.3) can be used in order to express several relevant constructs in ER^+ schemata, e.g., disjointness between entities and relationships, and non-participation of entities to relationships, but also more general ones. The new conceptual model which arises is called $\mathsf{ER}^+_\bot$. In fact, an $\mathsf{ER}^+_\bot$ schema $\mathscr{C}$ is an ER^+ schema with an additional set $\Sigma_\bot$ of NCs over $\mathscr{R}$, where $\mathscr{R}$ is the relational schema associated to $\mathscr{C}$.

Example 1.8. Consider the ER^+ schema $\mathscr{C}$ obtained from the one in Example 1.2 (see Figure 1.1) by adding the entity *Pension_scheme* and a relationship *Enrolled* among *Member* and *Pension_scheme*, without any cardinality constraints. The fact that students and professors are disjoint sets can be represented with the

NC $phd\_student(X), professor(X) \rightarrow \bot$ (entity disjointness). Moreover, the fact that a student cannot be enrolled in a pension scheme (i.e., it does not participate to *Enrolled* as the first component) can be represented by the NC $phd\_student(X)$, $enrolled(X,Y) \rightarrow \bot$ (non-participation of an entity to a relationship). ∎

A set of CDs and NCs is separable if the answers to queries can be computed by considering the TGDs only, and ignoring the KDs and the NCs, once it is known that the chase does not fail, and also that the chase satisfies the set of NCs.

Definition 1.5. Consider a set $\Sigma = \Sigma_T \cup \Sigma_K$ of CDs over a schema $\mathscr{R}$, where Σ_T are TGDs and Σ_K are KDs, and a set $\Sigma_\bot$ of NCs over $\mathscr{R}$. $\Sigma \cup \Sigma_\bot$ is *separable* if for every database D for $\mathscr{R}$, we have that either $chase(D,\Sigma)$ fails or $chase(D,\Sigma) \not\models \Sigma_\bot$ or, for every CQ q over $\mathscr{R}$, $ans(q,D,\Sigma) = ans(q,D,\Sigma_T)$.

It is straightforward to see that given a set Σ of CDs and a set $\Sigma_\bot$ of NCs, the fact that Σ is non-conflicting is sufficient for separability of $\Sigma \cup \Sigma_\bot$; this follows immediately from Theorem 1.2. The question that comes up is whether the non-conflicting property is also necessary for separability. It is not difficult to show that there exists a set Σ of CDs and a set $\Sigma_\bot$ of NCs such that Σ is not non-conflicting, but $\Sigma \cup \Sigma_\bot$ is separable.

Interestingly, if we consider only *strongly consistent* schemata [19], then the non-conflicting property is also necessary for separability. An $\mathsf{ER}^+_\bot$ schema $\mathscr{C}$, where Σ is the of CDs over $\mathscr{R}$ associated to $\mathscr{C}$ and $\Sigma_\bot$ is the of NCs over $\mathscr{R}$ associated to $\mathscr{C}$, is strongly consistent if there exists a (possibly infinite) instance I for $\mathscr{R}$ such that $I \models \Sigma \cup \Sigma_\bot$, and for each $e \in \mathsf{f}_e(\mathscr{R})$, $e(I) \neq \varnothing$. It is possible to show that the problem whether an $\mathsf{ER}^+_\bot$ schema $\mathscr{C}$, where the set of CDs associated to $\mathscr{C}$ is non-conflicting, is strongly consistent is PSPACE-complete. The formal proofs are omitted and can be found in [36].

We conclude this section by showing that the addition of NCs does not alter the complexity of CQ answering under non-conflicting CDs.

Theorem 1.9. *CQ-Ans under non-conflicting CDs and NCs is in* AC_0 *in data complexity, and* PSPACE*-complete in combined complexity. Also, CQ-Ans under non-conflicting IPCDs and BACDs is* NP*-complete in combined complexity.*

Proof. Consider a CQ q/n over a schema $\mathscr{R}$, a database D for $\mathscr{R}$, a set $\Sigma = \Sigma_T \cup \Sigma_K$ of non-conflicting CDs (resp., IPCDs, BACDs) over $\mathscr{R}$, where Σ_T are TGDs and Σ_K are KDs, a set $\Sigma_\bot$ of NCs over $\mathscr{R}$, and an n-tuple $\mathbf{t} \in \Gamma^n$. It holds that $\mathbf{t} \in ans(q,D,\Sigma \cup \Sigma_\bot)$ if and only if $chase(D,\Sigma)$ fails or $chase(D,\Sigma) \not\models \Sigma_\bot$ or $\mathbf{t} \in ans(q,D,\Sigma)$. Therefore, we can decide whether $\mathbf{t} \in ans(q,D,\Sigma \cup \Sigma_\bot)$ by applying the following simple algorithm; given a NC $\nu = \varphi(\mathbf{X}) \rightarrow \bot$, we denote by q_ν the Boolean CQ $p \leftarrow \varphi(\mathbf{X})$:

1. Construct the database D' from D and the union of Boolean CQs Q from Σ_K, as described in the proof of Lemma 1.2.
2. If $\langle\rangle \in Q(chase(D',\Sigma_T))$, then *accept*.
3. If there exists $\nu \in \Sigma_\bot$ such that $\langle\rangle \in ans(q_\nu,D,\Sigma)$, then *accept*.

4. If $\mathbf{t} \in ans(q, D, \Sigma)$, then *accept*; otherwise, *reject*.

Soundness and completeness of the above algorithm follows immediately from Lemma 1.2. Step 1 can be carried out in PTIME in general, and in AC_0 in case both $\mathscr{R}$ and Σ are fixed (see Lemma 1.2). Since Σ is a set of non-conflicting CDs (resp., IPCDs, BACDs), the claim follows immediately from Corollary 1.2. □

1.7 Discussion

In this chapter we have introduced an extension of Chen's Entity Relationship model, that is, the ER^+ family. ER^+ is a very natural formalism for modeling data, as it builds upon the ER model, which has been used for decades in database design. We argue that the ER model is a powerful tool for ontological query answering, which at the same time is not awkward to understand and use; in particular, practitioners and even persons without any technical knowledge can profitably use ER (and its extensions) to express ontologies.

We have presented three ER^+ languages which, by means of mild restrictions specified by efficiently-testable syntactic conditions, enjoy the separability property and first-order rewritability. The latter ensures not only that query answering under our languages is in the low complexity class AC_0, but that answering can be done by evaluating a suitable SQL query over the initial data. This opens the possibility of efficient implementation of query answering under non-conflicting ER^+ schemata on large data sets. Our study also pinpoints the complexity of query answering under our three non-conflicting ER^+ languages with respect to both data and combined complexity.

It is worth noticing that the problem we deal with in this chapter has several other applications. It is tightly related to the analogous problem of query answering on incomplete data under constraints [11, 12] and to semantic data integration [13]. Moreover, our problem of query answering is mutually reducible to the query containment problem (see, e.g., [14]), therefore all our results carry on to the latter.

Finally, we remind the reader that the complexity results for most DL-Lite languages come, in fact, as special cases of our results on ER^+. For instance, non-conflicting $\mathsf{ER}^+_{\mathscr{B}}$ with negative constraints is strictly more expressive than the languages *DL-Lite*$_{\mathscr{F}}$ and *DL-Lite*$_{\mathscr{R}}$ [5, 6]. This shows that, if DL-Lite languages are useful and important for modeling database schemata and ontologies, then *a fortiori* also ER^+ is.

References

1. A. Calì, G. Gottlob, A. Pieris (2009) Tractable query answering over conceptual schemata. In: Proc. of ER 175–190.

2. A. Calì, G. Gottlob, A. Pieris (2010) Query answering under expressive Entity-Relationship schemata. In: Proc. of ER 347–361.
3. P. P. Chen (1976) The Entity-Relationship model: Twards a unified view of data. In: ACM Trans. Database Syst. 1:9–36.
4. D. Calvanese, M. Lenzerini, D. Nardi (1998) Description logics for conceptual data modeling. In: Logics for Databases and Information Systems 229–263.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati (2007) Tractable reasoning and efficient query answering in description logics: The DL-Lite family. In: J. Autom. Reasoning 39:385–429.
6. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati (2008) Linking data to ontologies. In: J. Data Semantics 10:133–173.
7. D. Maier, A. O. Mendelzon, Y. Sagiv (1979) Testing implications of data dependencies. In: ACM Trans. Database Syst. 4:455–469.
8. D. S. Johnson, A. C. Klug (1984) Testing containment of conjunctive queries under functional and inclusion dependencies. In: J. Comput. Syst. Sci. 28:167–189.
9. A. Calì, D. Lembo, R. Rosati (2003) Query rewriting and answering under constraints in data integration systems. In: Proc. of IJCAI 16–21.
10. A. K. Chandra, P. M. Merlin (1977) Optimal implementation of conjunctive queries in relational data bases. In: Proc. of STOC 77–90.
11. A. Calì, D. Lembo, R. Rosati (2003) Decidability and complexity of query answering over incosistent and incomplete databases. In: Proc. of PODS 260–271.
12. R. van der Meyden (1998) Logical approaches to incomplete information: A survey. In: Logics for Databases and Information Systems 307–356.
13. M. Lenzerini (2002) Data integration: A theoretical perspective. In: Proc. of PODS 233–246.
14. A. Calì, G. Gottlob, M. Kifer (2008) Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. of KR 70–80.
15. M. Gogolla, U. Hohenstein (1991) Towards a semantic view of an extended Entity-Relationship model. In: ACM Trans. Database Syst. 16:369–416.
16. V. M. Markowitz, A. Shoshani (1992) Representing extended Entity-Relationship structures in relational databases: A modular approach. In: ACM Trans. Database Syst. 17:423–464.
17. V. M. Markowitz, J. A. Makowsky (1990) Identifying extended Entity-Relationship object structures in relational schemas. In: IEEE Trans. Software Eng. 16:777–790.
18. G. D. Battista, M. Lenzerini (1989) A deductive method for Entity-Relationship modeling. In: Proc. VLDB 13–21.
19. A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, M. Zakharyaschev (2007) Reasoning over extended ER models, In: Proc. of ER 277–292.
20. A. Calì, D. Calvanese, G. D. Giacomo, M. Lenzerini (2001) Accessing data integration systems through conceptual schemas, In: Proc. of ER 270–284.
21. M. Ortiz D. Calvanese, T. Eiter (2006) Characterizing data complexity for conjunctive query answering in expressive description logics. In: Proc. of AAAI.
22. D. Calvanese, G. D. Giacomo, M. Lenzerini (1998) On the decidability of query containment under constraints. In: Proc. of PODS 149–158.
23. R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa (2005) Data exchange: Semantics and query answering. In: Theor. Comput. Sci. 336:89–124.
24. A. Deutsch, A. Nash, J. B. Remmel (2008) The chase revisisted. In: Proc. of PODS 149–158.
25. B. Marnette (2009) Generalized schema-mappings: From termination to tractability. In: Proc. of PODS 13–22.
26. A. Calì, G. Gottlob, T. Lukasiewicz (2009) A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of PODS 77–86.
27. A. Calì, G. Gottlob, A. Pieris (2010) Advanced processing for ontological queries. In: PVLDB 3:554–565.
28. A. Calì, G. Gottlob, A. Pieris (2010) Query answering under non-guarded rules in Datalog+/-. In: Proc. of RR 1–17.
29. A. Calì, G. Gottlob, M. Kifer, T. Lukasiewicz, A. Pieris (2010) Ontological reasoning with F-Logic Lite and its extensions. In: Proc. of AAAI.

30. A. Calì, D. Martinenghi (2010) Querying incomplete data over extended ER schemata. In: TPLP 10:291–329.
31. M. Y. Vardi (1982) The complexity of relational query languages. In: Proc. of STOC 137–146.
32. S. Abiteboul, R. Hull, V. Vianu (1995) Foundations of Databases, Addison-Wesley.
33. M. Y. Vardi (1995) On the complexity of bounded-variable queries. In: Proc. of PODS 266–276.
34. C. H. Papadimitriou (1999) Computational Complexity, Addison-Wesley.
35. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati (2004) What to ask to a peer: Ontology-based query reformulation. In: Proc. of KR 469–478.
36. A. Calì, G. Gottlob, A. Pieris (2011) Ontological query answering under expressive Entity-Relationship schemata. Unpublished manuscript available from the authors.
37. D. Kozen (1977) Lower bounds for natural proof systems. In: Proc. of FOCS 254–266.
38. J. de Bruijn, S. Heymans (2007) Logical foundations of (e)RDF(S): Complexity and reasoning. In: Proc. of ISWC 86–99.