Reasoning about Entity Relationship Diagrams with Complex Attribute Dependencies

Carsten Lutz
LuFG Theoretical Computer Science, RWTH Aachen, Germany
Email: lutz@informatik.rwth-aachen.de

1 Motivation

Entity Relationship (ER) diagrams are among the most popular formalisms for the support of database design [6, 10, 5]. During the last years, the initially rather simple ER formalism has been extended by various means of expressivity to account for new, more complex application areas such as schema integration for data warehouses [10, 11]. Designing a conceptual model with such enriched ER diagrams is a nontrivial task: there exist complex interactions between the various means of expressivity, which quite often results in unnoticed inconsistencies in the ER schemas and in implicit ramifications of the modeling that have not been intended by the designer. To address this problem, Description Logics (DLs) have been proposed and successfully used as a tool for reasoning about ER diagrams and thereby detecting the aforementioned anomalies [4, 5, 7].

In the classical ER formalism, elementary properties of entities such as their size, age, or prize play only a minor role: these properties are represented by so-called attributes, which the database designer can merely list and associate with a type such as integer and string. However, in the actual database, there usually exist various constraints on attribute values that have to be satisfied. For example, if the Employee entity of a company is equipped with two attributes birth-year and recruitment-year, then we certainly want to enforce that, for all instances of Employee, the value of birth-year is smaller than the value of recruitment-year. Such "attribute dependencies" for relational databases are well-known and have been investigated in, e.g., [8, 3, 16]. However, research about this topic has focussed on the consistency and implication problems for sets of attribute dependencies. If reasoning about ER schemas (translated into an appropriate DL) is used to infer properties of the conceptual model, it seems more appropriate to strive for an *integrated* approach to reasoning with the conceptual model and attribute dependencies. Indeed, the presence of such dependencies can have a severe impact on, e.g., the consistency of an ER schema. In this paper, we propose such an integrated approach by extending ER diagrams with (various kinds of) attribute dependencies and showing how the extended ER formalism can be translated into Description Logics with concrete domains.

2 ER Diagrams

ER schemas are constructed from entities, relationships and attributes. We omit a more thorough discussion of their syntax for brevity and refer to the non-abridged version of this paper [15]. The semantics of ER schemas is described in terms of database states. Basic domains are represented by domain symbols D and associated sets Δ_D.

Definition 2.1. A *database state* is a tuple $(\Delta_{\mathcal{B}}, \cdot^{\mathcal{B}})$, where $\Delta_{\mathcal{B}}$ is a nonempty finite set disjoint from all basic domains and $\cdot^{\mathcal{B}}$ is a function mapping every entity E to a set $E^{\mathcal{B}} \subseteq \Delta_{\mathcal{B}}$, every attribute A to a partial function $A^{\mathcal{B}}$ from $\Delta_{\mathcal{B}}$ to the union, for all basic domains D, of Δ_D, and every relationship R to a set of *relation instances over* $\Delta_{\mathcal{B}}$, i.e., to a set of partial functions from the set of ER-roles to $\Delta^{\mathcal{B}}$. The relation instance r that maps ER-role U_i to e_i, for $i \in \{1, \ldots, k\}$, is denoted $[U_1 : e_1, \ldots, U_k : e_k]$.

To give a semantics to ER schemas, it remains to define the set of database states that they describe.

Definition 2.2. A database state $\mathcal{B}$ is *legal for* an ER schema $\mathcal{S}$ if it satisfies the following conditions:

- For each entity E, if E has an attribute A with basic domain D, then for each instance $e \in E^{\mathcal{B}}$, $A^{\mathcal{B}}(e)$ is defined and in Δ_D.
- For each relationship R of arity k between entities $E_1, \ldots, E_k$, to which R is connected by means of ER-roles $U_1, \ldots, U_k$ respectively, all instances of R are of the form $[U_1 : e_1, \ldots, U_k : e_k]$, where $e_i \in E_i^{\mathcal{B}}$, for $i \in \{1, \ldots, k\}$.
- For each ER-role U associated to relationship R and entity E, and for each instance e of E, it holds that $\mathsf{cmin}_{\mathcal{S}}(U) \leq \sharp\{r \in R^{\mathcal{B}} \mid r(U) = e\} \leq \mathsf{cmax}_{\mathcal{S}}(U)$, where $\mathsf{cmin}_{\mathcal{S}}(U)$ ($\mathsf{cmax}_{\mathcal{S}}(U)$) denotes the lower (upper) cardinality constraint on U in $\mathcal{S}$.
- For each pair of entities E_1, E_2 related by an ISA-link, we have $E_1^{\mathcal{B}} \subseteq E_2^{\mathcal{B}}$.

3 Reasoning with ER Diagrams

The Description Logic $\mathcal{ALCQI}$ extends the basic propositionally closed DL $\mathcal{ALC}$ with qualifying number restrictions and inverse roles. As described in [5], this logic is well-suited for reasoning about entity relationship schemas: ER schemas can be translated into general $\mathcal{ALCQI}$ TBoxes (also known as GCIs), and $\mathcal{ALCQI}$ reasoning procedures can then be used to check, for a given ER schema $\mathcal{S}$, (i) whether an entity or relationship G in $\mathcal{S}$ is consistent (i.e., whether there exists a legal database state $\mathcal{B}$ for $\mathcal{S}$ such that $G^{\mathcal{B}} \neq \emptyset$) and (ii) whether $\mathcal{S}$ implies ISA links and cardinality constraints that are not explicitly represented. Throughout this paper, when talking of "reasoning" with a DL, we mean deciding the satisfiability of concepts w.r.t. general TBoxes. Observe that, since database states are required to be *finite*, we are usually interested in *finite model* reasoning. It is known that finite model reasoning with $\mathcal{ALCQI}$ is decidable, more precisely in 2-ExpTime [4]. In this section, we describe the standard way of encoding ER schemas as $\mathcal{ALCQI}$ TBoxes. Later, this encoding will be modified such that attribute dependencies can be taken into account.

The TBox $\phi(\mathcal{S})$ derived from an ER schema $\mathcal{S}$ is defined as follows: we introduce a concept name $\phi(G)$ for each entity symbol, relationship symbol, and domain symbol G in $\mathcal{S}$, and a role name $\phi(H)$ for each ER role symbol and attribute symbol H in $\mathcal{S}$. The knowledge base $\phi(\mathcal{S})$ then contains the following concept equations:

1. For each entity E with attributes $A_1, \ldots, A_k$, with domains $D_1, \ldots, D_k$, the equation
$$\begin{array}{rcl}\phi(E) & \sqsubseteq & \exists \phi(A_1).\phi(D_1) \sqcap \cdots \sqcap \exists \phi(A_k).\phi(D_k) \\ & & \sqcap\, \exists^{=1} \phi(A_1) \sqcap \cdots \sqcap \exists^{=1} \phi(A_k).\end{array}$$

2. For each relationship R of arity k between entities $E_1, \ldots, E_k$, to which R is connected by means of ER-roles $U_1, \ldots, U_k$ respectively, the equations
$$\begin{array}{rcl}\phi(R) & \sqsubseteq & \forall \phi(U_1).\phi(E_1) \sqcap \cdots \sqcap \forall \phi(U_k).\phi(E_k) \\ & & \sqcap\, \exists^{=1} \phi(U_1) \sqcap \cdots \sqcap \exists^{=1} \phi(U_k) \\ \phi(E_i) & \sqsubseteq & \forall \phi(U_1)^{-}.\phi(R) \sqcap \cdots \sqcap \forall \phi(U_k)^{-}.\phi(R).\end{array}$$

3. For each ER-role U associated to relationship R and entity E, the equations
$$\begin{array}{rcl}\phi(E) & \sqsubseteq & \exists^{\geq \mathsf{cmin}_{\mathcal{S}}(U)} \phi(U)^{-} \\ \phi(E) & \sqsubseteq & \exists^{\leq \mathsf{cmax}_{\mathcal{S}}(U)} \phi(U)^{-} \text{ if } \mathsf{cmax}_{\mathcal{S}}(U) \neq \infty.\end{array}$$

4. For each pair of entities E_1, E_2 such that there is an ISA link between E_1 and E_2, the equation $E_1 \sqsubseteq E_2$.

5. For each pair of domain symbols C_1, C_2 such that $C_1 \neq C_2$, the equation $\phi(C_1) \sqsubseteq \neg \phi(C_2)$.

The correctness of this encoding is proved in [4]. Note that we generally view attributes to be single-valued rather than multi-valued. In the following, we will advocate a more sophisticated treatment of attributes and basic domains.

4 Adding Attribute Dependencies

We propose to extend ER schemas with three kinds of attribute dependencies: *entity attribute constraints (EADs)*, *relationship attribute constraints (RADs)*, and

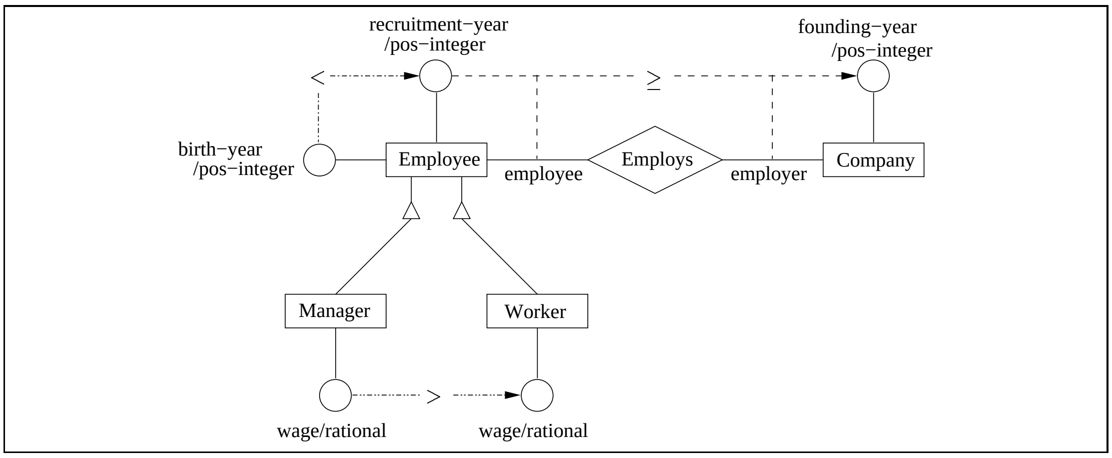

Figure 1: The augmented ER schema

global attribute constraints (GADs). Figure 1 shows an extended ER schema that illustrates all three types of attribute dependencies:

EAD. The dash-dotted edge between the birth-year and recruitment-year attributes of the Employee entity denotes an EAD. Note that this edge is labeled with a predicate ("$<$") whose arity coincides with the number of involved attributes. The EAD states that every Employee must be born before she is employed. Observe that, in EADs, all involved attributes are associated with the same entity.

RAD. The dashed edges between the attribute recruitment-year of entity Employee, the ER-roles employee and employer of the relationship Employs, and the attribute founding-year of entity Company denotes a RAD. It states that companies do not hire employees prior to their founding. In general, all ER-roles participating in a RAD must be associated with the same relationship. However, the arity of a RAD for a relationship R may be smaller than R's arity.

GAD. The dash-doubledotted edge between the wage attribute of the Manager entity and the wage attribute of the Worker entity denotes a GAD. It states that every manager earns more than any worker. For a GAD, we make no assumptions on whether and how the entities of the involved attributes are related in the ER schema.

We do not pose any restrictions on the arity of attribute dependencies. Note that (unless the involved predicate is unary), EADs are not special cases of GADs: if, for example, we replace the dash-dotted edge in Figure 1 by a dash-doubledotted edge, then the modified dependency states that every employee was born before any employee (i.e., not necessarily the same) was hired.

In the following, we refine the notion "legal database" to capture the semantics of attribute dependencies. We assume that every predicate P appearing in attribute dependencies is associated with an arity n, an n-tuple of basic domains $(D_1, \ldots, D_n)$ called P's *type*, and with a fixed extension $\mathcal{E}(P) \subseteq \Delta_{D_1} \times \cdots \times \Delta_{D_n}$.

In the presence of attribute dependencies, we require that any legal database instance $\mathcal{B}$ additionally satisfies the following conditions:

- For every EAD of arity k referring to (i) attributes A_i with basic domains D_i, for $i \in \{1, \ldots, k\}$, of an entity E, and (ii) a predicate P of arity k and type $(D_1, \ldots, D_k)$, the following holds: if $e \in E$, then $(A_1^{\mathcal{B}}(e), \ldots, A_k^{\mathcal{B}}(e)) \in \mathcal{E}(P)$.
- For every RAD of arity k referring to (i) a relationship R of arity n equipped with ER-roles $U_1, \ldots, U_n$ connecting entities $E_1, \ldots, E_n$, respectively, (ii) indexes h_i with $1 \leq h_i \leq n$, for $i \in \{1, \ldots, k\}$, (iii) attributes A_i of the entity E_{h_i} with basic domains D_i, for $i \in \{1, \ldots, k\}$, and (iv) a predicate P of arity k and type $(D_1, \ldots, D_k)$, the following holds: if $[U_1 : e_1, \ldots, U_n : e_n] \in R^{\mathcal{B}}$, then $(A_1^{\mathcal{B}}(e_{h_1}), \ldots, A_k^{\mathcal{B}}(e_{h_k})) \in \mathcal{E}(P)$.
- For every GAD of arity k referring to (i) attributes A_i of entities E_i with basic domains D_i, for $i \in \{1, \ldots, k\}$, and (ii) a predicate P of arity k and type $(D_1, \ldots, D_k)$, the following holds: if $e_i \in E_i$, for $i \in \{1, \ldots, k\}$, then $(A_1^{\mathcal{B}}(e_1), \ldots, A_k^{\mathcal{B}}(e_k)) \in \mathcal{E}(P)$.

How can attribute dependencies be captured by Description Logics? The general idea is to extend $\mathcal{ALCQI}$ with concrete domains [1, 12] and then to modify the standard encoding accordingly. Recall that a concrete domain $\mathcal{D}$ is a pair $(\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$, where $\Delta_{\mathcal{D}}$ is a set and $\Phi_{\mathcal{D}}$ a set of predicate names. Each predicate name $P \in \Phi_{\mathcal{D}}$ is associated with an arity n and an n-ary predicate $P^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}^n$. The logic $\mathcal{ALCQI}(\mathcal{D})$ is obtained from $\mathcal{ALCQI}$ by adding two concept constructors $\exists U_1, \ldots, U_n.P$ and $\forall U_1, \ldots, U_n.P$, where $P \in \Phi_{\mathcal{D}}$ is a predicate of arity n and the U_i are *paths*, i.e., sequences $R_1 \cdots R_k g$, with R_j a role name or the inverse of a role name, and g a concrete feature.[1] More details can be found in [15, 14].

We now modify the standard encoding of ER schemas from Section 3 to take into account attribute dependencies. This time, the target language is $\mathcal{ALCQI}(\mathcal{D})$. For a start, assume that we admit only EADs, but no RADs and GADs. Clearly, basic domains can be viewed as concrete domains. However, we assume that *all* basic domains are represented by a single concrete domain $\mathcal{D}$: if $D_1, \ldots, D_m$ are the basic domains, then we have $\Delta_{\mathcal{D}} = \Delta_{D_1} \cup \cdots \cup \Delta_{D_m}$ and $\Phi_{\mathcal{D}}$ contains (among others) a predicate $\top_{D_i}$ for each $1 \leq i \leq m$ such that $\top_{D_i}^{\mathcal{D}} = \Delta_{D_i}$. Unifying all basic domains in a single concrete domain is vital for admitting predicates that have a "mixed type", i.e., predicates P with extension $\mathcal{E}(P) \subseteq D_1 \times \cdots \times D_n$ such that $D_i \neq D_j$ for some $1 \leq i, j \leq n$.

In contrast to what was done in Section 3, we introduce a concrete feature $\phi(A)$ for each attribute A (instead of a role name). When translating ER schemas $\mathcal{S}$ to TBoxes, we replace Rule 1 from the translation in Section 3 by the following one:

[1]Note that we do not restrict ourselves to functional roles inside the concrete domain constructors.

1'. For each entity E with attributes $A_1, \ldots, A_k$, with domains $D_1, \ldots, D_k$, add the equation $\phi(E) \sqsubseteq \exists \phi(A_1).\top_{D_1} \sqcap \cdots \sqcap \exists \phi(A_k).\top_{D_k}$.

Moreover, we need additional concept equations to deal with EADs:

6. For every EAD of arity k referring to (i) attributes A_i, for $i \in \{1, \ldots, k\}$, of an entity E, and (ii) a predicate P of arity k, add the equation $\phi(E) \sqsubseteq \exists \phi(A_1), \ldots, \phi(A_k).P$.

It is rather straightforward to prove the correctness of this encoding by slightly modifying the correctness proof of the original encoding given in [4]. But is reasoning with $\mathcal{ALCQI}(\mathcal{D})$ still decidable? Let us defer this question for a moment and instead note that the described encoding uses only a restricted variant of the concrete domain constructors, as introduced in [9]: only concrete features are used inside these constructors, but no paths of length greater one. We use the name $\mathcal{ALCQI}^-(\mathcal{D})$ to denote the fragment of $\mathcal{ALCQI}(\mathcal{D})$ that is obtained by restricting the concrete domain constructors in the described way. The restrictedness allows us to make some claims about reasoning with *infinite models*: as proved in [2], the extension of any decidable Description Logic with the restricted concrete domain constructors is again decidable if the employed concrete domain satisfies some minor conditions. Moreover, it seems rather easy to prove an ExpTime upper bound for infinite model reasoning with $\mathcal{ALCQI}^-(\mathcal{D})$ analogous to the proof of Theorem 2.15 in [12]. Now what about *finite model* reasoning? It is not hard to see that, analogous to the proof of Theorem 2.14 in [12], we can reduce reasoning with $\mathcal{ALCQI}^-(\mathcal{D})$ to reasoning with $\mathcal{ALCQI}$ (at the cost of an exponential blow-up in the concept/TBox size). Since finite model reasoning with $\mathcal{ALCQI}$ is known to be in 2-ExpTime [4], we thus obtain that finite model reasoning with $\mathcal{ALCQI}^-(\mathcal{D})$ is decidable in 3-ExpTime (which is presumably not the smallest possible upper bound).

If RADs are admitted, we can neither offer general decidability results for infinite model reasoning nor for finite model reasoning. Let us consider the obvious way of encoding RADs:

7. For every RAD of arity k referring to (i) a relationship R of arity n equipped with ER-roles $U_1, \ldots, U_n$ connecting entities $E_1, \ldots, E_n$, respectively, (ii) indexes h_i with $1 \leq h_i \leq n$, for $i \in \{1, \ldots, k\}$, (iii) attributes A_i of the entity E_{h_i}, for $i \in \{1, \ldots, k\}$, and (iv) a predicate P of arity k, add the equation $\phi(R) \sqsubseteq \forall \phi(U_{h_1})\, \phi(A_1), \ldots, \phi(U_{h_k})\, \phi(A_1).P$.

In this encoding, paths of length 2 appear inside the concrete domain constructors. Thus, we cannot use $\mathcal{ALCQI}^-(\mathcal{D})$, but have to resort to $\mathcal{ALCQI}(\mathcal{D})$. Unfortunately, it is well-known that reasoning with $\mathcal{ALCQI}(\mathcal{D})$ and general TBoxes is undecidable for a large class of concrete domains [13]. Does this mean that reasoning with RADs and GADs is not possible? Certainly not! It just means that we have to be very careful in choosing our basic domains and the predicates admitted in RADs and GADs.

For the remainder of this section, assume that there exists only a single basic domain: the rational numbers. Moreover, assume that only the following predicates are available for EADs and RADs: (i) unary predicates P_q for $P \in \{<, \leq, =, \neq, \geq, >\}$ and $q \in \mathbb{Q}$, and (ii) binary predicates $<$, $\leq$, $=$, $\neq$, $\geq$, and $>$. Call the corresponding *concrete* domain Q. We can then translate EADs as before and RADs as follows: For each pair (A, E), with A attribute and E entity, introduce a concrete feature $g_{A,E}$. Then augment the encoding by the following rule:

7′. For every RAD of arity k referring to (i) a relationship R of arity n equipped with ER-roles $U_1, \ldots, U_n$ connecting entities $E_1, \ldots, E_n$, respectively, (ii) indexes h_i with $1 \leq h_i \leq n$, for $i \in \{1, \ldots, k\}$, (iii) attributes A_i of the entity E_{h_i}, for $i \in \{1, \ldots, k\}$, and (iv) a predicate P of arity k, add the equations

$$\begin{array}{lcll} \phi(R) & \sqsubseteq & \forall \phi(U_{h_i})\, \phi(A_i), g_{A_i,E_i}.= & \text{for } i \in \{1, \ldots, k\} \\ \phi(R) & \sqsubseteq & \exists g_{A_1,E_1}, \ldots, g_{A_k,E_k}.P & \end{array}$$

Note that k must be either 1 or 2 since we admit only unary and binary predicates. In this encoding, the concrete domain constructors appear only in the forms $\forall R g_1, g_2.P$ and $\exists g_1, \ldots, g_n.P$. Let $\mathcal{ALCQI}^*(\mathcal{D})$ be the restriction of $\mathcal{ALCQI}(\mathcal{D})$ to this form of concrete domain constructor. In [14], we show that infinite model reasoning with $\mathbb{Q}$-$\mathcal{SHIQ}$, of which $\mathcal{ALCQI}^*(\mathsf{Q})$ (i.e., $\mathcal{ALCQI}^*(\mathcal{D})$ instantiated with the concrete domain Q) is a fragment, is decidable in ExpTime. A decidability result for finite model reasoning is not known.

Let us now consider GADs. As we have already noted, unary GADs are nothing but EADs and can thus be translated using Rule 6. To deal with binary GADs, we have to exploit the *connected model property* of $\mathcal{ALCQI}^*(\mathsf{Q})$: if an $\mathcal{ALCQI}^*(\mathsf{Q})$-concept is satisfiable (finitely satisfiable) w.r.t. a general TBox $\mathcal{T}$, then C and $\mathcal{T}$ are satisfiable (finitely satisfiable) in a connected model, i.e., in a model that has a "root node" from which every other node can be reached by travelling along role relationships. The idea for dealing with binary GADs is now as follows: for each attribute A of an entity E, we introduce two new concrete features $g^{\mathsf{min}}_{A,E}$ and $g^{\mathsf{max}}_{A,E}$. Our concept equations will ensure that, for every connected model $\mathcal{I}$, there exist $q_1, q_2 \in \mathbb{Q}$ such that $g^{\mathsf{min}}_{A,E}(d) = q_1$ and $g^{\mathsf{max}}_{A,E}(d) = q_2$ for each $d \in \Delta_{\mathcal{I}}$. Moreover, we enforce that $\phi(A)^{\mathcal{I}}(d) \geq q_1$ and $\phi(A)^{\mathcal{I}}(d) \leq q_2$ for every $d \in \phi(E)^{\mathcal{I}}$. Finally, we can use the g^{min} and g^{max} features to ensure that the GADs are satisfied. For technical reasons, we must also introduce an additional concept name X_E for every entity E. Intuitively, X_E holds at every point of a connected model if the entity E has at least one instance in this model.

The exact rules for translating GADs can be found in the non-abridged version of this paper [15]. However, with the strategy sketched above we cannot deal with $\neq$ dependencies. Indeed, we have to leave the encoding of "$\neq$" in GADs as an open problem.

Acknowledgements. I would like to thank the anonymous referees for valuable comments.

References

[1] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, pages 452–457, 1991.

[2] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics and abstract description systems. *JAIR*, 16:1–58, 2002.

[3] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-generating dependencies. *Journal of Computer and System Sciences*, 59, 1999.

[4] D. Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* PhD Thesis, Università di Roma "La Sapienza", Italia, 1996.

[5] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

[6] P. P.-S. Chen. The entity-relationship model-toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[7] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modeling. In *Proc. of KRDB2000*, number 29 in CEUR-WS (http://ceur-ws.org/), pages 45–53, 2000.

[8] S. Ginsburg and R. Hull. Order dependency in the relational model. *Theoretical Computer Science*, 26(1-2):149–195, 1983.

[9] V. Haarslev, R. Möller, and M. Wessel. The description logic $\mathcal{ALCNH}_{R^+}$ extended with concrete domains: A practically motivated approach. In *Proc. of IJCAR'01*, number 2083 in LNAI, pages 29–44. Springer-Verlag, 2001.

[10] R. Hull and R. King. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys*, 19(3):201–260, 1987.

[11] M. Jarke, M. Lenzerini, Y. Vassilious, and P. Vassiliadis, editors. *Fundamentals of Data Warehousing.* Springer-Verlag, 2000.

[12] C. Lutz. *The Complexity of Reasoning with Concrete Domains.* PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2001.

[13] C. Lutz. NExpTime-complete description logics with concrete domains. In *Proc. of IJCAR'01*, number 2083 in LNAI, pages 45–60. Springer-Verlag, 2001.

[14] C. Lutz. Adding numbers to the $\mathcal{SHIQ}$ description logic—First results. In *Proc. of KR2002.* Morgan Kaufman, 2002. To appear.

[15] C. Lutz. Reasoning about entity relationship diagrams with complex attribute dependencies. LTCS-Report 02-01, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2002. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[16] D. Toman and G. Weddell. On attributes, roles, and dependencies in description logics and the ackermann case of the decision problem. In *Proc. of DL2001*, number 49 in CEUR-WS (http://ceur-ws.org/), pages 76–85, 2001.