Armstrong Relations for Ontology Design and Evaluation

Henriette Harmse[1], Katarina Britz[2], and Aurona Gerber[1]

[1] CSIR Meraka Institute and Department of Informatics, University of Pretoria, South Africa
[2] CSIR CAIR, Information Science Department, Stellenbosch University, South Africa

Abstract. A challenge in ontology design is to ensure that the ontology accurately represents the application domain and application constraints. Motivating scenarios provide the motivation for the representation choices made during design, and competency questions are subsequently used to evaluate the design. In this paper we show how the notion of Armstrong relations from relational database theory can be used to generate motivating scenarios and competency questions for uniqueness constraints and not null constraints.

1 Introduction

In this paper[3] our focus is on supporting an ontology engineer in designing an ontology that captures the relevant constraints of the application domain based on the input of a domain expert. The problem is that the expert might have omitted to mention constraints that are of importance. The question now is, how can the ontology engineer help the expert to determine all relevant constraints?

In an attempt to answer this, we translate Armstrong relations from relational database theory to ontologies, which is the main contribution of this paper. In relational database theory Armstrong relations are used to generate "perfect test data" for a set of constraints. "Perfect test data" satisfies all constraints deemed relevant and violates all constraints considered irrelevant. For arbitrary constraints Armstrong relations are undecidable [12], and we therefore limit ourselves to uniqueness constraints and not null constraints.

Adopting the ontology design approach of Grüninger, et al.[9], we consider each row in an an Armstrong relation as a potential motivating scenario since it satisfies each meaningful constraint. The Armstrong relation will also violate every irrelevant constraint and thus the Armstrong relation is a motivating scenario for irrelevant constraints that could be used by a ontology engineer to verify with an expert that the constraints that are violated, are indeed irrelevant. Once expert and ontology engineer agree on which constraints should be

[3] This work was partially funded by the National Research Foundation of South Africa under Grant No. 85482.

met and which not, these constraints and motivating scenarios can be used to evaluate the design of the ontology through formal competency questions (CQs).

The paper is structured as follows: Section 2 presents preliminaries and Section 3 develops the algorithm for generating an Armstrong ABox, which we relate to ontology design and evaluation in Section 4. We conclude with Section 5.

2 Preliminaries

2.1 Ontology Design and Evaluation

The ontology design and evaluation approach of Grüninger, et al. [9], aims to engineer an ontology based on first-order logic that matches a set of business requirements expressed as a number of motivating scenarios. Each motivating scenario gives rise to informal competency questions (CQ) which the ontology should answer. Based on the informal CQs a terminology can be constructed and the informal CQs can be translated into formal CQs. The last step in the approach is to define the conditions under which the competency questions are complete and to prove completeness.

Competency questions are defined by Grüninger, et al. [9] as follows:

Definition 1. *Let $T_{ontology}$ be a terminology with related axioms, T_{ground} a set of ground literals and Q a first-order sentence expressed in terms of $T_{ontology}$. A formal competency question then has one of two forms:*

Entailment CQ *Determine whether $T_{ontology} \cup T_{ground} \models Q$*
Consistency CQ *Determine whether $T_{ontology} \cup T_{ground} \not\models \neg Q$*

2.2 Armstrong Relations

Armstrong relations are the underlying notion to Armstrong ABoxes which we present in Section 3. In this section we explain the benefit of using Armstrong relations in relational database theory with some examples.

A challenge in database design is to determine the constraints that are relevant and the constraints that are irrelevant for a set of business requirements. This task is challenging due to the mismatch in expertise: typically domain experts have in-depth domain knowledge but limited modelling knowledge whilst ontology engineers have modelling experience without domain knowledge. Test data are therefore often used to validate, communicate and consolidate the conceptual model [12].

In relational database theory perfect test data can be generated using *Armstrong relations*. The generated test data are said to be perfect since an Armstrong relation for a table schema is a set of data that satisfy all constraints that are perceived to be relevant and violates all constraints that are perceived to be irrelevant [12][4]. As an example, consider the table schema `Schedule` used by

[4] The formal definition of an Armstrong relation can be found in the Link, et. al. paper, which is similar to that of an Armstrong ABox defined in Definition 8.

Link, et al. [12] (Figure 1), which defines a schedule of courses taught by lecturers at given times in given rooms. Note that the primary key over `C_ID` and `Time` implies two constraints, namely that (1) the combination of `C_ID` and `Time` is unique and that (2) neither `C_ID` nor `Time` is allowed to be null. An Armstrong relation for these 2 constraints is given in the table in Figure 1. Note that *ni* is used to represent a null marker or "no information".

```
CREATE TABLE Schedule (
    C_ID CHAR[5], L_Name VARCHAR,
    Time CHAR[15], Room VARCHAR,
    PRIMARY KEY (C_ID, Time);)
```

Row	C_ID	Time	L_Name	Room
1	11301	Mon, 10am	Church	Red
2	11301	Tue, 02pm	Church	Red
3	78200	Mon, 10am	Church	Red
4	99120	Wed, 04pm	ni	ni

Fig. 1. Table schema `Schedule` with related Armstrong relation

The table is an Armstrong relation for the table schema in Figure 1 because for all rows the combination of `C_ID` and `Time` is unique and neither of these columns contain any null values. Hence, it agrees with the uniqueness and not null constraints that are relevant and it also violates all uniqueness and not null constraints that are irrelevant. As an example it violates the uniqueness constraint for `C_ID` in rows 1 and 2.

An Armstrong relation is constructed with only a certain set of constraints in mind. The Armstrong relation of Figure 1 only considers uniqueness and not null constraints. Moreover, Armstrong relations are undecidable for arbitrary constraints [2], but under certain conditions Armstrong relations do exist for cardinality constraints, uniqueness constraints, functional dependencies, multivalued dependencies and not null constraints [8, 12]. In this paper we focus on uniqueness- and not null constraints.

Based on the example data in Figure 1, a domain expert subsequently realizes that having a lecturer teach two courses at the same time in the same room does not make sense. They decide to revise the table schema to that of Figure 2, which presents the related Armstrong relation as well.

```
CREATE TABLE Schedule (
    C_ID CHAR[5], L_Name VARCHAR,
    Time CHAR[15], Room VARCHAR,
    PRIMARY KEY (C_ID, Time),
    UNIQUE (Time, L_Name, Room);)
```

Row	C_ID	Time	L_Name	Room
1	11301	Mon, 10am	Church	Red
2	11301	Tue, 02pm	Church	Red
3	78200	Mon, 10am	Church	ni
4	99120	Mon, 10am	ni	Red

Fig. 2. Redesigned table schema `Schedule` with related Armstrong relation

2.3 Important Description Logic Notions

In this paper we will use OWL 2 [11, 13], which we assume readers are familiar with and therefore we only highlight the DL notions that are of relevance to this paper, namely n-ary relations and uniqueness constraints.

***n*-ary Relations** In general DLs only consider unary (concepts) and binary relations (roles) and hence n-ary relations cannot be expressed directly [1]. Variants of the DL $\mathcal{DLR}$ are the exception to the rule and explicitly include constructors

in their syntax for expressing n-ary relations [4]. However, n-ary relations can still be expressed in DLs that do not cater for n-ary relations through the notion of reification [1].

Uniqueness Constraints The requirement for expressing uniqueness is a need that has been expressed often by users of ontologies. This lead to various implementations of identification constraints in DLs with different expressivity and different behaviour [3, 5, 6, 13]. However, the underlying assumption on which these implementations agree is that if two individuals x and y participate in some inverse functional role r, then it follows that $x = y$. We will use Easy Keys as defined in Parsia, et. al. [13].

3 Generating an Armstrong ABox

In this section we adapt the definitions and algorithm of Link, et al. [12], which are applicable to relational database theory to ontologies. The translation has to consider differences between DLs and relational database theory, in particular the open world/closed world assumption. However, in constructing an Armstrong ABox we will restrict ourselves to known individuals.

The translation will be presented by first providing an informal discussion and/or example based on the table schemas and instances in Section 2.2. This approach is adopted so that the parallels between relational database theory and ontologies are clear, which will help to clarify why the Armstrong relation notion can be applied to ontologies. These informal explanations are followed by the formal DL related definitions and examples.

3.1 Ontology Structure for representing an Armstrong Relation

Intuitively a table schema maps to a TBox and a table instance to an ABox. Hence, an Armstrong relation for a table schema `Schema` will translate to an ontology $\mathcal{O}_{Schema}$ where $\mathcal{O}_{Schema} = \mathcal{T}_{Schema} \cup \mathcal{A}_{Schema}$.

The table schema of Figure 2 describes three different aspects of the table schema `Schedule` namely:

1. The structure of the table schema `Schedule`, which consists of columns `C_ID`, `Time`, `L_Name` and `Room`, where each column is associated with a countably infinite domain, which in this case is respectively `CHAR[5]`, `VARCHAR`, `CHAR[15]` and `VARCHAR`.
2. The primary key over `C_ID` and `Time`, which states that for these columns null values are not allowed, which we will refer to as *null-free* here.
3. The uniqueness constraint which is specified over `C_ID` and `Time` and the uniqueness constraint which is specified over `Time`, `L_Name` and `Room`.

The Armstrong algorithm of Link, et al. [12] depends on the ability to discern these different aspects of a table schema. We therefore partition $\mathcal{T}_{Schema}$ such that $\mathcal{T}_{Schema} = \mathcal{T}^{S}_{Schema} \cup \mathcal{T}^{nf}_{Schema} \cup \mathcal{T}^{\Sigma}_{Schema}$, where $\mathcal{T}^{S}_{Schema}$ defines the structural aspects, $\mathcal{T}^{nf}_{Schema}$ the null-free aspects and $\mathcal{T}^{\Sigma}_{Schema}$ the uniqueness constraints that apply to the `Schema` representation in DLs.

3.2 Characterization of n-ary Relations

The definitions presented in this section establishes the terminology that will be used to give a compact characterization of the DL representation of a table schema. The structure of a table schema consisting of n columns can be represented via reification, which we formalize in Definition 2 [1].

Definition 2. *An n-ary relation R^n over components $R_1, \ldots, R_n$ can be expressed as the reified concept C_{R^n} using n roles $r_1, \ldots, r_n$ such that*

$$C_{R^n} \sqsubseteq \leq 1 r_1.\top \sqcap \ldots \sqcap \leq 1 r_n.\top$$
$$\top \sqsubseteq \forall r_1.R_1 \sqcap \ldots \sqcap \top \sqsubseteq \forall r_n.R_n$$
$$C_{R^n} \ \texttt{hasKey}\ (r_1, \ldots, r_n)$$

where the key constraint ensures that no 2 individuals of C_{R^n} agrees on their participation in roles $r_1, \ldots, r_n$. For convenience we denote the DL axioms that represent the n-ary relation R^n as $\mathcal{T}^S_{R^n}$.

Example 1. The corresponding TBox $\mathcal{T}^S_{Sched}$ axioms for the table schema of Figure 2 are:

$$\begin{aligned}\mathcal{T}^S_{Sched} = \{&C_{Sched} \sqsubseteq \leq 1 r_{C\_ID}.\top \sqcap \leq 1 r_{Time}.\top \sqcap \leq 1 r_{L\_Name}.\top \sqcap \leq 1 r_{Room}.\top,\\ &\top \sqsubseteq \forall r_{C\_ID}.R_{char[5]}, \top \sqsubseteq \forall r_{Time}.R_{varchar}, \top \sqsubseteq \forall r_{L\_Name}.R_{char[15]},\\ &\top \sqsubseteq \forall r_{Room}.R_{varchar}, C_{Sched}\ \texttt{hasKey}\ (r_{C\_ID}, r_{Time}, r_{L\_Name}, r_{Room})\}\end{aligned}$$

where $R^{\mathcal{D}}_{char[5]}, R^{\mathcal{D}}_{varchar}, R^{\mathcal{D}}_{char[15]} \subseteq \triangle^{\mathcal{D}}$ and $\triangle^{\mathcal{D}}$ is the domain of all data types.

For a given table schema various table instances can exist. The table instance of Figure 2 is one possible table instance of the related table schema `Schedule`. When for a given row certain column values are not null, the row is said to be *total* for those columns. In the table instance of Figure 1, row 1 is total for columns `C_ID`, `Time`, `L_Name` and `Room`, while row 4 is total only for columns `C_ID`, `Time` and `Room`. When all rows for a table instance are total for a set of columns, the table instance is said to be total for those columns. Hence, the table instance is total for columns `C_ID` and `Time`.

A table instance is represented in DLs as an ABox, denoted by $\mathcal{A}_{R^n}$, corresponding to TBox $\mathcal{T}_{R^n}$. For a given $\mathcal{T}_{R^n}$ any number of $\mathcal{A}^0_{R^n}, \mathcal{A}^1_{R^n}, \ldots$ can exist, where each $\mathcal{A}^i_{R^n}$ represents a different table instance for a given table schema. The next definition defines $\mathcal{A}_{R^n}$ and related notions.

Definition 3. *An ABox $\mathcal{A}_{R^n}$ corresponding to $\mathcal{T}_{R^n}$ consists of a set of assertions $C_{R^n}(c_j), j \geq 1$. A known individual c of C_{R^n} is said to be X-**total**, $X \subseteq \{r_1, \ldots, r_n\}$, if there are known individuals h_i, such that $R_i(h_i)$ and $r_i(c, h_i)$, for any $r_i \in X$. $\mathcal{A}_{R^n}$ is said to be X-**total** if for all known individuals c_j, $j \geq 1$ such that $C_{R^n}(c_j) \in \mathcal{A}_{R^n}$ we have that c_j is X-**total**.*

Example 2. The corresponding $\mathcal{A}_{Sched}$ is given for $\mathcal{T}_{Sched}$ in Table 1 where c_1, c_2, c_3 and c_4 (for which the assertions are not explicitly shown) are individuals of C_{Sched}. Table 1 is the ABox representation of the table instance of Figure 2. Note

that no assertions are added for null markers since this represents the semantics of "no information" faithfully. Individual c_1 is $\{r_{C\_ID}, r_{Time}, r_{L\_Name}, r_{Room}\}$-total and $\mathcal{A}_{Sched}$ is $\{r_{C\_ID}, r_{Time}\}$-total.

The table schema of Figure 2 states that columns `C_ID` and `Time` together represent the primary key of `Schedule`. One implication of this is that columns `C_ID` and `Time` represent a *null-free* subschema of `Schedule` and hence the table instance of Figure 2 is {`C_ID`,`Time`}-total.

Table 1. Corresponding ABox, $\mathcal{A}_{Sched}$, for the table instance of Figure 2

C_ID	Time	L_Name	Room
$r_{C\_ID}(c_1, \text{“11301”})$	$r_{Time}(c_1, \text{“}Mon, 10am\text{”})$	$r_{L\_Name}(c_1, \text{“}Church\text{”})$	$r_{Room}(c_1, \text{“}Red\text{”})$
$r_{C\_ID}(c_2, \text{“11301”})$	$r_{Time}(c_2, \text{“}Tue, 02pm\text{”})$	$r_{L\_Name}(c_2, \text{“}Church\text{”})$	$r_{Room}(c_2, \text{“}Red\text{”})$
$r_{C\_ID}(c_3, \text{“78200”})$	$r_{Time}(c_3, \text{“}Mon, 10am\text{”})$	$r_{L\_Name}(c_3, \text{“}Church\text{”})$	
$r_{C\_ID}(c_4, \text{“99120”})$	$r_{Time}(c_4, \text{“}Mon, 10am\text{”})$		$r_{Room}(c_4, \text{“}Red\text{”})$

Definition 4. *Let $X \subseteq \{r_1, \ldots, r_n\}$ be a subset of the components of R^n that are null-free in the application domain, we define $\mathcal{T}^{nf}_{R^n}$ such that it consists of the axioms $C_{R^n} \sqsubseteq \exists r_i.\top$, for any $r_i \in X$. If $\mathcal{T}^{nf}_{R^n}$ is non-empty we say $\mathcal{T}_{R^n}$ is **null-free** for X, which we abbreviate by $nf(\mathcal{T}_{R^n}, X)$. When $\mathcal{A}_{R^n}$ is also X-**total** and therefore $\mathcal{A}_{R^n}$ satisfies $nf(\mathcal{T}_{R^n}, X)$, we say that $\mathcal{A}_{R^n}$ is **null-free** for X, which we denote by $\mathcal{A}_{R^n} \Vdash nf(\mathcal{T}_{R^n}, X)$.*

Example 3. When $\{C_{Sched} \sqsubseteq \exists r_{C\_ID}.\top, C_{Sched} \sqsubseteq \exists r_{C\_Time}.\top\} \subseteq \mathcal{T}^{nf}_{Sched}$, we say that the *Schedule* relation is null-free for r_{C_ID} and r_{Time}, which we denote by $nf(\mathcal{T}_{Sched}, \{r_{C\_ID}, r_{Time}\})$. From Table 1 we determine that $\mathcal{A}_{Sched}$ is null-free for r_{C_ID} and r_{Time} since $\mathcal{A}_{Sched}$ is $\{r_{C\_ID}, r_{Time}\}$-total. Hence, we have that $\mathcal{A}_{Sched} \Vdash nf(\mathcal{T}_{Sched}, \{r_{C\_ID}, r_{Time}\})$.

A cardinality constraint (CC) over columns of a table schema constrains how many times a combination of values can be duplicated for those columns in a table instance. This is expressed as $card(columns) \leq b$. As an example, the table instance of Figure 2 satisfies $card(L\_Name, Room) \leq 2$.

Definition 5. *A **cardinality constraint** (CC) over $\mathcal{T}^{S}_{R^n}$ is a statement $card(\mathcal{T}_{R^n}, X) \leq b$ where $X \subseteq \{r_1, \ldots, r_n\}$ and b is a positive integer. The CC $card(\mathcal{T}_{R^n}, X) \leq b$ is satisfied by $\mathcal{A}_{R^n}$, which we denote by $\mathcal{A}_{R^n} \Vdash card(\mathcal{T}_{R^n}, X) \leq b$, if and only if for each $r_k \in X$ there is a named individual h_k such that*

$$\sharp\{c_j | c_j \text{ is } X\text{-total}, C_{R^n}(c_j) \in \mathcal{A}_{R^n} \text{ and for each } r_i \in X, \exists h_i \text{ such that } R_i(h_i) \in \mathcal{A}_{R^n}, r_i(c_j, h_i) \in \mathcal{A}_{R^n}\} \leq b.$$

*In the special case where $b = 1$ for a CC, the CC is called a **uniqueness constraint** (UC) and is represented by $u(\mathcal{T}_{R^n}, X)$, which if satisfied by $\mathcal{A}_{R^n}$ is denoted by $\mathcal{A}_{R^n} \Vdash u(\mathcal{T}_{R^n}, X)$* [5].

Example 4. For Table 1 $\mathcal{A}_{Sched} \Vdash card(\mathcal{T}_{Sched}, \{r_{L\_Name}, r_{Room}\}) \leq 2$.

[5] We explicitly ignore non-standard CCs and UCs, that is where $X = \varnothing$.

3.3 $\mathcal{C}$-Armstrong ABoxes

Recall from Section 2.2 that an Armstrong relation satisfies all constraints that are relevant and violates all constraints that are irrelevant. I.e., the table schema of Figure 2 states that there is a UC over columns `C_ID` and `Time`. Hence, there is no row in the Armstrong relation of Figure 2 that violates this constraint.

Definition 6. *Let Σ be a set of constraints (i.e. UCs) over $\mathcal{T}^S_{R^n}$. We say that $\mathcal{A}_{R^n}$ **satisfies** Σ, denoted by $\mathcal{A}_{R^n} \Vdash \Sigma$, if $\mathcal{A}_{R^n}$ satisfies every element of Σ. If for some $\sigma \in \Sigma$, $\mathcal{A}_{R^n}$ does not satisfy σ we say $\mathcal{A}_{R^n}$ **violates** σ and write $\mathcal{A}_{R^n} \nVdash \sigma$. By extension we have that $\mathcal{A}_{R^n}$ also violates Σ, which we denote by $\mathcal{A}_{R^n} \nVdash \Sigma$. For convenience we represent the axioms that represent Σ as $\mathcal{T}^{\Sigma}_{R^n}$.*

Example 5. For $\mathcal{A}_{Sched}$ of Table 1 we have that $\mathcal{A}_{Sched} \Vdash \Sigma$ where $\Sigma = \{u(r_{C\_ID}, r_{Time}), u(r_{Time}, r_{L\_Name}, r_{Room})\}$. However, for $\Sigma = \{u(r_{C\_ID}), u(r_{L\_Name}, r_{Room})\}$ we have that $\mathcal{A}_{Sched} \nVdash \Sigma$.

Figure 2 states that there is a UC over columns `C_ID` and `Time`, which implies the following UCs: (1) the UC over `C_ID`, `Time` and `L_Name`, (2) the UC over `C_ID`, `Time` and `Room`, and (3) the UC over `C_ID`, `Time`, `L_Name` and `Room`.

Definition 7. *Assume $\mathcal{T}^S_{R^n}$ is given with $nf(\mathcal{T}_{R^n}, X), X \subseteq \{r_1, \ldots, r_n\}$, with $\Sigma \cup \{\varphi\}$ a finite set of constraints over $\mathcal{T}_{R^n}$. We say that Σ **implies** φ in the presence of $nf(\mathcal{T}_{R^n}, X)$, denoted by $\Sigma, nf(\mathcal{T}_{R^n}, X) \vDash \varphi$, if every X-total $\mathcal{A}_{R^n}$ that satisfies Σ also satisfies φ. If Σ does not imply φ in the presence of $nf(\mathcal{T}_{R^n}, X)$ it is denoted by $\Sigma, nf(\mathcal{T}_{R^n}, X) \nvDash \varphi$. We denote the **semantic closure** of Σ and $nf(\mathcal{T}_{R^n}, X)$ by $\Sigma^*_{nf(\mathcal{T}_{R^n}, X)}$, which we define as*

$$\Sigma^*_{nf(\mathcal{T}_{R^n}, X)} = \{\varphi | \Sigma, nf(\mathcal{T}_{R^n}, X) \vDash \varphi\}$$

Example 6. For $\mathcal{T}^S_{Sched}$, $nf(\mathcal{T}_{Sched}, \{r_{C\_ID}, r_{Time}\})$ and $\Sigma = \{u(r_{C\_ID}, r_{Time}), u(r_{Time}, r_{L\_Name}, r_{Room})\}$ we have that

$$\begin{aligned}\Sigma^*_{nf(\mathcal{T}_{Sched}, \{r_{C\_ID}, r_{Time}\})} = \{&u(r_{C\_ID}, r_{Time}), u(r_{Time}, r_{L\_Name}, r_{Room}),\\ &u(r_{C\_ID}, r_{Time}, r_{L\_Name}), u(r_{C\_ID}, r_{Time}, r_{Room}),\\ &u(r_{C\_ID}, r_{Time}, r_{L\_Name}, r_{Room})\}\end{aligned}$$

Given the previous definitions it is now possible to formalize the notion of an Armstrong ABox for an n-ary relation R^n.

Definition 8. *Let $\mathcal{C}$ denote a class of constraints and let Σ be a finite set of elements from $\mathcal{C}$. Assume $nf(\mathcal{T}_{R^n}, X)$, $X \subseteq S$ holds where $S = \{r_1, \ldots, r_n\}$. We say $\mathcal{A}_{R^n}$ is **$\mathcal{C}$-Armstrong** for Σ and $nf(\mathcal{T}_{R^n}, X)$ if and only if*

1. *$\mathcal{A}_{R^n}$ satisfies Σ, which we write as $\mathcal{A}_{R^n} \Vdash \Sigma$,*
2. *$\mathcal{A}_{R^n}$ violates every $\varphi \notin \Sigma^*_{nf(\mathcal{T}_{R^n}, X)}$, which we write as $\mathcal{A}_{R^n} \nVdash \varphi$ for each $\varphi \notin \Sigma^*_{nf(\mathcal{T}_{R^n}, X)}$,*
3. *$\mathcal{A}_{R^n}$ is X-total, which is written as $\mathcal{A}_{R^n} \Vdash nf(\mathcal{T}_{R^n}, X)$, and*
4. *for any $H \subseteq S - X$, $\mathcal{A}_{R^n}$ is not H-total, which we write as $\mathcal{A}_{R^n} \nVdash nf(\mathcal{T}_{R^n}, H)$.*

Example 7. For $\mathcal{T}^S_{Sched}$, $nf(\mathcal{T}_{Schedtheoremstar}, \{r_{C\_ID}, r_{Time}\})$ and $\Sigma = \{u(r_{C\_ID}, r_{Time}), u(r_{Time}, r_{L\_Name}, r_{Room})\}$, we can confirm that $\mathcal{A}_{Sched}$ of Table 1 is $\mathcal{C}$-Armstrong, where $\mathcal{C}$ represents the class of UCs, by verifying that:

1. $\mathcal{A}_{Sched}$ satisfies Σ,
2. $\mathcal{A}_{Sched}$ violates every $\varphi \notin \Sigma^*_{nf(\mathcal{T}_{Sched}, \{r_{C\_ID}, r_{Time}\})}$,
3. $\mathcal{A}_{Sched}$ is $\{r_{C\_ID}, r_{Time}\}$-total and
4. $\mathcal{A}_{Sched}$ is not $\{r_{L\_Name}\}$-total or $\{r_{Room}\}$-total.

3.4 Anti-keys and Strong Agreement Sets

Calculating an Armstrong relation for the class of UCs depends on the ability to determine those columns for which no UC is implied in order for them to be violated. Rather than violating all UCs not implied, it is sufficient to only violate those that are maximal with the property that they are not implied by Σ in the presence of the null-free subschema. Evidence for this intuition can be seen in Figure 2. The combination of `C_ID`, `L_Name` and `Room` as a potential UC is violated by rows 1 and 2, which implies that no subset of these columns can be a UC. In relational database theory the combination of `Time`, `L_Name` and `Room` is referred to as an *anti-key* [12].

An anti-key satisfies the following 3 conditions: (1) anti-keys are columns of a table schema, (2) anti-keys are non-empty sets of columns of a table schema for which no UC hold, and (3) anti-keys are maximal w.r.t. condition (2).

Definition 9. *Let Σ be a set of UCs, $S = \{r_1, \ldots, r_n\}$, $X \subseteq S$ and assume $nf(\mathcal{T}_{R^n}, X)$ holds for $\mathcal{T}^S_{R^n}$. For $Y \subseteq S$, we denote that Y is an anti-key by $a(Y)$. The set Σ^{-1} of all **anti-keys** is defined as*

$$\Sigma^{-1} = \{a(Y) | Y \subseteq S \text{ and } \Sigma, nf(\mathcal{T}_{R^n}, X) \not\models u(\mathcal{T}_{R^n}, Y) \text{ and } \text{for any } H \subseteq (S - Y)[\Sigma, nf(\mathcal{T}_{R^n}, X) \models u(\mathcal{T}_{R^n}, Y \cup H)]\}$$

Example 8. The anti-keys for $\mathcal{T}_{Sched}$ with $nf(\mathcal{T}_{R^n}, \{r_{C\_ID}, r_{Time}\})$ and UCs $\Sigma = \{u(r_{C\_ID}, r_{Time}), u(r_{Time}, r_{L\_Name}, r_{Room})\}$ is given by $\Sigma^{-1} = \{a(r_{C\_ID}, r_{L\_Name}, r_{Room}), a(r_{Time}, r_{L\_Name}), a(r_{Time}, r_{Room})\}$.

Anti-keys can be calculated using hypergraphs where the columns of the table schema and the sets of columns representing UCs are respectively the vertexes and the edges [12]. Since only minimal UCs are used, the hypergraph is a simple hypergraph. We illustrate the calculation of anti-keys through an example. The minimal transversal [7] of $\mathcal{H}$ is given by $Tr(\mathcal{H}) = \{\{\texttt{Time}\}, \{\texttt{C\_ID}, \texttt{Room}\}, \{\texttt{C\_ID}, \texttt{L\_Name}\}\}$. Anti-keys are maximal such that they are not UCs. Since each element of the transversal represent columns that form part of minimal UCs, the set difference between the set of columns representing the table schema and an element of the transversal delivers an anti-key. I.e., the set difference between the columns of `Schedule` and $\{\texttt{Time}\}$, the first element of the transversal, delivers the anti-key over `C_ID`, `L_Name` and `Room`.

Algorithm 1 Algorithm for calculating an Armstrong ABox

Input: TBox $\mathcal{T}^S_{R^n}$, $S = \{r_1, \ldots, r_n\}$, Σ a set of UCs and $nf(\mathcal{T}_{R^n}, X)$, $X \subseteq S$
Output: Armstrong ABox $\mathcal{A}_{R^n}$ for Σ and $nf(\mathcal{T}_{R^n}, X)$ where
$h_{r_i,j_{r_i}}, i = \{1, \ldots, n\}, j_{r_i} \geq 1,$ are individuals such that $R_i(h_{r_i,j_{r_i}})$

A1: $\mathcal{A}_{R^n} = \mathcal{A}_{R^n} \cup \{C_{R^n}(c_1), r_i(c_1, h_{r_i,1})\}, i = \{1, \ldots, n\}$
A2: Compute Σ^{-1} using hypergraph methods
A3: $k = 2, j_{r_i} = 2, \;\; i = \{1, \ldots, n\}$;
A4: **for all** Y such that $a(Y) \in \Sigma^{-1}$ **do**
$\mathcal{A}_{R^n} = \mathcal{A}_{R^n} \cup \{C_{R^n}(c_k), r_i(c_k, h_{r_i,j_{r_i}})\}, \;\; i = \{1, \ldots, n\}$ where

$$h_{r_i,j_{r_i}} = \begin{cases} h_{r_i,1}, \text{ if } r_i \in Y \\ h_{r_i,j_{r_i}}, j_{r_i} = j_{r_i} + 1, \text{ if } r_i \in X - Y \\ \textbf{do nothing}, \text{ else} \end{cases} ; k = k + 1;$$

A5: Let $Z \subseteq S$ be such that $\mathcal{A}_{R^n}$ is Z-total
if $Z - X \neq \varnothing$ **then**
$\mathcal{A}_{R^n} = \mathcal{A}_{R^n} \cup \{C_{R^n}(c_k), r_i(c_k, h_{r_i,j_{r_i}})\}, i = \{1, \ldots, n\}$ where

$$h_{r_i,j_{r_i}} = \begin{cases} \textbf{do nothing}, \text{ if } r_i \in X - Z \\ h_{r_i,j_{r_i}}, \text{ else} \end{cases} ; \textbf{return } \mathcal{A}_{R^n}$$

else **return** $\mathcal{A}_{R^n}$

Lemma 1. *Define the hypergraph $\mathcal{H}$ for $\mathcal{T}^S_{R^n}$, $S = \{r_1, \ldots, r_n\}$, with Σ a set of minimal UCs, with $nf(\mathcal{T}_{R^n}, X)$, $X \subseteq S$, as $\mathcal{H} = (V, E)$ where $V = S$ and $E = \{Z | u(Z) \in \Sigma\}$. Then the set of anti-keys is defined as*

$$\Sigma^{-1} = \{a(S - W) | W \in Tr(\mathcal{H})\}$$

where $Tr(\mathcal{H})$ denotes the minimal transversal of the hypergraph $\mathcal{H}$.

Example 9. For $\mathcal{T}^S_{Sched}$ and $\Sigma = \{u(r_{C\_ID}, r_{Time}), u(r_{Time}, r_{L\_Name}, r_{Room})\}$ we create the hypergraph $\mathcal{H} = (V, E)$ where $V = \{r_{C\_ID}, r_{Time}, r_{L\_Name}, r_{Room}\}$ and $E = \{\{r_{C\_ID}, r_{Time}\}, \{r_{Time}, r_{L\_Name}, r_{Room}\}\}$. We then have that $Tr(\mathcal{H}) = \{\{r_{Time}\}, \{r_{C\_ID}, r_{Room}\}, \{r_{C\_ID}, r_{L\_Name}\}\}$ from which follows that $\Sigma^{-1} = \{a(r_{C\_ID}, r_{L\_Name}, r_{Room}), a(r_{Time}, r_{L\_Name}), a(r_{Time}, r_{Room})\}$.

A notion related to that of anti-keys is *strong agreement sets*. Two rows of a table instance is said to *strongly agree* on a set of columns if for those columns the two rows agree on their total values. For the table instance in Figure 2 row 1 and 2 strongly agree on columns `C_ID`, `L_Name` and `Room`.

Definition 10. *Given $\mathcal{A}_{R^n}$ we say two individuals c_1 and c_2 of concept C_{R^n} **strongly agree** on the roles X, $X \subseteq S$, where $S = \{r_1, \ldots, r_n\}$, if they are X-total and the fillers of each $r_i \in X$ are filled by the same individuals. The **strong agreement set** of c_1 and c_2, denoted by $ag^s(c_1, c_2)$, is given by*

$$\begin{aligned} ag^s(c_1, c_2) = \{r_i | r_i\text{-total}, C_{R^n}(c_1) \in \mathcal{A}_{R^n}, C_{R^n}(c_2) \in \mathcal{A}_{R^n}, \text{ and } \exists y_i, z_i \text{ such that} \\ R_i(y_i) \in \mathcal{A}_{R^n}, R_i(z_i) \in \mathcal{A}_{R^n}, r_i(c_1, y_i) \in \mathcal{A}_{R^n}, r_i(c_2, z_i) \in \mathcal{A}_{R^n} \\ \text{and } y_i = z_i \text{ for any } r_i \in X\}. \end{aligned}$$

*The **strong agreement set** of $\mathcal{A}_{R^n}$ is given by $ag^s(\mathcal{A}_{R^n})$ which we define as*

$$ag^s(\mathcal{A}_{R^n}) = \{ag^s(c_1, c_2) | C_{R^n}(c_1) \wedge C_{R^n}(c_2) \wedge c_1 \neq c_2\}$$

Example 10. The **strong agreement set** of $\mathcal{A}_{R^n}$ of Table 1 is given by

$$ag^s(\mathcal{A}_{Sched}) = \{\{r_{C\_ID}, r_{L\_Name}, r_{Room}\}, \{r_{Time}, r_{L\_Name}\}, \{r_{Time}, r_{Room}\}, \{r_{L\_Name}\}, \{r_{Room}\}, \{r_{Time}\}\}$$

3.5 Algorithm for generating an Armstrong ABox

Anti-keys and strong agreement sets are useful to characterize an Armstrong relation for the class of UCs and a null-free subschema. A table instance is Armstrong for the class of UCs and a null-free subschema if and only if the following conditions hold:

1. Each anti-key is an element of a strong agreement set.
2. There is no uniqueness constraint such that the set of columns representing a uniqueness constraint is a subset of a strong agreement set.
3. The set of columns over which the table instance is total is the same as the null-free subschema.

Theorem 1. *For $\mathcal{T}^S_{R^n}$ let Σ be a set of UCs and assume $nf(\mathcal{T}_{R^n}, X)$, $X \subseteq S$ where $S = \{r_1, \ldots, r_n\}$ holds. An ABox $\mathcal{A}_{R^n}$ for $\mathcal{T}^S_{R^n}$, Σ and $nf(\mathcal{T}_{R^n}, X)$ is an Armstrong ABox if and only if the following conditions hold:*

1. *$\forall a(Y) \in \Sigma^{-1}[Y \in ag^s(\mathcal{A}_{R^n})]$,*
2. *$(\forall u(\mathcal{T}_{R^n}, Y) \in \Sigma)(\forall Z \in ag^s(\mathcal{A}_{R^n}))[(Y \nsubseteq Z)]$,*
3. *Let $Z \subseteq S$ be such that $\mathcal{A}_{R^n}$ is Z-total, then $Z = X$ [10, 12].*

Example 11. Examples 9 and 10 calculated Σ^{-1} and $ag^s(\mathcal{A}_{Sched})$ for $\mathcal{T}_{Sched}$ with $nf(\mathcal{T}_{R^n}, \{r_{C\_ID}, r_{Time}\})$ and $\Sigma = \{u(r_{C\_ID}, r_{Time}), u(r_{Time}, r_{L\_Name}, r_{Room})\}$. It therefore follows from Theorem 1 that $\mathcal{A}_{R^n}$ (Table 1) is an Armstrong ABox.

Table 2. $\mathcal{A}_{Sched}$ generated using Algorithm 1

C_ID	Time	L_Name	Room
$r_{C\_ID}(c_1, h_{r_{C\_ID},1})$	$r_{Time}(c_1, h_{r_{Time},1})$	$r_{L\_Name}(c_1, h_{r_{L\_Name},1})$	$r_{Room}(c_1, h_{r_{Room},1})$
$r_{C\_ID}(c_2, h_{r_{C\_ID},1})$	$r_{Time}(c_2, h_{r_{Time},2})$	$r_{L\_Name}(c_2, h_{r_{L\_Name},1})$	$r_{Room}(c_2, h_{r_{Room},1})$
$r_{C\_ID}(c_3, h_{r_{C\_ID},2})$	$r_{Time}(c_3, h_{r_{Time},1})$	$r_{L\_Name}(c_3, h_{r_{L\_Name},1})$	
$r_{C\_ID}(c_4, h_{r_{C\_ID},3})$	$r_{Time}(c_4, h_{r_{Time},1})$		$r_{Room}(c_4, h_{r_{Room},1})$

Algorithm 1 calculates an Armstrong ABox $\mathcal{A}_{R^n}$ for $\mathcal{T}^S_{R^n}$, $S = \{r_1, \ldots, r_n\}$, with $nf(\mathcal{T}_{R^n}, X)$ for $X \subseteq S$ and Σ is a set of UCs, using the following steps: (A1) generates an arbitrary first tuple. (A2) calculates the anti-keys. (A3) initializes indexes k and j_{r_i}, where k keeps track of the number of individuals representing tuples and indexes j_{r_i} keep track of the number of different individuals used as role fillers for each of the roles r_i. (A4) generates for each anti-key an individual c_k with related assertions where $h_{r_i,1}$ is re-used when r_i forms part of an anti-key. (A5) generates the equivalent of *ni* for any r_i that is not null-free.

Example 12. For $\mathcal{T}^S_{Sched}$, $\Sigma = \{u(r_{C\_ID}, r_{Time}), u(r_{Time}, r_{L\_Name}, r_{Room})\}$ and $nf(\mathcal{T}, \{r_{C\_ID}, r_{Time}\})$ we generate $\mathcal{A}_{Sched}$ (Table 2) using Algorithm 1, assuming $C_{R^n}(c_1), \ldots, C_{R^n}(c_4)$ are implied. A mapping of each $h_{r_i,j_{r_i}}$ delivers Table 1.

Theorem 2 follows from Lemma 1 and Theorem 1 [12].

Theorem 2. *On input* $(\mathcal{T}^S_{R^n}, \Sigma, nf(\mathcal{T}_{R^n}, X))$ *Algorithm 1 generates an Armstrong ABox* $\mathcal{A}_{R^n}$ *over* $\mathcal{T}^S_{R^n}$ *that is Armstrong for* Σ *and* $nf(\mathcal{T}_{R^n}, X)$.

The preceding definitions motivate the following definition:

Definition 11. *Given* $\mathcal{T}^S_{R^n}$ *with* Σ *a set of UCs and* $nf(\mathcal{T}_{R^n}, X)$, $X \subseteq S$ *where* $S = \{r_1, \ldots, r_n\}$, *we define the corresponding TBox* $\mathcal{T}_{R^n}$ *as* $\mathcal{T}_{R^n} = \mathcal{T}^S_{R^n} \cup \mathcal{T}^{\Sigma}_{R^n} \cup \mathcal{T}^{nf}_{R^n}$ *where* $\mathcal{T}^{\Sigma}_{R^n}$ *consists of the axioms representing* Σ *and* $\mathcal{T}^{nf}_{R^n}$ *consists of the axioms representing* $nf(\mathcal{T}_{R^n}, X)$. *The corresponding Armstrong ontology* $\mathcal{O}_{R^n}$ *is given by* $\mathcal{O}_{R^n} = \mathcal{T}_{R^n} \cup \mathcal{A}_{R^n}$ *where* $\mathcal{A}_{R^n}$ *is an Armstrong ABox for* $\mathcal{T}_{R^n}$.

4 Ontology Design and Evaluation using Armstrong ABoxes

In this section we discuss how an Armstrong ABox can be used to generate motivating scenarios and CQs.

Each row in Table 1 represents a scenario that supports the current design decisions for the ontology since it satisfies all relevant constraints. Each row in Table 1 is thus a motivating scenario. In order to ensure no latent UCs are missed, the complete table also needs to be considered. To communicate the current understanding of the relevant and irrelevant constraints, an ontology engineer can use an informal notation such as the table instance in Figure 2. In this way an Armstrong ABox can be used to systematically generate scenarios which can help an expert to validate the design.

The design of the ontology can be evaluated through generated CQs. $\mathcal{T}_{R^n}$ represent the axioms that have to be satisfied in every model. Hence, it follows that for every $\alpha \in \mathcal{T}_{R^n}$ we have that $\mathcal{T}_{R^n} \models \alpha$, which is referred to as an entailment CQ in Section 2.1. Each row of Table 1 must be consistent for $\mathcal{O}_{R^n}$, which we can express as $\mathcal{O}_{R^n} \not\models \neg Q$ where for all $i = \{1, \ldots n\}$ $Q = C_{R^n}(c_j) \wedge R(h_{ij}) \wedge r_i(c_j, h_{ij})$, which is referred to as a consistency CQ in Section 2.1. Note that this last CQ represents a means with which the motivating scenarios can be formally evaluated since, as stated earlier, $\mathcal{A}_{R^n}$ consists of motivating scenarios.

5 Conclusion

We presented a preliminary theoretical framework for applying Armstrong relations to ontology design within the context of the ontology design approach of Grüninger, et al. [9], where motivating scenarios inform design decisions and CQs are used to evaluate the ontology design. In future research we will investigate the applicability of Armstrong ABoxes to other DL constraints and aim to show how this framework can be used to revise existing populated ontologies.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press (2007), 2nd edition
2. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the Structure of Armstrong Relations for Functional Dependencies. J. ACM 31(1), 30–46 (1984)
3. Borgida, A., Weddell, G.E.: Adding Uniqueness Constraints to Description Logics (Preliminary Report). In: Bry, F., Ramakrishnan, R., Ramamohanarao, K. (eds.) DOOD. Lecture Notes in Computer Science, vol. 1341, pp. 85–102. Springer (1997)
4. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Description Logic Framework for Information Integration. In: Principles of Knowledge Representation and Reasoning. pp. 2–13. Citeseer (1998)
5. Calvanese, D., De Giacomo, G., Lenzerini, M.: Identification Constraints and Functional Dependencies in Description Logics. In: Proceedings International Joint Conference on Artificial Intelligence. vol. 17, pp. 155–160. Citeseer (2001)
6. Calvanese, D., Fischl, W., Pichler, R., Sallinger, E., Simkus, M.: Capturing relational schemas and functional dependencies in RDFS. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence. (2014)
7. Eiter, T., Gottlob, G.: Identifying the Minimal Transversals of a Hypergraph and Related Problems. In: SIAM Journal on Computing. pp. 1278–1304. Society for Industrial and Applied Mathematics (1995)
8. Fagin, R.: Armstrong Databases. Tech. rep. (May 1982)
9. Grüninger, M., Fox, M.: Methodology for the Design and Evaluation of Ontologies. In: Proceedings of International Joint Conference on Artificial Intelligence, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995 (1995)
10. Hartmann, S., Leck, U., Link, S.: On Codd Families of Keys over Incomplete Relations. Comput. J. 54(7), 1166–1180 (2011)
11. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning*, pages 57–67. AAAI Press, 2006.
12. Link, S.: Armstrong Databases: Validation, Communication and Consolidation of Conceptual Models with Perfect Test Data. Proceedings of the Eighth Asia-Pacific Conference on Conceptual Modelling (2012)
13. Parsia, B., Sattler, U., Schneider, T.: Easy Keys for OWL. In: Dolbear, C., Ruttenberg, A., Sattler, U. (eds.) Proceedings of the 5th Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 432. CEUR Workshop Proceedings (2008)