Generating Armstrong ABoxes for $\mathcal{ALC}$ TBoxes

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

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

Abstract. A challenge in ontology engineering is the mismatch in expertise between the ontology engineer and domain expert, which often leads to important constraints not being specified. Domain experts often only focus on specifying constraints that should hold and not on specifying constraints that could possibly be violated. In an attempt to bridge this gap we propose the use of "perfect test data". The generated test data is perfect in that it satisfies all the constraints of an application domain that are required, including ensuring that the test data violates constraints that can be violated. In the context of Description Logic ontologies we call this test data an "Armstrong ABox", a notion derived from Armstrong relations in relational database theory. In this paper we detail the theoretical development of Armstrong ABoxes for $\mathcal{ALC}$ TBoxes as well as an algorithm for generating such Armstrong ABoxes. The proposed algorithm is based, via the ontology completion algorithm of Baader et al., on attribute exploration in formal concept analysis.

1 Introduction

A challenge in ontology design is to know whether all the required constraints that correctly represent a domain of interest are specified. Any given set of constraints over a given domain can be classified into constraints that should hold and constraints that can possibly be violated. Ensuring that all constraints are classified explicitly avoids the situation where omission results in a constraint by default being classified as a constraint that can be violated.

The problem of incomplete specifications is well documented [5, 6] and could be the result of domain experts that concentrate on specifying facts that should hold (such as "all prime numbers are integers") and not on facts that could possibly be violated (such as "all prime numbers are odd"). The problem of incomplete specifications is often exacerbated by a mismatch in expertise between an ontology engineer and domain expert. An ontology engineer usually has limited knowledge about the application domain and domain experts have limited knowledge regarding ontology engineering. However, domain experts are often well versed in the data of their application domain [13]. This motivates the idea of presenting the domain expert with "perfect test data" that satisfy the required constraints and refute constraints that could be violated. In relational database theory this idea of perfect test data is realized as Armstrong relations [8], which we extend here to description logics (DLs) as Armstrong ABoxes.

Description logics (DLs) are syntactic variants of fragments of first-order logic that are specifically designed for the conceptual representation of an application domain in terms of concepts and relationships between concepts. A key design goal for DLs is to ensure that the basic reasoning procedures like satisfiability and classification are decidable. A DL ontology consists of a TBox and an ABox. The TBox is used to define concepts and relationships between concepts and the ABox is used to assert knowledge regarding the domain of interest [1]. In this paper we concentrate on the DL $\mathcal{ALC}$, which we define in Section 2.1.

Armstrong ABoxes assume that an ontology engineer has created a TBox in collaboration with a domain expert, but it is unclear whether the TBox describes the application domain faithfully. By generating "perfect test data" for a given TBox, constraints that follow, as well as constraints that do not follow from the TBox, can be made explicit through example data. The Armstrong ABox (example data) can be reviewed to ensure the TBox is not over- or underspecified, after which the TBox can be amended as needed: if the TBox is overspecified constraints can be removed and if the TBox is underspecified constraints can be added. A new Armstrong ABox can then be generated for the amended TBox, which again can be reviewed until the TBox and Armstrong ABox accurately specify constraints that should hold and constraints that could be violated.

Armstrong ABoxes are formalized relative to particular classes of constraints, with each class of constraints resulting in a different Armstrong ABox formalization. Different Armstrong ABox formalizations may need to use different algorithms to generate an Armstrong ABox for the particular formalization. In previous research we have defined Armstrong ABoxes for the class of constraints consisting of n-ary relations with uniqueness- and null-free constraints [12], of which the practical applicability has been illustrated on RDF datasets [11]. However, this did not address the broader problem of the class of constraints that can be formulated using the full expressivity of a given DL. The current paper addresses this limitation by providing (1) a formalization for Armstrong ABoxes for the class of constraints that can be formulated in $\mathcal{ALC}$ and (2) the algorithms necessary for generating an Armstrong ABox for an $\mathcal{ALC}$ TBox.

This paper is structured as follows: In Section 2 we review key definitions and results that are of importance in the development of Armstrong ABoxes, in Section 3 we provide the core definitions and proofs related to Armstrong ABoxes and Section 4 concludes this paper.

2 Preliminaries

We define the syntax and semantics for $\mathcal{ALC}$ in Section 2.1. The algorithm for generating an Armstrong ABox for an $\mathcal{ALC}$ TBox is based on the ontology completion algorithm of Baader et. al. [2] (Section 2.5), which is based on the attribute exploration algorithm of formal concept analysis (FCA) [9]. Core FCA definitions are reviewed in Section 2.3 and the attribute exploration algorithm is reviewed in Section 2.4. A key insight used in attribute exploration is to enumerate closed sets in lectic order, which we review in Section 2.2.

2.1 Description Logics (DLs)

The syntactic building blocks for an arbitrary DL are based on the disjoint sets N_C, N_R and N_I, where N_C is a set of concept names, N_R is a set of role names and N_I is a set of individual names. Concept names represent classes of entities (called concepts) that share common characteristics, roles names denote binary relations (called roles) that exist between individuals and individual names are used to refer to specific instances (called individuals) in a domain of interest [1].

$\mathcal{ALC}$ concept descriptions (referred to as concepts) are constructed using the following concept constructors

$$C := \top \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists r.C$$

where A is an atomic concept, C, C_1 and C_2 are (possibly complex) concepts and r is a role. The constructors $\sqcup$ and $\forall$, and the special concept $\bot$ are defined in terms of the others in the usual way [1].

The TBox consists of axioms $C_1 \sqsubseteq C_2$ called **general concept inclusions** (GCIs) stating that C_1 is subsumed by C_2. The ABox consists of **assertions** $C(x)$ and $r(x_1, x_2)$ stating respectively that individual x is an instance of C and that individuals x_1 and x_2 are associated via role r.

The semantics of concepts is given in terms of an interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\triangle^{\mathcal{I}}$ (the domain) is a non-empty set, and $\cdot^{\mathcal{I}}$ (the interpretation function) maps each concept name $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$, each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$, and each individual name $a \in N_I$ to an element $a^{\mathcal{I}} \in \triangle^{\mathcal{I}}$.

Given an interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$, the function $\cdot^{\mathcal{I}}$ is extended to interpret complex concepts in the following way:

$$\top^{\mathcal{I}} = \triangle^{\mathcal{I}}, (\neg C)^{\mathcal{I}} = \triangle^{\mathcal{I}} \backslash C^{\mathcal{I}}, (C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$$
$$(\exists r.C)^{\mathcal{I}} = \{x \in \triangle^{\mathcal{I}} | \text{A } y \text{ exists such that } (x, y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$$

When an interpretation $\mathcal{I}$ **satisfies** a GCI or assertion α it is denoted by $\mathcal{I} \Vdash \alpha$. Satisfaction of α is defined as follows: $\mathcal{I} \Vdash C_1 \sqsubseteq C_2$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, $\mathcal{I} \Vdash C(x)$ iff $x^{\mathcal{I}} \in C^{\mathcal{I}}$, and $\mathcal{I} \Vdash r(x, y)$ iff $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$. $\mathcal{I}$ is a **model** of a TBox $\mathcal{T}$ or an ABox $\mathcal{A}$ if it satisfies all its GCIs or assertions. In case $\mathcal{I}$ is a model of both $\mathcal{T}$ and $\mathcal{A}$, it is also called a model of the ontology $(\mathcal{T}, \mathcal{A})$ and $(\mathcal{T}, \mathcal{A})$ is said to be **consistent** if such a model exists.

An axiom or assertion α is said to be **entailed** by an ontology $\mathcal{O}$, written as $\mathcal{O} \vDash \alpha$, if every model of $\mathcal{O}$ is also a model of α. For a set of axioms $\Sigma = \{\sigma_0, \ldots, \sigma_n\}$, we abbreviate $\mathcal{O} \vDash \sigma_0, \ldots, \mathcal{O} \vDash \sigma_n$ with $\mathcal{O} \vDash \Sigma$. If $\mathcal{O}$ is empty, we abbreviate $\mathcal{O} \vDash \alpha$ as $\vDash \alpha$.

2.2 Closed Sets in Lectic Order

A set M with n elements has 2^n subsets. The set of all subsets of M is denoted by 2^M and is called the **powerset of** M. For S a finite set and $\mathcal{C} \subseteq 2^S$, $\mathcal{C}$ is a

set of subsets that is called a **closure system** on S if $S \in \mathcal{C}$, and $B_1, B_2 \in \mathcal{C}$ implies $B_1 \cap B_2 \in \mathcal{C}$ [7]. A mapping function $\varphi : 2^S \to 2^S$ is called a **closure operator** on S that assigns a **closure** $\varphi(B) \subseteq S$ to each subset $B \subseteq S$ if it is:

1. extensive: $B \subseteq \varphi(B)$ for all $B \subseteq S$,
2. monotone: $B_1 \subseteq B_2$ implies $\varphi(B_1) \subseteq \varphi(B_2)$, and
3. idempotent: $\varphi(\varphi(B)) = \varphi(B)$.

A subset $B \subseteq S$ is called a **closed set** w.r.t. φ if $\varphi(B) = B$ [7].

To generate all φ-closed sets for a set M, it is necessary to generate the closures for all $A \subseteq M$ w.r.t. φ. Assuming M consists of n elements, 2^n closures will be computed. Moreover, multiple sets $A \subseteq M$ may have the same closure. To only generate unique closures, lookups will have to be performed 2^n times. A more efficient means to generate closures for all subsets is to generate each closure only once. This can be achieved by generating closures in the lectic order [9].

Definition 1. *[9] Assume that $M = \{m_1, \ldots, m_n\}$ and fix some linear order $m_1 < \ldots < m_n$ on M. This order imposes a linear order on 2^M, called the **lectic order**, which is denoted by $<$: For $m_i \in M$ and $A, B \subseteq M$ the **order** $<_i$ is defined as*

$$A <_i B \text{ iff } m_i \in B, m_i \notin A \text{ and } \forall j < i.(m_j \in A \Leftrightarrow m_j \in B).$$

The order $<$ is the union of the orders $<_i$, i.e.,

$$A < B \text{ iff } A <_i B \text{ for some } m_i \in M.$$

Definition 1 states that A is lectically smaller than B if the smallest i for which the element m_i differs between sets A and B, m_i belongs to B and not to A. Note that $<$ extends the strict subset order because if $A \subset B$ it follows that $A < B$ since all the elements in which sets A and B differ belong to B. Thus $\emptyset$ is the smallest and M the largest set w.r.t. $<$.

All φ-closed sets can be generated exactly once for a set M when the closed sets are generated in the lectic order. Given a set $A \subseteq M$ it is possible to determine the next closed set in the lectic order, which is shown in Proposition 1.

Proposition 1. *[9] Given a closure operator φ on M and a φ-closed set $A \subset M$, the next φ-closed set following A in the lectic order is*

$$\varphi((A \cap \{m_1, \ldots, m_{i-1}\}) \cup \{m_i\})$$

where i is maximal such that $A <_i \varphi((A \cap \{m_1, \ldots, m_{i-1}\}) \cup \{m_i\})$.

The NextClosure algorithm (Algorithm 1) finds the next closed set in the lectic order for a given set $A \subseteq M$ and a closure operator φ. To find an m with maximal index i it traverses M in reverse linear order. Two cases need to be considered: either $m \in A$ or $m \notin A$. If $m \in A$ we can remove m from A since it will not result in a φ-closed set that is different from A. If $m \notin A$ we calculate $B := \varphi(A \cup \{m\})$ and if $B \setminus A$ has no element $< m$, we have found our next φ-closed set, otherwise we continue to the next m in reverse linear order. If no next φ-closed set can be found, the empty set is returned. All closed sets for M and φ can be generated by iterating through closed sets using NextClosure starting with $A := \varphi(\emptyset)$ and terminating when $A = M$.

2.3 Formal Concept Analysis (FCA)

FCA [9] is a field of applied mathematics that is based on a lattice-theoretic formalization of the notions of concept and conceptual hierarchy.

Definition 2. *[9] A **formal context** is a triple $\mathbb{K} = (G, M, I)$, where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a relation that associates each object g with the attributes satisfied by g. In order to express that an object g is in relation I with an attribute m, we write gIm.*

A formal context can be visualised as a crosstable, where the rows represent the objects, and the columns represent the attributes. A cross in column m of row g means that object g has attribute m, absence of a cross means that object g does not have attribute m.

Let $\mathbb{K} = (G, M, I)$ be a formal context. For a set of objects $A \subseteq G$, the **intent** A' of A is the set of attributes that are satisfied by all objects in A, which is defined as $A' := \{p \in M | \forall a \in A : aIp\}$. For a set of attributes $B \subseteq M$, the **extent** B' of B is the set of objects that satisfy all attributes in B, which is defined as $B' := \{o \in G \mid \forall b \in B : oIb\}$. A **formal concept** of $\mathbb{K}$ is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. The sets A and B are called the **concept extent** and the **concept intent** of the formal concept (A, B), respectively [9].

For the operators $\cdot'$ and $\cdot''$ ($\cdot'$ applied twice) the following statements hold [9]:

1. $A_1 \subseteq A_2$ implies $A_2' \subseteq A_1'$ (resp. $B_1 \subseteq B_2$ implies $B_2' \subseteq B_1'$),
2. $A_1 \subseteq A_1''$ (resp. $B_1 \subseteq B_1''$), and
3. the $\cdot''$-operator is a closure operator on both G and M and the set of concept intents (resp. concept extents) is a closure system on M (resp. G).

A formal context can be analyzed by studying the implications between attributes in the context, which motivates the next definition.

Definition 3. *[9] Let $\mathbb{K} = (G, M, I)$ be a formal context. An **implication** between the attributes in M is a pair of sets $L, R \subseteq M$, usually written as $L \to R$. An implication $L \to R$ **holds** in $\mathbb{K}$ if every object of $\mathbb{K}$ that has all the attributes in L also has all the attributes in R, i.e. if $L' \subseteq R'$. We denote the set of implications that hold in $\mathbb{K}$ by $Imp(\mathbb{K})$. A subset $X \subseteq M$ **respects an implication** $L \to R$ if $L \nsubseteq X$ or $R \subseteq X$. A subset $X \subseteq M$ **respects a set $\mathcal{L}$ of implications** if X respects every implication in $\mathcal{L}$. An implication $L \to R$ **follows** from a set of implications $\mathcal{L}$ if every subset $X \subseteq M$ that respects all implications in $\mathcal{L}$ also respects $L \to R$.*

Proposition 2. *If $\mathcal{L}$ is a set of implications over M, then $Mod(\mathcal{L}) := \{X \subseteq M \mid X \text{ respects } \mathcal{L}\}$ is a closure system on M. If $\mathcal{L} = Imp(\mathbb{K})$ for some formal context $\mathbb{K}$, then $Mod(\mathcal{L})$ is the system of all concept intents.*

Algorithm 1 NextClosure(M, A, φ)

for all $m \in M$ in reverse order **do**
  if $m \in A$ **then**
    $A := A \backslash \{m\}$
  else
    $B := \varphi(A \cup \{m\})$
    if $B \backslash A$ contains no element $< m$ **then**
      return B
    end if
  end if
end for
return $\emptyset$

Algorithm 2 ImplicationClosure(X, $\mathcal{L}$)

repeat
  stable := true
  for all $L \to R \in \mathcal{L}$ **do**
    if $L \subseteq X$ **then**
      $X := X \cup R$
      stable := false
      $\mathcal{L} := \mathcal{L} \backslash \{L \to R\}$
    end if
  end for
until stable
return X

Algorithm 3 AttributeExploration(M, $\mathbb{K}$)

$\mathcal{L} := \emptyset$; $L := \emptyset$
while $L \neq M$ **do**
  if $L \neq L''$ **then**
    Ask expert whether $L \to L''$ holds
    if yes **then**
      $\mathcal{L} := \mathcal{L} \cup \{L \to L''\}$
      $L :=$NextClosure($M, L,$ ImplicationClosure$(L, \mathcal{L})$)
    else
      Get an object o from the expert s.t. $L \subseteq o'$ and $L'' \nsubseteq o'$
      $\mathbb{K} := \mathbb{K} \cup \{o\}$
    end if
  else
    $L :=$NextClosure($M, L,$ ImplicationClosure$(L, \mathcal{L})$)
  end if
end while
return $\mathcal{L}, \mathbb{K}$

The **implication closure operator** for the closure system $\mathrm{Mod}(\mathcal{L})$ is given by $\mathcal{L} : 2^M \to 2^M$, which can be defined iteratively as

$$X^{\mathcal{L}} := X \cup \bigcup \{R \mid L \to R \in \mathcal{L}, L \subseteq X\}$$
$$X^{\mathcal{L}\mathcal{L}} := X^{\mathcal{L}} \cup \bigcup \{R \mid L \to R \in \mathcal{L}, L \subseteq X^{\mathcal{L}}\}$$
$$\vdots$$

From the sets $X^{\mathcal{L}}, X^{\mathcal{L}\mathcal{L}}, X^{\mathcal{L}\mathcal{L}\mathcal{L}}, \ldots$ a set $\mathcal{L}(X) := X^{\mathcal{L}\ldots\mathcal{L}}$ is obtained with $\mathcal{L}(X)^{\mathcal{L}} = \mathcal{L}(X)$, which can be calculated by Algorithm 2. For a given set X of attributes and a given set $\mathcal{L}$ of implications, it repeatedly iterates through the implications in $\mathcal{L}$, expanding X with R and removing $L \to R$ from $\mathcal{L}$ whenever $L \subseteq X$. Once X can no longer be expanded, the algorithm terminates [9].

2.4 Attribute Exploration

Attribute exploration is used where $\mathbb{K}$ is not known but can become known by posing questions to a domain expert. It is assumed that the domain expert is able to answer whether an implication holds in $\mathbb{K}$ or, in case an implication does not hold, give a counterexample of the implication. In order to make efficient use of an expert's time, attribute exploration uses minimal implication bases. The set of implications $\mathcal{L}$ is an **implication base** of $\mathbb{K}$ if

1. $\mathcal{L}$ is sound for $\mathbb{K}$, i. e. every implication from $\mathcal{L}$ holds in $\mathbb{K}$,
2. $\mathcal{L}$ is complete for $\mathbb{K}$, i. e. every implication that holds in $\mathbb{K}$ follows from $\mathcal{L}$, and
3. $\mathcal{L}$ is non-redundant for $\mathbb{K}$, i. e. no implication in $\mathcal{L}$ follows from other implications in $\mathcal{L}$.

For a given formal context $\mathbb{K}$ multiple implication bases can exist, of which the simplest is the implication base consisting of all implications. But because such an implication base can be too large to be practical, there is an interest in implication bases with minimal cardinality. Duquenne and Guigues showed that a minimal implication base can be constructed for every formal context. This implication base relies on the notion of pseudo-intents, which is defined recursively as follows – A subset of attributes $L \subseteq M$ is called a **pseudo-intent** of the context $\mathbb{K} = (G, M, I)$ iff (1) $L \neq L''$ (L is not a concept intent), and (2) if $L_0 \subsetneq L$ is a pseudo-intent and a proper subset of L, then $L_0'' \subseteq L$ holds [10, 9].

Theorem 1. *[10, 9]* $\{L \to L'' \mid L$ *is a pseudo-intent of* $\mathbb{K}\}$ *is a set of implications that is a minimal implication base of* $\mathbb{K} = (G, M, I)$*, called the* ***Duquenne-Guigues base*** *of* $\mathbb{K}$*.*

Algorithm 3 details the attribute exploration algorithm. It takes as input a set M of attributes and a context $\mathbb{K}$. The implication set $\mathcal{L}$ is initialized as the empty set. Based on Theorem 1 it iterates through the pseudo-intents of $\mathbb{K}$

(line 3), starting from the smallest $\mathcal{L}$-closed left-hand side (line 1), asking the expert implication questions of the form $L \to L''$ (line 4). If the expert answers "yes", the implication $L \to L''$ is added to the implication set $\mathcal{L}$ and the next closed set in the lectic order is determined (line 7). If the expert answers "no", the expert should provide a counterexample (line 9) with which the context $\mathbb{K}$ is expanded (line 10). For this reason L is kept constant while L'' is recalculated (line 3). When L is not a pseudo-intent, the next $\mathcal{L}$-closed set is considered (line 13). It is proven that Algorithm 3 terminates and on termination returns the completed context $\mathbb{K}$ and $\mathcal{L}$, where $\mathcal{L}$ is a Duquenne-Guigues base of $\mathbb{K}$ [9].

2.5 Ontology Completion

Classical attribute exploration assumes that an expert has complete information. I.e., the absence of a cross in a crosstable means that object g does not have attribute m. However, in practice experts often only have partial knowledge. This inspired the introduction of partial contexts [17, 2, 18, 9]. The formalization of partial contexts we will use here is based on Baader et al. [2, 18].

Definition 4. *[18] A **partial object description (pod)** is a tuple (A, S) where $A, S \subseteq M$ are such that $A \cap S = \emptyset$. We call such a pod a **full object description (fod)** if $A \cup S = M$. A set of pods is called a **partial context** and a set of fods a **full context**.*

The pod (A, S) states that the object it describes is known to satisfy all attributes from A and to not satisfy any attribute from S. A full context coincides with a formal context: a set of fods $\overline{\mathcal{K}}$ corresponds to the formal context $\mathbb{K}_{\overline{\mathcal{K}}} := (\overline{\mathcal{K}}, M, I)$, where $(\overline{A}, \overline{S}) I m$ if and only if $m \in \overline{A}$ for all $(\overline{A}, \overline{S}) \in \overline{\mathcal{K}}$. A partial context can be extended by adding new pods or extending existing pods.

Definition 5. *[18] Let $\mathcal{L}$ be a set of implications and $\mathcal{K}$ a partial context. An implication is called **undecided** w.r.t. $\mathcal{K}$ and $\mathcal{L}$ if it neither follows from $\mathcal{L}$ nor is refuted by $\mathcal{K}$. It is **decided** w.r.t. $\mathcal{K}$ and $\mathcal{L}$ if it is not undecided w.r.t. $\mathcal{K}$ and $\mathcal{L}$.*

The attribute exploration of partial contexts tries to decide all undecided implications by either adding the implication to $\mathcal{L}$ or extending $\mathcal{K}$ such that it refutes the implication. If all implications are decided, then the goal is achieved.

Let $(\mathcal{T}, \mathcal{A})$ be a consistent DL ontology and M a finite set of concept descriptions. Any individual name a occurring in $\mathcal{A}$ gives rise to the partial object description $pod_{\mathcal{T},\mathcal{A}}(a, M) := (A, S)$ where $A := \{C \in M | \mathcal{T}, \mathcal{A} \vDash C(a)\}$ and $S := \{C \in M | \mathcal{T}, \mathcal{A} \vDash \neg C(a)\}$, and the whole ABox induces the partial context

$$\mathcal{K}_{\mathcal{T},\mathcal{A}}(M) := \{pod_{\mathcal{T},\mathcal{A}}(a, M) | a \text{ is an individual name occurring in } \mathcal{A}\}.$$

Any element $d \in \triangle^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ gives rise to the full example

$$fod_{\mathcal{I}}(d, M) := (\overline{A}, \overline{S}) \text{ where } \overline{A} := \{C \in M \mid d \in C^{\mathcal{I}}\} \text{ and}$$
$$\overline{S} := \{C \in M \mid d \in (\neg C)^{\mathcal{I}}\},$$

and the whole interpretation induces the full context $\mathcal{K}_{\mathcal{I}}(M) := \{fod_{\mathcal{I}}(d, M) \mid d \in \triangle^{\mathcal{I}}\}$. Note that $fod_{\mathcal{I}}(d, M)$ is indeed a fod since every $d \in \triangle^{\mathcal{I}}$ satisfies either $d \in C^{\mathcal{I}}$ or $d \in \triangle^{\mathcal{I}} \backslash C^{\mathcal{I}} = (\neg C)^{\mathcal{I}}$.

Definition 6. *[18] The implication $L \to R$ over the attributes M is **refuted** by the ontology $(\mathcal{T}, \mathcal{A})$ if it is refuted by $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$, and it is **refuted** by the interpretation $\mathcal{I}$ if it is refuted by $\mathcal{K}_{\mathcal{I}}(M)$. If an implication is not refuted by $\mathcal{I}$, then we say that it **holds** in $\mathcal{I}$. The set of implications over M that hold in $\mathcal{I}$ is denoted by $Imp_M(\mathcal{I})$. In addition, we say that $L \to R$ **follows** from $\mathcal{T}$ if $\sqcap L \sqsubseteq_{\mathcal{T}} \sqcap R$, where $\sqcap L$ and $\sqcap R$ respectively stand for the conjunctions $\bigsqcap_{C \in L} C$ and $\bigsqcap_{D \in R} D$, and $\sqcap L \sqsubseteq_{\mathcal{T}} \sqcap R$ is a shorthand for $\mathcal{T} \vDash \sqcap L \sqsubseteq \sqcap R$.*

Similar to attribute exploration it is sufficient to only consider implications whose left-hand sides are $\mathcal{L}$-closed with the right-hand side the largest R such that $L \to R$ is not refuted by $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$:

Proposition 3. *[18] For a left-hand side L and a partial context $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$, the largest right-hand side such that $L \to \mathcal{K}_{\mathcal{T},\mathcal{A}}(L)$ is not refuted by $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$ is*

$$\mathcal{K}_{\mathcal{T},\mathcal{A}}(L) := M \backslash \bigcup \{D \in M \mid \exists a.\, L \subseteq \{C \mid \mathcal{T}, \mathcal{A} \vDash C(a)\} \wedge \mathcal{T}, \mathcal{A} \vDash \neg D(a)\}.$$

The aim is for the ontology to describe an intended model. For a fixed set M of concepts, the ontology is complete if it contains all the relevant knowledge about implications between these concepts: if an implication holds in the intended interpretation, then it should follow from the TBox, and if it does not hold in the intended interpretation, then the ABox should contain a counterexample.

Definition 7. *[18] Let $(\mathcal{T}, \mathcal{A})$ be a DL ontology, M a finite set of concept descriptions, and $\mathcal{I}$ a model of $(\mathcal{T}, \mathcal{A})$. Then $(\mathcal{T}, \mathcal{A})$ is M-complete w.r.t. $\mathcal{I}$ if the following three statements are equivalent for all implications $L \to R$ over M:*

1. *$L \to R$ holds in $\mathcal{I}$.*
2. *$L \to R$ follows from $\mathcal{T}$.*
3. *$L \to R$ is not refuted by $(\mathcal{T}, \mathcal{A})$.*

Let $(\mathcal{T}_0, \mathcal{A}_0)$ be a DL ontology that also has $\mathcal{I}$ as a model. Then $(\mathcal{T}, \mathcal{A})$ is a completion of $(\mathcal{T}_0, \mathcal{A}_0)$ if it is complete and extends $(\mathcal{T}_0, \mathcal{A}_0)$, i.e., $\mathcal{T}_0 \subseteq \mathcal{T}$ and $\mathcal{A}_0 \subseteq \mathcal{A}$.

The attribute exploration algorithm for partial contexts can be adapted for ontology completion [2, 18] for which Proposition 4 and Theorem 2 state important results.

Proposition 4. *[18] Let $(\mathcal{T}, \mathcal{A}) := (\mathcal{T}_0, \mathcal{A}_0)$ be an ontology, M a finite set of concept descriptions, and $\mathcal{I}$ a model of $(\mathcal{T}, \mathcal{A})$. Then the ontology completion algorithm terminates, and upon termination outputs an ontology $(\mathcal{T}, \mathcal{A}) := (\mathcal{T}_n, \mathcal{A}_n)$ and a set of implications $\mathcal{L}$ such that*

1. *$\mathcal{L}$ is sound and complete for $Imp_M(\mathcal{I})$, and*

2. $(\mathcal{T}, \mathcal{A}) := (\mathcal{T}_n, \mathcal{A}_n)$ *refutes every implication that is refuted by* $\mathcal{I}$*.*

Theorem 2. *[18] Let* $(\mathcal{T}, \mathcal{A}) := (\mathcal{T}_0, \mathcal{A}_0)$ *be a ontology,* M *a finite set of concept descriptions, and* $\mathcal{I}$ *a model of* $(\mathcal{T}, \mathcal{A}) := (\mathcal{T}_0, \mathcal{A}_0)$*, and let* $(\mathcal{T}, \mathcal{A}) := (\mathcal{T}_n, \mathcal{A}_n)$ *be the knowledge base computed by the ontology completion algorithm. Then* $(\mathcal{T}, \mathcal{A}) := (\mathcal{T}_n, \mathcal{A}_n)$ *is a completion of* $(\mathcal{T}, \mathcal{A}) := (\mathcal{T}_0, \mathcal{A}_0)$*.*

3 Armstrong ABoxes

In this section we introduce Armstrong ABoxes for $\mathcal{ALC}$ TBoxes. The intent of Armstrong ABoxes is to create example data that satisfy all required constraints and violate all constraints that do not necessarily hold for a specific application domain. In this way Armstrong ABoxes are the DL equivalent of Armstrong relations of relational database theory.

3.1 Formal Definitions

For convenience the notation $\bigsqcap C_i$ and $\bigsqcap D_j$ will respectively be used as shorthand for $C_{i_0} \sqcap \ldots \sqcap C_{i_n}$ and $D_{j_0} \sqcap \ldots \sqcap D_{j_m}$.

To determine whether there are axioms that have been fortuitously missed, the ontology engineer wants to add assertions to the Armstrong ABox that serve as test data that violates these candidate axioms. This idea is motivated based on experimental results for Armstrong relations, where it was shown that experts more readily recognize meaningful constraints that have been missed when the missed constraints are violated by test data [14].

Definition 8. *Let* $\mathcal{T}$ *be a consistent* $\mathcal{ALC}$ *TBox and let*

$$\sigma' := \bigsqcap C_i \sqsubseteq \bigsqcap D_j$$

for which $\mathcal{T} \not\models \sigma'$ *holds. An ABox* $\mathcal{A}'$ *is a* ***violating exemplar*** *of the entailment* $\mathcal{T} \models \sigma'$ *if*

$$\left\{ (\bigsqcap C_i)(x), (\neg \bigsqcap D_j)(x) \right\} \subseteq \mathcal{A}'$$

holds for some named individual x *that does not appear in any other assertions of* $\mathcal{A}'$*. This is denoted by* $\mathcal{A}' \nVdash \sigma'$*.*

Similar to Armstrong relations, Armstrong ABoxes include assertions that represent example data that satisfy the constraints of the TBox.

Definition 9. *Let* $\mathcal{T}$ *be a consistent* $\mathcal{ALC}$ *TBox, and let*

$$\sigma := \bigsqcap C_i \sqsubseteq \bigsqcap D_j$$

for which $\mathcal{T} \models \sigma$ *and* $\not\models \sigma$ *holds. An ABox* $\mathcal{A}$ *is a* ***satisfying exemplar*** *of the entailment* $\mathcal{T} \models \sigma$ *if*

$$\left\{ (\bigsqcap C_i)(x), (\bigsqcap D_j)(x) \right\} \subseteq \mathcal{A}$$

holds for some named individual x *that does not appear in any other assertions of* $\mathcal{A}$*. This is denoted by* $\mathcal{A} \Vdash \sigma$*.*

In general, for a given TBox $\mathcal{T}$, the number of non-entailments and entailments can be infinite. For this reason, similar to ontology completion, Armstrong ABoxes consider for a TBox $\mathcal{T}$ a finite set M of interesting concept descriptions for which a partial context $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$ can be constructed. The number of implications that can hold in such a context is finite. This is the reason why Definitions 8 and 9 are defined in terms of assertions contained in $\mathcal{A}$ and $\mathcal{A}'$ respectively, rather than assertions entailed by $(\mathcal{T},\mathcal{A})$ and $(\mathcal{T},\mathcal{A}')$ respectively.

In Section 3.3 a variation of ontology completion/attribute exploration will be used to generate Armstrong ABoxes. One of the aspects in which Armstrong ABoxes deviate from ontology completion is that for Armstrong ABoxes one of two possibilities must hold for each implication considered: either it follows from the TBox in which case a satisfying exemplar is added, or it does not follow from the TBox in which case a violating exemplar is added. To ensure that every implication not following from the TBox can indeed be refuted, we introduce the following definition.

Definition 10. *Let M be a set of concept descriptions. M is said to be **permissible** if it is finite and no concept in M is equivalent to $\top$. Furthermore, we define $M^{\rightarrow}$ to be the set of GCIs representing the finite set of all the implications $L \rightarrow R$ over M.*

Armstrong ABoxes assume that $\mathcal{T}$ is to the knowledge of the domain expert an accurate representation of the application domain. As such an interpretation $\mathcal{I}$ exists that is a model for $\mathcal{T}$ and is representative of the knowledge of the domain expert. Hence, in accordance with Definition 7, for all implications that hold over M it follows that $L \rightarrow R$ holds in $\mathcal{I}$ and $L \rightarrow R$ follows from $\mathcal{T}$.

Definition 11. *Let $\mathcal{T}$ be a consistent $\mathcal{ALC}$ TBox and let M be permissible. Let*

$$\Sigma' := \{\sigma' \mid \mathcal{T} \not\models \sigma' \text{ and } \sigma' \in M^{\rightarrow}\}.$$

*Σ' is called the **candidate axiom set** of $\mathcal{T}$ over M. Assume $\Sigma' = \{\sigma'_0, \ldots, \sigma'_n\}$. An ABox $\mathcal{A}'$ is a **violating exemplar** of $\mathcal{T} \models \Sigma'$ if $\mathcal{A}' \not\Vdash \sigma'_0, \ldots, \mathcal{A}' \not\Vdash \sigma'_n$ holds. An ABox $\mathcal{A}'$ is a **minimal violating exemplar** of $\mathcal{T} \models \Sigma'$ iff there is no ABox $\mathcal{A}'_0 \subset \mathcal{A}'$ that is violating exemplar of $\mathcal{T} \models \Sigma'$.*

Definition 12. *Let $\mathcal{T}$ be a consistent $\mathcal{ALC}$ TBox and let M be permissible. Let*

$$\Sigma := \{\sigma \mid \mathcal{T} \models \sigma, \ \not\models \sigma \text{ and } \sigma \in M^{\rightarrow}\}.$$

*Σ is called the **entailment set** of $\mathcal{T}$ over M. Assume $\Sigma = \{\sigma_0, \ldots, \sigma_n\}$. An ABox $\mathcal{A}$ is a **satisfying exemplar** of $\mathcal{T} \models \Sigma$ if $\mathcal{A} \Vdash \sigma_0, \ldots, \mathcal{A} \Vdash \sigma_n$ holds, which is denoted by $\mathcal{A} \Vdash \Sigma$. An ABox $\mathcal{A}$ is a **minimal satisfying exemplar** of $\mathcal{T} \models \Sigma$ iff there is no ABox $\mathcal{A}_0 \subset \mathcal{A}$ that is satisfying exemplar of $\mathcal{T} \models \Sigma$.*

For the sake of brevity "over M" will sometimes be dropped with the tacit understanding that candidate axiom- and entailment sets are per definition constrained to some set M of concepts that is permissible.

The notion of an Armstrong ABox can now be defined. An Armstrong ABox is denoted by $\mathcal{A}_{\cup}$, which is pronounced as "A-shield".

Definition 13. *Let $\mathcal{T}$ be a consistent $\mathcal{ALC}$ TBox with Σ and Σ' respectively the entailment- and candidate axiom sets of $\mathcal{T}$. $\mathcal{A}_\square$ is said to be an* ***Armstrong ABox*** *for $\mathcal{T}$ if and only if:*

1. *for every $\sigma \in \Sigma$, $\mathcal{A}_\square \Vdash \sigma$ holds,*
2. *for every $\sigma' \in \Sigma'$, $\mathcal{A}_\square \nVdash \sigma'$ holds and*
3. *there is no proper subset of $\mathcal{A}_\square$ such that properties (1) and (2) hold.*

$\mathcal{O}_\square = \mathcal{T} \cup \mathcal{A}_\square$ is called an ***Armstrong ontology****.*

3.2 Key Attributes of Armstrong ABoxes

Before it can be proved that an Armstrong ontology is consistent, some intermediate results need to be proven first.

Lemma 1. *Let $(\mathcal{T}, \mathcal{A})$ be a consistent $\mathcal{ALC}$ ontology. Let $\mathcal{A}_1$ be a set of assertions that is satisfiable w.r.t. $\mathcal{T}$. Further assume that there is no individual x that appears in both $\mathcal{A}$ and $\mathcal{A}_1$. Then $(\mathcal{T}, \mathcal{A} \cup \mathcal{A}_1)$ is consistent.*

Proof. Since $(\mathcal{T}, \mathcal{A})$ is consistent and $(\mathcal{T}, \mathcal{A}_1)$ is consistent, $\mathcal{A} \cup \mathcal{A}_1$ will only be inconsistent w.r.t. $\mathcal{T}$ if:

1. there is a clash in $\mathcal{A} \cup \mathcal{A}_1$, or
2. $\mathcal{A} \cup \mathcal{A}_1$ is unsatisfiable w.r.t. $\mathcal{T}$.

For (1) to be the reason for the inconsistency there must be an individual x such that $\mathcal{A} \cup \mathcal{A}_1 \vDash C(x)$ and $\mathcal{A} \cup \mathcal{A}_1 \vDash \neg C(x)$, which is impossible because of the assumption that no individuals are shared between $\mathcal{A}$ and $\mathcal{A}_1$. For (2) to be the reason for the inconsistency, there must be some entailment $\mathcal{T} \vDash C \sqsubseteq D$ for which there is an individual x such that $\mathcal{A} \cup \mathcal{A}_1 \vDash C(x)$ and $\mathcal{A} \cup \mathcal{A}_1 \vDash \neg D(x)$, which is impossible because no individuals are shared between $\mathcal{A}$ and $\mathcal{A}_1$. Since $\mathcal{A} \cup \mathcal{A}_1$ is consistent w.r.t. $\mathcal{T}$ the consistency of $\mathcal{T}$ is not affected by $\mathcal{A} \cup \mathcal{A}_1$ (see Proposition 3.6 of [16] or p. 142 of [3]). Hence, $(\mathcal{T}, \mathcal{A} \cup \mathcal{A}_1)$ must be consistent.

Lemma 2. *Let $\mathcal{T}$ be a consistent $\mathcal{ALC}$ TBox with $\Sigma' = \{\sigma'_0, \ldots, \sigma'_n\}$ the candidate axiom set of $\mathcal{T}$. Then $(\mathcal{T}, \mathcal{A}'_0 \cup \ldots \cup \mathcal{A}'_n)$ is consistent where $\mathcal{A}'_0, \ldots, \mathcal{A}'_n$ are minimal violating exemplars corresponding to Σ'.*

Proof. Let $\mathcal{A}'_0 = \left\{(\bigsqcap C_i)(x), (\neg \bigsqcap D_j)(x)\right\}$ be a violating exemplar. Then $\mathcal{A}'_0$ will only be unsatisfiable w.r.t. $\mathcal{T}$ if

1. $(\neg \bigsqcap D_j) \equiv \bot$, or
2. $\mathcal{A}'_0 \vDash C(y)$ and $\mathcal{A}'_0 \vDash \neg C(y)$ for a concept C and a named individual y.

(1) is in contradiction with M being permissible and (2) is in contradiction with $\mathcal{A}'_0$ being a minimal violating exemplar (Definition 11). Thus, $\mathcal{A}'_0$ is satisfiable w.r.t. $\mathcal{T}$ and from Lemma 1 it follows that $(\mathcal{T}, \mathcal{A}'_0)$ is consistent. The result follows by induction over the n minimal violating exemplars using Lemma 1 and the fact that no individuals are shared between exemplars (Definition 8).

Lemma 3. *Let $\mathcal{T}$ be a consistent $\mathcal{ALC}$ TBox with $\Sigma = \{\sigma_0, \ldots, \sigma_n\}$ the entailment set of $\mathcal{T}$. Then $(\mathcal{T}, \mathcal{A}_0 \cup \ldots \cup \mathcal{A}_n)$ is consistent where $\mathcal{A}_0, \ldots, \mathcal{A}_n$ are minimal satisfying exemplars corresponding to Σ.*

Proof. Let $\mathcal{A}_0 = \left\{(\bigsqcap C_i)(x), (\bigsqcap D_j)(x)\right\}$ be a satisfying exemplar. That $(\mathcal{T}, \mathcal{A}_0)$ is consistent follows from Lemma 1, the fact that M is permissible and $\mathcal{A}_0$ being a minimal satisfying exemplar. The result follows by induction over the n minimal satisfying exemplars using Lemma 1 and the fact that no individuals are shared between exemplars (Definition 9).

From the preceding definitions it follows that an Armstrong ABox can be derived from TBox entailments and non-entailments. Conversely, TBox entailments and non-entailments can be derived from Armstrong ABoxes.

Theorem 3. *Let $\mathcal{T}$ be a consistent $\mathcal{ALC}$ TBox and let $\mathcal{O}_\square := \mathcal{T} \cup \mathcal{A}_\square$. Then:*

1. *$\mathcal{O}_\square$ is consistent,*
2. *$\sigma' \in \Sigma'$ if and only if $\mathcal{A}_\square \nVdash \sigma'$ and*
3. *$\sigma \in \Sigma$ if and only if $\mathcal{A}_\square \Vdash \sigma$.*

Proof. (1) That $\mathcal{O}_\square$ is consistent is an immediate consequence of Lemmas 1 - 3, the fact that an Armstrong ABox is minimal (it cannot be extended to contain assertions that clash) and $\Sigma \cap \Sigma' = \emptyset$.

(2) Let $\sigma' := \bigsqcap C_i \sqsubseteq \bigsqcap D_j$ where $\sigma' \in M^{\rightarrow}$. Then $\mathcal{A}_\square \nVdash \sigma'$ if and only if $\{(\bigsqcap C_i)(x), (\neg \bigsqcap D_j)(x)\} \subseteq \mathcal{A}_\square$ for some new individual x such that $\mathcal{T} \nvDash \sigma'$ if and only if $\sigma' \in \Sigma'$.

(3) Let $\sigma := \bigsqcap C_i \sqsubseteq \bigsqcap D_j$ where $\sigma \in M^{\rightarrow}$. Then $\mathcal{A}_\square \Vdash \sigma$ if and only if $\{(\bigsqcap C_i)(x), (\bigsqcap D_j)(x)\} \subseteq \mathcal{A}_\square$ for some new individual x such that $\mathcal{T} \vDash \sigma$ if and only if $\sigma \in \Sigma$.

3.3 Algorithm

The ontology completion algorithm can be adapted to generate Armstrong ABoxes. However, there are some differences between the assumptions for ontology completion versus for Armstrong ABoxes.

Armstrong ABoxes start with a non-empty consistent $\mathcal{ALC}$ TBox $\mathcal{T}$ which has been constructed in collaboration with a domain expert. Moreover, the assumption is that $\mathcal{T}$ is to the knowledge of the domain expert an accurate representation of the application domain. As such an interpretation $\mathcal{I}$ exists that is a model for $\mathcal{T}$ and is representative of the knowledge of the domain expert. In this regard Armstrong ABoxes differ from ontology completion since a model $\mathcal{I}$ of $\mathcal{T}$ matches the interpretation $\mathcal{I}$ known to the domain expert. Since the assumption is that $\mathcal{T}$ is representative of the application domain, $\mathcal{T}$ is not extended during the generation of an Armstrong ABox.

The main objective of Armstrong ABoxes is to generate an $\mathcal{A}_\square$ corresponding to $\mathcal{T}$ in accordance with Definition 13. This can be achieved by checking whether $\sqcap L \sqsubseteq_{\mathcal{T}} \sqcap R$ follows from $\mathcal{T}$, in which case a satisfying exemplar (Definition 9) is

Algorithm 4 NewQuestion($(\mathcal{T}, \mathcal{A})$,$L$)

$R := \mathcal{K}_{\mathcal{T},\mathcal{A}}(L)$ ▷ Proposition 3
undecided := $L \neq R$
$C := \bigsqcap_{C \in L} C$
$D := \bigsqcap_{D \in \mathcal{K}_{\mathcal{T},\mathcal{A}}(L)} D$
return undecided, C, D

Algorithm 5 GenerateArmstrongABox($\mathcal{T}$,M)

$\mathcal{L} := \emptyset; \mathcal{A}_\square := \emptyset; L := \emptyset; i := 0$
while $L \neq M$ **do**
  undecided := NewQuestion($(\mathcal{T}, \mathcal{A}_\square)$,$L$)
  if undecided = true **then**
    if $C \sqsubseteq_{\mathcal{T}} D$ **then**
      $\mathcal{L} := \mathcal{L} \cup \{L \to R\}$
      $L :=$NextClosure($M, L,$ ImplicationClosure($L, \mathcal{L}$))
      $\mathcal{A}_\square := \mathcal{A}_\square \cup \{C(x_i), D(x_i)\}; i := i + 1$
    else
      $\mathcal{A}_\square := \mathcal{A}_\square \cup \{C(x_i), \neg D(x_i)\}; i := i + 1$
    end if
  else
    $L :=$NextClosure($M, L,$ ImplicationClosure($L, \mathcal{L}$))
  end if
end while
return $\mathcal{L}$, $(\mathcal{T}, \mathcal{A}_\square)$

added to $\mathcal{A}_\square$ and the implication base $\mathcal{L}$ is extended with $L \to R$. If $\sqcap L \sqsubseteq \sqcap R$ does not follow from $\mathcal{T}$, a related violating exemplar (Definition 8) is added to $\mathcal{A}_\square$. No questions are posed to an expert during Armstrong ABox generation.

To make the correspondence between Armstrong ABoxes and partial contexts induced by DL ontologies explicit, assume that $\mathcal{O} = (\mathcal{T}, \emptyset)$ represents the TBox for which an Armstrong ABox has to be generated. This will result in the ontology $\mathcal{O}_\square = (\mathcal{T}, \mathcal{A}_\square)$ which has a model (Theorem 3). Assume that $\mathcal{I}_\square$ is a model of $\mathcal{O}_\square$. Then $\mathcal{O}_\square$ induces the partial context

$$\mathcal{K}_{\mathcal{T},\mathcal{A}_\square} := \{pod_{\mathcal{T},\mathcal{A}_\square}(a, M) \mid a \text{ is an individual name occurring in } \mathcal{A}_\square\},$$

where M is a set of $\mathcal{ALC}$ concepts that is permissible, $pod_{\mathcal{T},\mathcal{A}_\square}$ is defined as before for $\mathcal{A}_\square$ instead of $\mathcal{A}$ and $pod_{\mathcal{T},\mathcal{A}_\square}$ is a pod since $\mathcal{O}_\square$ is consistent (Theorem 3).

Any element $d \in \triangle^{\mathcal{I}_\square}$ of an interpretation $\mathcal{I}_\square$ gives rise to the full example

$$fod_{\mathcal{I}_\square}(d, M) := (\overline{A}, \overline{S}) \text{ where } \overline{A} := \{C \in M \mid d \in C^{\mathcal{I}_\square}\} \text{ and}$$
$$\overline{S} := \{C \in M \mid d \in (\neg C)^{\mathcal{I}_\square}\},$$

and the whole interpretation induces the full context

$$\mathcal{K}_{\mathcal{I}_\square}(M) := \{fod_{\mathcal{I}_\square}(d, M) \mid d \in \triangle^{\mathcal{I}_\square}\}.$$

Note that $fod_{\mathcal{I}_\square}(d, M)$ is indeed a fod since every $d \in \triangle^{\mathcal{I}_\square}$ satisfies either $d \in C^{\mathcal{I}_\square}$ or $d \in \triangle^{\mathcal{I}_\square} \backslash C^{\mathcal{I}_\square} = (\neg C)^{\mathcal{I}_\square}$.

For this reason many of the results of ontology completion applies directly to Armstrong ontologies with the assumption that $\mathcal{I} = \mathcal{I}_\square$.

Algorithm 5 generates an Armstrong ABox $\mathcal{A}_\square$, given a TBox $\mathcal{T}$ and M a set of concepts that is permissible. It initializes the implication set $\mathcal{L}$ and the Armstrong ABox $\mathcal{A}_\square$ to be empty. L is initialized with the smallest set in the lectic order for $\mathcal{L}(\cdot)$ and a counter is initialized that is used in creating new individuals for which assertions are added to $\mathcal{A}_\square$ (line 1).

Algorithm 4, which is called in line 3 of Algorithm 5, is introduced to simplify dealing with exemplars. It takes as input the TBox $\mathcal{T}$, the Armstrong ABox $\mathcal{A}_\square$ calculated this far, and the left-hand side L of the current implication under consideration. In line 1 it calculates the largest R for L that is not refuted by the partial context $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$ (Proposition 3). Line 2 checks whether the implication is undecided in the context $\mathcal{K}_{\mathcal{T},\mathcal{A}}$. Lines 3 and 4 are introduced to simplify the notation used in Algorithm 5.

Line 4 of Algorithm 5 ensures that only undecided implications are considered, otherwise it moves on to the next L in the lectic order under $\mathcal{L}(\cdot)$(line 13). An implication for which the related subsumption already follows from the TBox $\mathcal{T}$, the implication is added to $\mathcal{L}$ and a satisfying exemplar is added to $\mathcal{A}_\square$ (lines 6-8), otherwise a violating exemplar is added to $\mathcal{A}_\square$ (line 10). Since $\mathcal{L}$ changed in line 6, the next L in the lectic order under $\mathcal{L}(\cdot)$ is determined (line 7).

At termination Algorithm 5 returns an Armstrong ontology and a set of implications $\mathcal{L}$ that is sound and complete for $Imp_M(\mathcal{I}_\square)$ (Theorem 4).

Theorem 4. *Let $\mathcal{T}$ be a consistent $\mathcal{ALC}$ TBox and let M be permissible. Algorithm 5 terminates and upon termination it outputs the ontology $(\mathcal{T}, \mathcal{A}_\square)$ which is an Armstrong ontology and a set of implications $\mathcal{L}$ such that*

1. *$\mathcal{L}$ is sound and complete for $Imp_M(\mathcal{I}_\square)$, $\mathcal{I}_\square$ a model of $(\mathcal{T}, \mathcal{A}_\square)$ and*
2. *$(\mathcal{T}, \mathcal{A}_\square)$ refutes every implication refuted by $\mathcal{I}_\square$.*

Proof. That Algorithm 5 terminates and upon termination it outputs the set of implications $\mathcal{L}$ such that (1) and (2) holds, follows from Proposition 4.

That $(\mathcal{T}, \mathcal{A}_\square)$ is an Armstrong ontology follows from the following facts:

1. conditions (1) and (2) hold for the set of implications $\mathcal{L}$,
2. whenever $\sigma \in \Sigma$, a corresponding satisfying exemplar is added to $\mathcal{A}_\square$ (line 8),
3. for every $\sigma' \in \Sigma'$, a violating exemplar is added to $\mathcal{A}_\square$ (line 10),
4. no other other assertions are added to $\mathcal{A}_\square$, $\Sigma \cap \Sigma' = \emptyset$ and no variables are shared between exemplars (lines 8 and 10).

3.4 Example

Starting with $M = \{\mathsf{Composite}, \mathsf{Even}, \mathsf{Odd}, \mathsf{Prime}, \mathsf{Square}\}$ and the TBox

$$\mathcal{T}_0 = \{\mathsf{Composite} \sqsubseteq \top, \mathsf{Even} \sqsubseteq \top, \mathsf{Odd} \sqsubseteq \top, \mathsf{Prime} \sqsubseteq \top, \mathsf{Square} \sqsubseteq \top\},$$

Table 1. Armstrong ABox for $\mathcal{T}_0$ and $M = \{\mathsf{Composite}, \mathsf{Even}, \mathsf{Odd}, \mathsf{Prime}, \mathsf{Square}\}$

Violating exemplars	
$\top(x_1)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_1)$
$\mathsf{Square}(x_2)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime})(x_2)$
$\mathsf{Prime}(x_3)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Square})(x_3)$
$(\mathsf{Prime} \sqcap \mathsf{Square})(x_4)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Odd})(x_4)$
$\mathsf{Odd}(x_5)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_5)$
$(\mathsf{Odd} \sqcap \mathsf{Square})(x_6)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Prime})(x_6)$
$(\mathsf{Odd} \sqcap \mathsf{Prime})(x_7)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Square})(x_7)$
$(\mathsf{Odd} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_8)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even})(x_8)$
$\mathsf{Even}(x_9)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_9)$
$(\mathsf{Even} \sqcap \mathsf{Square})(x_{10})$	$\neg(\mathsf{Composite} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime})(x_{10})$
$(\mathsf{Even} \sqcap \mathsf{Prime})(x_{11})$	$\neg(\mathsf{Composite} \sqcap \mathsf{Odd} \sqcap \mathsf{Square})(x_{11})$
$(\mathsf{Even} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{12})$	$\neg(\mathsf{Composite} \sqcap \mathsf{Odd})(x_{12})$
$(\mathsf{Even} \sqcap \mathsf{Odd})(x_{13})$	$\neg(\mathsf{Composite} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{13})$
$(\mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Square})(x_{14})$	$\neg(\mathsf{Composite} \sqcap \mathsf{Prime})(x_{14})$
$(\mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime})(x_{15})$	$\neg(\mathsf{Composite} \sqcap \mathsf{Square})(x_{15})$
$(\mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{16})$	$\neg\mathsf{Composite}(x_{16})$
$\mathsf{Composite}(x_{17})$	$\neg(\mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{17})$
$(\mathsf{Composite} \sqcap \mathsf{Square})(x_{18})$	$\neg(\mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime})(x_{18})$
$(\mathsf{Composite} \sqcap \mathsf{Prime})(x_{19})$	$\neg(\mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Square})(x_{19})$
$(\mathsf{Composite} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{20})$	$\neg(\mathsf{Even} \sqcap \mathsf{Odd})(x_{20})$
$(\mathsf{Composite} \sqcap \mathsf{Odd})(x_{21})$	$\neg(\mathsf{Even} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{21})$
$(\mathsf{Composite} \sqcap \mathsf{Odd} \sqcap \mathsf{Square})(x_{22})$	$\neg(\mathsf{Even} \sqcap \mathsf{Prime})(x_{22})$
$(\mathsf{Composite} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime})(x_{23})$	$\neg(\mathsf{Even} \sqcap \mathsf{Square})(x_{23})$
$(\mathsf{Composite} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{24})$	$\neg\mathsf{Even}(x_{24})$
$(\mathsf{Composite} \sqcap \mathsf{Even})(x_{25})$	$\neg(\mathsf{Odd} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{25})$
$(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Square})(x_{26})$	$\neg(\mathsf{Odd} \sqcap \mathsf{Prime})(x_{26})$
$(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Prime})(x_{27})$	$\neg(\mathsf{Odd} \sqcap \mathsf{Square})(x_{27})$
$(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Prime} \sqcap \mathsf{Square})(x_{28})$	$\neg\mathsf{Odd}(x_{28})$
$(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Odd})(x_{29})$	$\neg(\mathsf{Prime} \sqcap \mathsf{Square})(x_{29})$
$(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Square})(x_{30})$	$\neg\mathsf{Prime}(x_{30})$
$(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Odd} \sqcap \mathsf{Prime})(x_{31})$	$\neg\mathsf{Square}(x_{31})$

Table 2. Armstrong ABox for $\mathcal{T}_1$ and $M = \{\mathsf{Composite}, \mathsf{Even}, \mathsf{Square}\}$

Violating exemplars	
$\top(x_1)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even} \sqcap \mathsf{Square})(x_1)$
$\mathsf{Square}(x_2)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Even})(x_2)$
$\mathsf{Even}(x_3)$	$\neg(\mathsf{Composite} \sqcap \mathsf{Square})(x_3)$
$(\mathsf{Even} \sqcap \mathsf{Square})(x_4)$	$\neg\mathsf{Composite}(x_4)$
$\mathsf{Composite}(x_5)$	$\neg(\mathsf{Even} \sqcap \mathsf{Square})(x_5)$
$(\mathsf{Composite} \sqcap \mathsf{Square})(x_6)$	$\neg\mathsf{Even}(x_6)$
$(\mathsf{Composite} \sqcap \mathsf{Even})(x_7)$	$\neg\mathsf{Square}(x_7)$

Algorithm 5 generates no satisfying exemplars (because there are no implications that follow from $\mathcal{T}_0$ for $\mathcal{K}_{\mathcal{I}_\square}(M)$) and the violating exemplars given in Table 1. Looking at the assertions for x_1, the expert realizes that some of these attributes will never occur together. Hence, the expert adds the axioms $\{\mathsf{Even} \sqsubseteq \neg\mathsf{Odd}, \mathsf{Composite} \sqsubseteq \neg\mathsf{Prime}, \mathsf{Square} \sqsubseteq \neg\mathsf{Prime}\}$. Regenerating the Armstrong ABox for $M = \{\mathsf{Composite}, \mathsf{Even}, \mathsf{Square}\}$ and

$$\mathcal{T}_1 = \mathcal{T}_0 \cup \{\mathsf{Even} \sqsubseteq \neg\mathsf{Odd}, \mathsf{Composite} \sqsubseteq \neg\mathsf{Prime}, \mathsf{Square} \sqsubseteq \neg\mathsf{Prime}\}$$

results in no satisfying exemplars and the violating exemplars given in Table 2. Looking at the assertions for x_4 in Table 2 the expert realizes that every integer that is an Even and a Square, will necessarily be a $\mathsf{Composite}$. Therefore the expert adds the GCI $\mathsf{Even} \sqcap \mathsf{Square} \sqsubseteq \mathsf{Composite}$.

Generating an Armstrong ABox for $\mathcal{T}_2 = \mathcal{T}_1 \cup \{\mathsf{Even} \sqcap \mathsf{Square} \sqsubseteq \mathsf{Composite}\}$ with $M = \{\mathsf{Composite}, \mathsf{Even}, \mathsf{Square}\}$ will generate the same violating exemplars as in Table 2 except for x_4 that will be generated as a satisfying exemplar $\{(\mathsf{Even} \sqcap \mathsf{Square})(x_4), \mathsf{Composite}(x_4)\}$.

Generating an Armstrong ABox for $\mathcal{T}_2$ with $M = \{\mathsf{Composite}, \mathsf{Odd}, \mathsf{Square}\}$ will result in violating exemplars similar to Table 2, except that Even will be replaced with Odd. At this point the expert decides that $\mathcal{T}_2$ is sufficiently refined. Note that $\mathcal{T}_2$ defines constraints that should hold while the violating exemplars of the Armstrong ABoxes define constraints that can be violated.

3.5 Discussion

Our motivation for developing Armstrong ABoxes is to help identify ontologies of which the specification are incomplete, based on an idea that is inspired by Armstrong relations in relational database theory. Armstrong relations are used to assist domain experts to remedy incomplete specifications by identifying constraints that have been omitted, both with regard to constraints that should hold and constraints that do not necessarily hold [14]. By generating "perfect test data" in an Armstrong ABox we present the ontology engineer with exemplars that ensure that all the constraints of an application domain are specified. In particular the test data provides exemplars that violate constraints that do not necessarily hold.

Potential benefits of Armstrong ABoxes are that exemplars can make the meaning of entailments and non-entailments apparent, particularly for users that may not be well versed in DLs. Moreover, violating exemplars can alert a user to the fact that an entailment that should not follow from the TBox, does follow from the TBox. This can happen where $\mathcal{A}_\square$ is generated for $\mathcal{T}_0$ after which $\mathcal{T}_0$ is amended resulting in $\mathcal{T}_1$, without regenerating $\mathcal{A}_\square$. Assuming $\mathcal{T}_1$ is consistent and $(\mathcal{T}_1, \mathcal{A}_\square)$ is inconsistent, the reason for the inconsistency will be due to a σ' such that $\mathcal{T}_0 \not\models \sigma'$, $\mathcal{A}_\square \not\Vdash \sigma'$ and $\mathcal{T}_1 \models \sigma'$. For $\mathcal{T}_0$ σ' represents a constraint that does not hold and for $\mathcal{T}_1$ a constraint that does hold. The question that the expert has to resolve is: Should σ' hold or is it an unintended side-effect of the changes to $\mathcal{T}_0$?

An advantage of Armstrong ABoxes over ontology completion is that there is no need to reclassify the ontology because no GCIs are added when an Armstrong ABox is generated. In the case of ontology completion the ontology has to be reclassified every time a new GCI is added [18] but for Armstrong ABoxes the ontology only needs to be reclassified after the expert reviewed the Armstrong ABox and decided to add GCIs.

Armstrong ABoxes give an expert the flexibility to focus only on exemplars of interest. Ontology completion (resp. attribute exploration) can be a time consuming process even though it is mathematically designed to minimize the number of questions posed to an expert. This problem is exacerbated when experts don't provide mathematically optimal counterexamples, which results in the number of questions that need to be answered to reduce at a much slower rate than what is possible through optimal counterexamples. Moreover, an expert may not be able to provide an answer without further investigation [9, 18]. OntoComP, an implementation of ontology completion, deals with these challenges by trying to find a counterexample that may already be present in the ABox and it allows experts in certain situations to skip questions [4]. In contrast our Armstrong ABox algorithm will by definition always provide optimal counterexamples (Definition 8) and because such an Armstrong ABox is generated without expert interaction, it affords the expert the flexibility to choose which exemplars to focus on and which to ignore.

Limitations of our Armstrong ABox algorithm include its exponential complexity similar to attribute exploration/ontology completion. Less expressive DLs with favourable reasoning complexity like $\mathcal{FL}$, $\mathcal{AL}$ and $\mathcal{EL}$ are impractical given our current formalization of violating exemplars, which requires full negation. Moreover, ontology engineers and domain experts are likely to keep M small as to limit the cognitive load in reviewing an Armstrong ABox. This means the exponential complexity is likely to have a limited effect in practice, but this still needs to be evaluated in practice.

4 Conclusion

In order to assist with incomplete specifications when ontology engineers model a domain of interest, we propose the use of an Armstrong ABox, a notion derived from Armstrong relations in relational database theory. We formalized the notion of Armstrong ABoxes for TBoxes in the $\mathcal{ALC}$ DL, including an algorithm for generating such Armstrong ABoxes. The generated test data in the Armstrong ABox could be regarded as "perfect test data" that satisfies both the constraints of the domain that should hold, as well as constraints of the domain that do not hold. Our approach is novel in ontology engineering even though similar approaches have been used before in relational database specifications. Integrating our approach into ontology engineering tools will facilitate in detecting incomplete specifications.

References

1. F. Baader, D. Calvanese, D. L McGuinness, D. Nardi, and P. F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, 2007.
2. F. Baader, B. Ganter, B. Sertkaya, and U. Sattler, *Completing Description Logic Knowledge Bases Using Formal Concept Analysis.*, Proceedings of the 20th International Joint Conference on Artificial Intelligence (M. M. Veloso, ed.), Morgan Kaufmann Publishers Inc, 2007, pp. 230–235.
3. F. Baader, I. Horrocks, and U. Sattler, *Description Logics*, Handbook of Knowledge Representation (F. Van Harmelen, V. Lifschitz, and B. Porter, eds.), Elsevier, 2007, pp. 135–179.
4. F. Baader and B. Sertkaya, *Usability Issues in Description Logic Knowledge Base Completion.*, Proceedings of the 7th International Conference on Formal Concept Analysis (S. Ferré and S. Rudolph, eds.), LNCS, vol. 5548, Springer, 2009, pp. 1–21.
5. B. Boehm and V.R. Basili, *Software Defect Reduction Top 10 List*, Computer **34** (2001), no. 1, 135–137.
6. F. P. Brooks, *No Silver Bullet: Essence and Accidents of Software Engineering*, Computer **20** (1987), no. 4, 10–19.
7. B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*, second ed., Cambridge University Press, Cambridge, 2002.
8. R. Fagin and M. Y. Vardi, *Armstrong Databases for Functional and Inclusion Dependencies.*, Information Processing Letters **16** (1983), no. 1, 13–19.
9. B. Ganter and S. Obiedkov, *Conceptual Exploration*, Springer Verlag, 2016.
10. J. L. Guigues and V. Duquenne, *Famille Minimale d'implications Informatives Résultant d'un Tableau de Données Binaires*, Mathématiques et Sciences Humaines **24** (1986), no. 95, 5–18.
11. H. Harmse, K. Britz, and A. Gerber, *Informative Armstrong RDF Datasets for n-ary Relations*, Formal Ontology in Information Systems Proceedings of the Tenth International Conference (S. Borgo and P. Hitzler, eds.), Frontiers in Artificial Intelligence and Applications, IOS Press, 2018, To appear.
12. H. F. Harmse, A. Britz, and A. Gerber, *Armstrong Relations for Ontology Design and Evaluation*, Proceedings of the 29th International Workshop on Description Logics (M. Lenzerini and R. Peñaloza, eds.), vol. 1577, CEUR-WS.org, 2016.
13. B. Konev, A. Ozaki, and F. Wolter, *A Model for Learning Description Logic Ontologies Based on Exact Learning*, 13th AAAI Conference on Artificial Intelligence, AAAI Press, February 2016.
14. W. Langeveldt and S. Link, *Empirical Evidence for the Usefulness of Armstrong Relations in the Acquisition of Meaningful Functional Dependencies*, Information Systems **35** (2010), no. 3, 352–374.
15. S. Link, *Armstrong Databases: Validation, Communication and Consolidation of Conceptual Models with Perfect Test Data*, Proceedings of the 8th Asia-Pacific Conference on Conceptual Modelling (Melbourne, Australia) (A. Ghose and F. A. Ferrarotti, eds.), vol. 130, Australian Computer Society, Inc., 2012, pp. 3–22.
16. B. Nebel, *Reasoning and Revision in Hybrid Representation Systems.*, Ph.D. thesis, Saarland University, 1995.
17. S. Rudolph, *Relational Exploration - Combining Description Logics and Formal Concept Analysis for Knowledge Specification*, Ph.D. thesis, TU Dresden, 2006.
18. B. Sertkaya, *Formal Concept Analysis Methods for Description Logics*, Ph.D. thesis, TU Dresden, 2008.