Tableau methods for reasoning with link keys

Manuel Atencia, Jérôme Euzenat, Chan Le Duc, Khadija Jradeh

Reasoning for the description logic $\mathcal{ALC}$ with link keys

Manuel Atencia[a], Jérôme Euzenat[a], Chan Le Duc[b], Khadija Jradeh[a,b]

[a] *Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France*
[b] *Université Paris 8, LIASD, F-93526 Saint-Denis, France*

Abstract

Data interlinking is a critical task for widening and enhancing linked open data. One way to tackle data interlinking is to use link keys, which generalise keys to the case of two RDF datasets described using different ontologies. Link keys specify pairs of properties to compare for finding same-as links between instances of two classes of two different datasets. Hence, they can be used for finding links. Link keys can also be considered as logical axioms just like keys, ontologies and ontology alignments. We introduce the logic $\mathcal{ALC}$+$\mathcal{LK}$ extending the description logic $\mathcal{ALC}$ with link keys. It may be used to reason and infer entailed link keys that may be more useful for a particular data interlinking task. We show that link key entailment can be reduced to consistency checking without introducing the negation of link keys. For deciding the consistency of an $\mathcal{ALC}$+$\mathcal{LK}$ ontology, we introduce a new tableau-based algorithm. Contrary to the classical ones, the completion rules concerning link keys apply to pairs of individuals not directly related. We show that this algorithm is sound, complete and always terminates.

Keywords: data interlinking, link keys, reasoning, description logics, consistency, tableaux.

1. Introduction

Data interlinking is the task of discovering IRI references in different RDF datasets that refer to the same thing. The output of data interlinking is a set of identity links, typically specified using the owl:sameAs property.

Email addresses: Manuel.Atencia@inria.fr (Manuel Atencia), Jerome.Euzenat@inria.fr (Jérôme Euzenat), Chan.Leduc@iut.univ-paris8.fr (Chan Le Duc), Khadija.Jradeh@inria.fr (Khadija Jradeh)

owl:sameAs links are crucial to ensure interoperability in linked open data [12].

Different approaches to data interlinking have been proposed [9, 20]. Link keys are among them [3, 11]. Link keys generalise keys to the case of two RDF datasets described using different ontologies. An example of a key is the following:

$$(\{\mathsf{creator}, \mathsf{title}\} \text{ key } \mathsf{Work}) \tag{1}$$

stating that whenever two instances of the class Work share values for role creator and for role title, respectively, then they denote the same entity. An example of a link key is:

$$(\{\langle \mathsf{creator}, \mathsf{auteur}\rangle, \langle \mathsf{title}, \mathsf{titre}\rangle\} \text{ linkkey } \langle \mathsf{NonFiction}, \mathsf{Essai}\rangle) \tag{2}$$

stating that whenever an instance of the class NonFiction and an instance of the class Essai, share values for roles author and auteur, and for roles title and titre, respectively, they denote the same entity.

By nature, link keys can be used for data interlinking. For instance, the previous link key could be used to discover links between books of two bibliographic datasets, one using an English vocabulary and the other one using a French vocabulary. The problem is that, in practice, link keys are not given and need to be found.

One source of link keys is domain knowledge given by experts. Link keys may also be automatically extracted from RDF data [3**?**]. Another possibility is to infer link keys, which is the topic of this paper. Indeed, link keys can be considered as logical axioms, and, together with other kinds of knowledge such as keys, ontologies and ontology alignments, may entail new link keys.

Link key inference can complement link key extraction. The link key extraction algorithm described in [3] limits the search of link keys to link keys composed of named properties and named classes only. This leaves out complex link keys that may be helpful in practice. Reasoning can be used to combine automatically extracted link keys with other available knowledge to infer complex link keys that may be more useful for a particular data interlinking task. Additionally, link key inference can be used to confirm the belief of a domain expert that, given some knowledge, a set of property pairs constitute a link key for a pair of classes.

The following example illustrates this. Knowledge is modelled in description logics, which are the basis for semantic web languages such as OWL2.

Example 1. *Consider two library catalogs about books. In the first one the main class is* Work *and contains a subclass* NonFiction*.* creator *and* title *are a key in this ontology for the* Work *class as described in (1). In the second ontology, there is a class* Essai *with* auteur*,* lecteur *and* titre *roles and classes of people such as* Philosophe*. An alignment tells us that an* NonFiction *is more general than a* Essai *which has at least one* Philosophe *as* lecteur *(e.g. reader), that* creator *is equivalent to* auteur *and that* title *is equivalent to* titre*. This can be expressed in description logics as:*

$$\mathit{NonFiction} \sqsubseteq \mathit{Work} \tag{3}$$

$$\mathit{title} \equiv \mathit{titre} \tag{4}$$

$$\mathit{creator} \equiv \mathit{auteur} \tag{5}$$

$$\mathit{NonFiction} \sqsupseteq \mathit{Essai} \sqcap \exists \mathit{lecteur}.\mathit{Philosophe} \tag{6}$$

The key can be expressed as a link key:

$$(\{\langle \mathsf{creator}, \mathsf{creator}\rangle, \langle \mathsf{title}, \mathsf{title}\rangle\} \text{ linkkey } \langle \mathsf{Work}, \mathsf{Work}\rangle) \tag{7}$$

The set of statements (3–7) is sufficient for generating some links. However, for a user, it is not easy to find this out and a program requires a lot of inferences. It is thus useful to find more direct link keys entailed: they will be easier to check by a user and can be directly processed by a link generator. For instance, the link key (8) is entailed by (3–7):

$$\begin{aligned}(\{\langle \mathsf{creator}, \mathsf{auteur}\rangle, \langle \mathsf{title}, \mathsf{titre}\rangle\} \text{ linkkey} \\ \langle \mathsf{NonFiction}, \mathsf{Essai} \sqcap \exists \mathsf{lecteur}.\mathit{Philosophe}\rangle)\end{aligned} \tag{8}$$

though the more simple link key (2) is not entailed.

In this paper, we introduce a reasoning algorithm to determine whether an axiom — a subsumption relation between two concepts, an assertion or a link key — is entailed from other axioms. We model knowledge in description logics and we restrict ourselves to the description logic $\mathcal{ALC}$, which allows expressing negation, conjunction and disjunction of concepts, and existential and universal role restrictions. The presented algorithm will be the basis for extensions to deal with more expressive description logics. The language used in Example 1 is slightly more expressive, as it covers role name equivalence, but it can be rewritten as an $\mathcal{ALC}$ ontology to take these into account.

The proposed reasoning algorithm extends the standard tableau-based algorithm for reasoning in $\mathcal{ALC}$ [26]. In this algorithm, entailment is reduced to consistency checking: to decide if an axiom α is entailed by a knowledge base $\mathcal{O} = \langle \mathcal{A}, \mathcal{T} \rangle$, consisting of a set $\mathcal{A}$ of assertional axioms and a set $\mathcal{T}$ of terminological axioms, is equivalent to checking if $\mathcal{O}$ with the negation of α is inconsistent, *i.e.* it does not have a model. We extend the $\mathcal{ALC}$ tableau algorithm for deciding entailment to $\mathcal{ALC}$ knowledge bases $\mathcal{O} = \langle \mathcal{A}, \mathcal{T}, \mathcal{LK} \rangle$ equipped with link keys. We show that link key entailment can be reduced to consistency checking by expressing the negation of a link key as a set of assertional axioms and we provide the necessary tableau completion rules to deal with link keys. We prove that the algorithm is sound, complete and that it always terminates. For that purpose, we use unravelled interpretations because the canonical $\mathcal{ALC}$ interpretations may not satisfy link keys.

The remainder of the paper is organised as follows. Section 2 positions our work with respect to data interlinking and works on reasoning with keys in description logics. Section 3 describes the tableau-based algorithm and proves its termination, soundness and correctness. Section 4 provides examples of the use of the algorithm. Section 5 proves the termination, soundness, correctness and complexity of the proposed algorithm. Section 6 concludes the paper and presents future work.

2. Related work

Data interlinking. Different approaches to data interlinking can be found in the literature. They can be divided into two main categories: numerical methods [28, 16, 27, 21] and logical methods [24, 13, 2, 1]. Numerical methods compute a similarity between resources based on their property values and establish links between those which are highly similar. Logical methods use an axiomatic characterisation of what makes two resources identical in order to find links between different datasets. Link keys fall into this category.

The added value of logical methods is to profit from logical reasoning. The works [24, 13, 2, 1] propose rule-based approaches to infer same-as links that are logically entailed from an input set of domain constraints and facts. In [13], Hogan *et al.* use a subset of OWL 2 RL/RDF rules to derive owl:sameAs relations within the whole linked open data corpora. In [2], Al-Bakri *et al.* propose a different method that queries the linked open data cloud to import only the necessary specific data for inferring or contradicting given target same-as facts. The method is based on the query-subquery algorithm for

answering Datalog queries over deductive databases. In [24], Saïs *et al.* introduce a logical method that translates into rules schema constraints of RDFS, extended with OWL-DL and SWRL primitives, and infers 100% correct decisions of reference reconciliation and no reconciliation. In [1], Al-Bakri *et al.* present a probabilistic framework to model and reason over uncertain RDF facts and rules that is based on probabilistic Datalog [10]. The authors report on experiments that demonstrate the gain of using reasoning (rule chaining) for data interlinking by comparing their method with Silk [28].

The above-mentioned approaches focus on link inference, whereas the main focus of the method described in this paper is to infer link keys. In this sense, it will complement these approaches, as link key inference will provide input knowledge to be translated into rules for inferring links.

Reasoning with keys and link keys in description logics. Keys have been introduced in description logics as global constraints in a specific KBox [7, 18] and as a new concept constructor [6]. Calvanese *et al.* [8] have shown how to formalize keys in the $\mathcal{DLR}$ logic, and proved that the reasoning problems such as satisfiability, entailment in that logic are ExpTime-complete. The authors have indicated that $\mathcal{DLR}$ allowing for an arity of relations greater than two and unary functional dependencies is undecidable. Keys based on features (functional roles whose value belongs to a concrete domain) have been introduced within the $\mathcal{ALCOK}(\mathcal{D})$ logic [18] and an extension of the tableau method has been provided to deal with these logics. All undecidable cases identified in this work are related to the presence of a concrete domain. Motik *et al.* [19] have proposed a combination of OWL-DL with a kind of rules, namely, DL-safe rules, which are restricted such that decidability is guaranteed. This restriction imposes that each variable occurring in the premise (body) of such a rule must be bound to an individual explicitly introduced in the initial ABox. Obviously, DL-safe rules allow to express keys whose variables refer only to initial ABox individuals.

Link keys generalise keys to the case of different RDF datasets which can be interpreted as description logics. The tableau method for reasoning with link keys in $\mathcal{ALC}$ provided in this paper modifies and extends the algorithm described in [11]. These modifications concern the addition of new completion rules and avoiding merging nodes. They have been applied to ensure soundness, completeness and termination of the algorithm.

3. Tableau method for $\mathcal{ALC}$+$\mathcal{LK}$

This section describes a tableau-based algorithm for reasoning with link keys in a centralised context where link keys are considered as specific axioms stored in an ontology. We restrict ourselves to the extension of the description logic $\mathcal{ALC}$ with link keys, denoted by $\mathcal{ALC}$+$\mathcal{LK}$.

First, we start in Section 3.1 by giving the necessary preliminaries to introduce the algorithm. Then we show in Section 3.2 that $\mathcal{ALC}$+$\mathcal{LK}$ ontology entailment can be reduced to ontology consistency checking. Finally, the algorithm described in Section 3.3 decides consistency of $\mathcal{ALC}$+$\mathcal{LK}$ ontologies.

3.1. Preliminaries

The logic $\mathcal{ALC}$+$\mathcal{LK}$ extends $\mathcal{ALC}$ with link keys made up of $\mathcal{ALC}$-concepts and role names. This is defined below.

Definition 1 (Syntax of $\mathcal{ALC}$+$\mathcal{LK}$)**.** *Let* $\mathbf{C}$*,* $\mathbf{R}$ *and* $\mathbf{I}$ *be non-empty sets of* concept names*,* role names *and* individuals*, respectively. The set of* $\mathcal{ALC}$-concepts *(or simply concepts) is the smallest set such that*

- *every concept name in* $\mathbf{C}$*,* $\top$ *and* $\bot$ *are concepts, and*
- *if* C, D *are concepts and* R *is a role name in* $\mathbf{R}$ *then* $C \sqcap D$*,* $C \sqcup D$*,* $\neg C$*,* $\forall R.C$ *and* $\exists R.C$ *are concepts.*

A general concept inclusion *(GCI) is an expression of the form* $C \sqsubseteq D$ *where* C, D *are concepts. A* terminology *or* TBox *is a finite set of GCIs.*

An ABox assertion *is an expression of the form* $C(a)$*,* $R(a, b)$*,* $a \approx b$ *or* $a \not\approx b$ *where* C *is a concept,* R *is a role name in* $\mathbf{R}$ *and* a, b *are individuals in* $\mathbf{I}$*. An* ABox *is a finite set of ABox assertions.*

An $\mathcal{ALC} + \mathcal{LK}$ link key[1] *(simply called link key) is an expression of the form* $(\{\langle P_1, Q_1\rangle, \ldots, \langle P_n, Q_n\rangle\}$ $\mathsf{linkkey}$ $\langle C, D\rangle)$ *such that* $\langle C, D\rangle$ *is a pair of* $\mathcal{ALC}$*-concepts and* $\{\langle P_1, Q_1\rangle, \ldots, \langle P_n, Q_n\rangle\}$ *is a non-empty sequence of pairs of role names in* $\mathbf{R}$*. An* LKBox *is a finite set of link keys.*

A triple $\mathcal{O} = (\mathcal{A}, \mathcal{T}, \mathcal{LK})$*, where* $\mathcal{T}$ *is a TBox,* $\mathcal{A}$ *is an ABox and* $\mathcal{LK}$ *is an LKBox, is called an* $\mathcal{ALC}$+$\mathcal{LK}$ ontology*.*

[1] Although it is possible to consider other types of link keys [**?**], in this paper, we restrict ourselves to this one. Strictly speaking, this is rather $\mathcal{ALC} + \mathcal{LK}^{\mathrm{in},*}$.

By abuse of notation, we will write $(\{\langle P_i, Q_i\rangle\}_{i=1}^{n}$ linkkey $\langle C, D\rangle)$ instead of $(\{\langle P_1, Q_1\rangle, \ldots, \langle P_n, Q_n\rangle\}$ linkkey $\langle C, D\rangle)$.

Below we define the semantics of $\mathcal{ALC}$+$\mathcal{LK}$.

Definition 2 (Semantics of $\mathcal{ALC}$+$\mathcal{LK}$)**.** *An* interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *is composed of a non-empty set* $\Delta^{\mathcal{I}}$*, called the* domain *of* $\mathcal{I}$*, and a* valuation $\cdot^{\mathcal{I}}$ *which maps every concept name to a subset of* $\Delta^{\mathcal{I}}$*, every role name to a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *and each individual to an element of* $\Delta^{\mathcal{I}}$*. The valuation is extended to constructed concepts such that, for all concepts* C, D *and role name* R*, the following is satisfied:*

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}
\end{aligned}$$

An interpretation $\mathcal{I}$ satisfies *a* GCI $C \sqsubseteq D$*, denoted by* $\mathcal{I} \models C \sqsubseteq D$*, if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$*.* $\mathcal{I}$ *is a* model *of a* TBox $\mathcal{T}$ *if* $\mathcal{I}$ *satisfies every GCI in* $\mathcal{T}$*.*

An interpretation $\mathcal{I}$ *satisfies the ABox assertions*

$$\begin{aligned}
C(a) &\text{ if } a^{\mathcal{I}} \in C^{\mathcal{I}} \\
R(a, b) &\text{ if } \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}} \\
a \approx b &\text{ if } a^{\mathcal{I}} = b^{\mathcal{I}} \\
a \not\approx b &\text{ if } a^{\mathcal{I}} \neq b^{\mathcal{I}}
\end{aligned}$$

Given an ABox assertion α*,* $\mathcal{I} \models \alpha$ *denotes that* $\mathcal{I}$ *satisfies* α*.* $\mathcal{I}$ *is a model of an ABox* $\mathcal{A}$ *if it satisfies every ABox assertion in* $\mathcal{A}$*.*

An interpretation $\mathcal{I}$ *satisfies a link key* $(\{\langle P_i, Q_i\rangle\}_{i=1}^{n}$ linkkey $\langle C, D\rangle)$*, which will be denoted by* $\mathcal{I} \models (\{\langle P_i, Q_i\rangle\}_{i=1}^{n}$ linkkey $\langle C, D\rangle)$*, if*

$$\begin{aligned}
&\forall \delta, \eta, x_1, \cdots, x_n \in \Delta^{\mathcal{I}}, \\
&\qquad \delta \in C^{\mathcal{I}} \wedge \eta \in D^{\mathcal{I}} \wedge \bigwedge_{1 \leq i \leq n} ((\delta, x_i) \in P_i^{\mathcal{I}} \wedge (\eta, x_i) \in Q_i^{\mathcal{I}}) \Rightarrow \delta = \eta
\end{aligned}$$

$\mathcal{I}$ is a model of an LKBox $\mathcal{LK}$ if $\mathcal{I}$ satisfies every link key in $\mathcal{LK}$.

An interpretation $\mathcal{I}$ is a model of an $\mathcal{ALC}$+$\mathcal{LK}$ ontology $\mathcal{O} = (\mathcal{A}, \mathcal{T}, \mathcal{LK})$ if $\mathcal{I}$ is a model of $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{LK}$. An ontology $\mathcal{O}$ is consistent *if there exists a model of $\mathcal{O}$. An ontology $\mathcal{O}$* entails *a GCI, an ABox assertion or a link key α, written $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α.*

We finish these preliminaries by introducing notations and definitions that will be used in the paper. We use $|S|$ to denote the cardinality of a set S. Given an $\mathcal{ALC}$+$\mathcal{LK}$ ontology $\mathcal{O} = \langle \mathcal{A}, \mathcal{T}, \mathcal{LK} \rangle$, we denote by $\mathsf{sub}(\mathcal{O}) = \mathsf{sub}(\mathcal{A}, \mathcal{T}, \mathcal{LK})$ the set of all sub-concepts occurring in $\mathcal{A}, \mathcal{T}$ and $\mathcal{LK}$. The size of an ontology $\mathcal{O}$ is denoted by $\|\mathcal{O}\| = \|\mathcal{A}\| + \|\mathcal{T}\| + \|\mathcal{LK}\|$ where $\|\mathcal{A}\|$ is the size of all assertions, $\|\mathcal{T}\|$ the size of all GCIs and $\|\mathcal{LK}\|$ the size of all link keys. It holds that $|\mathsf{sub}(\mathcal{O})|$ is polynomially bounded by $\|\mathcal{O}\|$ since if a concept is represented as string then a sub-concept is a substring.

Finally, given two individuals s, t in $\mathbf{I}$, we define the label of s as $L(s) = \{C \in \mathsf{sub}(\mathcal{O}) \mid C(s) \in \mathcal{A}\}$ and the label of $\langle s, t \rangle$ as $L(s,t) = \{R \in \mathbf{R} \mid R(s,t) \in \mathcal{A}\}$. We assume hereafter, without loss of generality, that the individuals of all ABoxes are labelled in this way.

3.2. Reduction of ontology entailment to ontology consistency

In $\mathcal{ALC}$+$\mathcal{LK}$, ontology entailment of GCIs, concept assertions, equality and inequality statements, and link keys is reducible to ontology consistency checking. Indeed, given an $\mathcal{ALC}$+$\mathcal{LK}$ ontology $\mathcal{O} = \langle \mathcal{A}, \mathcal{T}, \mathcal{LK} \rangle$, two concepts C, D and two individuals a, b:

$$\begin{aligned}
\mathcal{O} \models C \sqsubseteq D &\text{ iff } \langle \mathcal{A} \cup \{(C \sqcap \neg D)(x)\}, \mathcal{T}, \mathcal{LK} \rangle \text{ is inconsistent} \\
\mathcal{O} \models C(a) &\text{ iff } \langle \mathcal{A} \cup \{\neg C(a)\}, \mathcal{T}, \mathcal{LK} \rangle \text{ is inconsistent} \\
\mathcal{O} \models a \approx b &\text{ iff } \langle \mathcal{A} \cup \{a \not\approx b\}, \mathcal{T}, \mathcal{LK} \rangle \text{ is inconsistent} \\
\mathcal{O} \models a \not\approx b &\text{ iff } \langle \mathcal{A} \cup \{a \approx b\}, \mathcal{T}, \mathcal{LK} \rangle \text{ is inconsistent}
\end{aligned}$$

where x is a new individual not present in $\mathcal{O}$. Notice that ontology entailment of role assertions may require considering negation of roles, which go beyond $\mathcal{ALC}$+$\mathcal{LK}$ expressivity.

This result can be extended to link keys. It is not necessary to express link key negation, but sufficient to provide an ABox witnessing this negation. Lemma 1 below proves that link key entailment can be reduced to consistency checking: given a link key λ, $\mathcal{O} \models \lambda$ if and only if $\langle \mathcal{A} \cup \mathcal{A}', \mathcal{T}, \mathcal{LK} \rangle$ is inconsistent, where $\mathcal{A}'$ represents the negation of λ.

Lemma 1 (Reduction of ontology entailment to consistency)**.** *Let* $\mathcal{O} = \langle \mathcal{A}, \mathcal{T}, \mathcal{LK} \rangle$ *be an* $\mathcal{ALC}$+$\mathcal{LK}$ *ontology. It holds that*

$$\mathcal{O} \models (\{\langle P_i, Q_i \rangle\}_{i=1}^{n} \ \mathsf{linkkey} \ \langle C, D \rangle) \text{ iff } \langle \mathcal{A} \cup \mathcal{A}', \mathcal{T}, \mathcal{LK} \rangle \text{ is inconsistent}$$

with $\mathcal{A}' = \{C(x), D(y), x \not\approx y\} \cup \bigcup_{i=1}^{n} \{P_i(x, z_i), Q_i(y, z_i)\}$ *and* $x, y, z_1, \cdots, z_n$ *are new individuals not present in* $\mathcal{O}$*.*

Proof. Let $\lambda = (\{\langle P_i, Q_i \rangle\}_{i=1}^{n} \ \mathsf{linkkey} \ \langle C, D \rangle)$. Assume first that $\mathcal{O} \models \lambda$. Let us show that $\mathcal{O}' = \langle \mathcal{A} \cup \mathcal{A}', \mathcal{T}, \mathcal{LK} \rangle$ is inconsistent. By contradiction, assume that $\mathcal{O}'$ has a model $\mathcal{I}$. Since $\mathcal{A} \subseteq \mathcal{A} \cup \mathcal{A}'$, then $\mathcal{I}$ must be a model of $\mathcal{O}$ too. Moreover, since $\mathcal{I}$ is a model of $\mathcal{O}'$, $\mathcal{I}$ must be a model of $\mathcal{A}'$, which means that $x^{\mathcal{I}} \in C^{\mathcal{I}}$, $y^{\mathcal{I}} \in D^{\mathcal{I}}$, $\langle x^{\mathcal{I}}, z_i^{\mathcal{I}} \rangle \in P_i^{\mathcal{I}}$, $\langle y^{\mathcal{I}}, z_i^{\mathcal{I}} \rangle \in Q_i^{\mathcal{I}}$ and $x^{\mathcal{I}} \neq y^{\mathcal{I}}$. This implies that $\mathcal{I} \not\models \lambda$. Thus, we have a model $\mathcal{I}$ of $\mathcal{O}$ such that $\mathcal{I} \not\models \lambda$. Therefore, $\mathcal{O} \not\models \lambda$, which contradicts the assumption.

Assume now that $\mathcal{O} \not\models \lambda$. Let us show that $\mathcal{O}' = \langle \mathcal{A} \cup \mathcal{A}', \mathcal{T}, \mathcal{LK} \rangle$ is consistent. Since $\mathcal{O} \not\models \lambda$, then there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \not\models \lambda$ (otherwise λ would be entailed). Since $\mathcal{I} \not\models \lambda$, by the semantics of link keys, there exists $\delta, \delta', \delta_1, \ldots, \delta_n \in \Delta^{\mathcal{I}}$ such that $\delta \in C^{\mathcal{I}}$, $\delta' \in D^{\mathcal{I}}$, $(\delta, \delta_1) \in P_1^{\mathcal{I}}, (\delta', \delta_1) \in Q_1^{\mathcal{I}}, \ldots, (\delta, \delta_n) \in P_n^{\mathcal{I}}, (\delta', \delta_n) \in Q_n^{\mathcal{I}}$ and $\delta \neq \delta'$. Let us extend $\mathcal{I}$ by defining $x^{\mathcal{I}} = \delta$, $y^{\mathcal{I}} = \delta', z_1^{\mathcal{I}} = \delta_1, \ldots, z_n^{\mathcal{I}} = \delta_n$. Then, $\mathcal{I}$ is a model of $\mathcal{A}'$. $\mathcal{I}$ is still a model of $\mathcal{O}$. Therefore, $\mathcal{I}$ is a model of $\mathcal{O}'$ and, thus, $\mathcal{O}'$ is consistent. □

Thanks to Lemma 1, ontology entailment in $\mathcal{ALC}$+$\mathcal{LK}$ can be reduced to ontology consistency. The following section describes a tableau algorithm for checking the consistency of an ontology in $\mathcal{ALC}$+$\mathcal{LK}$.

3.3. Tableau algorithm for $\mathcal{ALC}$+$\mathcal{LK}$

The algorithm to decide if an ontology with link keys $\mathcal{O}_0 = \langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$ is consistent, starts with $\mathcal{A}_0$ and applies the completion rules listed in Figure 1, guided by $\mathcal{T}$ and $\mathcal{LK}$. The completion rules generate new ABoxes. If no more rule is applicable to a generated ABox and this ABox does not contain any obvious contradiction (called clash) then there exists a model of $\mathcal{O}_0$ that can be built from the ABox, otherwise no model exist. This algorithm is based on the standard tableau algorithm for reasoning in $\mathcal{ALC}$ [26] to which we have added specific completion rules for dealing with link keys.

More precisely, we use $\mathbf{A}$ to denote a set of ABoxes and $\langle \mathbf{A}, \mathcal{T}, \mathcal{LK} \rangle$ is a *generalised ontology* to be used by the method. At the beginning, $\mathbf{A}$ is

initialised with $\mathbf{A}_0 = \{\mathcal{A}_0\}$. $\langle \mathbf{A}, \mathcal{T}, \mathcal{LK} \rangle$ is said to be *consistent* if there exists $\mathcal{A} \in \mathbf{A}$ such that $\langle \mathcal{A}, \mathcal{T}, \mathcal{LK} \rangle$ is consistent.

The application of a completion rule transforms the set of ABoxes into another set of ABoxes. There are two types of rules: deterministic and non-deterministic rules. Each application of a deterministic rule replaces an ABox $\mathcal{A} \in \mathbf{A}_k$ by a new ABox $\mathcal{A}' \in \mathbf{A}_{k+1}$. However, the application of a non-deterministic rule replaces an ABox $\mathcal{A} \in \mathbf{A}_k$ by several new ABoxes $\mathcal{A}'_1 \ldots \mathcal{A}'_n \in \mathbf{A}_{k+1}$.

The algorithm then generates a sequence of sets of ABoxes:

$$\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \ldots \tag{9}$$

such that $\mathbf{A}_{k+1}$ is obtained from $\mathbf{A}_k$ by applying a completion rule. An ontology $\langle \mathcal{A}, \mathcal{T}, \mathcal{LK} \rangle$ with $\mathcal{A} \in \mathbf{A}_k$ is called a *derived ontology* from $\langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$ and $\mathcal{A}$ a *derived ABox*. Such a derived ABox $\mathcal{A}$, and the corresponding ontology, is called *complete* if no completion rule is applicable.

3.3.1. Preprocessing

As usual, to ease the description of the completion rules, we start with a preprocessing step. All concepts occurring in the initial ontology are expressed into negation normal form (NNF), *i.e.* negation only occurs in front of concept names. Any $\mathcal{ALC}$-concept can be transformed to an equivalent one in NNF by using De Morgan's laws and the duality between existential and universal restrictions. In addition, all concepts occurring in all link keys are in NNF as well. Note that the NNF of a concept C can be computed in polynomial time in the size of C [5]. For a concept C, $\sim C$ will denote the negation normal form of $\neg C$.

3.3.2. Blocking

As for $\mathcal{ALC}$ with GCIs, blocking (cycle detection) is necessary to ensure the termination of the algorithm. Before giving the definition of blocking, we make a distinction between old and new individuals. Let $\mathcal{O}_k = \langle \mathcal{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ be an $\mathcal{ALC}$+$\mathcal{LK}$ ontology with set of individuals $I \neq \emptyset$. Assume that $\mathcal{O}_k$ is derived from an initial ontology $\mathcal{O}_0 = \langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$ with a set of individuals I_0. We have $I_0 \subseteq I$. An individual $a \in I$ is called *old* if $a \in I_0$, and *new* otherwise. New individuals result from applying specific rules (in Figure 1, $\rightarrow_\exists$ is the only such rule). We will write $I = I_{old} \uplus I_{new}$ where $I_{old} = I_0$ and $I_{new} = I \setminus I_0$. In particular, $\mathcal{O}_0$ has old individuals only, and no new individuals.

We assume that there is a total order over $I_{old} = \{s_1, \ldots, s_n\}$ with $s_i < s_j$ for all $1 \leq i < j \leq n$. If a rule adds a new individual s to an ABox, then $<$ is extended by setting $s_i < s$ for all $1 \leq i \leq n$ and $t < s$ if t was added to the ontology prior to s. By construction, $<$ is a total order over $I = I_{old} \uplus I_{new}$.

For the sake of simplicity, we assume that, if an equality assertion $x \approx y$ or an inequality assertion $x \not\approx y$ belongs to $\mathcal{A}_0$ then $x < y$. This has no impact on consistency checking because $\approx$ and $\not\approx$ are symmetric.

Definition 3 (Order and equivalence among individuals)**.** *Let $\mathcal{O} = \langle \mathcal{A}, \mathcal{T}, \mathcal{LK} \rangle$ be an $\mathcal{ALC}$+$\mathcal{LK}$ ontology with a set of individuals I and an order relation $<$ over I. For each individual $s \in I$, we use s^+ to denote the transitive closure of s with respect to the relation $\approx$ (appearing in assertions), i.e. s^+ is the smallest set such that $s \in s^+$, and if $c \approx b \in \mathcal{A}$ or $b \approx c \in \mathcal{A}$ with some $c \in s^+$ then $b \in s^+$. The function $\mathsf{e}(s)$ associates to each individual s the smallest element of s^+ with respect to the order relation $<$.*

Below we give the definition of a blocked element.

Definition 4 (Blocking)**.** *Let $\mathcal{O}_k = \langle \mathcal{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ be a derived $\mathcal{ALC}$+$\mathcal{LK}$ ontology with a set of individuals $I = I_{old} \uplus I_{new}$. An individual $s \in I_{new}$ is* blocked *by an individual $t \in I_{new}$ if $t < s$ and $L(s) \subseteq L(t)$. We denote by $\mathsf{b}(s)$ the least individual (with respect to the total order $<$) that blocks s.*

Notice that only new individuals may be blocked. Also, given a blocked element $s \in I_{new}$, the existence and uniqueness of $\mathsf{b}(s)$ is guaranteed by the fact that $<$ is a finite strict total order (and, thereby, a well-order), so the set of blocking elements of s, which is not empty, has a least element in $<$ which is unique. The following lemma proves that $\mathsf{b}(s)$ is always non blocked.

Lemma 2. *Let $\mathcal{O}_k = \langle \mathcal{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ be a derived $\mathcal{ALC}$+$\mathcal{LK}$ ontology with a set of individuals I. If $s \in I$ is a blocked individual then $\mathsf{b}(s)$ is not blocked.*

Proof. By contradiction, assume that $\mathsf{b}(s)$ is blocked by an individual $t \in I$. Then, $t < \mathsf{b}(s)$ and $L(\mathsf{b}(s)) \subseteq L(t)$. Since s is blocked by $\mathsf{b}(s)$, we have $\mathsf{b}(s) < s$ and $L(s) \subseteq L(\mathsf{b}(s))$. Hence, $t < \mathsf{b}(s) < s$ and $L(s) \subseteq L(t)$, which contradicts the definition of $\mathsf{b}(s)$. □

3.3.3. Clashes

Clashes are atomic contradictions. Given an ontology with link keys $\mathcal{O} = \langle \mathcal{A}, \mathcal{T}, \mathcal{LK} \rangle$, we will say that $\mathcal{A}$ contains a clash if one of the two following situations occurs:

- **$\neg$-clash**: $\{A(s), \neg A(s)\} \subseteq \mathcal{A}$ for some individual name s and a concept name A, or

- **$\not\approx$-clash**: $\{x \not\approx y\} \subseteq \mathcal{A}$ with $x \in y^+$ for some individuals x, y.

If $\mathcal{A}$ contains no clash, we say that $\mathcal{A}$, and $\mathcal{O}$, is *clash-free*.

The case when $\{\bot(s)\} \subseteq \mathcal{A}$, for some individual s, will be considered a $\neg$-clash too (implicitly, $\{\bot \sqsubseteq \neg\top\} \subseteq \mathcal{T}$ and $\top(t) \in \mathcal{A}$ for all t). We will write $\mathcal{A} \to \neg$-clash and $\mathcal{A} \to \not\approx$-clash if $\mathcal{A}$ contains, respectively, a $\neg$-clash or a $\not\approx$-clash.

3.3.4. Completion rules

Completion rules transform the ABox of a generalised ontology. They leave the TBox and LKBox unchanged. This transformation is monotonic, *i.e.* it only adds new assertions and never removes anything from the ontology.

Figure 1 shows the list of completion rules of the algorithm. They are standard completion rules for reasoning in $\mathcal{ALC}$ together with three more rules to deal with link keys ($\to_{\text{chooseLK1}}$, $\to_{\text{chooseLK2}}$ and $\to_{\text{LK}}$) and a rule to handle equality ($\to_{\approx}$). The $\to_{\text{LK}}$ rule translates the semantics of link keys. The $\to_{\text{chooseLK1}}$ and $\to_{\text{chooseLK2}}$ rules make it explicit whether two individuals a and b that satisfy the condition of a link key should be set as equal or not. Certainly, given an interpretation $\mathcal{I}$, the absence of an assertion $C(a)$ (resp. $D(b)$) from an ontology does not necessarily imply that $a^{\mathcal{I}} \notin C^{\mathcal{I}}$ (resp. $b^{\mathcal{I}} \notin D^{\mathcal{I}}$). For this purpose, we need to add $\sim C(a)$ (resp. $\sim D(b)$) explicitly.

Contrary to [11], the $\to_{\approx}$ rule does not remove any assertion from ABoxes. It just makes $L(x) = L(y)$, $L(x, z) = L(y, z)$, $L(z, x) = L(z, y)$ for some individual z if $x \approx y$ belongs to the ABox $\mathcal{A}$.

A derived ABox is *closed* if it is either complete or contains a clash. A generalised ontology $\langle \mathbf{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ is called *closed* if each $\mathcal{A} \in \mathbf{A}_k$ is closed. A closed generalised ontology $\langle \mathbf{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ is called *successful* if there exists $\mathcal{A} \in \mathbf{A}_k$ which is complete and clash-free.

At this point, we have all the necessary elements to present the algorithm for checking ontology consistency. Algorithm 1 below returns YES if it builds a successful generalised ontology $\langle \mathbf{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ from a generalised ontology $\langle \{\mathcal{A}_0\}, \mathcal{T}, \mathcal{LK} \rangle$, and NO otherwise.

Before proving termination, soundness and completeness of the algorithm, we illustrate it with examples.

Rule $\rightarrow_{\sqcap}$
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(s)$, but it does not contain both $C_1(s)$ and $C_2(s)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(s), C_2(s)\}$
Rule $\rightarrow_{\sqcup}$
Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(s)$, but neither $C_1(s)$ nor $C_2(s)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(s)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{C_2(s)\}$
Rule $\rightarrow_{\forall}$
Condition: $\mathcal{A}$ contains $(\forall R.C)(s)$ and $R(s,t)$, but it does not contain $C(t)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(t)\}$
Rule $\rightarrow_{\exists}$
Condition: $\mathcal{A}$ contains $(\exists R.C)(s)$ but there is no individual name t such that $\mathcal{A}$ contains $R(s,t)$ and $C(t)$, and s is not blocked.
Action: $\mathcal{A}' := \mathcal{A} \cup \{R(s,t), C(t)\}$ where t is an individual not occurring in $\mathcal{A}$. Set $x < t$ for all individuals x in $\mathcal{A}$.
Rule $\rightarrow_{\mathbf{choose}}$
Condition: $\mathcal{T}$ contains $C \sqsubseteq D$ and there is an individual name s such that $\mathcal{A}$ does contain neither $\sim C(s)$ nor $D(s)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{\sim C(s)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{D(s)\}$
Rule $\rightarrow_{\mathbf{chooseLK1}}$
Condition: $\mathcal{LK}$ contains $(\{\langle P_i, Q_i\rangle\}_{i=1}^n$ $\mathsf{linkkey}$ $\langle C, D\rangle)$, and there exist individual names $x, y, z_1, \ldots, z_n$ such that $P_i(x, z_i), Q_i(y, z_i) \in \mathcal{A}$ for $1 \leq i \leq n$ and $\{C(x), \sim C(x)\} \cap \mathcal{A} = \emptyset$
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{\sim C(x)\}$
Rule $\rightarrow_{\mathbf{chooseLK2}}$
Condition: $\mathcal{LK}$ contains $(\{\langle P_i, Q_i\rangle\}_{i=1}^n$ $\mathsf{linkkey}$ $\langle C, D\rangle)$, and there exist individual names $x, y, z_1, \ldots, z_n$ such that $P_i(x, z_i), Q_i(y, z_i) \in \mathcal{A}$ for $1 \leq i \leq n$ and $\{D(y), \sim D(y)\} \cap \mathcal{A} = \emptyset$
Action: $\mathcal{A}' := \mathcal{A} \cup \{D(y)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{\sim D(y)\}$
Rule $\rightarrow_{\mathbf{LK}}$
Condition: $\mathcal{LK}$ contains $(\{\langle P_i, Q_i\rangle\}_{i=1}^n$ $\mathsf{linkkey}$ $\langle C, D\rangle)$, and there exist individual names $x, y, z_1, \ldots, z_n$ such that $C(x)$, $D(y), P_i(x, z_i), Q_i(y, z_i) \in \mathcal{A}$ for $1 \leq i \leq n$, and $\mathcal{A} \cap \{x \approx y, y \approx x\} = \emptyset$
Action: $\mathcal{A}' := \mathcal{A} \cup \{x \approx y\}$ if $x < y$, and $\mathcal{A}' := \mathcal{A} \cup \{y \approx x\}$ otherwise.
Rule $\rightarrow_{\approx}$
Condition: $\mathcal{A}$ contains $y \approx x$ (with $y \neq x$), and $\Sigma \cap \mathcal{A} \neq \emptyset, \Sigma \setminus \mathcal{A} \neq \emptyset$ where Σ is one of the following sets of assertions: $\{C(x), C(y)\}$, $\{R(x,z), R(y,z)\}$, $\{R(z,x), R(z,y)\}$, for some concept C, or some individual z and some role R
Action: $\mathcal{A}' := \mathcal{A} \cup \Sigma$.

Figure 1: Completion rules for $\mathcal{ALC}$+$\mathcal{LK}$.

Algorithm 1: Checking ontology consistency

Input : An $\mathcal{ALC}$+$\mathcal{LK}$ ontology $\langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$
Output: Consistency of $\langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$

Initialize a set of ABoxes $\mathbf{A} = \{\mathcal{A}_0\}$;
while *there is a completion rule* r *in Figure 1 which is applicable to an individual* s *in some* $\mathcal{A} \in \mathbf{A}$ **do**
    Apply r to s;
if *there is a clash-free ABox* $\mathcal{A} \in \mathbf{A}$ **then**
    return YES;
else
    return NO;

4. Examples

This section provides a few examples of the use of the tableau-based algorithm described in Section 3. Example 2, derived from [11], illustrates a link inference. Example 3 shows the effect of the new $\rightarrow_{\mathbf{chooseLK}}$ rule. Examples 4 and 5 show how the validity and non validity of the link keys of Example 1 may be obtained. Finally, Example 6 shows the effect of blocking and will be further used to illustrate the proofs of properties in Section 5.

Each example displays the initial entailment to check (when applicable), the initial knowledge base corresponding to the reduction of the problem to a unsatisfiability test and the application of the rules of the algorithm. Each line corresponds to the application of a rule to an ABox. It identifies the rule applied, the resulting ABox and the clashes ($\neg$, $\not\approx$) or completion ($\Box$) of the ABox.

Example 2 (Chained link generation)**.**

Entailment: $(\langle P, R\rangle \ \mathsf{linkkey}\ \langle C, D\rangle), (\langle Q, S\rangle \ \mathsf{linkkey}\ \langle E, F\rangle),$
$C(a), P(a, c), E(c), Q(c, v), D(b), R(b, d), F(d), S(d, v) \models a \approx b$

Initial knowledge base:

$$\mathcal{A}_0 = \{C(a), P(a, c), E(c), Q(c, v), D(b), R(b, d), F(d), S(d, v), a \not\approx b\}$$
$$\mathcal{T} = \emptyset$$
$$\mathcal{LK} = \{(\langle P, R\rangle \ \mathsf{linkkey}\ \langle C, D\rangle), (\langle Q, S\rangle \ \mathsf{linkkey}\ \langle E, F\rangle)\}$$

Algorithm:

$$\begin{array}{lll}
\mathcal{A}_0 \rightarrow_{\mathbf{LK}} & \mathcal{A}_1 := \mathcal{A}_0 \cup \{c \approx d\} & \\
\mathcal{A}_1 \rightarrow_{\approx} & \mathcal{A}_2 := \mathcal{A}_1 \cup \{P(a,d), E(d), Q(d,v), R(b,c), F(c), S(c,v)\} & \\
\mathcal{A}_2 \rightarrow_{\mathbf{LK}} & \mathcal{A}_3 := \mathcal{A}_2 \cup \{a \approx b\} & \not\approx
\end{array}$$

The unique closed ABox contains a clash. Hence, the entailment is valid.

Example 3 (ChooseLK in action)**.**

Entailment: $(\langle P, Q\rangle \text{ linkkey } \langle C, D\rangle), (\langle P, R\rangle \text{ linkkey } \langle C, \neg D\rangle)$
$C(a), P(a,v), P(a,w), Q(b,v), R(b,w) \models a \approx b$

Initial knowledge base:

$$\begin{aligned}
\mathcal{A}_0 &= \{C(a), P(a,v), P(a,w), Q(b,v), R(b,w), a \not\approx b\} \\
\mathcal{T} &= \emptyset \\
\mathcal{LK} &= \{(\langle P, Q\rangle \text{ linkkey } \langle C, D\rangle), (\langle P, R\rangle \text{ linkkey } \langle C, \neg D\rangle)\}
\end{aligned}$$

Algorithm:

$$\begin{array}{lll}
\mathcal{A}_0 \rightarrow_{\mathbf{chooseLK2}} & \mathcal{A}_{01} := \mathcal{A}_0 \cup \{D(b)\} & \\
 & \mathcal{A}_{02} := \mathcal{A}_0 \cup \{\neg D(b)\} & \\
\mathcal{A}_{01} \rightarrow_{\mathbf{LK}} & \mathcal{A}_{03} := \mathcal{A}_{01} \cup \{a \approx b\} & \not\approx \\
\mathcal{A}_{02} \rightarrow_{\mathbf{LK}} & \mathcal{A}_{04} := \mathcal{A}_{02} \cup \{a \approx b\} & \not\approx
\end{array}$$

All closed ABoxes contain a clash. Hence, the entailment is valid.

Example 4 (Link key inference)**.** *This is the inference of the link key (8) given in Example 1 (the concepts and role names correspond to the initials of those of the example). The example is not expressed in $\mathcal{ALC}$ because it contains role equivalence statements. However, an equivalent $\mathcal{ALC}$+$\mathcal{LK}$ ontology may be obtained through rewriting the ontology. Here, we encode it by duplicating the ABox statements containing the equivalent properties.*

Entailment: $(\langle C, C\rangle, \langle T, T\rangle \text{ linkkey } \langle W, W\rangle), N \sqsubseteq W, T \equiv T', C \equiv A, N \sqsupseteq E \sqcap \exists L.P \models (\langle C, A\rangle, \langle T, T'\rangle \text{ linkkey } \langle N, E \sqcap \exists L.P\rangle)$

Initial knowledge base:

$$\begin{aligned}\mathcal{A}_0 &= \{N(d), C(d,v), T(d,w), A(d,v), T'(d,w), (E \sqcap \exists L.P)(b), A(b,v),\\ &\qquad T'(b,w), C(b,v), T(b,w), d \not\approx b\}\\ \mathcal{T} &= \{N \sqsubseteq W, N \sqsupseteq E \sqcap \exists L.P\}\\ \mathcal{LK} &= \{(\langle C, C\rangle, \langle T, T\rangle \text{ linkkey } \langle W, W\rangle)\}\end{aligned}$$

Algorithm:

$\mathcal{A}_0 \rightarrow_{\sqcap}$	$\mathcal{A}_1 := \mathcal{A}_0 \cup \{E(b), (\exists L.P)(b)\}$	
$\mathcal{A}_1 \rightarrow_{\exists}$	$\mathcal{A}_2 := \mathcal{A}_1 \cup \{L(b, v'), P(v')\}$	
$\mathcal{A}_2 \rightarrow_{\mathbf{choose}}$	$\mathcal{A}_{21} := \mathcal{A}_2 \cup \{(\neg E \sqcup \forall L.\neg P)(b)\}$	
	$\mathcal{A}_{22} := \mathcal{A}_2 \cup \{N(b)\}$	
$\mathcal{A}_{21} \rightarrow_{\sqcup}$	$\mathcal{A}_{211} := \mathcal{A}_{21} \cup \{(\neg E)(b)\}$	$\neg$
	$\mathcal{A}_{212} := \mathcal{A}_{21} \cup \{(\forall L.\neg P)(b)\}$	
$\mathcal{A}_{212} \rightarrow_{\forall}$	$\mathcal{A}_{213} := \mathcal{A}_{212} \cup \{(\neg P)(v')\}$	$\neg$
$\mathcal{A}_{22} \rightarrow_{\mathbf{choose}}$	$\mathcal{A}_{221} := \mathcal{A}_{22} \cup \{\neg N(d)\}$	$\neg$
	$\mathcal{A}_{222} := \mathcal{A}_{22} \cup \{W(d)\}$	
$\mathcal{A}_{222} \rightarrow_{\mathbf{choose}}$	$\mathcal{A}_{2221} := \mathcal{A}_{222} \cup \{\neg N(b)\}$	$\neg$
	$\mathcal{A}_{2222} := \mathcal{A}_{222} \cup \{W(b)\}$	
$\mathcal{A}_{2222} \rightarrow_{\mathbf{LK}}$	$\mathcal{A}_{2223} := \mathcal{A}_{2222} \cup \{d \approx b\}$	$\not\approx$

All closed ABoxes contain a clash. Hence, the entailment is valid.

Example 5 (Link key non inference)**.** *This is the non-inference of the link key (2) in Example 1. The same comments as in Example 4 apply. For readability, we adopted an ABox numbering scheme different from that of other examples in which only the path leading to a complete and clash-free ABox is numbered.*

Entailment: $(\langle C, C\rangle, \langle T, T\rangle \text{ linkkey } \langle W, W\rangle), N \sqsubseteq W, T \equiv T', C \equiv A, N \sqsupseteq E \sqcap \exists L.P \models (\langle C, A\rangle, \langle T, T'\rangle \text{ linkkey } \langle N, E\rangle)$

Initial knowledge base:

$$\begin{aligned}\mathcal{A}_0 &= \{N(d), C(d,v), T(d,w), A(d,v), T'(d,w), E(b), A(b,v), T'(b,w), C(b,v),\\ &\qquad T(b,w), d \not\approx b\}\\ \mathcal{T} &= \{N \sqsubseteq W, N \sqsupseteq E \sqcap \exists L.P\}\\ \mathcal{LK} &= \{(\langle C, C\rangle, \langle T, T\rangle \ \mathsf{linkkey}\ \langle W, W\rangle)\}\end{aligned}$$

Algorithm:

$$\begin{array}{lll}
\mathcal{A}_0 \rightarrow_{\mathbf{choose}} & \mathcal{A}_* := \mathcal{A}_0 \cup \{\neg N(d)\} & \neg\\
& \mathcal{A}_1 := \mathcal{A}_0 \cup \{W(d)\} & \\
\mathcal{A}_1 \rightarrow_{\mathbf{choose}} & \mathcal{A}_* := \mathcal{A}_1 \cup \{N(b)\} & \\
& \mathcal{A}_2 := \mathcal{A}_1 \cup \{(\neg E \sqcup \forall L.\neg P)(b)\} & \\
\mathcal{A}_2 \rightarrow_{\sqcup} & \mathcal{A}_* := \mathcal{A}_2 \cup \{\neg E(b)\} & \neg\\
& \mathcal{A}_3 := \mathcal{A}_2 \cup \{(\forall L.\neg P)(b)\} & \\
\mathcal{A}_3 \rightarrow_{\mathbf{choose}} & \mathcal{A}_4 := \mathcal{A}_3 \cup \{\neg N(b)\} & \\
& \mathcal{A}_* := \mathcal{A}_3 \cup \{W(b)\} & \\
\mathcal{A}_4 \rightarrow_{\mathbf{choose}} & \mathcal{A}_* := \mathcal{A}_4 \cup \{\neg N(v)\} & \\
& \mathcal{A}_5 := \mathcal{A}_4 \cup \{W(v)\} & \\
\mathcal{A}_5 \rightarrow_{\mathbf{choose}} & \mathcal{A}_6 := \mathcal{A}_5 \cup \{N(v)\} & \\
& \mathcal{A}_* := \mathcal{A}_5 \cup \{(\neg E \sqcup \forall L.\neg P)(v)\} & \\
\mathcal{A}_6 \rightarrow_{\mathbf{choose}} & \mathcal{A}_* := \mathcal{A}_6 \cup \{\neg N(w)\} & \\
& \mathcal{A}_7 := \mathcal{A}_6 \cup \{W(w)\} & \\
\mathcal{A}_7 \rightarrow_{\mathbf{choose}} & \mathcal{A}_8 := \mathcal{A}_7 \cup \{N(w)\} & \\
& \mathcal{A}_* := \mathcal{A}_7 \cup \{(\neg E \sqcup \forall L.\neg P)(w)\} & \\
\mathcal{A}_8 \rightarrow_{\mathbf{chooseLK2}} & \mathcal{A}_* := \mathcal{A}_8 \cup \{W(b)\} & \\
& \mathcal{A}_9 := \mathcal{A}_8 \cup \{\neg W(b)\} & \square
\end{array}$$

$\mathcal{A}_9$ is a complete and clash-free derived ABox. Hence, the entailment is invalid.

Example 6 (Knowledge base consistency)**.** *This example is particular, since it is only concerned with the consistency of a knowledge base. It is used in the remainder for illustrating the proofs.*

Initial knowledge base:

$$\begin{aligned}\mathcal{A}_0 &= \{(\exists W.(\exists R.\top \sqcap \exists P.\exists R.\top \sqcap \exists Q.\exists R.\top))(a), P(s,a), Q(a,s)\}\\ \mathcal{T} &= \emptyset\\ \mathcal{LK} &= \{(\langle R, R\rangle \text{ linkkey } \langle \top, \top\rangle)\}\end{aligned}$$

Algorithm:

$$\begin{aligned}
&\mathcal{A}_0 \to_\exists \quad \mathcal{A}_1 := \mathcal{A}_0 \cup \{W(a,b), (\exists R.\top \sqcap \exists P.\exists R.\top \sqcap \exists Q.\exists R.\top)(b)\}\\
&\mathcal{A}_1 \to_\sqcap \quad \mathcal{A}_2 := \mathcal{A}_1 \cup \{(\exists R.\top)(b), (\exists P.\exists R.\top \sqcap \exists Q.\exists R.\top)(b)\}\\
&\mathcal{A}_2 \to_\sqcap \quad \mathcal{A}_3 := \mathcal{A}_2 \cup \{(\exists P.\exists R.\top)(b), (\exists Q.\exists R.\top)(b)\}\\
&\mathcal{A}_3 \to_\exists \quad \mathcal{A}_4 := \mathcal{A}_3 \cup \{R(b,c), \top(c)\}\\
&\mathcal{A}_4 \to_\exists \quad \mathcal{A}_5 := \mathcal{A}_4 \cup \{P(b,d), (\exists R.\top)(d)\} \qquad b \text{ blocks } d\\
&\mathcal{A}_5 \to_\exists \quad \mathcal{A}_6 := \mathcal{A}_5 \cup \{Q(b,e), (\exists R.\top)(e)\} \qquad b \text{ blocks } e \ \square
\end{aligned}$$

The unique closed ABox is complete and clash-free. Hence, the initial knowledge base is consistent.

Figure 2 (p. 22) displays the derived ABox corresponding to $\mathcal{A}_6$.

5. Properties of the method

We establish the termination (§5.2), soundness (§5.3), completeness (§5.4) and complexity (§5.5) of the proposed method (Algorithm 1). But first, we have to introduce properties which are necessary for the proof of soundness and completeness (§5.1).

5.1. Some properties of derived ontologies

The following lemma shows a property of an ABox which is derived by completion rules.

Lemma 3. *Let $\mathcal{O}_k = \langle \mathcal{A}_k, \mathcal{T}, \mathcal{LK}\rangle$ be a derived $\mathcal{ALC}$+$\mathcal{LK}$ ontology with a set of individuals $I = I_{old} \uplus I_{new}$. It holds that*

1. *If $R(a,c), S(b,c) \in \mathcal{A}_k$ and $a \neq b$ then $a, b, c \in I_{old}$.*
2. *If $(a \approx b) \in \mathcal{A}_k$ then $a, b \in I_{old}$.*

3. If $(a \not\approx b) \in \mathcal{A}_k$ *then* $a, b \in I_{old}$.

Proof. Assume first that the $\rightarrow_{\approx}$ rule was not used in the derivation of $\mathcal{A}_k$. In this case, by the behaviour of the $\rightarrow_{\exists}$ rule, the only kinds of role assertions that may be included in $\mathcal{A}_k$ are: $R(u, v)$ with $u, v \in I_{old}$, $R(u, s)$ with $u \in I_{old}$ and $s \in I_{new}$, and $R(s, t)$ with $s, t \in I_{new}$, where R is a role name. Therefore, $(*)$ if $R(u, v) \in \mathcal{A}_k$ and $v \in I_{old}$ then $u \in I_{old}$. Also, since the $\rightarrow_{\exists}$ rule always adds new individual names, we have $(**)$ if $R(u, s), S(v, s) \in \mathcal{A}_k$ and $u \neq v$ then $s \in I_{old}$. Item 1 of the lemma follows from $(*)$ and $(**)$.

Now, assume that $(x \approx y) \in \mathcal{A}_k$. If $(x \approx y) \in \mathcal{A}_0$ then $x, y \in I_{old}$. Assume that $(x \approx y) \notin \mathcal{A}_0$. Then $x \approx y$ was added to $\mathcal{A}$ by applying the $\rightarrow_{\text{LK}}$ rule. This means that there are $C, D, P_1, Q_1, z_1, \ldots, P_n, Q_n, z_n$ such that $(\{\langle P_i, Q_i\rangle\}_{i=1}^{n}$ $\mathsf{linkkey}$ $\langle C, D\rangle), P_i(x, z_i), Q_i(y, z_i) \in \mathcal{A}_k$. The same argument as used above allows to conclude that $x, y \in I_{old}$.

Assume now that the $\rightarrow_{\approx}$ rule was used in the derivation of $\mathcal{A}_k$. Imagine that this derivation was

$$\mathcal{A}_0 \rightarrow \mathcal{A}_1 \rightarrow \ldots \rightarrow \mathcal{A}_k$$

and that $\mathcal{A}_n \rightarrow_{\approx} \mathcal{A}_{n+1}$ (for $0 \leq n \leq k-1$) was the first application of the $\rightarrow_{\approx}$ rule. As before, the only role assertions that $\mathcal{A}_n$ may include are: $R(u, v)$ with $u, v \in I_{old}$, $R(u, s)$ with $u \in I_{old}$ and $s \in I_{new}$, and $R(s, t)$ with $s, t \in I_{new}$ where R is a role name. Also, if $(x \approx y) \in \mathcal{A}_n$ then $x, y \in I_{old}$. By the behaviour of the $\rightarrow_{\approx}$ rule, the same holds in $\mathcal{A}_{n+1}$. Then, the same holds in $\mathcal{A}_k$ too, and the same argument used before proves Item 2 of the lemma.

Finally, since no completion rule adds an inequality assertion to a derived ontology, Item 3 holds too. □

Lemma 4 is a consequence of Lemma 3 simply stating that the $\rightarrow_{\text{LK}}$ rule can only be applied to individuals of $\mathcal{A}_0$.

Lemma 4. *Let* $\mathcal{O}_k = \langle \mathcal{A}_k, \mathcal{T}, \mathcal{LK}\rangle$ *be a derived* $\mathcal{ALC}$+$\mathcal{LK}$ *ontology with set of individuals* $I = I_{old} \uplus I_{new}$. *If there are distinct individuals* $x, y, z_1, \ldots, z_n \in I$ *with* $C(x), D(y), P_i(x, z_i), Q_i(y, z_i) \in \mathcal{A}_k$ *for* $1 \leq i \leq n$, *then* $x, y, z_1, \ldots, z_n \in I_{old}$.

Proof. This is a direct consequence of Lemma 3, more precisely of the proof of Item 2. □

This lemma may seem surprising. It owes to the fact that, contrary to constraints such as role-value-maps [25], link keys work backwards: they take advantage of role value equality to identify role bearers. Role-value-maps take advantage of role bearer equality to identify role values. Hence, as soon as there cannot be role equality among individuals generated by the tableau method, these individuals (in I_{new}) cannot be identified.

This does not render link keys useless: on the contrary, their role is to identify individuals among the ABox, not those generated by the method.

5.2. Termination

To prove termination of Algorithm 1, we need to prove that it returns YES or NO after performing a finite number of ontology transformations, *i.e.* the loop between Lines 2-3 in Algorithm 1 is finite.

Proposition 1 (Termination)**.** *Let $\mathcal{O}_0$ be an $\mathcal{ALC}$+$\mathcal{LK}$ ontology. Algorithm 1 terminates on $\mathcal{O}_0$.*

Proof. There are three factors that can affect the termination of Algorithm 1: the generation of new ABoxes by the non deterministic rules ($\rightarrow_{\sqcup}$, $\rightarrow_{\text{chooseLK1}}$, $\rightarrow_{\text{chooseLK2}}$ and $\rightarrow_{\text{choose}}$), the generation of new assertions by all rules and especially of new individuals by the $\rightarrow_{\exists}$ rule, and the possible non monotonically increasing behaviour of these rule application. We address the three issues.

First, Algorithm 1 adds an assertion to an ABox when a completion rule is applicable, and never removes anything from them. This behavior of Algorithm 1 is a consequence of the completion rules. Hence the ABoxes can only grow. Similarly, the number of generated ABoxes can only increase. Now let us prove that these are bounded.

Let $\mathbf{A}_k$ be a set of ABoxes built by Algorithm 1 from an $\mathcal{ALC}$+$\mathcal{LK}$ ontology $\mathcal{O}_0 = \langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$. It holds that each ABox $\mathcal{A} \in \mathbf{A}_k$ contains (i) the initial assertions coming from $\langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$, (ii) individuals $I = I_{old} \uplus I_{new}$, (iii) concept assertions $C(x)$ associated to each individual x, and (iv) role assertions $R(x, y)$ associated to two individuals x, y. Let $\ell = \|\mathcal{O}_0\|$, we have $|\mathsf{sub}(\mathcal{O}_0)| \leq O(\ell)$. By the blocking condition, we have $L(d) \neq L(d')$ for all new individuals $d, d' \in I_{new}$ with $d \neq d'$. Since $L(d) \subseteq \mathsf{sub}(\mathcal{O}_0)$, we obtain $|I| \leq O(2^\ell)$. Hence, $|\mathcal{A}| \leq O(2^\ell)$ for all $\mathcal{A} \in \mathbf{A}_k$.

Finally, the number of generated ABoxes is bounded. From each ABox $\mathcal{A}$, for each individual d and each concept $C \in L(d)$ or an axiom $C \sqsubseteq$

D there is at most one new ABox that is created and added by the $\rightarrow_{\sqcup}$, $\rightarrow_{\text{chooseLK1}}$, $\rightarrow_{\text{chooseLK2}}$ and $\rightarrow_{\text{choose}}$ rules. Moreover, when an application of a nondeterministic rule to an individual d in $\mathcal{A}$ due to a concept $C \in L(d)$ leads to add a new ABox $\mathcal{A}'$, C no longer triggers another application by the same nondeterministic rule to the individual d in $\mathcal{A}'$ copied from $\mathcal{A}$. Therefore, the number of generated ABoxes is bounded by $|I|^{\ell \times |I|} \leq O(2^{2^\ell})$.

Hence, Algorithm 1 can only generate a generalised ontology comprising a finite number of bounded-size ABoxes and it only adds assertions and never removes anything from the generalised ontology. Therefore, Algorithm 1 terminates. □

5.3. Soundness

Since Algorithm 1 is a decision procedure, it is sound if it is ensured that when it returns YES the input ontology is consistent. Thus, for soundness, we have to prove that, if Algorithm 1 is able to derive a successful generalised ontology $\mathcal{O}_k = \langle \mathbf{A}_k, \mathcal{T}, \mathcal{LK} \rangle$, then $\mathcal{O}_0 = \langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$ has a model. For this, we will use $\mathcal{A} \in \mathbf{A}_k$ to define an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, and show that $\mathcal{I}$ is a model of $\mathcal{O}_0$.

As usual, derived ABoxes containing a clash do not represent models. One may define $\mathcal{I}$ by interpreting all individuals in $\mathcal{A}$ as themselves, and then, for every concept name A, $s \in A^{\mathcal{I}}$ iff $A(s) \in \mathcal{A}$, and, for every role name R, $\langle s, t \rangle \in R^{\mathcal{I}}$ iff s is not blocked and $R(s,t) \in \mathcal{A}$, or s is blocked and $R(\mathsf{b}(s), t) \in \mathcal{A}$. This simple interpretation, called the canonical interpretation [4], is used in the case of $\mathcal{ALC}$. It also builds a model of $\mathcal{O}_0$ in case of complete clash-free non blocked derived ABoxes.

It turns out that this does not work for complete clash-free blocked derived ABoxes. Indeed, it may lead to a situation where $\mathcal{I}$ does not satisfy a link key even though $\mathcal{A}$ is complete. This is illustrated by Example 7.

Example 7 (Inadequacy of the canonical interpretation)**.** *Figure 2 depicts the single derived ABox $\mathcal{A}_6$ at the end of Example 6. d and e are labelled with $\exists R.\top$, but they do not have R-offspring since they are blocked by b which is also labelled by $\exists R.\top$. The canonical interpretation $\mathcal{I}$ associated with such a situation would simply be defined such that $\langle d^{\mathcal{I}}, c^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, $\langle e^{\mathcal{I}}, c^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ (and of course $\langle b^{\mathcal{I}}, c^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$). It is depicted in the left-hand side of Figure 3. The problem is that $\mathcal{I}$ does not satisfy $\mathcal{LK}$ because d, e and b are different, through they all share a value for role R.*

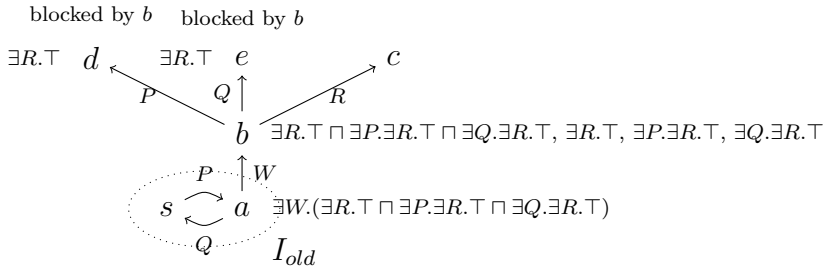

Figure 2: Derived ABox corresponding to $\mathcal{A}_6$ in Example 6. It is complete, clash-free and blocked.

In order to overcome the problem of Example 7, we will consider a different interpretation $\mathcal{I}$, that we call the *unravelled interpretation*. It is inspired from the unravelling technique used in [15] to devise a (possibly infinite) tree-like model from a derived ABox for the expressive description logic $\mathcal{SHIQ}$. We will show that the unravelled interpretation is a model of $\mathcal{O}_0$ independently from whether the derived ABox is blocked or not.

The unravelled interpretation associates paths to individuals in $\mathcal{A}_k$. These paths are sequences of names of individuals in the derived ABox. For instance, $\mathsf{p} = \langle a, b, c\rangle$ is such a path. Its last element (c) is called its tail and we write $\mathsf{tail}(\mathsf{p}) = c$; its first element ($a$) is called its root. A path containing only one element is called a root path.

Below we give the formal definition of the unravelled interpretation.

Definition 5 (Unravelled interpretation)**.** *Let $\mathcal{O}_0 = \langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK}\rangle$ be an $\mathcal{ALC}$+$\mathcal{LK}$ ontology; let $\mathcal{A}_k$ be a complete and clash-free ABox derived from $\mathcal{O}_0$ with set of individuals $I = I_{old} \uplus I_{new}$. The interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$ of $\mathcal{O}_0$ unravelled from $\mathcal{A}_k$ (or* unravelled *interpretation from $\mathcal{A}_k$) is defined as follows:*

1. *$\Delta^{\mathcal{I}}$ is the smallest set of* paths *built as follows:*
   (a) *$\Delta^{\mathcal{I}}$ contains a path $\mathsf{p}_a = \langle \mathsf{e}(a)\rangle$ for each $a \in I_{old}$. In this case, $a^{\mathcal{I}} = \mathsf{p}_a$.*
   (b) *For each $\mathsf{p} \in \Delta^{\mathcal{I}}$ such that $R(\mathsf{tail}(\mathsf{p}), a) \in \mathcal{A}_k$, $a \in I_{new}$ and a is not blocked, $\Delta^{\mathcal{I}}$ contains a path $\mathsf{p}' = \langle \mathsf{p}, a\rangle$.*
   (c) *For each $\mathsf{p} \in \Delta^{\mathcal{I}}$ such that $R(\mathsf{tail}(\mathsf{p}), a) \in \mathcal{A}_k$, $a \in I_{new}$ and a is blocked, $\Delta^{\mathcal{I}}$ contains a path $\mathsf{p}' = \langle \mathsf{p}, \mathsf{b}(a)\rangle$.*

2. *For each concept name* A, $A^{\mathcal{I}} = \{\mathsf{p} \in \Delta^{\mathcal{I}} \mid A(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k\}$

3. *For each role name* R,

$$R^{\mathcal{I}} = \{\langle \mathsf{p}_a, \mathsf{p}_b \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid R(a,b) \in \mathcal{A}_k\} \cup$$
$$\{\langle \mathsf{p}, \mathsf{p}' \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \mathsf{p}' = \langle \mathsf{p}, a \rangle, R(\mathsf{tail}(\mathsf{p}), a) \in \mathcal{A}_k, a \text{ is not blocked}\} \cup$$
$$\{\langle \mathsf{p}, \mathsf{p}' \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \mathsf{p}' = \langle \mathsf{p}, \mathsf{b}(a) \rangle, R(\mathsf{tail}(\mathsf{p}), a) \in \mathcal{A}_k, a \text{ is blocked}\}$$

From Definition 5, individuals of I_{old} are assigned a root path and equivalent individuals are interpreted as the same root path. Each individual of I_{new} generates paths in the unravelled interpretation obtained by concatenating the path associated to their ancestor in the application of the $\rightarrow_{\exists}$ rule to its name if it is not blocked and the name of its blocking node otherwise. Hence, each path in the domain $\Delta^{\mathcal{I}}$ of the unravelled interpretation is rooted at a path corresponding to an old individual of the ontology, *i.e.* an individual of $\mathcal{A}_0$.

It is possible to display the unravelled interpretation as an edge-labelled directed graph such that each element of $\Delta^{\mathcal{I}}$ is a node and there is an edge between two nodes if the pair of nodes belongs to the interpretation of a relation. Edges are labelled by the set of roles in which the corresponding pair appears. Figure 3 (right) displays such a graph.

The domain of the unravelled interpretation $\Delta^{\mathcal{I}}$ may be infinite because, as illustrated in Example 7, paths for blocked nodes may end with one of their ancestor in the derived ABox.

Example 7 (Unravelled interpretation)**.** *If* $\mathcal{I}$ *is the unravelled interpretation from* $\mathcal{A}_6$ *of Example 6,* $\mathcal{I}$ *interprets* d *and* e *as the same path individual* $\langle a, b, b \rangle$*. In this way, the link key is satisfied by* $\mathcal{I}$*. The unravelled interpretation* $\mathcal{I}$ *is depicted at the right-hand side of Figure 3 as a graph. It is a tree, albeit infinite, rooted in a root node (*$\langle a \rangle$*). Actually, the graph of unravelled interpretations corresponds to the image of a forest made of trees whose branches extend to the sky, while underground their roots can be connected and interleaved in an arbitrary way. Notice that* $\langle a, b, b \rangle, \langle a, b, b, b \rangle, \langle a, b, b, b, b \rangle \ldots$ *belong to* $\Delta^{\mathcal{I}}$*, i.e.* $\Delta^{\mathcal{I}}$ *is infinite. Also note that, there is no pair of elements of* $\Delta^{\mathcal{I}}$ *have the same value for* R *because, at each stage of the tree, the* R*-value is different:* $\langle a, b, c \rangle, \langle a, b, b, c \rangle, \langle a, b, b, b, c \rangle \ldots$ *Hence, the link key cannot apply.*

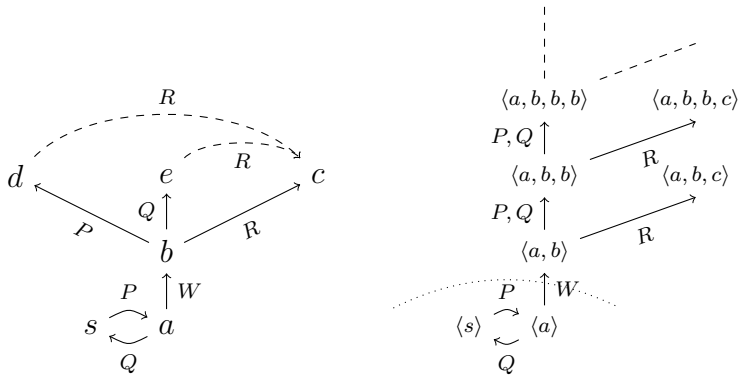

Figure 3: Left: Canonical interpretation corresponding to the derived ABox of Figure 2. It does not satisfy $\mathcal{LK}$ since d, e and b have a common R-value and are different. Right: Unravelled interpretation from the same derived ABox. It is infinite but satisfies $\mathcal{LK}$ since all R-values are different due to different prefixes.

The unravelled interpretation from $\mathcal{A}_k$ is not an interpretation of $\mathcal{O}_k$, as it does not interpret individuals in I_{new}, but it is an interpretation of $\mathcal{O}_0$. Proposition 2 shows that it is a model of $\mathcal{O}_0$.

The argument of the proof goes as follows: The interpretation unravelled from $\langle \mathcal{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ satisfies the $\mathcal{ALC}$ ontology $\langle \mathcal{A}_0, \mathcal{T} \rangle$. By Lemma 4, link keys only apply to statements involving individuals of $\mathcal{A}_0$. Since the derived ABox $\mathcal{A}_k$ is complete, it contains the result of the application of all link keys (individuals from $\mathcal{A}_0$ cannot be blocked). Unravelling does two things: (a) merging all $\mathcal{A}_0$ individuals related by $\approx$, and (b) expanding blocked individuals into (possibly infinite) trees. Hence, in both cases, all link keys are satisfied because no different individuals satisfy the link key conditions.

Proposition 2 (Soundness)**.** *If Algorithm 1 derives a successful generalised ontology from an $\mathcal{ALC}$+$\mathcal{LK}$ ontology $\mathcal{O}_0$, then $\mathcal{O}_0$ is consistent.*

Proof. Let $\mathcal{O}_0 = \langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$ be an $\mathcal{ALC}$+$\mathcal{LK}$ ontology and $\mathcal{O}_k = \langle \mathbf{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ be the successful generalised ontology derived from $\mathcal{O}_0$. This means that there exists a complete and clash-free ABox $\mathcal{A}_k \in \mathbf{A}_k$.

To prove the lemma, we show that the unravelled interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ from $\mathcal{A}_k$ according to Definition 5 is a model of $\mathcal{O}_0$. Recall that $I = I_{old} \uplus I_{new}$ where I_{old} is the set of the individuals in $\mathcal{O}_0$ and I is the set of all individuals in $\mathcal{O}_k$.

Apart from equalities $x \approx y \in \mathcal{A}_0$, an application of the $\rightarrow_{\text{LK}}$ rule can add a new equality while no rule can remove any equality. Therefore, each transitive closure a^+ for some individual a changes monotonically, *i.e.* $a^+(\mathcal{A}_{k-1}) \subseteq a^+(\mathcal{A}_k)$ for every individual a where $a^+(\mathcal{X})$ denotes the transitive closure a^+ defined over an ABox $\mathcal{X}$ (*cf.* Definition 3). In the sequel, we write a^+ for $a^+(\mathcal{A}_k)$. We rely on the following claims:

$$a \in I_{old} \Longrightarrow a^{\mathcal{I}} = \mathsf{e}(a)^{\mathcal{I}} = \mathsf{p}_{\mathsf{e}(a)} \tag{10}$$

$$a \in I_{old} \Longrightarrow a^+ \subseteq I_{old} \Longrightarrow \mathsf{e}(a) \in I_{old} \tag{11}$$

$$a \in I_{new} \Longrightarrow a^+ = \{a\} \tag{12}$$

$$\mathcal{A}_k \text{ is complete and clash-free} \Longrightarrow \begin{cases} L(x) = L(\mathsf{e}(x)), \text{and} \\ L(x,y) = L(\mathsf{e}(x), \mathsf{e}(y)) \end{cases} \tag{13}$$

The claim (10) is due to Definition 5 while the claims (11) and (12) are direct consequences of Lemma 3(2). The claim (13) is a consequence of the non-applicability of the $\rightarrow_{\approx}$ rule.

To prove that $\mathcal{I}$ is a model of $\langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$, we have to prove that $\mathcal{I}$ satisfies all assertions in $\mathcal{A}_0$, all GCIs in $\mathcal{T}$ and all link keys in $\mathcal{LK}$.

Assume $a \approx b \in \mathcal{A}_0$. This implies that $a, b \in I_{old}$ and $a, b \in a^+$. By the claim (10), $a^{\mathcal{I}} = \mathsf{p}_{\mathsf{e}(a)}$ and $b^{\mathcal{I}} = \mathsf{p}_{\mathsf{e}(b)}$. Since, $a, b \in a^+$ then $\mathsf{e}(a) = \mathsf{e}(b)$, and we have $a^{\mathcal{I}} = b^{\mathcal{I}} = \mathsf{p}_{\mathsf{e}(a)}$. Thus $a^{\mathcal{I}} = b^{\mathcal{I}}$.

Assume that $R(a, b) \in \mathcal{A}_0$. This implies that $a, b \in I_{old}$. We have $a^{\mathcal{I}} = e(a)^{\mathcal{I}} = \mathsf{p}_{e(a)}$ and $b^{\mathcal{I}} = e(b)^{\mathcal{I}} = \mathsf{p}_{e(b)}$ due to the claim (10), and $R(\mathsf{e}(a), \mathsf{e}(b)) \in \mathcal{A}_k$ due to the claim (13). By Definition 5, $\langle \mathsf{p}_{e(a)}, \mathsf{p}_{e(b)} \rangle \in R^{\mathcal{I}}$, and thus $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$.

Assume $a \not\approx b \in \mathcal{A}_0$ with $a < b$. We have $a, b \in I_{old}$ due to Lemma 3. By the claim (10), we have $a^{\mathcal{I}} = \mathsf{e}(a)^{\mathcal{I}} = \mathsf{p}_{\mathsf{e}(a)}$ and $b^{\mathcal{I}} = \mathsf{e}(b)^{\mathcal{I}} = \mathsf{p}_{\mathsf{e}(b)}$. By contradiction, assume that $\mathsf{p}_{\mathsf{e}(a)} = \mathsf{p}_{\mathsf{e}(b)}$. This implies that $\mathsf{e}(a) = \mathsf{e}(b)$ and thus $b \in a^+$, which is a $\not\approx$-clash. This contradicts clash-freeness of $\mathcal{A}_k$. Therefore, $\mathsf{p}_{\mathsf{e}(a)} \neq \mathsf{p}_{\mathsf{e}(b)}$ and $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

Assume $E(w) \in \mathcal{A}_0$. To show $w^{\mathcal{I}} \in E^{\mathcal{I}}$, we need to show a stronger claim:

$$\text{For all } \mathsf{p} \in \Delta^{\mathcal{I}}, \text{ if } C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k \text{ then } \mathsf{p} \in C^{\mathcal{I}} \tag{14}$$

Indeed, $E(w) \in \mathcal{A}_0$ and the claim (13) imply $E(\mathsf{e}(w)) \in \mathcal{A}_k$. In addition, $E(w) \in \mathcal{A}_0$ and the claim (11) imply that $w, \mathsf{e}(w) \in I_{old}$. By the definition of $\mathcal{I}$, there is some $\mathsf{p} \in \Delta^{\mathcal{I}}$ such that $\mathsf{p} = w^{\mathcal{I}} = \mathsf{e}(w)^{\mathcal{I}}$ and $\mathsf{tail}(\mathsf{p}) = \mathsf{e}(w)$. From the claim (14), we obtain $w^{\mathcal{I}} \in E^{\mathcal{I}}$. We now show the claim (14). Let us proceed by induction on the length of the concept C.

1. Assume that $C = A$ with a concept name A and $C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$. We have $C^{\mathcal{I}} = A^{\mathcal{I}} = \{\mathsf{p}' \in \Delta^{\mathcal{I}} \mid A(\mathsf{tail}(\mathsf{p}')) \in \mathcal{A}_k\}$ by the definition of $\mathcal{I}$. Hence, $A(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$ implies $\mathsf{p} \in A^{\mathcal{I}}$.

2. Assume that $C = C_1 \sqcap C_2$ and $C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$. Since $\mathcal{A}_k$ is complete then the $\rightarrow_{\sqcap}$ rule is not applicable, hence $C_1(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$, $C_2(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$. By induction hypothesis, $\mathsf{p} \in C_1^{\mathcal{I}}$ and $\mathsf{p} \in C_2^{\mathcal{I}}$. Then, $\mathsf{p} \in C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} = (C_1 \sqcap C_2)^{\mathcal{I}}$.

3. Assume that $C = C_1 \sqcup C_2$ and $C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$. Since $\mathcal{A}_k$ is complete then the $\rightarrow_{\sqcap}$ rule is not applicable, hence $C_1(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$ or $C_2(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$. By induction hypothesis, $\mathsf{p} \in C_1^{\mathcal{I}}$ or $\mathsf{p} \in C_2^{\mathcal{I}}$. Then, $\mathsf{p} \in C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} = (C_1 \sqcup C_2)^{\mathcal{I}}$.

4. Assume now that $C = \forall R.D$ and $C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$.

   Let $\mathsf{p}' \in \Delta^{\mathcal{I}}$ such that $(\mathsf{p}, \mathsf{p}') \in R^{\mathcal{I}}$. From the definition of $\mathcal{I}$, we consider the following two cases:

   - $R(\mathsf{tail}(\mathsf{p}), t) \in \mathcal{A}_k$ and t is not blocked with $t = \mathsf{tail}(\mathsf{p}')$. Since $\mathcal{A}_k$ is complete, the $\rightarrow_{\forall}$ rule is not applicable, thus $D(t) \in \mathcal{A}_k$. By induction hypothesis, $\mathsf{p}' \in D^{\mathcal{I}}$. Hence, $\mathsf{p} \in C^{\mathcal{I}}$.
   - $R(\mathsf{tail}(\mathsf{p}), t) \in \mathcal{A}_k$ and t is blocked with $\mathsf{b}(t) = \mathsf{tail}(\mathsf{p}')$. Since $\mathcal{A}_k$ is complete, the $\rightarrow_{\forall}$ rule is not applicable, thus $D(t) \in \mathcal{A}_k$. Since $\mathsf{tail}(\mathsf{p}')$ blocks t, we have $L(t) \subseteq L(\mathsf{tail}(\mathsf{p}'))$, and thus $D(\mathsf{tail}(\mathsf{p}')) \in \mathcal{A}_k$. By induction hypothesis, $\mathsf{p}' \in D^{\mathcal{I}}$. Hence, $\mathsf{p} \in C^{\mathcal{I}}$.

5. Assume that $C = \exists R.D$ and $C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$. Since $\mathsf{tail}(\mathsf{p})$ is never blocked and $\mathcal{A}_k$ is complete, the $\rightarrow_{\exists}$ rule is not applicable, and thus there exists $t \in I$ such that $R(\mathsf{tail}(\mathsf{p}), t) \in \mathcal{A}_k$, $D(t) \in \mathcal{A}_k$. By claim (13), we have $R(\mathsf{e}(\mathsf{tail}(\mathsf{p})), \mathsf{e}(t)), D(\mathsf{e}(t)) \in \mathcal{A}_k$. By the claims (10), (12) and Definition 5, $\mathsf{e}(\mathsf{tail}(\mathsf{p})) = \mathsf{tail}(\mathsf{p})$, and thus, $R(\mathsf{tail}(\mathsf{p}), \mathsf{e}(t)), D(\mathsf{e}(t)) \in \mathcal{A}_k$. We distinguish the following two cases:

   - Assume that $\mathsf{e}(t)$ is not blocked. By the definition of $\mathcal{I}$ and $R(\mathsf{tail}(\mathsf{p}), \mathsf{e}(t)) \in \mathcal{A}_k$, there is some $\mathsf{p}' \in \Delta^{\mathcal{I}}$ such that $\mathsf{tail}(\mathsf{p}') = \mathsf{e}(t)$ and $(\mathsf{p}, \mathsf{p}') \in R^{\mathcal{I}}$. Moreover, since $D(\mathsf{e}(t)) \in \mathcal{A}_k$ and $\mathsf{tail}(\mathsf{p}') = \mathsf{e}(t)$, by induction hypothesis, $\mathsf{p}' \in D^{\mathcal{I}}$. Hence, $\mathsf{p} \in C^{\mathcal{I}}$.
   - Assume that $\mathsf{e}(t)$ is blocked. According to the definition of $\mathcal{I}$ and $R(\mathsf{tail}(\mathsf{p}), \mathsf{e}(t)) \in \mathcal{A}_k$, there is some $\mathsf{p}' \in \Delta^{\mathcal{I}}$ such that $\mathsf{tail}(\mathsf{p}') =$

$\mathsf{b}(\mathsf{e}(t))$ and $(\mathsf{p}, \mathsf{p}') \in R^{\mathcal{I}}$. We have $D(\mathsf{e}(t)) \in \mathcal{A}_k$ implies $D(\mathsf{b}(\mathsf{e}(t))) \in \mathcal{A}_k$. By induction hypothesis, we have $\mathsf{p}' \in D^{\mathcal{I}}$. Hence, $\mathsf{p} \in C^{\mathcal{I}}$.

6. Assume that $C = \sim D$ and $C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$. We have to show that $\mathsf{p} \notin D^{\mathcal{I}}$. We proceed by induction on the length of D. If D is a concept name then $D(\mathsf{tail}(\mathsf{p})) \notin \mathcal{A}_k$ since $\mathcal{A}_k$ is clash-free. By the definition of $D^{\mathcal{I}}$, $\mathsf{p} \notin D^{\mathcal{I}}$. Assume that $D = C_1 \sqcap C_2$. This implies that $\sim D = \sim C_1 \sqcup \sim C_2$. Due to completeness, $\mathcal{A}_k$ must contain either $\sim C_1(\mathsf{tail}(\mathsf{p}))$ or $\sim C_2(\mathsf{tail}(\mathsf{p}))$. By induction hypothesis, we have $\mathsf{p} \notin C_1^{\mathcal{I}}$ or $\mathsf{p} \notin C_2^{\mathcal{I}}$. Hence, $\mathsf{p} \notin C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$, and thus $\mathsf{p} \notin D^{\mathcal{I}}$. Analogously, we can prove for the case of $D = C_1 \sqcup C_2$.

   Now assume that $D = \exists R.E$. This implies that $\sim D = \forall R.\sim E$. Let $\mathsf{p}' \in \Delta^{\mathcal{I}}$ with $(\mathsf{p}, \mathsf{p}') \in R^{\mathcal{I}}$. By Item 4, we have showed that $\mathsf{p} \in \sim D^{\mathcal{I}}$, and thus $\mathsf{p} \notin D^{\mathcal{I}}$. Analogously, we can prove for the case of $D = \forall R.E$.

We now show that $\mathcal{I}$ satisfies all GCIs in $\mathcal{T}$. Let $C \sqsubseteq D \in \mathcal{T}$ and $\mathsf{p} \in C^{\mathcal{I}}$. We have to show $\mathsf{p} \in D^{\mathcal{I}}$. Due to the completeness of $\mathcal{A}_k$, *i.e.* the $\rightarrow_{\text{choose}}$ rule is not applicable, we have either $\sim C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$ or $D(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$. If $\sim C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$, then $\mathsf{p} \notin C^{\mathcal{I}}$ due to Item 6, which contradicts $\mathsf{p} \in C^{\mathcal{I}}$. Hence, $D(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$ and thus, $\mathsf{p} \in D^{\mathcal{I}}$.

We now show that $\mathcal{I}$ satisfies link keys in $\mathcal{LK}$. Assume that $\lambda = (\{\langle P_i, Q_i \rangle\}_{i=1}^{n}$ linkkey $\langle C, D \rangle) \in \mathcal{LK}$. Let us prove that $\mathcal{I}$ satisfies λ. Let $\mathsf{p}, \mathsf{q}, \mathsf{p}_1, \ldots, \mathsf{p}_n \in \Delta^{\mathcal{I}}$ such that $\mathsf{p} \in C^{\mathcal{I}}$, $\mathsf{q} \in D^{\mathcal{I}}$, and $(\mathsf{p}, \mathsf{p}_i) \in P_i^{\mathcal{I}}$ and $(\mathsf{q}, \mathsf{p}_i) \in Q_i^{\mathcal{I}}$ for $1 \leq i \leq n$. We have to prove that $\mathsf{p} = \mathsf{q}$. Since $\mathcal{A}_k$ is complete, then neither the $\rightarrow_{\text{chooseLK1}}$ rule nor the $\rightarrow_{\text{chooseLK2}}$ rule may be applied, which means that $\mathcal{A}_k$ contains either $C(\mathsf{tail}(\mathsf{p}))$ or $\sim C(\mathsf{tail}(\mathsf{p}))$, and either $D(\mathsf{tail}(\mathsf{q}))$ or $\sim D(\mathsf{tail}(\mathsf{q}))$. If $\sim C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$ or $\sim D(\mathsf{tail}(\mathsf{q})) \in \mathcal{A}_k$ then $\mathsf{p} \notin C^{\mathcal{I}}$ or $\mathsf{q} \notin D^{\mathcal{I}}$ by the claim (14), which contradicts $\mathsf{p} \in C^{\mathcal{I}}$ or $\mathsf{q} \in D^{\mathcal{I}}$. Therefore, $C(\mathsf{tail}(\mathsf{p})) \in \mathcal{A}_k$, $D(\mathsf{tail}(\mathsf{q})) \in \mathcal{A}_k$. We consider the following cases:

Assume that $\mathsf{tail}(\mathsf{p}_i) \in I_{old}$ for all $1 \leq i \leq n$. We obtain $\mathsf{tail}(\mathsf{p}), \mathsf{tail}(\mathsf{q}) \in I_{old}$, $P_i(\mathsf{tail}(\mathsf{p}), \mathsf{tail}(\mathsf{p}_i))$, $Q_i(\mathsf{tail}(\mathsf{q}), \mathsf{tail}(\mathsf{p}_i)) \in \mathcal{A}_k$ from the definition of $\mathcal{I}$, $(\mathsf{p}, \mathsf{p}_i) \in P_i^{\mathcal{I}}$ and $(\mathsf{q}, \mathsf{p}_i) \in Q_i^{\mathcal{I}}$. Since $\mathcal{A}_k$ is complete, the satisfaction of the link key implies $\mathsf{tail}(\mathsf{p}) = \mathsf{tail}(\mathsf{q})$. Hence, $\mathsf{p}_{\mathsf{tail}(\mathsf{p})} = \mathsf{p}_{\mathsf{tail}(\mathsf{q})}$. From $\mathsf{p} = \mathsf{p}_{\mathsf{tail}(\mathsf{p})}$ and $\mathsf{q} = \mathsf{p}_{\mathsf{tail}(\mathsf{q})}$, we obtain $\mathsf{p} = \mathsf{q}$.

Assume that $\mathsf{tail}(\mathsf{p}_i) \in I_{new}$ for some $1 \leq i \leq n$. From the definition of $\mathcal{I}$, $(\mathsf{p}, \mathsf{p}_i) \in P_i^{\mathcal{I}}$ and $(\mathsf{q}, \mathsf{p}_i) \in Q_i^{\mathcal{I}}$, we obtain $\mathsf{p}_i = \langle \mathsf{p}, \mathsf{tail}(\mathsf{p}_i) \rangle = \langle \mathsf{q}, \mathsf{tail}(\mathsf{p}_i) \rangle$. Thus, $\mathsf{p} = \mathsf{q}$. □

5.4. Completeness

Since Algorithm 1 is a decision procedure, Algorithm 1 is complete if it is ensured that when the initial ontology is consistent, the algorithm returns YES. Thus, for completeness, we have to prove that if the initial ontology $\langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$ is consistent then Algorithm 1 is able to build a successful generalised ontology $\langle \mathbf{A}, \mathcal{T}, \mathcal{LK} \rangle$.

Proposition 3 (Completeness)**.** *If an $\mathcal{ALC}$+$\mathcal{LK}$ ontology $\mathcal{O}_0$ is consistent, then Algorithm 1 derives a successful generalised ontology from $\mathcal{O}_0$.*

Proof. Assume that $\mathcal{O}_0 = \langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$ and that $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ is a model of $\mathcal{O}_0$. We show that Algorithm 1 can build a generalised ontology $\langle \mathbf{A}_k, \mathcal{T}, \mathcal{LK} \rangle$ with a complete and clash-free ABox $\mathcal{A}_k \in \mathbf{A}_k$.

We maintain a function π which associates each individual s of an ABox $\mathcal{A}_k \in \mathbf{A}_k$ to an individual in $\Delta^{\mathcal{I}}$, *i.e.* $\pi(s) \in \Delta^{\mathcal{I}}$.

After applying a completion rule, we must update π in such a way that π satisfies the following conditions:

$$C(s) \in \mathcal{A}_k \text{ implies } \pi(s) \in C^{\mathcal{I}} \text{ or } \pi(\mathsf{b}(s)) \in C^{\mathcal{I}} \tag{15}$$

$$R(s,t) \in \mathcal{A}_k \text{ implies } \langle \pi(s), \pi(t) \rangle \in R^{\mathcal{I}} \text{ or } \langle \pi(s), \pi(\mathsf{b}(t)) \rangle \in R^{\mathcal{I}} \tag{16}$$

$$s \not\approx t \in \mathcal{A}_k \text{ implies } \pi(s) \neq \pi(t) \tag{17}$$

$$s \approx t \in \mathcal{A}_k \text{ implies } \pi(s) = \pi(t) \tag{18}$$

According to Proposition 1, Algorithm 1 always terminates at some $\mathbf{A}_n$. Thanks to the function π with Conditions (15-18) which helps to choose a "good" ABox $\mathcal{A}_k$ among several ABoxes $\mathbf{A}_k$ at each step $k \leq n$, we will show that there is an ABox $\mathcal{A}_n \in \mathbf{A}_n$ which is mapped to $\Delta^{\mathcal{I}}$ by π such that $\langle \mathcal{A}_n, \mathcal{T}, \mathcal{LK} \rangle$ is clash-free.

Assume that there exists such a function π. We show that $\mathcal{A}_n$ is complete and clash-free. When Algorithm 1 terminates, $\mathcal{A}_n$ must be complete. Assume that $A(s), \neg A(s) \in \mathcal{A}_n$. By Condition (15), we have $\pi(s) \in A^{\mathcal{I}}$ and $\pi(s) \in (\neg A)^{\mathcal{I}}$. It is not possible since $\mathcal{I}$ is a model. If $x \not\approx x \in \mathcal{A}_n$ then $\pi(x) \neq \pi(x)$ due to Condition (17), which is a contradiction. Assume that $x \not\approx y \in \mathcal{A}_n$ with $x \in y^+$. This implies that $\pi(x) \neq \pi(y)$ and there are $x \approx x_1, \cdots, x_n \approx y \in \mathcal{A}_n$. From Condition (18), we obtain $\pi(x) = \pi(y)$ which is a contradiction. Therefore, $\mathcal{A}_n$ is clash-free.

Now, let us define π. For each $s \in I_{old}$, there is some $s^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ since $\mathcal{I}$ is a model of $\mathcal{O}_0$. We define $\pi(s) = s^{\mathcal{I}}$, and $\pi(s) = \pi(s')$ if $s \approx s' \in \mathcal{A}_0$. Let

$R(s,t) \in \mathcal{A}_0$. We have $s, t \in I_{old}$, and thus $\pi(s), \pi(t)$ are defined. This implies that $\langle \pi(s), \pi(t) \rangle \in R^{\mathcal{I}}$ since $\mathcal{I}$ is a model of $\mathcal{O}_0$. For individual assertions, it holds that $s \not\approx t \in \mathcal{A}_0$ implies $\pi(s) \neq \pi(t)$ since $\mathcal{I}$ is a model of $\mathcal{O}_0$, and $s \approx t \in \mathcal{A}_0$ implies $\pi(s) = \pi(t)$ by the definition of π. Let $C(s) \in \mathcal{A}_0$. We have $s \in I_{old}$, and thus $\pi(s)$ is defined. This implies that $\pi(s) \in C^{\mathcal{I}}$ since $\mathcal{I}$ is a model of C. Therefore, Conditions (15-17) are verified for $\mathcal{A}_0$.

In the sequel, we consider each possible transformation performed by a completion rule on $\mathbf{A}_k$. Assume that there is an ABox $\mathcal{A}_k \in \mathbf{A}_k$ such that $\pi(s) \in \Delta^{\mathcal{I}}$ for each individual s occurring in $\mathcal{A}_k$, and π satisfies Conditions (15-18).

- The $\rightarrow_{\sqcap}$ rule is applied to $(C_1 \sqcap C_2)(s) \in \mathcal{A}_k$. Thus, $C_1(s), C_2(s) \in \mathcal{A}_{k+1}$. By Condition (15) and $\pi(s) \in \Delta^{\mathcal{I}}$, we have $\pi(s) \in (C_1 \sqcap C_2)^{\mathcal{I}}$. We obtain $\pi(s) \in C_1^{\mathcal{I}}$ and $\pi(s) \in C_2^{\mathcal{I}}$ since $\pi(s) \in (C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$. Therefore, Condition (15) is preserved.

- The $\rightarrow_{\exists}$ rule is applied to $\exists R.C(s) \in \mathcal{A}_k$ where s is not blocked. Thus, the rule adds an individual t and $C(t), R(s,t)$ to $\mathcal{A}_k$. Thus, $C(t), R(s,t) \in \mathcal{A}_{k+1}$. By Condition (15) and $\pi(s) \in \Delta^{\mathcal{I}}$, we have $\pi(s) \in (\exists R.C)^{\mathcal{I}}$. Since $\mathcal{I}$ is a model of $(\exists R.C)$, there is some $t' \in \Delta^{\mathcal{I}}$ such that $\langle \pi(s), t' \rangle \in R^{\mathcal{I}}$ and $t' \in C^{\mathcal{I}}$. If t is not blocked, we define $\pi(t) = t'$. Thus, Condition (15) and (16) are preserved. If t is blocked by $\mathsf{b}(t)$, we define $\pi(t) = \pi(\mathsf{b}(t))$. From Condition (16), we obtain $\langle \pi(s), \pi(t) \rangle \in R^{\mathcal{I}}$. Moreover, $L(t) \subseteq L(\mathsf{b}(t))$ implies $C(\mathsf{b}(t)) \in \mathcal{A}_{k+1}$. From Condition (15), it follows $\pi(\mathsf{b}(t)) = \pi(t) \in C^{\mathcal{I}}$. Hence, Conditions (15) and (16) are preserved.

- The $\rightarrow_{\forall}$ rule is applied to $\forall R.C(s) \in \mathcal{A}_k$. If s is blocked then $\forall R.C(\mathsf{b}(s)) \in \mathcal{A}_k$ and $\pi(s) = \pi(\mathsf{b}(s))$. Hence, it suffices to consider s that is not blocked. By Condition (15) and $\pi(s) \in \Delta^{\mathcal{I}}$, we have $\pi(s) \in (\forall R.C)^{\mathcal{I}}$. Assume that there is an individual t in $\mathcal{A}_k$ such that $R(s,t) \in \mathcal{A}_k$. In this case, the rule adds $C(t)$ to $\mathcal{A}_k$. Thus, $C(t) \in \mathcal{A}_{k+1}$. Assume that t is not blocked, by Condition (16) and $\pi(s) \in \Delta^{\mathcal{I}}$, we have $\langle \pi(s), \pi(t) \rangle \in R^{\mathcal{I}}$. Since $\mathcal{I}$ is a model of $\forall R.C$, we obtain $\pi(t) \in C^{\mathcal{I}}$. Thus, Condition (15) is preserved. Assume that t is blocked by $\mathsf{b}(t)$. We define $\pi(t) = \pi(\mathsf{b}(t))$. We have $L(t) \subseteq L(\mathsf{b}(t))$, and thus, $C(\mathsf{b}(t)) \in \mathcal{A}_{k+1}$. From Condition (15), it follows $\pi(\mathsf{b}(t)) = \pi(t) \in C^{\mathcal{I}}$. Hence, Condition (15) is preserved.

- The $\rightarrow_{\mathrm{LK}}$ rule is applied to individuals x, y, z_i with $C(x), D(y) \in \mathcal{A}_k$, $P_i(x, z_i), Q_i(y, z_i) \in \mathcal{A}_k$ for $1 \leq i \leq m$. According to Lemma 4, we have $x, y, z_i \in I_{old}$ for $1 \leq i \leq n$. Thus they are not blocked. By Condition (15) and (16), we have $\pi(x) \in C^{\mathcal{I}}$, $\pi(y) \in D^{\mathcal{I}}$, $\langle \pi(x), \pi(z_i) \rangle \in P_i^{\mathcal{I}}$ and $\langle \pi(y), \pi(z_i) \rangle \in Q_i^{\mathcal{I}}$ for $1 \leq i \leq n$.

  The $\rightarrow_{\mathrm{LK}}$ rule adds $x \approx y$ to $\mathcal{A}_k$. We obtain $(x \approx y) \in \mathcal{A}_{k+1}$. Since $\mathcal{I}$ is a model of $\mathcal{O}_0$, $\mathcal{I}$ must satisfy the link key. Hence, $x^{\mathcal{I}} = y^{\mathcal{I}}$, and thus $\pi(x) = \pi(y)$. Therefore, Condition (18) is preserved.

- The $\rightarrow_{\approx}$ rule is applied when $(x \approx y) \in \mathcal{A}_k$. It makes $L(x) = L(y)$ and $L(x, y) = L(\mathsf{e}(x), \mathsf{e}(y))$. This rule does not change individuals, Condition (17) and Condition (18) are preserved. If it adds $C(x)$ to $\mathcal{A}_k$ when $C(y) \in \mathcal{A}_k$ (or vice versa) then Condition (15) is preserved since $\pi(x) = \pi(y)$ and $\pi(y) \in C^{\mathcal{I}}$ imply $\pi(x) \in C^{\mathcal{I}}$. If it adds $R(x, z)$ (resp. $R(z, x)$) to $\mathcal{A}_k$ when $R(y, z) \in \mathcal{A}_k$ (resp. $R(z, y)$) then Condition (16) is preserved since $\pi(x) = \pi(y)$ and $\langle \pi(y), \pi(z) \rangle \in R^{\mathcal{I}}$ (resp. $\langle \pi(z), \pi(y) \rangle \in R^{\mathcal{I}}$) imply $\langle \pi(x), \pi(z) \rangle \in R^{\mathcal{I}}$ (resp. $\langle \pi(z), \pi(x) \rangle \in R^{\mathcal{I}}$).

- The $\rightarrow_{\sqcup}$ rule is applied to $(C_1 \sqcup C_2)(s) \in \mathcal{A}_k$. It transforms $\mathcal{A}_k$ to $\mathcal{A}_{k+1}$ with $C_1(s) \in \mathcal{A}_{k+1}$, and adds a new ABox $\mathcal{A}'_{k+1}$ with $C_2(s) \in \mathcal{A}'_{k+1}$. By Condition (15) and $\pi(s) \in \Delta^{\mathcal{I}}$, we have $\pi(s) \in (C_1 \sqcup C_2)^{\mathcal{I}}$, and thus, either $\pi(s) \in C_1^{\mathcal{I}}$ or $\pi(s) \in C_2^{\mathcal{I}}$. Assume that $\pi(s) \in C_1^{\mathcal{I}}$. In this case, we choose $\mathcal{A}_{k+1}$ including s with $\pi(s) \in C_1^{\mathcal{I}}$. This implies that Condition (15) is preserved in $\mathcal{A}_{k+1}$. Assume that $\pi(s) \in C_2^{\mathcal{I}}$. In this case, we choose $\mathcal{A}'_{k+1}$ including $C_2(s)$.This implies that Condition (15) is preserved in $\mathcal{A}'_{k+1}$.

- The $\rightarrow_{\mathrm{choose}}$ rule is applied to $(\sim C \sqcup D)(s) \in \mathcal{A}_k$ with $C \sqsubseteq D \in \mathcal{T}$. In the same way, we can choose an ABox among $\mathcal{A}_{k+1}$ and $\mathcal{A}'_{k+1}$ such that Condition (15) is preserved.

- the $\rightarrow_{\mathrm{chooseLK1}}$ rule is applied to individuals x, y, z_i with $y < x$, $C(x) \in \mathcal{A}_k$, $D(y) \in \mathcal{A}_k$, $P_i(x, z_i), Q_i(y, z_i) \in \mathcal{A}_k$ for $1 \leq i \leq m$. This rule transforms $\mathcal{A}_k$ to $\mathcal{A}_{k+1}$ with $C(x) \in \mathcal{A}_{k+1}$, and adds a new ABox $\mathcal{A}'_{k+1}$ with $\sim C(x) \in \mathcal{A}'_{k+1}$.Since $\mathcal{I}$ is a model, we have either $\pi(x) \in C^{\mathcal{I}}$ or $\pi(x) \in \sim C^{\mathcal{I}}$. Assume that $\pi(x) \in C^{\mathcal{I}}$. In this case, we choose $\mathcal{A}_{k+1}$ including x with $\pi(x) \in C^{\mathcal{I}}$. This implies that Condition (15) is preserved in $\mathcal{A}_{k+1}$. Assume that $\pi(x) \in (\sim C)^{\mathcal{I}}$. In this case, we choose

$\mathcal{A}'_{k+1}$ including $\sim C(x)$.This implies that Condition (15) is preserved in $\mathcal{A}'_{k+1}$.

- the $\rightarrow_{\text{chooseLK2}}$ rule. Analogously.

This completes the proof of preservation of Conditions 15-18 for each application of a completion rule. □

5.5. Complexity

Proposition 4 (Complexity)**.** *Let $\mathcal{O}_0 = \langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle$ be an $\mathcal{ALC}$+$\mathcal{LK}$ ontology. Algorithm 1 runs in doubly exponential time in the size of $\mathcal{O}_0$.*

Proof. According to the proof of Proposition 1, Algorithm 1 generates a collection $\mathbf{A}_k$ of ABoxes such that $|\mathbf{A}_k| \leq O(2^{2^\ell})$ and $|\mathcal{A}| \leq O(2^\ell)$ for all $\mathcal{A} \in \mathbf{A}_k$ where $\ell = \left|\left|\langle \mathcal{A}_0, \mathcal{T}, \mathcal{LK} \rangle\right|\right|$. Since Algorithm 1 never removes anything from an intermediate ABox, the complexity is bounded by $O(2^{2^\ell})$. Therefore, it runs in deterministic doubly exponential time in the worst case (2ExpTime). □

It is known that $\mathcal{ALC}$ with general concept axioms is ExpTime-complete [22]. This result provides a lower bound of the reasoning problem in $\mathcal{ALC}$+$\mathcal{LK}$. The doubly exponential complexity of Algorithm 1 is caused by the interaction between the nondeterministic behavior, *i.e.* a new ABox is duplicated by non deterministic rules such as the $\rightarrow_\sqcup$ rule, and exponential generation of new individuals by the $\rightarrow_\exists$ rule. Moreover, we know from Lemma 4 that the completion rules related to the application of link keys are applied only to old individuals I_{old} whose cardinality is polynomial in the size of the ontology. This means that link keys are not responsible of the doubly exponential complexity resulting from Algorithm 1. An open question is whether ExpTime is the tight lower bound of consistency checking in $\mathcal{ALC}$+$\mathcal{LK}$.

The following theorem is a consequence of all propositions established until now.

Theorem 1. *$\mathcal{ALC}$+$\mathcal{LK}$ consistency can be decided in doubly exponential time in the size of ontologies.*

6. Conclusions and Future Work

Link keys are a generalisation of keys in RDF datasets to different RDF datasets described using different vocabularies. As such, they can be used for data interlinking. In previous work, we showed that link keys can be extracted from RDF data and used effectively for interlinking datasets. In this paper, we have addressed link key reasoning. Reasoning with link keys can be used to combine automatically extracted link keys with other different kinds of knowledge to infer new axioms. In particular, it can infer new link keys that may be better adapted to a specific data interlinking task.

We have proposed a tableau-based algorithm for reasoning in the $\mathcal{ALC}$+$\mathcal{LK}$ logic, an extension of $\mathcal{ALC}$ with link keys. We have provided proofs of its soundness, completeness and termination.

Reasoning in $\mathcal{ALC}$+$\mathcal{LK}$ is more challenging than $\mathcal{ALC}$. It requires the introduction of new completion rules: the $\rightarrow_{\text{chooseLK1}}$, $\rightarrow_{\text{chooseLK2}}$ and $\rightarrow_{\text{LK}}$ rules to deal with link keys, and the $\rightarrow_{\approx}$ rule to handle equality. In addition, the canonical interpretation used for proving the soundness of the standard $\mathcal{ALC}$ algorithm cannot be directly used for $\mathcal{ALC}$+$\mathcal{LK}$. We have introduced the unravelled interpretation to prove it.

In the future, we plan to study extensions of the algorithm for reasoning with link keys in more expressive description logics allowing for inverse roles and number restrictions. We also plan to extend the expressiveness of link keys by considering roles (beyond role names) and link key covering eq-conditions [**?**].

Also, we will study if ExpTime is the tight lower complexity bound for consistency checking in $\mathcal{ALC}$+$\mathcal{LK}$, and, if so, design a worst-case optimal tableau algorithm. For this, one possibility is to use an exponential structure for representing ontology models inspired from compressed-tableau [23, 17].

We plan to implement the algorithm for $\mathcal{ALC}$+$\mathcal{LK}$ with basic techniques of optimisation known in the literature such as absorption and backtracking [14]. These techniques will allow to reduce the number of useless ABox axioms generated by the current algorithm, *e.g.* useless equality statements generated by the $\rightarrow_{\approx}$ rule.

Last but not least, we will evaluate the impact of link key inference on data interlinking. We plan to combine link key inference with link inference based on rules to ensure scalability, and we will use RDF datasets described by semantically rich ontologies such as Insee COG and GeoNames in the geographic domain, or British Library and BNF (National Library of France)

in the bibliographic domain.

Acknowledgements

This work has been partially supported by the ANR project Elker (ANR-17-CE23-0007-01).

[1] Al-Bakri, M., Atencia, M., David, J., Lalande, S., and Rousset, M. (2016). Uncertainty-sensitive reasoning for inferring sameAs facts in linked data. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 698–706. IOS Press.

[2] Al-Bakri, M., Atencia, M., Lalande, S., and Rousset, M.-C. (2015). Inferring same-as facts from linked data: an iterative import-by-query approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 9–15. AAAI Press.

[3] Atencia, M., David, J., and Euzenat, J. (2014). Data interlinking through robust linkkey extraction. In Schaub, T., Friedrich, G., and O'Sullivan, B., editors, *Proc. 21st european conference on artificial intelligence (ECAI), Praha (CZ)*, pages 15–20, Amsterdam (NL). IOS press.

[4] Baader, F., Buchheit, M., and Hollunder, B. (1996). Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1-2):195–213.

[5] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2007). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

[6] Borgida, A. and Weddell, G. (1997). Adding uniqueness constraints to description logics (preliminary report). In *Deductive and Object-Oriented Databases, 5th International Conference, DOOD'97, Montreux, Switzerland, December 8-12, 1997, Proceedings*, volume 1341 of *Lecture Notes in Computer Science*, pages 85–102. Springer.

[7] Calvanese, D., De Giacomo, G., and Lenzerini, M. (2000). Keys for free in description logics. In *Proceedings of the 2000 International Workshop on Description Logics (DL2000), Aachen, Germany, August 17-19, 2000*, CEUR Workshop Proceedings, pages 79–88. CEUR-WS.org.

[8] Calvanese, D., De Giacomo, G., and Lenzerini, M. (2001). Identification constraints and functional dependencies in description logics. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 155–160. Morgan Kaufmann.

[9] Ferrara, A., Nikolov, A., and Scharffe, F. (2011). Data linking for the semantic web. *International Journal of Semantic Web and Information Systems*, 7(3):46–76.

[10] Fuhr, N. (2000). Probabilistic datalog: implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110.

[11] Gmati, M., Atencia, M., and Euzenat, J. (2016). Tableau extensions for reasoning with link keys. In *Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 18, 2016.*, CEUR Workshop Proceedings, pages 37–48. CEUR-WS.org.

[12] Heath, T. and Bizer, C. (2011). *Linked Data : Evolving the Web into a Global Data Space*. Morgan and Claypool.

[13] Hogan, A., Zimmermann, A., Umbrich, J., Polleres, A., and Decker, S. (2012). Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *Web Semantics: Science, Services and Agents on the World Wide Web*, 10(0):76–110.

[14] Horrocks, I. (2007). Implementation and optimization techniques. In *The Description Logic Handbook: Theory, Implementation and Applications (2nd edition)*, pages 329–373. Cambridge University Press.

[15] Horrocks, I., Sattler, U., and Tobies, S. (1999). Practical reasoning for expressive description logics. In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 1999)*. Springer.

[16] Isele, R., Jentzsch, A., and Bizer, C. (2011). Efficient multidimensional blocking for link discovery without losing recall. In *Proceedings of the 14th International Workshop on the Web and Databases 2011, WebDB 2011, Athens, Greece, June 12, 2011.*

[17] Le Duc, C., Dong, T., Lamolle, M., and Bossard, A. (2016). Raisonnement fondé sur un tableau compressé pour les logiques de description. In *https://www.supagro.fr/jfpc_jiaf_2016/Articles.IAF.2016/LeDuc_IAF_2016.pdf.* Acte de collogue, JIAF 2016.

[18] Lutz, C., Areces, C., Horrocks, I., and Sattler, U. (2005). Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research*, 23:667–726.

[19] Motik, B., Sattler, U., and Studer, R. (2005). Query answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60.

[20] Nentwig, M., Hartung, M., Ngonga Ngomo, A.-C., and Rahm, E. (2017). A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436.

[21] Ngomo, A. N. and Auer, S. (2011). LIMES - A time-efficient approach for large-scale link discovery on the web of data. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2312–2317. IJCAI/AAAI.

[22] Pratt, V. R. (1978). A practical decision method for propositional dynamic logic. In *Proceedings of the tenth annual ACM symposium on Theory of Computing*, pages 326–337.

[23] Pratt-Hartmann, I. (2005). Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395.

[24] Saïs, F., Pernelle, N., and Rousset, M. (2007). L2R: A logical method for reference reconciliation. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 329–334. AAAI Press.

[25] Schmidt-Schauß, M. (1989). Subsumption in KL-ONE is undecidable. In *Proc. 1st conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 421–431. Morgan Kaufmann.

[26] Schmidt-Schauß, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26.

[27] Suchanek, F. M., Abiteboul, S., and Senellart, P. (2011). PARIS: probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3):157–168.

[28] Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G. (2009). Discovering and maintaining links on the web of data. In *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, volume 5823 of *Lecture Notes in Computer Science*, pages 650–665. Springer.