Concept Abduction in Description logics with cardinality restrictions

Viet-Hoang VU, Nhan LE-THANH
KEWI - Laboratory I3S - CNRS
Nice-Sophia Antipolis University, France

Abstract—Recently the usefulness of Concept Abduction, a novel non-monotonic inference service for Description Logics (DLs), has been argued in the context of ontology-based applications such as semantic matchmaking and resource retrieval. Based on tableau calculus, a method has been proposed to realize this reasoning task in $\mathcal{ALN}$, a description logic that supports simple cardinality restrictions as well as other basic constructors. However, in many ontology-based systems, the representation of ontology would require expressive formalisms for capturing domain-specific constraints, this language is not sufficient. In order to increase the applicability of the abductive reasoning method in such contexts, we would like to present in the scope of this paper an extension of the tableaux-based algorithm for dealing with concepts represented in$\mathcal{ALCQ}$, the description logic that extends $\mathcal{ALN}$ with full concept negation and quantified number restrictions.

Keywords—abductive reasoning, description logics, semantic matchmaking, non-monotonic inference, tableaux-based method.

I. Introduction

Description Logics (DLs) [1] are a family of knowledge representation formalisms that are used widely for building ontology, especially in the context of Semantic Web. Descendants from earlier semantic networks and frame-based languages, but equipped with a logic-based semantics, these languages allow to represent the knowledge of an application domain in a well-structured and formal way. Besides, to better support the ontology engineering, description logics systems provides as well tools to reason about its knowledge bases efficiently. There are two kinds: standard reasoning services, mainly monotonic, like concept satisfiability or concept subsumption; and non-standard reasoning services, usually non-monotonic, such as least-common subsumer or concept approximation.

Abduction is a popular technique of common sense reasoning[12] to find back *explanation*s from *observations*. Despite it was well studied in the context of classical logics, just recently this inference method have been begun to gain the attention of the description logics community because of its usefulness to deal with inferences that new ontology-based applications would require [7], [6], [11]. In one of such contexts [11], *concept abduction* has been proposed as a novel inference service for $\mathcal{ALN}$, the description logic that, in addition to basic constructors like conjunction of concepts ($\sqcap$) and value restriction on role ($\forall$), supports unqualified number restrictions. Derived from propositional abduction, the non-monotonic reasoning method allows to, given two non-disjoint concepts C, D and a TBox $\mathcal{T}$, find a hypothesis H such that $C \sqcap H \equiv_{\mathcal{T}} \bot$ and $C \sqcap H \sqsubseteq_{\mathcal{T}} D$.

There are many applications that this abductive inference service would be useful, ranging from resource retrieval [5] to semantic matchmaking [10] as well as many others [7]. In order to realize it, an uniform tableaux-based method has been proposed in [4] for the description logic $\mathcal{ALN}$. However, this language is rather inexpressive, in many domains the representation of ontology would require more expressive formalisms. Taking a tourism ontology, for example, according to the classification of hotels in France [9], a three stars hotel is an accommodation that must have :

1) at least 6 chambers of all categories,
2) 1 salon and elevators if the building has more than 3 floors,
3) 1 table or desk, 1 telephone and 1 color television in every room,
4) private bathrooms with shower or bathtub, .etc.

Clearly, for capturing fully semantics of these constraints, at least the quantified cardinality and the disjointness should be used as shown below :

$$
\begin{array}{rcl}
\mathsf{Accommodation} & \sqsubseteq & \geqslant 1 hasRoom.\mathsf{Room} \\
\mathsf{Hotel} & \sqsubseteq & \mathsf{Accommodation} \\
\mathsf{3StarsHotel} & \sqsubseteq & \mathsf{Hotel} \sqcap \geqslant 6 hasRoom.\mathsf{3StarsRoom} \\
\mathsf{3StarsRoom} & \sqsubseteq & \mathsf{Room} \\
 & & \sqcap \geqslant 1 hasEquipment.\ (\mathsf{Table} \sqcup \mathsf{Desk}) \\
 & & \sqcap \geqslant 1 hasEquipment.\mathsf{Telephone} \\
 & & \sqcap \geqslant 1 hasEquipment.\mathsf{ColorTelevison} \\
 & & \sqcap \geqslant 1 hasBathRoom.\mathsf{PrivateBathRoom} \\
\mathsf{PrivateBathRoom} & \sqsubseteq & \mathsf{BathRoom} \\
 & & \sqcap \geqslant 1 hasEquipment.\ (\mathsf{Shower} \sqcup \mathsf{BathTub}) \\
\mathsf{Shower} & \sqsubseteq & \neg \mathsf{BathTub} \\
\mathsf{3StarsHotel} & & \\
\sqcap \geqslant 3 hasFloor.\top & \sqsubseteq & \geqslant 1 hasEvelator.\mathsf{Evelator}
\end{array}
$$

As a consequence, in order to increase the applicability of abductive reasoning methods in such situations, there is a need to extend the tableaux-based technique for more expressive description logics. In the scope of this paper, we would like to present an extension of the approach for solving the concept abduction problem in $\mathcal{ALCQ}$, the language that extends $\mathcal{ALN}$ with full concept negation and quantified number restrictions.

The rest of this paper is organized as follows. Firstly, in the section 2, we recall some preliminaries on the description logic $\mathcal{ALCQ}$. Next, we give an overview of abductive reasoning tasks in description logics in the section 3. Then, the problem of concept abduction in $\mathcal{ALCQ}$ as well as the tableaux-based

algorithm for computing its solutions are presented in the section 4. An example to illustrate how the algorithm works is presented in the section 5. And finally, the section 6 is reserved for some conclusions.

II. Description logic $\mathcal{ALCQ}$.

Like every other description logics, in $\mathcal{ALCQ}$, knowledge of domains is represented by means of *concept descriptions* which denotes a group of individuals sharing some common characteristics. Starting with a set NC of *concept names* and a set NR of *role names*, these formulas are inductively constructed by using a set of *concept constructors*.

For $\mathcal{ALCQ}$, concepts are formed as follows :

$$C, D ::= \top \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall r.C \mid \exists r.C \mid \geqslant nr.C \mid \leqslant nr.C$$

As usual, the semantics of $\mathcal{ALCQ}$ concept descriptions is defined in a model-theoretic way. Formally, an *interpretation* is defined as a pair $\mathcal{I} = \left(\triangle^{\mathcal{I}}, .^{\mathcal{I}}\right)$, where $\triangle^{\mathcal{I}}$, a non-empty set, is the *domain* of the interpretation and $.^{\mathcal{I}}$ is an *interpretation function* which assigns to every concept name $A \in NC$ a set $A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$ and to every role name $R \in NR$ a binary relation $R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$. The function is then extended to more complex concept descriptions according to the semantics of specific constructors

$$\begin{array}{rcl}
\top^{\mathcal{I}} & = & \triangle^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(A)^{\mathcal{I}} & = & A^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \triangle^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall r.C)^{\mathcal{I}} & = & \left\{a \in \triangle^{\mathcal{I}} \mid \forall b.\, (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\right\} \\
(\exists r.C)^{\mathcal{I}} & = & \left\{a \in \triangle^{\mathcal{I}} \mid \exists b.\, (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\right\} \\
(\geqslant nr.C)^{\mathcal{I}} & = & \left\{a \in \triangle^{\mathcal{I}} \mid \left|b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\right| \geq n\right\} \\
(\leqslant nr.C)^{\mathcal{I}} & = & \left\{a \in \triangle^{\mathcal{I}} \mid \left|b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\right| \leq n\right\}
\end{array}$$

Knowledge is represented in DLs by means of *terminological axioms* and *individual assertions*. Terminological axioms express relations between concept descriptions, they can be either *concept inclusions* ($C \sqsubseteq D$) or *concept equivalence* ($C \equiv D$). An axiom is *simple*, if on the left-side of it, there is only one concept name. The axiom is *general*, if both of its sides are arbitrary complex descriptions. A simple concept equivalence is also called a *concept definition*. Then, a finite set of concept definitions $\mathcal{T}$ is a *terminology* , or a *TBox*, if no concept name is defined more than once. In the TBox $\mathcal{T}$, we say a concept C *directly uses* a concept D if D appears in the right side of C's definition. Then, $\mathcal{T}$ is *cyclic* if there is a concept name in $\mathcal{T}$ that transitively uses itself. Otherwise, it's *acyclic*.

Named individuals are introduced in the knowledge base by assertions. There are two kinds, concept and role assertions denoted respectively by $C(a)$ and $R(b, c)$, where C is some concept, R is a role and a, b, c are named distinct individuals. A finite set of assertions $\mathcal{A}$ is also called an *ABox*. The interpretation $\mathcal{I}$ is extended straightforwardly to provide a semantics for the ABox's assertions. Accordingly, each individual name a is mapped to an element $a^{\mathcal{I}} \in \triangle^{\mathcal{I}}$ such that, if a, b are distinct names, then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

Then, the interpretation $\mathcal{I}$ *satisfies* a terminological axiom $C \sqsubseteq D$ ($C \equiv D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($C^{\mathcal{I}} = D^{\mathcal{I}}$). $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ iff it satisfies all of its axioms. $\mathcal{I}$ satisfies an assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and it satisfies $R(b, c)$ if $\left(a^{\mathcal{I}}, b^{\mathcal{I}}\right) \in R^{\mathcal{I}}$. The interpretation is a model of the ABox $\mathcal{A}$ if it satisfies all its assertions. Finally, $\mathcal{I}$ satisfies an assertion α or an ABox $\mathcal{A}$ with respect to a TBox $\mathcal{T}$ if in addition to being a model of α or of $\mathcal{A}$, it's also a model of $\mathcal{T}$.

In addition to the means for representing and storing knowledge about concepts, roles and individuals in the application domain, DLs systems provide a set of standard inference services to reason about them :

1) *Concept satisfiability*: Given a TBox $\mathcal{T}$, a concept C is *satisfiable w.r.t* $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$.
2) *Concept subsumption*: Given a TBox $\mathcal{T}$, a concept C is *subsumed* by a concept D *w.r.t* $\mathcal{T}$, denoted as $C \sqsubseteq_{\mathcal{T}} D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$.
3) *Entailment*: Given an ABox $\mathcal{A}$, an assertion α is *entailed* by $\mathcal{A}$, denoted as $\mathcal{A} \models \alpha$, if every model of $\mathcal{A}$ satisfies α.
4) *Consistency* : Given an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. $\mathcal{A}$ is *consistent w.r.t* $\mathcal{T}$, if there is an interpretation $\mathcal{I}$ that is a model of both $\mathcal{A}$ and $\mathcal{T}$. $\mathcal{A}$ is called simply consistent if it is consistent w.r.t the empty TBox.

Importantly, since full concept negation ($\neg$) is allowed, the problem of concept satisfiability and concept subsumption can be reduced to each other, because :

- C is *satisfiable* iff $C \sqsubseteq A \sqcap \neg A$ is not hold, where A is any concept name.
- And vice versa, $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is *unsatisfiable* w.r.t $\mathcal{T}$.

Most popular reasoning techniques in DLs are tableaux-based methods [2]. Accordingly, in order to verify the satisfiability of a concept C w.r.t a TBox $\mathcal{T}$, the algorithms try to construct a consistent model $\mathcal{I}$ for C and $C_{\mathcal{T}}$, where $C_{\mathcal{T}}$ is the *internalized concept* of $\mathcal{T}$ which is defined as follows

$$C_{\mathcal{T}} := \bigsqcap_{(C_i \sqsubseteq D_i) \in \mathcal{T}} (\neg C_i \sqcup D_i)$$

To do that, *tableaux* or *completion graphs* are used for representing candidates for $\mathcal{I}$. These graph structures contains nodes and edges that represent respectively individuals and role instances in the domain of models. Each node and edge is labeled with a set $L()$ of concepts and roles of which they are instances. If the tableau does not contain any *clash* (obvious contradiction) in all its labels, then it's *open*, otherwise it's *closed*. As any clash-free tableau τ corresponds to a consistent canonical interpretation $\mathcal{I}_\tau$, if the algorithms can find an open tableau for C and $\mathcal{T}$ then C is satisfiable w.r.t $\mathcal{T}$, on the contrary, C is unsatisfiable.

III. Abductive reasoning in description logics

Abduction is a well-known form of commonsense reasoning that gives the ability to infer from *observations* to *explanatory*

hypotheses. This technique was introduced firstly by Charles Sander Peirce in [12], since then it has been intensively studied in classical logics. Recently, abduction has begun to gain the attention of the DLs community because of its usefulness to deal with inferences that new applications using DLs would require [7], [6], [11].

In [7], abductive reasoning in DLs are divided into four tasks :

1) **Concept abduction** : Given two concepts C and D represented in some description logic $\mathcal{L}$, Γ is a knowledge-base in $\mathcal{L}$ and supposing that C, D are satisfiable w.r.t Γ. A solution to the *(conditionalized) concept abduction problem* for $\langle \Gamma, C, D \rangle$ is any concept H in $\mathcal{L}'$ such that :
$$C \sqcap H \not\equiv_\Gamma \bot \text{ and } C \sqcap H \sqsubseteq_\Gamma D$$
The set of all such solutions H is denoted by $\mathcal{S}_{CCA} \langle \Gamma, C, D \rangle$.
2) **ABox abduction** : Let Γ be a knowledge-base in some description logic $\mathcal{L}$ and α be an ABox assertion in $\mathcal{L}$ such that $\Gamma \cup \alpha$ in consistent. A solution to the *ABox abduction problem* for given $\langle \Gamma, \alpha \rangle$ is any finite set $\mathcal{S}_A = \{\beta\}$ of ABox assertions in $\mathcal{L}'$ such that :
$$\Gamma \cup \mathcal{S}_A \models \alpha$$
The set of all such solutions is denoted by $\mathcal{S}_A \langle \Gamma, \alpha \rangle$.
3) **TBox abduction** : Let Γ be a knowledge-base in some description logic $\mathcal{L}$, C and D be two concepts that are satisfiable w.r.t Γ and supposing that $\Gamma \cup \{C \sqsubseteq D\}$ is consistent. A solution to the *TBox abduction problem* for give $\langle \Gamma, C, D \rangle$ is any set $\mathcal{S}_T = \{E_i \sqsubseteq F_j\}$ of terminological axioms in $\mathcal{L}'$ such that :
$$C \sqsubseteq_{\Gamma \cup \mathcal{S}_T} D$$
The set of all such solutions is denoted by $\mathcal{S}_T \langle \Gamma, C, D \rangle$.
4) **Knowledge-base abduction** : Let $\mathcal{L}$ be some description logic, Γ be a knowledge-base in $\mathcal{L}$ and γ be a TBox axiom or ABox assertion such that $\Gamma \cup \{\gamma\}$ is consistent. A solution to the *knowledge-base abduction problem* for $\langle \Gamma, \gamma \rangle$ is any finite set $\mathcal{S} = \{\delta\}$ of TBox axioms or ABox assertions such that :
$$\Gamma \cup \mathcal{S} \models \gamma$$
The set of all such solutions is denoted by $\mathcal{S}_K \langle \Gamma, \gamma \rangle$.

In the definitions, $\mathcal{L}$ is the *source logic* in which concept descriptions are represented and $\mathcal{L}'$ is the *target logic* in which the solutions are searched. Without any restriction, solutions to abductive reasoning problems can be very general, trivial and even inconsistent. To prevent undesirable solutions, several conditions have to be imposed in order to obtain worth solutions [7].

There are many applications of abductive reasoning methods in ontology-based systems using description logics, ranging from resource retrieval [5], semantic matchmaking [10] to medical diagnosis as well as many others [7]. In the scope of this paper, we concentrate only on the concept reasoning task.

Concept abduction is proposed as a novel inference service for description logics firstly in the context of semantic matchmaking[11], [10], an ontology-based process aimed to find best matches for a *request* or a *demand* among available *supplies* in the market. Formalizing this operation in the context of description logics, matches between the request D and a supply C with respect to a common ontology $\mathcal{T}$ can be categorized into five classes which are listed below in the order of the most preferable to the less :

1) **Exact match** corresponding to $D \equiv_\mathcal{T} C$ implies that the match is perfect because D and C are equivalent concepts.
2) **Full match** corresponding to $C \sqsubseteq_\mathcal{T} D$ suggests that all features required by D can be entirely fulfilled by C .
3) **Plug-in match** corresponding to $D \sqsubseteq_\mathcal{T} C$ indicates that what is demanded by D is more specific than C.
4) **Potential match** corresponds to $C \sqcap D \not\sqsubseteq_\mathcal{T} \bot$ and thus C and D share something in common.
5) **Partial match** corresponding to $C \sqcap D \sqsubseteq_\mathcal{T} \bot$ implies the conflict between C and D.

From the traditional point of view, only the first three classes are really interesting [8], [13], as a consequence the last two ones are less considered. There is no method to rank potential matches and partial matches are just ignored.

It turns out that abductive reasoning methods can bring new measurements for potential matching. In fact, if there is a potential match between C and D w.r.t $\mathcal{T}$, the *maximal* solution H to the abduction problem $\mathcal{P} = \langle \mathcal{T}, C, D \rangle$ can be viewed as the *semantic distance* between C and D w.r.t $\mathcal{T}$. If there are several potential supplies C, then H can be used to rank them for finding most promising ones[10].

For solving the concept abduction problem $\mathcal{P} = \langle \mathcal{T}, C, D \rangle$ when the logics are $\mathcal{ALN}$, an uniform tableaux-based algorithm has been proposed [4]. However, this language is quite inexpressive, in many domains the representation of ontology would require more expressive languages. That is the reason why we would like to extend this method for $\mathcal{ALCQ}$, an extension of $\mathcal{ALN}$ with the full concept negation and quantified cardinality. It will be the subject of discussion in the next section.

IV. Concept abduction in $\mathcal{ALCQ}$.

We consider the problem of concept abduction $\mathcal{P}$ for given $\langle \mathcal{T}, C, D \rangle$ where concepts C, D and the TBox $\mathcal{T}$ are expressed in $\mathcal{ALCQ}$, solutions H are computed in the target logic $\mathcal{ALN}$. To recall, in the uniform tableaux-based method [4], instead of using one labeling function $L()$ as normally, two different functions are employed , $T()$ and $F()$, to denote respectively concepts and roles that individuals and pairs of individuals belong and do not belong to.

Hence, given a tableau τ and let $\mathcal{I}_\tau$ be its canonical interpretation, for each node v and each edge $e = \langle v, w \rangle$ of τ, we have :

- if $C \in T(v)$ and $D \in F(v)$ then $v^{\mathcal{I}_\tau} \in C^{\mathcal{I}_\tau}$ and $v^{\mathcal{I}_\tau} \notin D^{\mathcal{I}_\tau}$,
- if $R \in T(e)$ and $S \in F(e)$ then $\left\langle v^{\mathcal{I}_\tau}, w^{\mathcal{I}_\tau} \right\rangle \in R^{\mathcal{I}_\tau}$ and $\left\langle v^{\mathcal{I}_\tau}, w^{\mathcal{I}_\tau} \right\rangle \notin R^{\mathcal{I}_\tau}$.

From these semantics, two kinds of clashes can be then distinguished in τ :

1) *Homogeneous clashes* that occur inside T's or F's labels and
2) *Heterogeneous clashes* that happen due to the intersection of two different T's and F's labels.

If the tableau contains a clash, $\mathcal{I}_\tau$ will not be consistent and thus the knowledge-base can not have a model with respect to the tableau's constraints. But more than that, separating two types of clashes allow to determine if the unsatisfiability is caused either by single concepts (homogeneous clashes) or by both of them (heterogeneous clashes), which will be translated to either by the self-contradiction of concepts or by the subsumption between them.

Consequently, in order to build the solution H, we only need to find concept descriptions that, when added to the root of the original tableau, will generate clashes in every of its derivations, in which at least one clash is heterogeneous. For discovering such formulas, we perform abduction on the labels of tableaux's nodes.

For illustrating, let's consider the following abduction problem where $C := \exists r.A \sqcap \forall r.B$, $D := \exists r.C$ and $\mathcal{T} = \emptyset$. Using uniform tableaux transformation rules, a tableau for $C \sqsubseteq D$ would be obtained as follows :

$$\begin{array}{ll} & T(v_0) = \{\exists r.A \sqcap \forall r.B\} \\ v_0 : & F(v_0) = \{\exists r.C\} \\ \Big\downarrow r & \\ & T(v_1) = \{A, B\} \\ v_1 : & F(v_1) = \{C\} \end{array}$$

Fig. 1. A tableau for $\exists r.A \sqcap \forall r.B \sqsubseteq \exists r.C$

Since the tableau is open, $C \not\sqsubseteq D$. To engender a heterogeneous clash in it, obviously $H := \forall r.C$ must be the finding solution.

A. An uniform tableaux-based method for $\mathcal{ALCQ}$

We start by extending uniform tableaux calculus for the description logic $\mathcal{ALCQ}$. To do that, we define a *prefixed tableau* as a tree-like structure $\tau = (V, E, v^r, T, F)$, where :

- V is a finite set of nodes, $v^r \in V$ is the root of the tableau.
- E is a finite set of edges.
- $T(\,)$ and $F(\,)$ are two labeling functions.

To distinguish, elements included in $T(\,)$ and $F(\,)$ are called respectively *T-labels* and *F-labels*. Sometimes, we will use $w_i \neq w_j$ when necessary for denoting that w_i and w_j are two distinct individuals in τ.

For any $e = \langle v, w\rangle \in E$, if either $R \in T(\langle v, w\rangle)$ or $\neg R \in F(\langle v, w\rangle)$, then w is called a *R-successor* of v, and v is called the *predecessor* of w in τ. Then, as usual, *ancestor* is the transitive closure of predecessors and *descendant* is the transitive closure of successors.

Principally, from defined semantics of $T(\,)$ and $F(\,)$, let $\mathcal{I}_\tau$ be the canonical interpretation of a prefixed tableau τ, the followings hold :

1) If $(C_1 \sqcap C_1) \in T(v)$ then $v^{\mathcal{I}_\tau} \in (C_1)^{\mathcal{I}_\tau} \wedge v^{\mathcal{I}_\tau} \in (C_2)^{\mathcal{I}_\tau}$.
   If $(C_1 \sqcap C_2) \in F(v)$ then $v^{\mathcal{I}_\tau} \in (\neg C_1)^{\mathcal{I}_\tau} \vee v^{\mathcal{I}_\tau} \in (\neg C_2)^{\mathcal{I}_\tau}$.
2) If $(C_1 \sqcup C_2) \in T(v)$ then $v^{\mathcal{I}_\tau} \in (C_1)^{\mathcal{I}_\tau} \vee v^{\mathcal{I}_\tau} \in (C_2)^{\mathcal{I}_\tau}$.
   If $(C_1 \sqcup C_2) \in F(v)$ then $v^{\mathcal{I}_\tau} \in (\neg C_1)^{\mathcal{I}_\tau} \wedge v^{\mathcal{I}_\tau} \in (\neg C_2)^{\mathcal{I}_\tau}$.
3) If $(\forall R.C) \in T(v)$ and there is R-successor w of v then $w^{\mathcal{I}_\tau} \in (C)^{\mathcal{I}_\tau}$.
   If $(\forall R.C) \in F(v)$ then there must be a R-successor w of v such that $w^{\mathcal{I}_\tau} \in (\neg C)^{\mathcal{I}_\tau}$.
4) If $(\exists R.C) \in T(v)$ then there must be a R-successor w of v such that $w^{\mathcal{I}_\tau} \in (C)^{\mathcal{I}_\tau}$.
   If $(\exists R.C) \in F(v)$ and there is R-successor w of v then $w^{\mathcal{I}_\tau} \in (\neg C)^{\mathcal{I}_\tau}$.
5) If $(\leqslant nR.C) \in T(v)$ then there must not be $(n+1)$ distinct R-successors $w_1, \ldots, w_{n+1}$ of v such that $w_i^{\mathcal{I}_\tau} \in (C)^{\mathcal{I}_\tau}$ for $1 \leq i \leq n+1$.
   If $(\leqslant nR.C) \in F(v)$ then there must be at least $(n+1)$ distinct R-successors $w_1, \ldots, w_{n+1}$ of v such that $w_i^{\mathcal{I}_\tau} \in (C)^{\mathcal{I}_\tau}$ for $1 \leq i \leq n+1$.
6) If $(\geqslant nR.C) \in T(v)$ then there must be at least n distinct R-successors $w_1, \ldots, w_n$ of v such that $w_i^{\mathcal{I}_\tau} \in (C)^{\mathcal{I}_\tau}$ for $1 \leq i \leq n$.
   If $(\geqslant nR.C) \in F(v)$ then there must not be n distinct R-successors $w_1, \ldots, w_n$ such that $w_i^{\mathcal{I}_\tau} \in (C)^{\mathcal{I}_\tau}$ for $1 \leq i \leq n$.

Then, given two concepts C, D and a TBox $\mathcal{T}$ in $\mathcal{ALCQ}$, for verifying if $C \sqsubseteq_\mathcal{T} D$, the uniform tableaux-based algorithm works by starting with an initial $\tau_0 = \{V, E, v^r, T, F\}$ where :

- $V = \{v^r\}$,
- $E = \emptyset$,
- $T(v^r) = \{C \sqcap C_\mathcal{T}\}$,
- $F(v^r) = \{D\}$.

Then, the following transformation rules are applied to some tableau τ until no rule can be used. We assume that all concepts are already in the NNF, i.e the normal form in which negations occur only in front of concept names. Using de Morgan's rules and other rules for quantifier, the normalization can be done in linear time

$$\begin{array}{rcl} \neg(C \sqcup D) & \mapsto & \neg C \sqcap \neg D \\ \neg(C \sqcap D) & \mapsto & \neg C \sqcup \neg D \\ \neg\forall R.C & \mapsto & \exists R.\neg C \\ \neg\exists R.C & \mapsto & \forall R.\neg C \end{array}$$

Thus, in the following, $\dot{\neg}C$ will be used to denote the NNF of $\neg C$.

1) $\sqcap$-**rules**:
   - **if** $(C_1 \sqcap C_2) \in T(v)$, v is not blocked and $\{C_1, C_2\} \not\subseteq T(v)$ **then** $T(v) = T(v) \cup \{C_1, C_2\}$.

- **if** $(C_1 \sqcup C_2) \in F(v)$, v is not blocked and $\{C_1, C_2\} \nsubseteq T(v)$ **then** $F(v) = F(v) \cup \{C_1, C_2\}$.

2) $\sqcup$-**rule**:

- **if** $(C_1 \sqcup C_2) \in T(v)$, v is not blocked and $\{C_1, C_2\} \cap T(v) = \emptyset$ **then**:
  - $\tau_1 = \{V, E, v^r, T, F\}$ where $T(v) = T(v) \cup \{C_1\}$.
  - $\tau_2 = \{V, E, v^r, T, F\}$ where $T(v) = T(v) \cup \{C_2\}$.
- **if** $(C_1 \sqcap C_2) \in F(v)$, v is not blocked and $\{C_1, C_2\} \cap F(v) = \emptyset$ **then**:
  - $\tau_1 = \{V, E, v^r, T, F\}$ where $F(v) = F(v) \cup \{C_1\}$.
  - $\tau_2 = \{V, E, v^r, T, F\}$ where $F(v) = F(v) \cup \{C_2\}$.

3) $\exists$-**rule**:

- **if** $(\exists R.C) \in T(v)$, v is not blocked and there is no R-successor w of v with either $C \in T(w)$ or $\neg C \in F(w)$ **then** :
  - $V = V \cup \{w\}$, $E = E \cup \{\langle v, w\rangle\}$, $T(\langle v, w\rangle) = \{R\}$ and $T(w) = \{C\}$.
- **if** $(\forall R.C) \in F(v)$, v is not blocked and there is no R-successor w of v with either $C \in T(w)$ or $\neg C \in F(w)$ **then** :
  - $V = V \cup \{w\}$, $E = E \cup \{\langle v, w\rangle\}$, $F(\langle v, w\rangle) = \{\neg R\}$ and $F(w) = \{C\}$.

4) $\forall$-**rules**:

- **if** $(\forall R.C) \in T(v)$, v is not blocked and there is a R-successor w of v such that $C \notin T(w)$ **then** :
  - $T(w) = T(w) \cup \{C\}$.
- **if** $(\exists R.C) \in F(v)$, v is not blocked and there is a R-successor w of v such that $C \notin F(w)$ **then** :
  - $F(w) = F(w) \cup \{C\}$.

5) *choose*-**rule**:

- **if** $(\leqslant nR.C) \in T(v)$**,** v is not blocked and there is a R-successor w of v such that $\{C, \doteq\neg C\} \cap T(w) = \emptyset$ **then** :
  - $\tau_1 = \{V, E, v^r, T, F\}$ where $T(w) = T(w) \cup \{C\}$.
  - $\tau_2 = \{V, E, v^r, T, F\}$ where $T(w) = T(w) \cup \{\doteq\neg C\}$.
- **if** $(\geqslant nR.C) \in F(v)$**,** v is not blocked and there is a R-successor w of v such that $\{C, \doteq\neg C\} \cap F(w) = \emptyset$ **then** :
  - $\tau_1 = \{V, E, v^r, T, F\}$ where $F(w) = F(w) \cup \{C\}$.
  - $\tau_2 = \{V, E, v^r, T, F\}$, where $F(w) = F(w) \cup \{\doteq\neg C\}$.

6) $\geqslant$-**rules**:

- **if** $(\geqslant nR.C) \in T(v)$, v is not blocked and there are not n R-successors $w_1, \ldots, w_n$ of v such that $C \in T(w_k)$ for $1 \leq k \leq n$ and $w_i \neq w_j$ for $1 \leq i < j \leq n$ **then** :
  - $V = V \cup \{w_i\}$, $E = E \cup \{\langle v, w_i\rangle\}$, $T(\langle v, w_i\rangle) = \{R\}$, let $T(w_i) = \{C\}$ if $C \neq \top$ and add $w_i \neq w_j$ for $1 \leq i < j \leq n$ to τ.
- **if** $(\leqslant nR.C) \in F(v)$**,** v is not blocked and there are not $n+1$ local R-successors $w_1, \ldots, w_{n+1}$ of v such that $\doteq C \in F(w_k)$ for $1 \leq k \leq n+1$ and $w_i \neq w_j$ for $1 \leq i < j \leq n+1$ **then** :
  - $V = V \cup \{w_i\}$, $E = E \cup \{\langle v, w_i\rangle\}$, $F(\langle v, w_i\rangle) = \{\neg R\}$, let $F(w_i) = \{\doteq C\}$ if $C \neq \top$ and add $w_i \neq w_j$ for $1 \leq i \leq n+1$ to τ.

7) $\leqslant$-**rules**:

- **if** $(\leqslant nR.C) \in T(v)$, v is not blocked and there are $n+1$ R-successors $w_1, \ldots, w_{n+1}$ of v such that either $C \in T(w_k)$ or $\doteq C \in F(w_k)$ for $1 \leq k \leq n+1$ and there is not $w_i \neq w_j$ for some $i \neq j$ **then** :
  - for each pair w_i, w_j such that $i < j$, either $C \in T(w_i) \cap T(w_j)$ or $\doteq C \in F(w_i) \cap F(w_j)$ and there is not $w_i \neq w_j$ in τ, $\mathrm{Merge}(w_i, w_j)$.
- **if** $(\geqslant nR.C) \in F(v)$, v is not blocked and there are n R-successors $w_1, \ldots, w_n$ of v such that either $C \in T(w_k)$ or $\doteq C \in F(w_k)$ for $1 \leq k \leq n$ and there is not $w_i \neq w_j$ for some $i \neq j$ **then** :
  - for each pair w_i, w_j such that $i < j$, either $C \in T(w_i) \cap T(w_j)$ or $\doteq C \in F(w_i) \cap F(w_j)$ and there is not $w_i \neq w_j$ in τ, $\mathrm{Merge}(w_i, w_j)$.

In $\leqslant$-**rules**, we have used the operator $\mathrm{Merge}()$ in order to merge a node with another :

- $\mathrm{Merge}(v, v')$ where v and v' are two nodes such that there is not $v \neq v'$ in τ :
  - let $T(v) = T(v) \cup T(v')$, $F(v) = F(v) \cup F(v')$ and $E = E \cup \{\langle v, w\rangle\}$;
  - let $T(\langle v, w\rangle) = T(\langle v', w\rangle)$ and $F(\langle v, w\rangle) = F(\langle v', w\rangle)$ for each w is a successor of v'.
  - let $V = V \setminus \{v'\}$ and $E = E \setminus \{\langle u, v'\rangle\}$, where u is the predecessor of v and v'.
  - add $v \neq v''$ for each v'', $v' \neq v''$ in τ .

We assign to each rule a priority in such a way that if there is more than one can be applied, those on T-labels are always executed first. To ensure the termination of the transformation, a blocking strategy is required as usual: for any node $v \in V$, v is *blocked by* u if :

- u comes before v in some enumeration and
- either $T(v) \subseteq T(u)$ or $F(v) \subseteq F(u)$.

Some rules are non-deterministic, as a result a tableau τ can be transformed into new tableaux $\tau_1, \ldots, \tau_m$. Because the rules preserve the consistency, the canonical interpretation $\mathcal{I}_\tau$ of τ is consistent iff one of $\mathcal{I}_{\tau_1}, \ldots, \mathcal{I}_{\tau_m}$ is so.

Definition 1: (**Clashes**) A prefixed tableau $\tau = (V, E, v^r, T, F)$ contains a ***homogeneous clash*** if one of the followings holds for some node $v \in V$:

- either $\{A, \neg A\} \subseteq T(v)$or $\{A, \neg A\} \subseteq F(v)$;
- $(\leqslant nR.C) \in T(v)$, v has mR-successors $w_1, \ldots, w_m$ such that :
  - $m > n$,

- $C \in T(w_i)$ for $1 \leq i \leq m$ and
- $w_i \neq w_j$ for $1 \leq i < j \leq m$;

- $(\geqslant nR.C) \in F(v)$, v has m R-successors $w_1, \ldots, w_m$ such that :
  - $m \geq n$,
  - $\doteq C \in F(w_i)$ for $1 \leq i \leq m$ and
  - $w_i \neq w_j$ for $1 \leq i < j \leq m$;

The tableau contains a ***heterogeneous clash*** if $T(v) \cap F(v) \neq \emptyset$.

Clashes are defined very similarly as in [4]. To distinguish, we call homogeneous clashes happening in T-labels and F-labels respectively *T-homogeneous* and *F-homogeneous* clashes. Then, the tableau τ is *complete* if no rule can be applied to it. It's *closed* if it contains a clash, otherwise it's *open*. Finally, the following theorem establishes the soundness and completeness of the algorithm :

Theorem 1: Let C, D be two $\mathcal{ALCQ}$ concepts, $\mathcal{T}$ be an acyclic general TBox in $\mathcal{ALCQ}$. Then $C \sqsubseteq_{\mathcal{T}} D$ iff all prefixed tableau constructed by the uniform algorithm starting with $\tau_0 = (V, E, v^r, T, F)$ where $V = \{v^r\}$, $E = \emptyset$, $T(v) = \{C\}$ and $F(v) = \{D\}$ are closed. Particularly, if every tableau contains at least one T-homogeneous clash then $C \sqsubseteq_{\mathcal{T}} \bot$.

The complexity of the algorithm is derived directly from the hardness of the normal tableaux-based algorithm for $\mathcal{ALCQ}$. Indeed, if we look at dual rules in detail, each of them applied to $T(v) = \{C\}$ and $F(v) = \{D\}$ is equivalent to a normal rule applied to $L(v) = \{C \sqcup \neg D\}$. Thus, the uniform algorithm performs neither worse nor better than the ordinary one.

Theorem 2: Let C and D be two $\mathcal{ALCQ}$ concepts, $\mathcal{T}$ be a acyclic general TBox in $\mathcal{ALCQ}$. Deciding if $C \sqsubseteq_{\mathcal{T}} D$ using the uniform tableaux algorithm is EXPTIME-complete.

B. Computing abductive solutions.

As mentioned previously, in order to construct a solution H to the concept abduction problem $\mathcal{P}$ for given $\langle \mathcal{T}, C, D \rangle$, we try to find concept descriptions that would generate clashes in every tableau for $C \sqsubseteq_{\mathcal{T}} D$ such that among them at least one clash is heterogeneous. To do that, let $\theta = \{\tau_1, \ldots, \tau_n\}$ be the set of all open tableaux for verifying $C \sqsubseteq_{\mathcal{T}} D$, if θ is not empty, to compute H, we start by building for each $\tau_i \in \theta$, two *closing sets* $cl^T(\tau_i)$ and $cl^H(\tau_i)$, which contains formulas that will eventually close it :

Definition 2: (**Closing sets**) Let $\tau = (V, E, v^r, T, F)$ be any open and complete tableau. We define $cl^T(\tau)$ and $cl^H(\tau)$ as two disjoint sets of concept descriptions built from labels of τ such that, if we let τ_T and τ_H be respectively complete tableaux derived from τ by adding any element of $ct^T(\tau)$ and $cl^H(\tau)$ into $T(v^r)$, then τ_T and τ_H are closed. Moreover,

- every clash in τ_T is T-homogeneous and,
- every clash in τ_H is heterogeneous.

It can be seen that any conjunctive expression H built from elements of $cl^T()$ and $cl^H()$ such that, for each open tableau at least one of its closing elements is appeared in H, will be the finding solution if $C \sqcap H \not\sqsubseteq_{\mathcal{T}} \bot$. We characterize these conditions in the following theorem :

Theorem 3: Let $\mathcal{P} = \langle \mathcal{T}, C, D \rangle$ be a concept abduction problem in the description logic $\mathcal{ALCQ}$. Let $\theta = \{\tau_1, \ldots, \tau_n\}$ be the set of open and complete prefixed tableaux for $C \sqsubseteq_{\mathcal{T}} D$. If θ is not empty, for each $\tau_i \in \theta$, let $cl(\tau_i) = cl^T(\tau_i) \cup cl^H(\tau_i)$ and $\mathsf{choice}()$ be some choice function.

For any set $S = \langle H_1 = \mathsf{choice}(cl(\tau_i)), \ldots, H_n = \mathsf{choice}(cl(\tau_i)) \rangle$, let H be a conjunctive expression built from elements of S, $H ::= H_1 \sqcap \cdots \sqcap H_n$. If :

1) for every $\tau_i \in \theta$, $S \cap cl(\tau_i) \neq \emptyset$ and
2) $\exists \tau_j \in \theta$ such that $S \cap cl^T(\tau_j) = \emptyset$.

then H is a solution to the problem $\mathcal{P}$.

For building the closing sets, from the definition of clashes, it can be seen that there are two ways to generate a clash in some node v of the tableau, the first one is the *contradiction of concept names*. Thus,

- if A is a (maybe negated) concept name and $A \in T(v)$, then $\doteq A \in cl^T(v)$;
- if B is a (maybe negated) concept name, $B \in F(v)$ and $\doteq B \notin T(v)$, then $B \in cl^H(v)$.
- if v has a R-successor w and there is not another R-successor w' of v such that $w' \neq w$ is included in τ then :
  - if $R \in T(\langle v, w \rangle)$ then $\forall R.E \in cl^T(v)$, for each $E \in cl^T(w)$,
  - otherwise $\forall R.F \in cl^H(v)$, for each $F \in cl^H(w)$ if $\forall R.F \notin cl^T(v)$.

The other way comes from the *incompatibility of qualified number restrictions*. So,

- if v has n R-successors $w_1, \ldots, w_n$ such that $n > 1$ and $w_i \neq w_j$ for $1 \leq i < j \leq n$ then :
  - if $R \in T(\langle v, w_i \rangle)$ for every $1 \leq i \leq n$ then :
    - $\leqslant (k-1)R.\top \in cl^T(v)$;
  - otherwise :
    - $\leqslant (k-1)R.\top \in cl^H(v)$.

From the semantics of tableaux rules and the definition of clashes, it can be verified that if one of these expressions is added into $T(v)$, a clash will be eventually produced in some label of derived tableaux. Based on these descriptions, we design a recursive algorithms for computing $cl^T(v)$ as follow.

Algorithm 1 compute $cl^T(v)$

Input: v, a node in a prefixed tableau τ.
Output: The closing set $cl^T(v)$.

1: $cl^T(v) \leftarrow \emptyset$.
2: **if** $T(v) \neq \emptyset$ **then**
3: **for each** $A \in T(v)$, A is a (maybe negated) concept name **do**
4: $cl^T(v) \leftarrow cl^T(v) \cup \{\dot{\neg}A\}$.
5: **end for**
6: **end if**
7: **if** v has n R-successor $w_1, \ldots, w_n$ such that $n \geq 2$, $w_i \neq w_j$ for $1 \leq i < j \leq n$ and $R \in T(\langle v, w_i \rangle)$ for every $1 \leq i \leq n$ **then**
8: $cl^T(v) \leftarrow cl^T(v) \cup \{\leqslant (k-1)R.\top\}$
9: **end if**
10: **for each** w, w is a R-successor of v and there is not w' such that $w \neq w'$ is present in τ **do**
11: **for each** $E \in cl^T(w)$ **do**
12: $cl^T(v) \leftarrow cl^T(v) \cup \{\forall R.E\}$.
13: **end for**
14: **end for**
15: **return** $cl^T(v)$.

In the same manner, $cl^H(v)$ is built by the following procedure.

Algorithm 2 compute $cl^H(v)$

Input: v, a node in an open and complete prefixed tableau τ.
Output: The closing set $cl^H(v)$.

1: $cl^H(v) \leftarrow \emptyset$.
2: **if** $F(v) \neq \emptyset$ **then**
3: **for each** $B \in F(v)$, B is a (maybe negated) concept name **do**
4: $cl^H(v) \leftarrow cl^H(v) \cup \{B\}$.
5: **end for**
6: **end if**
7: **if** v has m R-successor $w_1, \ldots, w_m$ such that $m \geq 2$, $w_i \neq w_j$ for $1 \leq i < j \leq m$ and $\neg R \in F(\langle v, w_i \rangle)$ for some i, $1 \leq i \leq m$ **then**
8: $cl^H(v) \leftarrow cl^H(v) \cup \{\leqslant (k-1)R.\top\}$.
9: **end if**
10: **for each** w, w is a R-successor of v and there is not w' such that $w \neq w'$ is present in τ **do**
11: **for each** $F \in cl^H(w)$ **do**
12: $cl^H(v) \leftarrow cl^H(v) \cup \{\forall R.F\}$.
13: **end for**
14: **end for**
15: $cl^H(v) \leftarrow cl^H(v) \setminus \big(cl^H(v) \cap cl^T(v)\big)$.
16: **return** $cl^H(v)$.

Theorem 4: Given τ a prefixed tableaux for the description logic $\mathcal{ALCQ}$. The sets$cl^T(v^r)$ and $cl^H(v^r)$ computed respectively by the algorithm 1 and 2are the closing sets of τ.
Let $c_T := \bigcup_{\tau_i \in \theta} cl^T(\tau_i)$ and $c_H := \bigcup_{\tau_i \in \theta} cl^H(\tau_i)$. For each closing element $H_i \in c_T \cup c_H$, we define $\tau - set(H_i) = \{\tau \mid H_i \in cl(\tau)\}$. Let S be a *minimal set* of $c_T \cup c_H$ such that :

- $\theta \subseteq \bigcup_{H_i \in S} \tau - set(H_i)$,
- $\{H_i, \dot{\neg}H_i\} \not\subseteq S$,
- $S \cap c_H \neq \emptyset$.

Then, $H := \bigsqcap_{H_i \in S} H_i$ is a solution to the abduction problem $\mathcal{P} = \langle \mathcal{T}, C, D \rangle$. We implement this process in the algorithm 3 below :

Algorithm 3 conceptAbduction

Input: C and D, two $\mathcal{ALCQ}$ concepts.
Input: $\mathcal{T}$, a general acyclic TBox in $\mathcal{ALCQ}$.
Output: H, a solution to the concept abduction problem $(\mathcal{T}, C, D)$.

1: $S \leftarrow \emptyset$ and $H \leftarrow \epsilon$.
2: Build $\theta = \{\tau_1, \ldots, \tau_n\}$, the set of open and complete prefixed tableaux for $C \sqsubseteq_{\mathcal{T}} D$.
3: **if** $\theta = \emptyset$ **then**
4: */*$C \sqsubseteq_{\mathcal{T}} D$ and thus no abductive solution is needed*/*.
5: **else**
6: $c_T \leftarrow \emptyset$ and $c_H \leftarrow \emptyset$.
7: **for each** $\tau_j \in \theta$ **do**
8: $c_T \leftarrow c_T \cup cl^T(\tau_j)$.
9: $c_H \leftarrow c_H \cup cl^H(\tau_j)$.
10: **end for**
11: Build S, the minimal subset of $c_T \cup c_H$ such that :
(i) $\theta \subseteq \bigcup_{H_i \in S} \tau - set(H_i)$,
(ii) $\{H_i, \dot{\neg}H_i\} \not\subseteq S$ and
(iii) $S \cap c_H \neq \emptyset$.
12: **for each** $H_i \in S$ **do**
13: $H \leftarrow H \sqcap H_i$
14: **end for**
15: **end if**
16: **return** H.

Theorem 5: The concept description H returned by the algorithm 3 is a solution the concept abduction problem $\mathcal{P}$ for given $\langle \mathcal{T}, C, D \rangle$

In [11], the complexity of any abductive algorithm is proved to be bound by the complexity of the subsumption problem in the corresponding description logics. For the description $\mathcal{ALCQ}$, it's EXPTIME-complete.

It turns out that our algorithm do not perform worse than this bound. Indeed, we observe that the complexity of algorithms 1 and 2 is equal to the complexity of verifying if a tableau specifies a consistent model, which is shown to be exponential in [1]. Then, for finding of the minimal set S which contains sub-descriptions of the solution H, at the worst, we have to investigate through all subsets of $c_T \cup c_H$. As this process does not exceed the exponential bound neither, the complexity of the algorithm 3can not be worse than EXPTIME.

V. Example

To illustrate how the algorithms1, 2 and 3 work, let's consider the following abduction problem $\mathcal{P} = \langle \mathcal{T}, C, D \rangle$

where :

$$\mathcal{T} = \left\{ \begin{array}{lcl} \mathsf{Room1} & \sqsubseteq & \exists hasEquipment.\,(\mathsf{Desk} \sqcup \mathsf{Table}) \\ \mathsf{Room2} & \sqsubseteq & \exists hasInternet.\,(\mathsf{WiFi} \sqcup \mathsf{ADSL}) \end{array} \right\}$$

$$C ::= \mathsf{Room1} \sqcup \mathsf{Room2}$$

$$D ::= \exists hasEquipment.\mathsf{Desk} \sqcap \exists hasInternet.\mathsf{WiFi}$$

For computing the solution to $\mathcal{P}$, the algorithm starts by building tableaux for $C \sqsubseteq_{\mathcal{T}} D$. The whole transformation of tableaux is illustrated in the appendix. Accordingly, the set of open and complete tableaux is

$$\theta = \{\tau_8, \tau_9, \tau_{10}, \tau_{18}, \tau_{19}, \tau_{21}, \tau_{22}, \tau_{28}, \tau_{29}, \tau_{30}, \tau_{38}, \tau_{39}, \tau_{41}, \tau_{42}\}$$

As θ is not empty, $C \not\sqsubseteq_{\mathcal{T}} D$. We compute then closing sets for these tableaux. The result is listed and then summarized in two tables I and II :

Table I
CLOSING SETS OF OPEN TABLEAUX

τ_i	$cl^T(\tau_i)$	$cl^H(\tau_i)$
τ_8	$\neg\mathsf{Room}_1, \mathsf{Room}_2, \forall hasEquipment.\neg\mathsf{Desk}$	$\emptyset$
τ_9	$\neg\mathsf{Room}_1, \mathsf{Room}_2, \forall hasEquipment.\neg\mathsf{Table}$	$\forall hasEquipment.\mathsf{Desk}$
τ_{10}	$\neg\mathsf{Room}_1, \mathsf{Room}_2, \forall hasEquipment.\neg\mathsf{Table}$	$\emptyset$
τ_{18}	$\neg\mathsf{Room}_1, \forall hasEquipment.\neg\mathsf{Table}, \forall hasInternet.\neg\mathsf{WiFi}$	$\forall hasEquipment.\mathsf{Desk}$
τ_{19}	$\neg\mathsf{Room}_1, \forall hasEquipment.\neg\mathsf{Desk}, \forall hasInternet.\neg\mathsf{ADSL}$	$\forall hasInternet.\mathsf{WiFi}$
τ_{21}	$\neg\mathsf{Room}_1, \forall hasEquipment.\neg\mathsf{Table}, \forall hasInternet.\neg\mathsf{ADSL}$	$\forall hasEquipment.\mathsf{Desk}$
τ_{22}	$\neg\mathsf{Room}_1, \forall hasEquipment.\neg\mathsf{Table}, \forall hasInternet.\neg\mathsf{ADSL}$	$\forall hasInternet.\mathsf{WiFi}$
τ_{28}	$\neg\mathsf{Room}_2, \mathsf{Room}_1, \forall hasInternet.\neg\mathsf{WiFi}$	$\emptyset$
τ_{29}	$\neg\mathsf{Room}_2, \mathsf{Room}_1, \forall hasInternet.\neg\mathsf{ADSL}$	$\forall hasInternet.\mathsf{WiFi}$
τ_{30}	$\neg\mathsf{Room}_2, \mathsf{Room}_1, \forall hasInternet.\neg\mathsf{ADSL}$	$\emptyset$
τ_{38}	$\neg\mathsf{Room}_2, \forall hasEquipment.\neg\mathsf{Table}, \forall hasInternet.\neg\mathsf{WiFi}$	$\forall hasEquipment.\mathsf{Desk}$
τ_{39}	$\neg\mathsf{Room}_2, \forall hasEquipment.\neg\mathsf{Desk}, \forall hasInternet.\neg\mathsf{ADSL}$	$\forall hasInternet.\mathsf{WiFi}$
τ_{41}	$\neg\mathsf{Room}_2, \forall hasEquipment.\neg\mathsf{Table}, \forall hasInternet.\neg\mathsf{ADSL}$	$\forall hasEquipment.\mathsf{Desk}$
τ_{42}	$\neg\mathsf{Room}_2, \forall hasEquipment.\neg\mathsf{Table}, \forall hasInternet.\neg\mathsf{ADSL}$	$\forall hasInternet.\mathsf{WiFi}$

Table II
CLOSING ELEMENTS AND ITS $\tau - set\,()$

Closing element (H_i)	$\tau - set\,(H_i)$
$\neg\mathsf{Room}_1$	$\{\tau_{12}, \tau_{13}, \tau_{14}, \tau_{18}, \tau_{19}, \tau_{21}, \tau_{22}\}$
Room_2	$\{\tau_{12}, \tau_{13}, \tau_{14}\}$
$\forall hasEquipment.\neg\mathsf{Desk}$	$\{\tau_{12}, \tau_{18}, \tau_{30}\}$
$\forall hasEquipment.\neg\mathsf{Table}$	$\{\tau_{13}, \tau_{14}, \tau_{19}, \tau_{21}, \tau_{22}, \tau_{31}, \tau_{33}, \tau_{34}\}$
$\forall hasInternet.\neg\mathsf{ADSL}$	$\{\tau_{18}, \tau_{21}, \tau_{22}, \tau_{25}, \tau_{26}, \tau_{30}, \tau_{33}, \tau_{34}\}$
$\forall hasInternet.\neg\mathsf{WiFi}$	$\{\tau_{19}, \tau_{24}, \tau_{31}\}$
$\neg\mathsf{Room}_2$	$\{\tau_{24}, \tau_{25}, \tau_{26}, \tau_{30}, \tau_{31}, \tau_{33}, \tau_{34}\}$
Room_1	$\{\tau_{24}, \tau_{25}, \tau_{26}\}$
$\forall hasEquipment.\mathsf{Desk}$	$\{\tau_{13}, \tau_{19}, \tau_{21}, \tau_{31}, \tau_{33}\}$
$\forall hasInternet.\mathsf{WiFi}$	$\{\tau_{18}, \tau_{22}, \tau_{25}, \tau_{30}, \tau_{34}\}$

From these elements, we build a concept description $H = \mathsf{Room}_1 \sqcap \mathsf{Room}_2 \sqcap \forall hasEquipment.\mathsf{Desk} \sqcap \forall hasInternet.\mathsf{WiFi}$. It can be verified that H is a solution to the abduction problem $\mathcal{P}$.

VI. CONCLUSIONS

We have studied in the scope of this paper the problem of concept abduction in the context of $\mathcal{ALCQ}$, the description logic that is enough expressive for many ontology-based applications with the support of full concept negation and qualified number restrictions. Based on the already developed tableaux-based method for solving the problem in $\mathcal{ALN}$, we have developed an extension of this technique for dealing with concepts represented in $\mathcal{ALCQ}$.

To do that, we have introduced firstly new expansion rules to handle expressive formulas under the scope of full concept negations and quantified cardinality. Then, using complements of constraints in open tableaux, we compute concept descriptions that will eventually lead to their closure. Finally, a solution to the problem of concept abduction is built as a consistent conjunction of such formulas.

We would like to develop the result of this paper in two directions. Firstly, we would like to improve algorithms 1 and 2to obtain abductive solutions in more expressive languages, such as $\mathcal{ALQ}$. In the other direction, we want to extend the concept abductive reasoning task to the distributed modular formalisms, such as Package-based Description logics [3]. Thereby, semantic matchmaking can be performed in more general contexts where requests and offers are specified in different, distributed but related ontologies, which are quite common in domains such as eTourism.

REFERENCES

[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:2001, 2000.

[3] Jie Bao, George Voutsadakis, Giora Slutzki, and Vasant Honavar. Package-based description logics.

[4] S. Colucci, T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. A uniform tableaux-based approach to concept abduction and contraction in aln. In *Contraction in ALN . In Proc. of DL-04 (2004), CEUR Workshop Proceedings*, page 104, 2004.

[5] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, Agnese Pinto, and Azzurra Ragone. Semantic-based resource retrieval using non-standard inference services in description logics.

[6] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Marina Mongiello, and Francesco M. Donini. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In *ICEC '04: Proceedings of the 6th international conference on Electronic commerce*, pages 41–50, New York, NY, USA, 2004. ACM.

[7] Corinna Elsenbroich, Oliver Kutz, and Ulrike Sattler. A case for abductive reasoning over ontologies. In *in 'Proc. OWL: Experiences and Directions*, pages 10–11, 2006.

[8] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. pages 331–339. ACM Press, 2003.

[9] Industry Minister of the Economy and Employment. Nouvelle grille de classification hôtelière, 2009.

[10] Tommaso Di Noia, Eugenio Di Sciascio, and Francesco M. Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research*, 29:307, 2007.

[11] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. Abductive matchmaking using description logics. In *In Proc. of IJCAI 2003*, pages 337–342. Morgan Kaufmann, 2003.

[12] Charles Sanders Peirce. Abduction and induction. In Justus Buchler, editor, *Philosophical writings of Pierce*, pages 150–156. New York: Dover Books, 1955.

[13] Katia Sycara, Widoff S, Klusch M, and Jianguo Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. In *in Cyberspace. Autonomous Agents and Multi-Agent Systems*, pages 173–203, 2002.

APPENDIX

The initial tableau is :

$\tau_0 = \{V_0, E_0, v^r, T, F\}$ where $V_0 = \{v^r\}$, $E_0 = \emptyset$ with :

- $T(v^r) = \left\{ \begin{array}{l} \mathsf{Room}_1 \sqcup \mathsf{Room}_2, \\ \neg\mathsf{Room}_1 \\ \quad \sqcup \exists hasEquipment.(\mathsf{Desk} \sqcup \mathsf{Table}), \\ \neg\mathsf{Room}_2 \\ \quad \sqcup \exists hasInternet.(\mathsf{ADSL} \sqcup \mathsf{WiFi}) \end{array} \right\}$;
- $F(v^r) = \{\exists hasEquipment.\mathsf{Desk} \sqcap \exists hasInternet.\mathsf{WiFi}\}$.

Tableaux developed on the first branch are :

1) $\tau_1 = \{V, E, v^r, T, F\}$ with :
   - $T(v^r) = \{\ldots, \mathsf{Room}_1\}$;
   - $F(v^r) = \{\ldots\}$.
2) $\tau_2 = \{V, E, v^r, T, F\}$ with :
   - $T(v^r) = \{\ldots, \mathsf{Room}_2\}$;
   - $F(v^r) = \{\ldots\}$.
3) $\tau_3 = \{V, E, v^r, T, F\}$ (CLASH) with :
   - $T(v^r) = \{\ldots, \mathsf{Room}_1, \neg\mathsf{Room}_1\}$;
   - $F(v^r) = \{\ldots\}$.
4) $\tau_4 = \{V, E, v^r, T, F\}$ with :
   - $T(v^r) = \{\ldots, \exists hasEquipment.(\mathsf{Desk} \sqcup \mathsf{Table})\}$;
   - $F(v^r) = \{\ldots\}$.
5) $\tau_5 = \{V, E, v^r, T, F\}$ with :
   - $T(v^r) = \left\{ \begin{array}{c} \ldots, \\ \exists hasEquipment.(\mathsf{Desk} \sqcup \mathsf{Table}), \\ \neg\mathsf{Room}_2 \end{array} \right\}$;
   - $F(v^r) = \{\ldots\}$.
6) $\tau_6 = \{V, E, v^r, T, F\}$ with :
   - $T(v^r) = \left\{ \begin{array}{c} \ldots, \\ \exists hasEquipment.(\mathsf{Desk} \sqcup \mathsf{Table}), \\ \exists hasInternet.(\mathsf{ADSL} \sqcup \mathsf{WiFi}) \end{array} \right\}$;
   - $F(v^r) = \{\ldots\}$.
7) $\tau_7 = \{V_1, E_1, v^r, T, F\}$ (CLASH) with :
   - $T(v^r) = \{\ldots\}$;
   - $F(v^r) = \{\ldots, \exists hasEquipment.\mathsf{Desk}\}$;
   - $T(v_1) = \{\mathsf{Desk} \sqcup \mathsf{Table}, \mathsf{Desk}\}$;
   - $F(v_1) = \{\mathsf{Desk}\}$;
   - $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
   - $F(\langle v^r, v_1\rangle) = \emptyset$.
8) $\tau_8 = \{V_1, E_1, v^r, T, F\}$ with :
   - $T(v^r) = \{\ldots\}$;
   - $F(v^r) = \{\ldots, \exists hasInternet.\mathsf{WiFi}\}$;
   - $T(v_1) = \{\mathsf{Desk} \sqcup \mathsf{Table}, \mathsf{Desk}\}$;
   - $F(v_1) = \emptyset$;
   - $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
   - $F(\langle v^r, v_1\rangle) = \emptyset$.
9) $\tau_9 = \{V_1, E_1, v^r, T, F\}$with :
   - $T(v^r) = \{\ldots\}$;
   - $F(v^r) = \{\ldots, \exists hasEquipment.\mathsf{Desk}\}$;
   - $T(v_1) = \{\mathsf{Desk} \sqcup \mathsf{Table}, \mathsf{Table}\}$;
   - $F(v_1) = \{\mathsf{Desk}\}$;
   - $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
   - $F(\langle v^r, v_1\rangle) = \emptyset$.
10) $\tau_{10} = \{V_1, E_1, v^r, T, F\}$ with :
    - $T(v^r) = \{\ldots\}$;
    - $F(v^r) = \{\ldots, \exists hasInternet.\mathsf{WiFi}\}$;
    - $T(v_1) = \{\mathsf{Desk} \sqcup \mathsf{Table}, \mathsf{Table}\}$;
    - $F(v_1) = \emptyset$;
    - $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
    - $F(\langle v^r, v_1\rangle) = \emptyset$.
11) $\tau_{11} = \{V_2, E_2, v^r, T, F\}$ where $V_2 = \{v^r, v_1, v_2\}$, $E_2 = \{\langle v^r, v_1\rangle, \langle v^r, v_2\rangle\}$ with :
    - $T(v^r) = \{\ldots\}$;
    - $F(v^r) = \{\ldots\}$;
    - $T(v_1) = \{\mathsf{Desk}\}$;
    - $F(v_1) = \emptyset$;
    - $T(v_2) = \{\mathsf{WiFi}\}$;
    - $F(v_2) = \emptyset$;
    - $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
    - $F(\langle v^r, v_1\rangle) = \emptyset$;
    - $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
    - $F(\langle v^r, v_2\rangle) = \emptyset$.
12) $\tau_{12} = \{V_2, E_2, v^r, T, F\}$ with :
    - $T(v^r) = \{\ldots\}$;
    - $F(v^r) = \{\ldots\}$;
    - $T(v_1) = \{\mathsf{Table}\}$;
    - $F(v_1) = \emptyset$;
    - $T(v_2) = \{\mathsf{WiFi}\}$;
    - $F(v_2) = \emptyset$;
    - $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
    - $F(\langle v^r, v_1\rangle) = \emptyset$;
    - $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
    - $F(\langle v^r, v_2\rangle) = \emptyset$.
13) $\tau_{13} = \{V_2, E_2, v^r, T, F\}$ with :
    - $T(v^r) = \{\ldots\}$;
    - $F(v^r) = \{\ldots\}$;
    - $T(v_1) = \{\mathsf{Desk}\}$;
    - $F(v_1) = \emptyset$;
    - $T(v_2) = \{\mathsf{ADSL}\}$;
    - $F(v_2) = \emptyset$;
    - $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
    - $F(\langle v^r, v_1\rangle) = \emptyset$;
    - $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
    - $F(\langle v^r, v_2\rangle) = \emptyset$.
14) $\tau_{14} = \{V_2, E_2, v^r, T, F\}$ with :
    - $T(v^r) = \{\ldots\}$;
    - $F(v^r) = \{\ldots\}$;
    - $T(v_1) = \{\mathsf{Table}\}$;
    - $F(v_1) = \emptyset$;
    - $T(v_2) = \{\mathsf{ADSL}\}$;
    - $F(v_2) = \emptyset$;
    - $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
    - $F(\langle v^r, v_1\rangle) = \emptyset$;
    - $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
    - $F(\langle v^r, v_2\rangle) = \emptyset$.
15) $\tau_{15} = \{V_2, E_2, v^r, T, F\}$ (CLASH) with :
    - $T(v^r) = \{\ldots\}$;
    - $F(v^r) = \{\ldots, \exists hasEquipment.\mathsf{Desk}\}$,

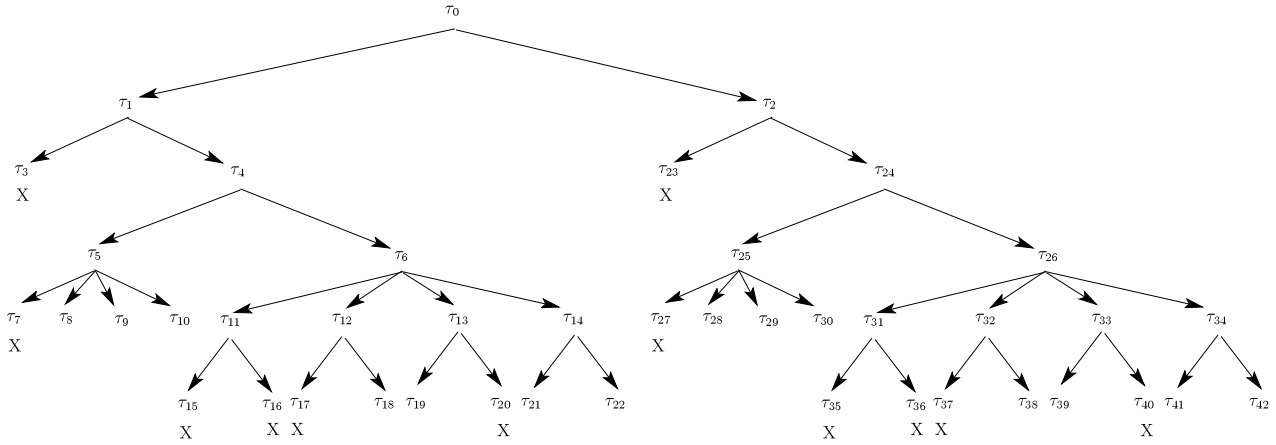

Fig. 2. Transformation of tableaux

- $T(v_1) = \{\mathsf{Desk}\}$;
- $F(v_1) = \{\mathsf{Desk}\}$;
- $T(v_2) = \{\mathsf{WiFi}\}$;
- $F(v_2) = \emptyset$;
- $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
- $F(\langle v^r, v_1\rangle) = \emptyset$;
- $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
- $F(\langle v^r, v_2\rangle) = \emptyset$.

16) $\tau_{16} = \{V_2, E_2, v^r, T, F\}$ (CLASH) with :
- $T(v^r) = \{\dots\}$;
- $F(v^r) = \{\dots, \exists hasInternet.\mathsf{WiFi}\}$;
- $T(v_1) = \{\mathsf{Desk}\}$;
- $F(v_1) = \emptyset$;
- $T(v_2) = \{\mathsf{WiFi}\}$;
- $F(v_2) = \{\mathsf{WiFi}\}$;
- $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
- $F(\langle v^r, v_1\rangle) = \emptyset$;
- $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
- $F(\langle v^r, v_2\rangle) = \emptyset$.

17) $\tau_{17} = \{V_2, E_2, v^r, T, F\}$ (CLASH) with :
- $T(v^r) = \{\dots\}$;
- $F(v^r) = \{\dots, \exists hasInternet.\mathsf{WiFi}\}$;
- $T(v_1) = \{\mathsf{Table}\}$;
- $F(v_1) = \emptyset$;
- $T(v_2) = \{\mathsf{WiFi}\}$;
- $F(v_2) = \{\mathsf{WiFi}\}$;
- $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$
- ; $F(\langle v^r, v_1\rangle) = \emptyset$;
- $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
- $F(\langle v^r, v_2\rangle) = \emptyset$.

18) $\tau_{18} = \{V_2, E_2, v^r, T, F\}$ with :
- $T(v^r) = \{\dots\}$;
- $F(v^r) = \{\dots, \exists hasEquipment.\mathsf{Desk}\}$;
- $T(v_1) = \{\mathsf{Table}\}$;
- $F(v_1) = \{\mathsf{Desk}\}$;
- $T(v_2) = \{\mathsf{WiFi}\}$;
- $F(v_2) = \emptyset$;
- $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
- $F(\langle v^r, v_1\rangle) = \emptyset$;
- $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
- $F(\langle v^r, v_2\rangle) = \emptyset$.

19) $\tau_{19} = \{V_2, E_2, v^r, T, F\}$ with :
- $T(v^r) = \{\dots\}$;
- $F(v^r) = \{\dots, \exists hasInternet.\mathsf{WiFi}\}$;
- $T(v_1) = \{\mathsf{Desk}\}$;
- $F(v_1) = \emptyset$;
- $T(v_2) = \{\mathsf{ADSL}\}$;
- $F(v_2) = \{\mathsf{WiFi}\}$;
- $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
- $F(\langle v^r, v_1\rangle) = \emptyset$;
- $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
- $F(\langle v^r, v_2\rangle) = \emptyset$.

20) $\tau_{20} = \{V_2, E_2, v^r, T, F\}$ (CLASH) with :
- $T(v^r) = \{\dots\}$;
- $F(v^r) = \{\dots, \exists hasEquipment.\mathsf{Desk}\}$;
- $T(v_1) = \{\mathsf{Desk}\}$;
- $F(v_1) = \{\mathsf{Desk}\}$;
- $T(v_2) = \{\mathsf{ADSL}\}$;
- $F(v_2) = \emptyset$;
- $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
- $F(\langle v^r, v_1\rangle) = \emptyset$;
- $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
- $F(\langle v^r, v_2\rangle) = \emptyset$.

21) $\tau_{21} = \{V_2, E_2, v^r, T, F\}$ with :
- $T(v^r) = \{\dots\}$;
- $F(v^r) = \{\dots, \exists hasEquipment.\mathsf{Desk}\}$;
- $T(v_1) = \{\mathsf{Table}\}$;
- $F(v_1) = \{\mathsf{Desk}\}$;
- $T(v_2) = \{\mathsf{ADSL}\}$;
- $F(v_2) = \emptyset$;
- $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
- $F(\langle v^r, v_1\rangle) = \emptyset$;
- $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
- $F(\langle v^r, v_2\rangle) = \emptyset$.

22) $\tau_{22} = \{V_2, E_2, v^r, T, F\}$ with :
- $T(v^r) = \{\dots\}$;
- $F(v^r) = \{\dots, \exists hasInternet.\mathsf{WiFi}\}$;
- $T(v_1) = \{\mathsf{Table}\}$;
- $F(v_1) = \emptyset$;

- $T(v_2) = \{\mathsf{ADSL}\}$;
- $F(v_2) = \{\mathsf{WiFi}\}$;
- $T(\langle v^r, v_1\rangle) = \{hasEquipment\}$;
- $F(\langle v^r, v_1\rangle) = \emptyset$;
- $T(\langle v^r, v_2\rangle) = \{hasInternet\}$;
- $F(\langle v^r, v_2\rangle) = \emptyset$.

In the very similar manner, τ_2, τ_{23} , ..., τ_{42} are developed on the second branch.