Comenius university in bratislava

Faculty of Mathematics, Physics and Informatics

ABDUCTION SOLVER BASED ON HIGHLY EFFICIENT C++ DL REASONER

Master thesis

2018

Bc. Drahomír Mrózek

Comenius university in bratislava

Faculty of Mathematics, Physics and Informatics

ABDUCTION SOLVER BASED ON HIGHLY EFFICIENT C++ DL REASONER

Master thesis

Study programme:	Applied informatics
Field of study:	2511 Aplikovaná informatika
Study department:	Department of Applied Informatics
Advisor:	RNDr. Martin Homola, PhD.
Consultant:	Mgr. Júlia Pukancová

Bratislava, 2017 Bc. Drahomír Mrózek

todo:zadanie

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa.

....

Bratislava, 2017 Bc. Drahomír Mrózek

Acknowledgments

TODO

Abstract

This work will start from an existing solution of an abduction solver based on Pellet reasoner for description logics, implemented in Java. The implementation in C++ opens different ways for improvement and effectivization, starting with the utilization of a more effective C++ inference system for description logics.
Keywords: abduction, description logic

Abstrakt

Práca bude vychádzať z existujúceho riešenia abduktívneho systému založeného na reasoneri pre deskripčné logiky Pellet implementovaného v Jave. Pri implementácii v C++ sa otvárajú možnosti na zlepšenie a zefektívnenie existujúceho návrhu abduktívneho solvera, počnúc využitím efektívnejšieho C++ inferenčného systému pre deskripčné logiky.

Kľúčové slová: abdukcia, deskripčná logika

Contents

Chapter 1

Introduction

Story No. 1 TODO Úvod, troska kontextu, asi na 1,5 strany Cieľom práce je návrh a vývoj abduktívneho systému pre deskripčné logiky založený na existujúcom reasoneri s dôrazom na optimalizačné techniky a efektívnosť.

Chapter 2

Description logic

2.1 Introduction to description logic

This chapter is based on Handbook on ontologies [7] and Handbook of knowledge representation [8].

The word “ontology” is used with different meanings in different communities.
In philosophy, Aristotle in his Metaphysics defined **Ontology** as the study of attributes that belong to things because of their very nature.
Ontology focuses on the nature and structure of things independently of any further considerations, and even independently of their actual existence.
For example, it makes perfect sense to study the Ontology of unicorns and other fictitious entities: although they do not have actual existence, their nature and structure can be described in terms of general categories and relations.

in Computer Science, we refer to an **ontology** as a special kind of information object or computational artifact. Computational ontologies are a means to formally model the structure of a system, that is the relevant entities and relations that emerge from its observation, and which are useful to our purposes. An example of such a system can be a company with all its employees and their interrelationships. The ontology engineer analyzes relevant entities and organizes them into concepts and relations, being represented, respectively, by unary and binary predicates.

Description logics (**DLs**) are a family of knowledge representation languages that can be used to represent an ontology in a structured and formally well-understood way. The ”description” part of their name is based on how the important notions of the domain are described by concept descriptions (unary predicates) and atomic roles (binary predicates). The ”logic” part comes from their formal, logic-based semantics, unlike some other methods of representation of ontologies, for example semantic networks. In **DLs**, ontology is represented by a knowledge base.

Definition 2.1.1 *(Knowledge base) A Knowledge base is an ordered pair $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is the terminological part (**TBox**) and $\mathcal{A}$ is the assertional part (**ABox**).*

Definition 2.1.2 *(GCI, TBox) The terminological part of a knowledge base, **Tbox**, is a finite set of general concept inclusions. A general concept inclusion (**GCI**) is a statement of the form $A \sqsubseteq B$, where A and B are concepts.*
*The definition of a concept varies depending on the **DL**, for a definition in ALC, see 2.2.1.*

Example 2.1.1 *(GCI) Examples of **GCIs** in description logic ALC, and their equivalents in natural language (note that conventionally, concept names start with an uppercase letter, while individual and role names start with a* *lowercase letter) :*

- $Human \sqsubseteq \forall hasChild.Human$

  "If someone is a human, then all their children are also human"

- $Human \sqcap \neg Female \sqcap (\exists married.\top) \sqcap (\forall hasChild.(Doctor \sqcup Professor)) \sqsubseteq HappyMan$

  "A human who is not female, is married, and every child he has is either a doctor or a proffesor, is a happy man."

  Note that $\top$ is a special concept, meaning "everything" - every individual is an instance of this concept. Similarily, there is a special concept $\bot$, or "nothing" - no individual is an instance of this concept.

- $Painter \equiv \exists hasPainted.Picture$

  $A \equiv B$ is a shorter version of $(A \sqsubseteq B) \wedge (B \sqsubseteq A)$, so this statement means "If someone has painted a picture, they are a painter, and if someone is a painter, they have painted a picture".

Definition 2.1.3 *(ABoxEx)* ***ABox*** *consists of 2 types of statements about individuals : concept assertions and role assertions.*
Concept assertions have the form $a : C$ where a is an individual and C is a concept.
Role assertions have the form $a, b : R$ where a,b are individuals and R is a role.

Example 2.1.2 *Some examples of ABox statements, with no natural language translations as their meaning is obvious:*

- $Alice : Human$

- $michelangelo : Painter \sqcap Sculptor$
- $daVinci, monaLisa : hasPainted$

Interpretation of description logics is done using sets. Although formal definitions vary based on the type of description logic, the general idea is that we create a set of objects, assign one object to every individual, interpret concepts as sets of objects and roles as sets of ordered doubles of objects. A formal definition can be seen in the next section, 2.2.2.

2.2 Selected description logics

2.2.1 $\mathcal{ALC}$

In this thesis, we will be using a widely used description logic $\mathcal{ALC}$ and it's extensions. Although $\mathcal{ALC}$ stands for "Attributive concept Language with Complements". It's one of the less expressive languages, for example, it can't express the concept " someone who has 2 children ". You can see examples of TBox (2.1.1) and ABox (**??**) statements in $\mathcal{ALC}$ in the previous section, now it is time for formal definitions.

Definition 2.2.1 *[8] (Syntax of $\mathcal{ALC}$ concepts and roles). Let N_C be a set of concept names and N_R be a set of role names. The set of Concepts is the smallest set such that*

1. *$\top, \bot$, and every concept name $A \in N_C$ is an Concept,*
2. *If C and D are Concepts and $r \in N_R$, then $C \sqcap D$, $C \sqcup D, \neg C, \forall r.C$, and $\exists r.C$ are Concepts.*

The set of roles is the set of role names N_R. (Note: this is not true for some simple extensions of $\mathcal{ALC}$, for example $\mathcal{ALC}_{(\cap)}$ - $\mathcal{ALC}$ with role intersection)

The semantics of $\mathcal{ALC}$ (and of DLs in general) are given as interpretations.

Definition 2.2.2 *[8] ($\mathcal{ALC}$ semantics). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty set $\Delta^{\mathcal{I}}$, called the domain of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ that maps every $\mathcal{ALC}$ Concept to a subset of $\Delta^{\mathcal{I}}$, and every $\mathcal{ALC}$ role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that, for all $\mathcal{ALC}$ Concepts C, D and all role names r:*

$$\begin{array}{ll}
\top^{\mathcal{I}} = \Delta^{\mathcal{I}} & \bot^{\mathcal{I}} = \emptyset, \\
(C \sqcap D)^{\mathcal{I}} & = C^{\mathcal{I}} \cap D^{\mathcal{I}}, \\
(C \sqcup D)^{\mathcal{I}} & = C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
\neg C^{\mathcal{I}} & = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}, \\
(\exists r.C)^{\mathcal{I}} & = \{x \in \Delta^{\mathcal{I}} | \exists y \in \Delta^{\mathcal{I}} \text{ with } \langle x, y \rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}, \\
(\forall r.C)^{\mathcal{I}} & = \{x \in \Delta^{\mathcal{I}} | \forall y \in \Delta^{\mathcal{I}}, \text{ if } \langle x, y \rangle \in r^{\mathcal{I}}, \text{ then } y \in C^{\mathcal{I}}\}.
\end{array}$$

You may notice that an interpretation does not have to comply with any TBox or ABox assertions. If some assertion A is "true" for an interpretation $\mathcal{I}$, we say that $\mathcal{I}$ **satisfies** A.

Definition 2.2.3 *(satisfiability) Given an interpretation $\mathcal{I}$:*

- *A GCI $A \sqsubseteq B$ is satisfied by $\mathcal{I}$ iff $A^{\mathcal{I}} \subset B^{\mathcal{I}}$.*
- *A concept assertion $a : C$ is satisfied by $\mathcal{I}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$.*
- *A role assertion $a, b : R$ is satisfied by $\mathcal{I}$ iff $\langle a^m athcalI, b^m athcalI \rangle \in R^{\mathcal{I}}$*

An assertion is satisfiable iff an interpretation exists that satisfies it.

Definition 2.2.4 *(model, consistency) An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K}$ if it satisfies every assertion in the TBox and ABox of $\mathcal{K}$. A knowledge base is consistent if at least one model of it exists.*

An ontology can have multiple models, some less intuitive than other.

Example 2.2.1 *(model)*

Let there be a knowledge base $\mathcal{K} = \{\mathcal{T}, \mathcal{A}\}$, $\mathcal{T} = \{$

$A \sqsubseteq B$

$A \sqsubseteq \exists r.C$

$C \sqsubseteq \forall r.D$

$\}$

$\mathcal{A} = \{$

$a : A$

$c,d : r$

$\}$

One possible model of $\mathcal{K}$ *would be* $\mathcal{M}_1 = \{$

$\Delta^{\mathcal{I}} = \{\ a_x, c_x, d_x\ \}$

$a^{\mathcal{I}} = a_x, c^{\mathcal{I}} = c_x, d^{\mathcal{I}} = d_x$

$A^{\mathcal{I}} = \{a_x\}, B^{\mathcal{I}} = \{a_x\}, C^{\mathcal{I}} = \{c_x\}, D^{\mathcal{I}} = \{d_x\}$

$r^{\mathcal{I}} = \{\langle a_x, b_x\rangle, \langle a_x, c_X\rangle, \langle c_x, d_x\rangle\}$

$\}$

But other, less intuitive models also exist, for example $\mathcal{M}_2$ *and* $\mathcal{M}_3$. $\mathcal{M}_2 = \{$

$\Delta^{\mathcal{I}} = \{\ a_x, c_x, d_x, c_n\ \}$

$a^{\mathcal{I}} = a_x, c^{\mathcal{I}} = c_x, d^{\mathcal{I}} = d_x$

$A^{\mathcal{I}} = \{a_x\}, B^{\mathcal{I}} = \{a_x\}, C^{\mathcal{I}} = \{c_n\}, D^{\mathcal{I}} = \{\}$

$r^{\mathcal{I}} = \{\langle a_x, b_x\rangle, \langle a_x, c_n\rangle, \langle c_x, d_x\rangle\}\ \}$

$\mathcal{M}_3 = \{$

$\Delta^{\mathcal{I}} = \{i_x\}$

$a^{\mathcal{I}} = i_x, c^{\mathcal{I}} = i_x, d^{\mathcal{I}} = i_x$

$A^{\mathcal{I}} = B^{\mathcal{I}} = C^{\mathcal{I}} = D^{\mathcal{I}} = \{i_x\}$

$r^{\mathcal{I}} = \{\langle i_x, i_x \rangle\}$

$\}$

2.2.2 $\mathcal{SHIQ}$

A new description logic can be defined by adding some syntax to an existing description logic, increasing it's expressibility. As such, description logics are usually named as a list of abbreviations (TODO: abbreviations? acronyms? nenapada ma ako preformulovat tuto vetu) of the allowed expressions.
The language $\mathcal{SHIQ}$ is one of the more expressive description logics. $\mathcal{S}$ is an abbreviation of "$\mathcal{ALC}$ with transitive roles" - meaning we can use any syntax from $\mathcal{ALC}$, and also declare a role to be transitive. $\mathcal{R}$ stands for "role hierarchy", meaning we can define a role to be a subrole of another role. $\mathcal{I}$ stands for "inverse roles" (TODO spytat sa: wiki vravi "inverse properties", ale mne sa zda ze property je OWL termin a role je DL termin, a teraz vravime o DL a preto vraviet o properties bez definovania by bolo metuce) , meaning for every role R, there is an inverse role R^{-}. Lastly, $\mathcal{Q}$ stand for "qualified cardinality restriction"which allows us to specify the number of objects an object relates to by a role, for example we can say that a painter is someone who has painted at least 10 paintings.

Example 2.2.2 *Some examples of TBox statements that can be made in $\mathcal{SHIQ}$ but not in $\mathcal{ALC}$:*

- $Painter \sqsubseteq Human \sqcap \geq 10 hasPainted.Picture$
- $Painting \sqsubseteq Picture \sqcap \exists hasPainted^{-}.\top$
  "A Painting is a picture that was painted."
- $isParent \sqsubseteq isAncestor$

Meaning isParent is a subrole of hasAncestor - if someone is a parent of someone, they are also their ancestor.

- $transitive(isAncestor)$
  The role isAncestor is transitive - if A is an ancestor of B and B is an ancestor of C, then A is an ancestor of C.

Note that the last 2 examples introduce statements that are not general concept inclusion to the TBox. For this reason, some authors prefer defining a third part of the knowledge base, RBox, consisting of statements about roles.

The definitions of syntax and semantics of $\mathcal{SHIQ}$ are similar to those of $\mathcal{ALC}$.

Definition 2.2.5 *[7] ($\mathcal{SHIQ}$ concept and role syntax) Let R be a set of role names, which is partitioned into a set R_+ of transitive roles and a set R_p of normal roles. The set of all $\mathcal{SHIQ}$ roles is $R \cup \{r^- | r \in R\}$, where r^- is called the inverse of the role r.*
Let N_C be a set of concept names. The set of $\mathcal{SHIQ}$ concepts is the smallest set such that:

1. *every concept $A \in N_C$ is a $\mathcal{SHIQ}$ concept.*
2. *if A and B are $\mathcal{SHIQ}$ concepts and r is a $\mathcal{SHIQ}$ role, then $A \sqcap B, A \sqcup B, \neg A, \forall r.A, and \exists r.A$ are $\mathcal{SHIQ}$ concepts.*
3. *if A is a $\mathcal{SHIQ}$ concept and r is a simple $\mathcal{SHIQ}$ role (simple role is neither transitive nor has a transitive subrole), and $n \in \mathbb{N}$, then $(\leq nr.A)$ and $(\geq nr.A)$ are $\mathcal{SHIQ}$ concepts.*

$\mathcal{SHIQ}$ semantics can be described as $\mathcal{ALC}$ semantics with same additions.

Definition 2.2.6 *($\mathcal{SHIQ}$ semantics) [7] in addition to definition* **??**, *for all $p \in R$ and $r \in R_+$:*
$\langle x, y\rangle \in p^{\mathcal{I}}$ *iff* $\langle y, x\rangle \in (p^-)^{\mathcal{I}}$.
if $\langle x, y\rangle \in r^{\mathcal{I}}$ *and* $\langle y, z\rangle \in r^{\mathcal{I}}$ *then* $\langle x, z\rangle \in r^{\mathcal{I}}$.
$(\leq nr.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \#r^{\mathcal{I}}(x, C) \leq n\}$,
$(\geq nr.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \#r^{\mathcal{I}}(x, C) \geq n\}$,

where $\#M$ denotes the cardinality of the set M, and $r^{\mathcal{I}}(x, C) := \{y | \langle x, y\rangle \in r^{\mathcal{I}}$ *and* $y \in C^{\mathcal{I}}\}$.

$\mathcal{SHIQ}$ ABox and it's model are the same as in $\mathcal{ALC}$ (definition **??**). For the TBox, we have to add role subsumption axioms. Some authors define TBox as only containing GCIs, and use a new structure, RBox, to contain role inclusions. For this thesis, role inclusion axioms are a part of TBox.

Definition 2.2.7 *[7] ($\mathcal{SHIQ}$ TBox)*
A role inclusion axiom is of the form $r \sqsubseteq s$, where r,s are roles. An interpretation $\mathcal{I}$ is a model of a role inclusion axiom $r \sqsubseteq s$ if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$.
$\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it it is a model of every role inclusion axiom and GCI axiom (definition **??***) in TBox.*

2.2.3 $\mathcal{ALCHO}$

$\mathcal{ALC}$ with role hierarchy and Nominals.
Nominals are concepts with exacly one specific instance. For example, {john} is a concept with its only instance being the individual 'john'.
Nominals can be used to express enumerations, for example [3]:
$Beatle \equiv \{john\} \sqcup \{paul\} \sqcup \{george\} \sqcup \{ringo\}$

Role hirerarchy was already described in section 2.2.2 (example **??**).

Definition 2.2.8 *(Syntax and semantics of $\mathcal{ALCHO}$)*
Syntax of $\mathcal{ALCHO}$ is the syntax of $\mathcal{ALC}$ (definition 2.2.1), with {a} added to the set of concepts C for each individual 'a'.
Similarly, semantics of $\mathcal{ALCHO}$ are the semantics of $\mathcal{ALC}$ (definition 2.2.2) with the addition of $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$,
where 'a' is an individual and $\mathcal{I}$ is the interpretation.

In $\mathcal{ALCHO}$, the definition of an ABox is the same for $\mathcal{ALC}$ (definition **??**) and the definition of a TBox is the same as for $\mathcal{SHIQ}$ (definition 2.2.7.

2.2.4 $\mathcal{SROIQV}(\mathcal{D})$

Jazyk koncludu, TODO asi ked bude hotova implementacia.

2.3 Tableau algorithm

In our algorithm, we heavily make use of tableau algorithm. Tableau algorithm is a method of constructing a model of a knowledge base $\mathcal{K}$ if $\mathcal{K}$ is consistent, and stops if no model of $\mathcal{K}$ exists and thererfore $\mathcal{K}$ is inconsistent.

Tableau algorithm uses knowledge base in negation normal form (NNF), that is, every concept complement $\neg$ applies only to a concept name [8]. Any $\mathcal{ALC}$ concept can be transformed to an equivalent concept in NNF by using de Morgan's laws and the duality between existential and universal restrictions ($\neg\exists r.C \equiv \forall r.\neg C$). For example, the concept $\neg(\exists r.A \sqcap \forall s.B)$, where A, B are concept names, can be transformed using de Morgan's laws to $\neg\exists r.C \sqcup \neg\forall s.B$, and this can then be transformed using the existential-universal duality into $(\forall r.\neg A) \sqcup (\exists s.\neg B)$.

The idea behind the tableau algorithm for $\mathcal{K} = \{\mathcal{T}, \mathcal{A}\}$is to start with the concrete situation described in $\mathcal{A}$ and expand based on what can be inferred from $\mathcal{T}$ and currently known ABox statements. This is done using something called completion graph, which is a graph where nodes represent individuals, directed edges represent relations between individuals, each node has a label containing concepts the individual belongs to, and each edge has a label consisting of the names of it's roles .

Definition 2.3.1 *(Completion graph)*
A completion graph is a pair $(G, \mathcal{L})$, where G is a directed finite graph and $\mathcal{L}$ is a labeling function mapping each node from G to a set of concepts, and each edge to a set of roles.

Tableau algorithm for $\mathcal{K}(\mathcal{T}, \mathcal{A})$ first creates a completion graph based on $\mathcal{A}$, and then expands it using tableau expansion rules.

Definition 2.3.2 *[8] ($\mathcal{ALC}$ tableau expansion rules)*

- $\sqcap$*-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not blocked (definition 2.3.4), and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$*

- $\sqcup$*-rule: if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not blocked, and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$, then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$, for some $C \in \{C_1, C_2\}$*

- $\exists$*-rule: if $\exists r.C \in \mathcal{L}(x)$, x is not blocked, and x has no r-successor y with $C \in \mathcal{L}(y)$, then create a new node y with $\mathcal{L}(\langle x, y\rangle) = \{r\}$ and $\mathcal{L}(y) = \{C\}$.*

- *$\forall$-rule: if $\forall r.C \in \mathcal{L}(x)$, x is not blocked, and there is an r-successor y of x with $C \notin \mathcal{L}(y)$ then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$.*

- *$\sqsubseteq$-rule: if $C_1 \sqsubseteq C_2 \in \mathcal{T}$, x is not blocked, and $C_2 \sqcup$NNF($\neg C_1$) $\notin \mathcal{L}(x)$, then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_2 \sqcup$ NNF($\neg C_1$) $\}$*

Where C, C_1, C_2 are concepts, r is role, $\mathcal{T}$ is TBox, and NNF is normal negation form.

Why are we checking whether nodes are blocked? And what are blocked nodes?
If we removed ”x is not blocked” from all expansion rules, then the algorithm could generate an infinite graph using the $\exists$-rule, for example for $\mathcal{K} = (\mathcal{T}, \mathcal{A}), \mathcal{T} = \{C \sqsubseteq \exists r.C\}, \mathcal{A} = \{a : C\}$ the algorithm would create a node a_2 with $\langle a, a_2 \rangle : r$, then a node a_3 with $\langle a_2, a_3 \rangle : r$ and so on. To guarantee termination, we define node blocking.

Definition 2.3.3 *(ancestor) In completion graph $\mathcal{CG} = (G, \mathcal{L}), G = (N, E)$, nodes $x, y \in N$, and edge $\langle x, y \rangle \in E$, then y is a successor of x and x is an ancestor of y.*
If $\mathcal{L}(\langle x, y \rangle) = r$, then y is an r-successor of x.
Ancestry is transitive (if node x is an ancestor of node y and y is an an ancestor of node z, then x is is an ancestor of z).

Definition 2.3.4 *(blocking) A node x is blocked if it has an ancestor y that is either blocked, or $\mathcal{L}(x) \subseteq \mathcal{L}(y)$.*

And now we can show the pseudocode of (nondeterministic) tableau algorithm.

```
function TableauAlg( Knowledge base $\mathcal{K}$ =$(\mathcal{T}, \mathcal{A})$)
    Convert $\mathcal{T}$ to NNF.
    Completion graph $\mathcal{CG} = (G, \mathcal{L}), G = (N, E), N = E = \mathcal{L} = \emptyset$
    for Each individual $i \in \mathcal{A}$ do
        $N = N \cup n_i$
        for Each assertional axiom $(i : C) \in \mathcal{A}$ do
            $\mathcal{L}(n_i) = \mathcal{L}(n_i) \cup C$
        end for
        for Each assertional axiom $(i, x : r) \in \mathcal{A}$ do
            $E = E \cup \langle n_i, n_x \rangle$
            $\mathcal{L}(\langle n_i, n_x \rangle) = \mathcal{L}(\langle n_i, n_x \rangle) \cup r$
        end for
    end for
    while A tableau expansion rule can be applied on $\mathcal{CG}$ do
        Apply a rule on $CG$
        if Clash exists in $\mathcal{CG}$ then
            return "$\mathcal{K}$ is inconsistent" (assuming algorithm undeterministically always picks the 'correct' decision in $\sqcup$-rule if one exists)
        end if
    end while
    return "$\mathcal{K}$ is consistent"
end function
```

Definition 2.3.5 *(clash)*

In completion graph $\mathcal{CG}$=$((N, E), \mathcal{L})$ *, if there is a node* $n \in N$ *and a concept* C *where* $\{C, \neg C\} \in \mathcal{L}(n)$*, then* $\mathcal{CG}$ *has a clash.*

If the algorithm cannot apply any expansion rules, then $\mathcal{K}$ is consistent and $\mathcal{CG}$ is its model, or a finite part of an infinite model due to blocking. Our abduction algorithm only needs this finite part.

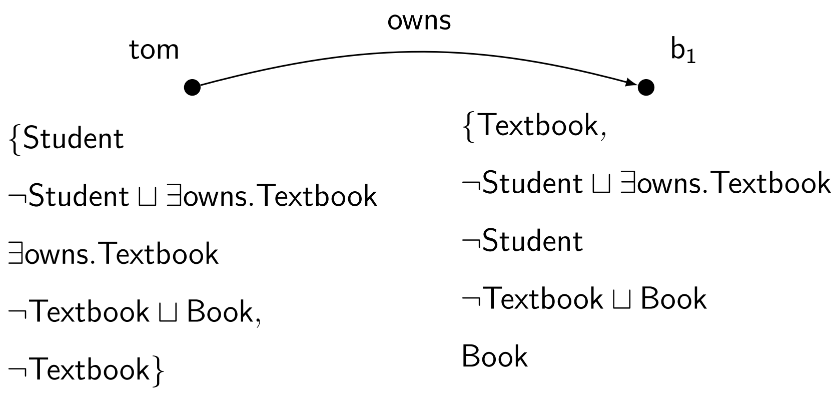
The tableau algorithm has 2 sources of nondeterminism : picking which rule to apply, and how to apply the $\sqcup$-rule. The order of rule applications doesn't matter for the sake of determining consistency, but the application of the $\sqcup$-rule does, as one choice can lead to a clash while the other not.

If a clash happens in a deterministic implementation, we can backtrack to

a previous $\sqcup$ decision with an unexplored choice and continue from there. If there are no unexplored $\sqcup$ decisions and we find a clash, $\mathcal{K}$ is inconsistent (no model exists). If there is no decision with an unexplored possibility we can backtrack to, we can declare $\mathcal{K}$ to be inconsistent. I think it would be useful to show an example of tableau completion graph:
obrazok odstraneny, nejake problemy s hboxom a nemoznostou lomit riadok.

Example 2.3.1

$$\mathcal{K} = \left\{ \begin{array}{c} \mathsf{Student} \sqsubseteq \exists \mathsf{owns.Textbook} \\ \mathsf{Textbook} \sqsubseteq \mathsf{Book} \\ \mathsf{tom}: \mathsf{Student} \end{array} \right\}$$



In this case, the tableau algorithm stops without a clash, therefore the knowledge base is consistent.

Chapter 3

Abduction

Logical thinking can be divided into 3 categories: **Deduction**, **Induction** and **Abduction**.

Deduction is the process of using known causes and rules to infer the results of a rule. For example, if we know the fact that a floor is wet, and the rule that this particular floor is slippery when wet, we can arrive at a conclusion that the floor must be slippery. Deduction is the only one of the 3 types of logical reasoning where if the premises are true and the rules used are true, then the conclusion must also be true.
Induction is the process of knowing the cause and effect and creating a plausible rule. For example, we know that whenever we've seen this floor wet, it was also slippery, therefore, the floor is slippery when wet. Unlike deduction, induction the result of induction isn't necessarily correct.
Abduction, also known as hypothesis or diagnosis, is when we know the rules and observe an unexpected effect, and try to find an explanation for it - a plausible cause. For example, we if we see someone slip on the floor and know that the floor is slippery when wet, we may assume that the floor

is wet. Another example of abductive reasoning is when a doctor observes patients symptoms, knows which diseases cause which symptoms, and based on these rules guess which disease the patient may have. Like with induction, there are often multiple plausible explanation, and the true explanation may not be among them even if all the information we have is correct; the person who slipped on the floor may have slippery shoes, or the patient could have a new diease unknown to the doctor.

3.1 Abduction in description logic

If the only condition for $\mathcal{E}$ being an abductive explanation for observation O in knowledge base $\mathcal{K}$ was that $\mathcal{K} \cup \mathcal{E} \models O$, we would reach some unintuitive explanations, for example for O="The floor is slippery" and $\mathcal{K}$ = "The floor is slippery when wet" , $\mathcal{E}$ could be "The floor is slippery" (as in first order logic $A \rightarrow A$) or "The sun is shining and the sun is not shining" ($A \cap \neg A \models B$, since $X \models Y$ if there is no model of $X \cup \neg Y$) or "The floor is wet and a bird is chirping" (irrelevant information) . To limit ourselves only to more usefull explanations, we will also add some restrictions to our definition of logical abduction for the purpose of our work:

Definition 3.1.1 *(Abductive explanation) [1]*
Set of axioms $\mathcal{E}$ is an explanation of observation O in knowledge base $\mathcal{K}$, iff:

- $\mathcal{K} \cup \mathcal{E} \models O$
- *Consistency :* $\mathcal{K} \cup \mathcal{E} \not\models \bot$
- *Relevance :* $\mathcal{E} \not\models O$
- *Explanatoriness :* $\mathcal{K} \not\models O$

- *Minimality: There is no other explanation $\mathcal{F}$ of O in $\mathcal{K}$, where $\mathcal{F} \subset \mathcal{E}$*

Elsenbroich et. al. [1] defined several types of abduction in description logig : Concept abduction, TBox abduction, ABox abduction and Knowledge base abduction. We are mainly interested in the ABox abduction, but there we will also mention shortly describe.

Concept abduction is the simplest of the 4: for a knowledge base $\mathcal{K}$ and the observation - concept $\mathcal{C}$, we are looking for a concept $\mathcal{D}$ so that $\mathcal{K} \models \mathcal{D} \sqsubseteq \mathcal{C}$.

TBox abduction is the process of finding a finite set of TBox rules $\mathcal{S}_t$ for knowledge base $\mathcal{K}$ and concepts $\mathcal{C}, \mathcal{D}$ satisfiable w.r.t. $\mathcal{K}$, so that $\mathcal{K} \cup \mathcal{S}_t \models \mathcal{C} \sqsubseteq \mathcal{D}$. This type of abduction is used for ontology debugging : If an ontology engineer expects $\mathcal{C} \sqsubseteq \mathcal{D}$ to follow from $\mathcal{K}$ but finds it doesn't, they can look for TBox rules to add to $\mathcal{K}$. On the other hand, if an undesirable $\mathcal{C} \sqsubseteq \mathcal{D}$ follows from the ontology, they can find such $\mathcal{S}_t$ from $\mathcal{K}$ so that $\mathcal{K} - \mathcal{S}_t \not\models \mathcal{C} \sqsubseteq \mathcal{D}$.

Knowledge base abduction is an abduction where both the observation and the explanation can contain both TBox and ABox axioms. (TODO: najst sposob ako povedat ze sme nenasli algoritmus alebo aplikaciu)

3.1.1 ABox abduction

In ABox abduction, we are looking for an explanation $\mathcal{E}$ of observation O in knowledge base $\mathcal{K}$ according to rules 3.1.1, where both $\mathcal{E}$ abd O are limited to ABox axioms.

For example, for a simple ontology where the Abox of the knowledge base is empty and the Tbox=$\{\mathcal{A} \sqsubseteq \mathcal{C}; \mathcal{B} \sqsubseteq \mathcal{C}\}$, and an observation Abox=$\{a : \mathcal{C}\}$, there are explanations: $\{\{a : \mathcal{A}\}, \{a : \mathcal{B}\}\}$. $\{a : \mathcal{C}\}$ is not an explanation due to the relevancy requirement, $\{a : \mathcal{A}, a : \mathcal{B}\}$ is not and explanation due to the minimality requirement.

TODO: relevancy testing in ABox abduction without using reasoner (dat to sem alebo do implementacie?)

TODO uses: automatic planning, medical (paper)

TODO: Abduction as set cover, abduction as probability

3.2 Uses

TODO: Medical, automatic code checking, automatic planning, any type of diagnosis

Chapter 4

Previous abduction solvers

4.1 theoretical approaches

TODO: abdukcia v FOL TODO: bol nejaky pokus o abdukciu v deskripcnej logike nezalozeny na hitting setoch? - ten co som nasiel na mobile by sa mozno dal spomenut, nepouziva termin "hitting set", pouziva substitucie na uzatvaranie tableaux s dvojitymi labelmi ‘

4.2 implemented solutions

TODO: Racer

Chapter 5

Our approach

The approach most often used to perform abductive reasoning in description logic without translating to other formal logics is based on the following approach: Let there be a knowledge base $\mathcal{K}$, observation O, and a set of axioms $\mathcal{S}$. If $\mathcal{K} \sqcup \mathcal{S}$ is consistent, then by definition there is at least one model of $\mathcal{K} \sqcup \mathcal{S}$. If $\mathcal{K} \sqcup \mathcal{S} \sqcup \neg O$ is inconsistent, then there is no model for $\mathcal{K} \sqcup \mathcal{S}$ where $\neg O$ is not true, that is every model of $\mathcal{K} \sqcup \mathcal{S}$ contains O, so $\mathcal{K} \sqcup \mathcal{S} \models O$.

We can find such a set $\mathcal{S}$ by generating every model of $\mathcal{K} \sqcup \neg O$, and picking a set of complements of axiom in these models so that every model has at least one axiom complement in $\mathcal{S}$. This can be formulated as the hitting set (defined below 5.0.1) problem - for each model in the set of models M of $\mathcal{K} \sqcup \neg O$, we create an **antimodel** (5.0.2)consisting of negations of every axiom in the model, the set of these antimodels we call M'. Our goal is finding a minimal (inclusion-wise) set $\mathcal{S}$ containing at least one axiom from each antimodel in M' - $\mathcal{S}$ a hitting set 5.0.1 for M'.

Additionally, if $\mathcal{S}$ is relevant and explanatory (**??**) and $\mathcal{K} \sqcup \mathcal{S}$ is consistent, it's an explanation for observation O.

Definition 5.0.1 *(Hitting set (TODO: cituj paper))*
A hitting set for a collection of sets F (in our case, F is the collection of antimodels) is a set H s.t. $H \cap S \neq \{\}$ *for every* $S \in F$*. A hitting set H for F is minimal if there is no other hitting set H' for F s.t.* $H' \not\subseteq H$*.*

Definition 5.0.2 *To better explain our algorithm, it will be useful to define the term 'antimodel'. Let M be a model of knowledge base* $\mathcal{K}$*. Then M' is an antimodel of* $\mathcal{K}$ *iff it contains negations of every axiom from M and nothing else - for every axiom of form* $a : C$ *in M, M' has* $a : \neg C$*, and for every axiom* $a, b : R$ *in M it has* $a, b : \neg R$ *(a and b are not in role R). (TODO: toto asi pre niektore deskripcne logiky nieje mozne vyjadrit, (role complements))*

This idea was first introduced by Raymond Reiter in "A Theory of Diagnosis from First Principles"[5] as a general method for abductive reasoning in any formal logic with binary semantics (every statement is either true or false) and operands $\wedge, \vee, \neg$ with their usual semantic meaning, including first order logic. Additionally, Reiter proposes using a hitting set tree, which we will describe later.

Ken Halland and Katarina Britz in " ABox abduction in ALC using a DL tableau"[2] proposed an algorithm using the idea of hitting sets for abduction in description logic ALC, using a modified tableaux algorithm - their algorithm first develops all possible completion graphs (multiple graphs resulting from the use of $\sqcup$ rule), based on the knowledge base, and once there are no more rules to apply, they add an axiom from the observation complement $\neg O$ to the knowledge base. If there is no rule to apply or unused observation, the model is added to the list of models from which minimal

hitting set is generated.
This algorithm generates every model reachable by tableaux algorithm for $\mathcal{K} \sqcup O$, which as we will show, is not necessary.

Our work is mainly based on the algorithm by Martin Homola and Júlia Pukancová, using the idea of a hitting set tree from Reiter [5], which make generating every model of $\mathcal{K} \sqcup \neg O$ not necessary.

Definition 5.0.3 *(Hitting set tree (TODO: reference)) A hitting set tree (HS-tree) for a collection of sets F is T = (V, E, L, H), where (V, E) is the smallest tree in which the labelling function L labels the nodes of V by elements of F, the edges of E by elements of sets in F, and H(n) is the set of edge-labels from the root node to n ∈ V , s.t.:*

- *(a) for the root r ∈ V : L(r) = S for some S ∈ F, if F $\neq \{\}$, otherwise L(r) = {};*
- *(b) for each n ∈ V : L(n) = S for some S ∈ F s.t. S ∩ H(n) = {}, if such S ∈ F exists, otherwise L(n) = {};*
- *(c) each n ∈ V has a successor n_σ for each $\sigma \in L(n)$ with $L(n, n_\sigma) = \sigma$.*

We can generate the HS-tree inf the following way: let F be the list of antimodels for $\mathcal{K} \cap \neg O$. The label of each vertex n is either an antimodel of $\mathcal{K} \cap H(n) \cap \neg O$ (which is also an antimodel of $\mathcal{K} \cap \neg O$) or empty, if no such model exists - in this case H(n) is a hitting set of F, and $\mathcal{K} \cap H(n) \models O$ - H(n) is an explanation of H(n) if it also fulfills the other conditions (**??**).
If the vertex n is labeled by an antimodel, we create a child vertex for each axiom of antimodel L(n) - we label the edge by that axiom.
By using a HS-tree, we can find a hitting set that hits every antimodel of

$\mathcal{K} \cap \neg O$ while having to generate only a few of these models - each choice of edge label usually also eliminates many models that were not generated. The H and P (TODO ref) algorithm uses this algorithm, combined with the pruning methods (for example, eliminate a vertex n if H(n) contains axioms {A , ¬ A } (inconsistent with itself) or if there is a vertex n' with emply label (either due to pruning or being a hitting set) and $H(n') \subseteq H(n)$, or if there is a vertex n' where H(n) = H(n')).

This algorithm works for single observations, like {a:C} , {a,b:R} or { $a : (A \sqcap B) \sqcup C$}, but not for multiple observations, for example $\mathcal{MO}$ {a:A,b:B}. We can obtain their explanations by finding explanations of every single observation, performing carthesian multiplication (TODO: spravne vyjadrenie? mozno konjunkcia ako kartezianskym sucinom?) over the resulting sets of explanations, and finally checking the resulting explanations for consistency with $\mathcal{K} \cup \mathcal{MO}$ and relevancy with $\mathcal{K} \cup O$ for every single observation $O \in \mathcal{MO}$, and the resulting observations are then checked for minimality against each other. For example, in the knowledge base $\mathcal{K}$, if the explanations for {a:A} are {{b:B;a:C} , {a:D} } and the explanations for {b:B} are {{b:C},{b:D;b,a:R}}, the explanation candidates before consistency and relevancy checking for observation {a:A,b:B} are {{b:B,a:C,b:C},{b:B;a:C;b:D;b,a:R},{a:D;b:C}, {a:D;b:D;b,a:R}}. since {b:B} ⊨ {b:B} , the explanation candidates {b:B,a:C,b:C}, {b:B;a:C;b:D;b,a:R} are cut for relevancy. Let's say the remaning candidates ,{a:D;b:C},{a:D;b:D;b,a:R} are consistent with $\mathcal{K}$, then they are explanations of $\mathcal{MO} = \{a : A; b : B\}$.
TODO: the paradox ($a : A \sqcup B$ nie je to iste ako {a:A, a:B})

5.1 Our algorithm

Our algorithm is based on the H and P algorithm (TODO: ref), some main differences:

1. We do not use the formalism of the HS-tree, instead we only remember the hitting set candidates (the equivalent of vertex in HS-tree) for depth d and are building a set of hitting set candidates for depth d+1
2. When generating antimodels, we do not generate axioms when either they or their complements are in the $\mathcal{K} \cup O$
3. We check for relevancy and consistency of HS candidates when they are generated, instead of at the end of the algorithm.
4. (TODO: Minimum inconsistent set)

Chapter 6

Implementation

TODO: reasonery, logiky

```
function ABDUCTION( Knowledge base $\mathcal{K}$, observation O, maximum Depth maxD)
    Output: Set of minimum explanations $\mathcal{S}$
    if $\mathcal{K} \cup O$ is inconsistent then
        return $\emptyset$ //observation not consistent with knowledge base
    end if
    if $\mathcal{K} \cup \neg O$ is inconsistent then
        return $\{\{\}\}$ //observation can be inferred from knowledge base
    end if
    $\mathcal{C} = \{\{\}\}$ //hitting set candidates for this iteration (one empty set)
    $\mathcal{NC} = \emptyset$ //hitting set candidates for next iteration
    $\mathcal{S} = \emptyset$
    D = 1 //depth
    while $\mathcal{C} \neq \emptyset \wedge D \leq maxD$ do
        for each candidate $c \in \mathcal{C}$ //TODO: malo by c tiez byt v mathcal? do
            for each hitting set $s \in \mathcal{S}$ do
                if $s \in c$ then
                    Continue to next candidate //expalanation wouldn't be minimal
                end if
            end for
            if $\mathcal{K} \cup \{\neg O\} \cup c$ is inconsistent then
                if $\mathcal{K} \cup c$ is consistent then
                    $\mathcal{S} = \mathcal{S} \cup c$ //c is a hitting set (explanation)
                end if
                Continue to next candidate
            end if
            if D = maxD then
                Continue to next candidate // no need to create candidates for next while iteration
            end if
            Axioms $\mathcal{AX}$ = getAntiModel($\mathcal{K}, O, c$)
            for each axiom $ax \in \mathcal{AX}$ do
                if $ax \in c$ then
                    Continue to next axiom //$c \cup ax \equiv c$
                end if
                $nc = c \cup ax$
                if $O \in nc$ then
                    Continue to next axiom //explanation would be trivial
                end if
                $\mathcal{NC} = \mathcal{NC} \cup nc$
            end for
        end for
        $\mathcal{C} = \mathcal{NC}$
        $\mathcal{NC} = \emptyset$
    end while
end function
```

```
1: function getAntiModel( Knowledge base $\mathcal{K}$, observation O, set of
   axioms $\mathcal{AX}$)
2:     Output: Antimodel $\mathcal{AM}$ of a model $\mathcal{M}$ of $\mathcal{K} \cup \{\neg O\} \cup \mathcal{AX}$
3:     $\mathcal{AM} = \emptyset$
4:     $\mathcal{M}$ = model of $\mathcal{K} \cup \{\neg O\} \cup \mathcal{AX}$
5:     for each individual $\mathcal{I} \in \mathcal{K} \cup O \cup \mathcal{AX}$ : do
6:         $\mathcal{C}_k$ =set of concepts $\{\forall C|\mathcal{I} : C \in \mathcal{K}\ \}$ //known concepts
7:         $\mathcal{C}_a$ = set of all concepts $\in \mathcal{K}$ //all concepts
8:         $\mathcal{C}_i$ = set of concepts $\{\forall C|\mathcal{I} : C \in \mathcal{M}\ \}$ //inferred concepts
9:         for each concept $C \in C_i$ do
10:            if not $C \in \mathcal{C}_k$ then
11:                $\mathcal{AM} = \mathcal{AM} \cup (\mathcal{I} : C)$
12:            end if
13:        end for
14:        for each concept $C \in \mathcal{C}_a$ do
15:            if not $C \in \mathcal{C}_i$ then
16:                $\mathcal{AM} = \mathcal{AM} \cup (\mathcal{I} : C)$
17:            end if
18:        end for
19:    end for
20:    return $\mathcal{AM}$
21: end function
```

Chapter 7

Results

Chapter 8

Conclusion

Literatúra

[1] C. Elsenbroich, O. Kutz, and U. Sattler. A case for abductive reasoning over ontologies. CEUR, 2006.

[2] K. Halland and K. Britz. Naive abox abduction in alc using a dl tableau. In Y. Kazakov, D. Lembo, and F. Wolter, editors, *Description Logics*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[3] M. Krötzsch, F. Simancik, and I. Horrocks. A description logic primer. *arXiv preprint arXiv:1201.4089v3*, 2012.

[4] J. Pukancová and M. Homola. Tableau-based abox abduction for description logics, 2017.

[5] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[6] R. Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.

[7] S. Staab and R. E. Studer. *Handbook on Ontologies*. Springer, 2010.

[8] H. Van, V. L. Frank, and P. e. Bruce. *Kandbook of knowledge representation*. Elsevier, 2008.

Zoznam obrázkov