Interpolation based Assertion Mining

Stefan Schlobach*
Department of Computer Science, King's College London, Strand
London WC2R 2LS, UK
Email: schlobac@dcs.kcl.ac.uk

Abstract

In this paper we describe a new method for learning terminological knowledge from assertions in description logic based knowledge bases. We present *assertion mining* as the search for *generalised decision concepts (GDC)* which can be used to conceptually define classes of ABox individuals. We show that GDCs can be constructed from ABox interpolants and present tableau based algorithms.

In the recent past learning and knowledge discovery using description logics (DL) have attracted much research. *Least common subsumer (lcs)*–learning has been the most popular approach [3, 1]. lcs–learner construct minimal descriptions for the common properties of a set of positively classified objects in the ABox which do not instantiate any negative example. Unfortunately it is not always possible to construct such a minimal description, it can easily be seen that the *most specific concept* does not exist in DLs with existential quantification and many authors discuss approximations of the knowledge about individuals in knowledge base of different expressivity [8].

In this paper we present *assertion mining* for $\mathcal{ALC}$ knowledge bases as an alternative learning method which shifts the focus from learning of common properties of a set of positive examples towards the construction of discerning concepts which separate positive from negative examples.An advantage of such an approach is that these discerning concepts exist for $\mathcal{ALC}$ even if the most specific concept does not. As in lcs–learning we consider a supervised learning scenario: the ABox of a DL knowledge base contains a number of objects which are classified into positive and negative examples. For each of these classes (which we call decisions) we try to find terminological definitions which can be used to classify new data using instance checking. A concept which preserves the

*This research has been supported by EPSRC grant GR/L91818.

classification properties of a decision will be called *generalised decision concept (GDC).*

GDCs can be constructed from ABox interpolants for the positive and negative examples. An ABox interpolant is a concept which discerns two elements a and b in a knowledge base Σ, i.e. a concept for which a is an instance of, and for the negation of which b is. We define tableau based algorithms to calculate ABox interpolants for positive and negative examples. It can be shown that if such concepts exist in $\mathcal{ALC}$, they can effectively be calculated even if the most specific concepts do not exist.

Consider the following scenario: A hybrid knowledge base Σ^{Arr} contains information about patients suffering from cardiac arrhythmia. A patient record might include general assertions about gender, age or habits and family information, e.g. PAT_1: $\textsc{Male} \sqcap \neg\textsc{Smoker} \sqcap \neg\textsc{Old} \sqcap \exists\textsc{hasrelative}.\textsc{Arrhythmia}$, but also technical details about ECG measures such as $(\mathrm{PAT}_1,\mathrm{PW})$:has_pwave; PW:$\neg$OK; PAT_2:$\forall$has_pwave.OK, and additional knowledge describing the patients' conditions as diagnosed by some medical experts: PAT_1:Tachycardic; PAT_2:Healthy. Assertion mining is the search for terminological axioms which formally define a medical condition and possible diagnostic criteria for these conditions.

We assume familiarity with description logic representation and reasoning and will only briefly introduce $\mathcal{ALC}$. For concepts, which are interpreted as subsets of a universe U, we will usually use the letters C and D, furthermore R for roles, which are interpreted as binary relations over U. The constructors conjunction $C \sqcap D$, disjunction $C \sqcup D$, negation $\neg C$ and existential $(\exists R.C)$ and universal $(\forall R.C)$ quantification have the usual set theoretic interpretation. An ABox $\mathcal{A}$ is a finite collection of role $((a, b) : R)$ or concept $(a : C)$ assertions, where a and b denote individual objects of U. A TBox $\mathcal{T}$ is a set of axioms $C \doteq D$ or $C \dot\sqsubseteq D$. A knowledge base Σ is now defined as the pair $(\mathcal{T}, \mathcal{A})$. The reasoning services which we use in this paper are subsumption $\Sigma \models C \sqsubseteq D$ and instance checking $a \in_\Sigma C$ which semantically correspond to the subset and element relation. In $\mathcal{ALC}$, both subsumption and instance checking can be reduced to ABox inconsistency $\Sigma \models \mathcal{A} = \bot$. Example 1 describes an $\mathcal{ALC}$ knowledge base Σ^{arr} with some standard reasoning.

1 Assertion Mining

We will briefly define the main terminology for assertion mining, but refer to a detailed discussion in [12]. We assume that the ABox of a knowledge base[1] consists of a significant amount of possibly noisy information about individuals, a description of the knowledge about the concrete elements in the world. Assertion mining is the search for useful information in this data which is going to be represented as terminological knowledge.

[1] To simplify the presentation we assume from now on that $\mathcal{T}$ is empty or unfoldable.

Example 1 A knowledge base Σ^{arr} for arrhythmia diseases

For the knowledge base $\Sigma^{\text{arr}} = (\mathcal{T}_{\text{arr}}, \mathcal{A}_{\text{arr}})$ there are some simple but non-trivial examples of the reasoning processes:

$$\begin{aligned}
\mathcal{A}_{\text{arr}} = \{ \; & \textsc{pat}_1\text{: } \textsc{Male} \sqcap \neg\textsc{Smoker} \sqcap \forall\textsc{hasrelative}.\textsc{Tachycardic}, \\
& (\textsc{pat}_1,\textsc{pat}_1)\text{:}\textsc{hasrelative},\ \textsc{pat}_1\text{:}\forall\textsc{has_pwave}.\text{OK} \\
& (\textsc{pat}_2,\textsc{pw})\text{:}\textsc{has_pwave},\ \textsc{pw}\text{:}\neg\text{OK},\ \textsc{pat}_2\text{:}\textsc{Arrhythmia} \; \} \\
\mathcal{T}_{\text{arr}} = \{ \; & \textsc{Tachycardic} \mathrel{\dot\sqsubseteq} \textsc{Arrhythmia} \sqcap \neg\textsc{LowHeartRate}, \\
& \textsc{Hypertrophic} \doteq \textsc{Arrhythmia} \sqcap \exists\textsc{has_pwave}.\neg\text{OK} \; \}
\end{aligned}$$

1. *ABox consistency:* $\Sigma^{\text{arr}} \models \mathcal{A} \neq \bot$ but $\Sigma^{\text{arr}} \models \mathcal{A} \cup \{\textsc{pat}_1\text{:}\neg\textsc{Arrhythmia}\} = \bot$.
2. *Subs.:* $\Sigma^{\text{arr}} \models \forall\textsc{hasrelative}.\textsc{Tachycardic} \sqsubseteq \forall\textsc{hasrelative}.\textsc{LowHeartRate}$.
3. *Instance checking:* $\textsc{pat}_1 \in_{\Sigma^{\text{arr}}} \textsc{Arrhythmia}$

For this purpose we consider a supervised learning approach on data which is classified into decision classes. Here classification corresponds to instance checking with respect to a set $\mathcal{D} = \{D_1, \ldots, D_n\}$ of concepts (which we call *decisions*). Each object o in an ABox $\mathcal{A}$ which is an instance of at least one (but possibly more) decisions $o \in_\Sigma D_i$ in $\mathcal{A}$ is called classifiable. Let $class(\mathcal{A})$ denote the set of all classifiable objects in $\mathcal{A}$. In Σ^{arr} the ABox objects $\textsc{pat}_1$ and $\textsc{pat}_2$ are classifiable w.r.t. $\mathcal{D}_{\text{arr}} = \{\textsc{Tachycardic}, \textsc{Hypertrophic}\}$, $\textsc{pw}$ is not.

ABox mining is now the search for a formal definition for each of the decisions in $\mathcal{D}$ which might eventually be added to the TBox after being evaluated and assessed. In the process of the generalisation of a decision D_i, we will call the instances of D_i in $\mathcal{A}$ the *positive examples* and all instances of the remaining decisions *negative examples.*

Learning Criteria: Generalised Decision Concepts Whether a learned concept L_D can be used as a formal definition of a decision D depends on the knowledge which is represented in the ABox $\mathcal{A}$, the TBox $\mathcal{T}$ and on some additional *learning criteria.* The primary criterion we identified in [12] is exclusiveness. A concept L_D is *exclusive* w.r.t. a decision D if for all objects a in $class(\mathcal{A})$: $a \notin_\Sigma D \Rightarrow a \in_\Sigma \neg L_D$. Exclusiveness implies "correctness" of classification w.r.t. the original decision, i.e. $\forall a \in_\Sigma L_D \Rightarrow a \in_\Sigma D$, a condition traditionally known as *covering.* Furthermore, exclusiveness garanties coveredness even for the case that more information about elements in the ABox become available. Given the Open World Assumption which usually underlies description logic based knowledge representation, this seems to be a crucial requirement. Based on Rough Set theory [10] we also show in [12] that exclusiveness provides a theoretical notion of *safe data* which ensures good properties w.r.t. noisy data.

A learned concept L_D should also be supported by the fact that there is at least one example in the ABox, which is an instance of L_D and we define a *witness* for L_D as an individual $a \in D$ in $class(\mathcal{A})$ such that $a \in_\Sigma L_D$.

Definition 1.1 *A* generalised decision concept (GDC) *for a decision D with respect to a knowledge base $\Sigma = (\mathcal{T}, \mathcal{A})$ and a set of decisions $\mathcal{D}$ is a concept, which is exclusive w.r.t. D and $\mathcal{D}$ and for which a witness exists in $\mathcal{A}$.*

Given the definition of exclusiveness, the set of all GDCs for a decision D_i is:

$$\underline{G}(D_i) = \{C \in \mathcal{DL} \mid \begin{array}{l} \exists o \in \mathcal{A} : o \in_\Sigma C \text{ and} \\ \forall a \in \mathcal{A} : a \notin_\Sigma D_i \Rightarrow \forall_{k \neq i} D_k \in \mathcal{D}\ (a \in_\Sigma D_k \Rightarrow a \in_\Sigma \neg C). \end{array}$$

There are several GDCs for the decision Tachycardic w.r.t. Σ^{arr} among them $\forall$has_pwave.OK and $\forall$has_pwave.OK $\sqcup \neg$Arrhythmia.

Since there is still a possibly infinite number of GDCs we need to identify an inductive bias to be able to formally define the necessary restriction of the hypothesis space.

Inductive Bias: Common Vocabulary and Polarity. There are different aspects related to the choice of the inductive bias underlying our approach and we will discuss them briefly.

Object abstraction. We use the fact that DLs have language features (quantification) to reason about objects in an abstract way, generalising from particular examples to more general facts (e.g. existence of a role successor).

Common language with Discerning polarity. We take the designer of the knowledge base and the witnesses they provided seriously, and relate the learned concepts to the vocabulary of each example. The GDCs should consist of the vocabulary which is both used in the examples and the counterexamples and which is used in the same way (i.e. with the same polarity) as it was in the examples. Consider the GDCs for Tachycardic as given above. The second one contains the negated atomic concept Arrhythmia. But there is no reason to assume that people without arrhythmia are likely to have a Tachycardic condition. Since we know on the other hand that all of pat_1's p_waves are o.k as opposed to pat_2's it seems to be a legitimate inductive leap to assume that $\forall$has_pwave.OK is a *good* GDC for the decision Tachycardic. This bias is usually not discussed in the learning literature, because it automatically applies to most learning methods or is enforced using algorithms for missing attribute values.

Maximality. In addition to these syntactical restrictions we have to discuss the semantical choice of GDCs according to the subsumption ordering. In assertion mining the search space for possible learned concepts is defined by comparison of examples with counterexamples which is then restricted through the syntactical bias as described above. Since the learned concepts are defined explicitely as to exclude the counterexamples it seems appropriate to choose the maximal such concepts. This is a more or less arbitrary choice[2] which is related

[2]The choice is arbitrary because you can either minimise the description of the positive or of the negative examples. The first choice corresponds to the least common subsumer approach,

to ongoing research to deconstruct GDCs into sub-GDCs as an evaluation procedure. Such subconcepts of the GDCs will simplify the constructed GDCs and can be used to fine-tune the required level of generalisation.

It has to be mentioned that the algorithms presented in this paper do *not* calculate maximal GDCs w.r.t. the subsumption hierarchy. For technical reasons related to the definition of the common language of two examples we calculate *big* but not *maximal* concepts. For a more detailed discussion we refer to [12].

In the remaining sections we will introduce interpolation for hybrid knowledge representation systems and show how to use algorithms for interpolation to calculate such big GDCs with common vocabulary.

2 Interpolation Methods for Assertion Mining

Having defined learning targets and an inductive bias to restrict the hypothesis space it remains to provide effective algorithms to find some of these GDCs. We sketch the mining process from input (a knowledge base and a set of decisions) to the final output of new terminological axioms in the following section. This process is based on ABox interpolation. Vaguely speaking, interpolants are intermediate formulas semantically linking two other formulas using common vocabulary. We extend this notion to incorporate implicit knowledge about objects and polarity. The idea for the definition of the common language is based on the traditional notion of Lyndon interpolation for first order logic [9].

The language $occ(C)$ is a set of concept names occurring in C, labelled with the quantifier depth. A concept name A has positive (negative) polarity if it is embedded in an even (odd) number of negations. The quantifier depth describes the sequence of roles the concept is quantified over. $\tilde{occ}(C)$ is a dual notion in the sense that $\tilde{occ}(C) = occ(\neg C)$. Formally:

$$
\begin{array}{ll}
occ(C)^s = \{(A,+)^s\},\ \tilde{occ}(C)^s = \{(A,-)^s\} & \text{if } C = A \text{ and } A \text{ is an atom.}\\
occ(C)^s = \tilde{occ}(D)^s,\ \tilde{occ}(C)^s = occ(D)^s & \text{if } C = \neg D\\
occ(C)^s = occ(C_1)^s \cup occ(C_2)^s & \text{if } C = C_1 \sqcap C_2 \text{ or } C = C_1 \sqcup C_2\\
\tilde{occ}(C)^s = \tilde{occ}(C_1)^s \cup \tilde{occ}(C_2)^s & \text{if } C = C_1 \sqcap C_2 \text{ or } C = C_1 \sqcup C_2\\
occ(C)^s = occ(D)^{sR} & \text{if } C = \exists R.D \text{ or } C = \forall R.D\\
\tilde{occ}(C)^s = \tilde{occ}(D)^{sR} & \text{if } C = \exists R.D \text{ or } C = \forall R.D
\end{array}
$$

Based on this language we define concept interpolation, the basis for ABox interpolation. From now on we will use the letters I and L for (Lyndon) interpolants. *Concept interpolation* holds for two concepts C and D where $C \sqsubseteq D$ if there is a concept I such that $C \sqsubseteq I$ and $I \sqsubseteq D$ and $occ(I) \subseteq occ(C) \cap occ(D)$.

ABox interpolants connect the implicit positive knowledge about an objects a with the negative information about an object b, thus highlighting the differences

the second to assertion mining.

between them. A concept I is an *ABox interpolant* for a and b iff $a \in_\Sigma I$ and $b \in_\Sigma \neg I$ and where $occ(I) \subseteq occ^{\mathcal{A}}(a) \cap \tilde{occ}^{\mathcal{A}}(b)$. The language for an instance a is the smallest set $occ^{\mathcal{A}}(a)$ such that $occ(C) \subseteq occ^{\mathcal{A}}(a)$ for all conceptual axioms $(a : C) \in \mathcal{A}$, $occ^{\mathcal{A}}(b)^R \subseteq occ^{\mathcal{A}}(a)$ for all $(a, b) : R \in \mathcal{A}$ and $occ(C) \subseteq occ^{\mathcal{A}}(a)$ if $(b, a) : R \in \mathcal{A}$ and $b : \forall R.C \in \mathcal{A}$. $\tilde{o}cc^{\mathcal{A}}(a)$ is defined similarily.

Partial ABox interpolation plays an intermediate role between concept and ABox interpolation and relates knowledge about an object with a concept. A concept L is a *partial interpolant* for an object o, an ABox $\mathcal{A}$ and a concept C where $o \in_\Sigma C$ if and only if: $C \sqsubseteq L$ & $o \in_{\mathcal{A}} \neg L$ and $occ(L) \subseteq \tilde{occ}^{\mathcal{A}}(o) \cap occ(C)$.

Partial interpolation for an ABox $\mathcal{A}$ can be reduced to concept interpolation by construction of a preprocessing complete ABox $\mathcal{A}'$ for $\mathcal{A}$ [6]. Furthermore ABox interpolation can be reduced to partial interpolation by propagation of some properties of objects into the ABox.

2.1 Assertion Mining and Interpolation

It is easy to see that the set of ABox interpolants for a classified object $a \in_\Sigma D$ (the examples for D in $\mathcal{A}$) with all other classified individuals $b \notin_\Sigma D$ (the counterexamples) coincides with the set of generalised decision concepts.

Theorem 2.1 *If there are any classified individuals for a decision D and a knowledge base Σ, $\underline{G}(D)$ is the set of all ABox interpolants for the classified objects $a \in_\Sigma D$ with respect to the set of all classified instances $b \notin_\Sigma D$.*

Every generalised decision concept C for a decision D is an ABox interpolant for an $a \in_\Sigma D$ and all $b \notin D$ because there is a witness and because $b \in_\Sigma \neg C$ for all $b \notin_\Sigma D$. But every ABox interpolant I for a and all b obviously instantiates every b into $\neg I$ and since $a \in_\Sigma I$ is a witness, I is a generalised decision concept.

This theorem is the theoretical foundation for Assertion Mining using interpolation methods. The idea to use interpolants for assertion mining is simple: the input is an ABox $\mathcal{A}$ and a set of decisions $\mathcal{D}$ and an empty or unfoldable TBox. Preprocessing of the ABox might consist of unfolding, restriction to fewer elements, retrieval of the instances of each decision, etc. For each decision D in $\mathcal{D}$ a generalised decision concept for each decision (if defined) is calculated as the disjunction over the GDCs for each possible witness. The GDCs for each witness are just the conjunctions over the ABox interpolants of the witness with all the negative examples. The evaluation of the GDCs could comprise statistical analysis, decomposition into minimal subconcepts or involve human expertise.

Algorithm 2.1 summarizes the assertion mining procedure in pseudo code where $\texttt{ABox\_LI}(\mathcal{A}, a, b)$ is a method to calculate an ABox interpolant for two elements a and b with respect to an ABox $\mathcal{A}$. Such a procedure for $\mathcal{ALC}$ will be presented in the following section based on tableau calculi.

Algorithm 2.1 `disc_aboxmine(`$\mathcal{A}, \mathcal{D}$`)` Discernibility based Assertion Mining

Input: An ABox $\mathcal{A}$ and a set of decisions $\mathcal{D}$.
Output: A TBox $\mathcal{T} = \bigcup_{D \in \mathcal{D}} \{D \doteq G_D\}$, where all G_D are "almost" maximal GDCs.

$\mathcal{A}^* :=$ `preprocess`$(\mathcal{A})$;
$\mathcal{T} := \varnothing$;
for all $D \in \mathcal{D}$
    for all $a \in class(\mathcal{A}^*)$
        if $a \in D$
            for all $b \in class(\mathcal{A}^*)$
                if not $b \in D$
                    if `ABox_LI`$(\mathcal{A}^*, a, b)$ is defined
                        $LI^a := LI^a \sqcap$ `ABox_LI`$(\mathcal{A}^*, a, b)$;
    $GDC := GDC \sqcup LI^a$;
    `evaluate`(GDC);
    $\mathcal{T} := \mathcal{T} \cup \{D \doteq GDC\}$;
return $\mathcal{T}$;

2.2 Tableau Methods for Interpolation.

The algorithms to calculate interpolants using logical tableaux presented here follow the lines of [7]. Interpolants for $\mathcal{ALC}$ concept subsumption can be constructed from a fully expanded closed tableau collecting contradicting literals on each branch using construction rules corresponding to traditional tableaux. For ABox interpolants the more complex interaction between role and concept assertions has to be taken into account. The solution is to preprocess the ABox (according to [6]) and propagate the result of all possible inference steps into a concept. This concept can then be used in an intermediate interpolation step to represent the complete knowledge about one of the objects. Application of the same preprocessing steps then allow a further reduction of the interpolation problem to concept interpolation.

The algorithms described in this paper are based on tableau proof calculi. A tableau is a set of branches, where each branch is a set of formulas. A formula is a term of the form $(a : C)$ or $(a, b) : R$ where a and b are individual variables, C an $\mathcal{ALC}$ concept and R a role name. A branch is closed if it contains two formulas $a : C$ and $a : \neg C$. The notions of open branch and closed and open tableau are defined as usual. We will identify a branch B with the set of formulas ϕ on the branch and write $\phi \in B$. To simplify the presentation of our rules, we assume that all formulas are in *negation normal form*, i.e. negation is always pushed down to the atomic level [13]. The algorithms to calculate ABox interpolant are based on concept interpolation. Since concept interpolation is similar to interpolation in modal logic K and because of the close connection between $\mathcal{ALC}$ and K [11] we omit further details and assume that there is a procedure

$\texttt{concept\_LI}(C_1, C_2)$ which calculates a concept interpolant for C_1 and C_2. Please consult [12, 7] for further details.

$(\sqcap)$:	**if**	$(a : C_1 \sqcap C_2) \in B$, but not both $(a : C_1) \in B$ and $(a : C_2) \in B$
	then	$B' := B \cup \{(a : C_1), (a : C_2)\}$.
$(\sqcup)$:	**if**	$(a : C_1 \sqcup C_2) \in B$, but neither $(a : C_1) \in B$ nor $(a : C_2) \in B$.
	then	$B' := B \cup \{a : C_1\}$ and $B'' := B \cup \{a : C_2\}$.
$(\forall)$:	**if**	$(a : \forall R.C) \in B$ and $((a, b) : R) \in B$ but not $(b : C) \in B$.
	then	$B' := B \cup \{b : C\}$.

Figure 1: Preprocessing Rules

Algorithms for ABox interpolation. To calculate ABox interpolants we adapt the reduction introduced by Hollunder in [6]. In order to reduce ABox interpolation to concept interpolation and partial ABox interpolation, we have to preprocess the ABox using the rules in Fig. 1 until no more rule can be applied. In this case the ABox is called preprocessing complete (ppc). Each application of a rule triggers a construction rule for ABox Interpolants defined in Algorithm 2.2, where the concepts B_a and B_c defined in Definition 2.2 and $\texttt{inconsistent}(C)$ is the standard procedure for concept consistency.

Algorithm 2.2 $\texttt{ABox\_LI}(B, a, b)$: ABox Lyndon Interpolation

Input: A branch B and two individuals a and b.
Output: ABox Lyndon interpolant for a and b.

```
apply(rule,B);
if rule = (⊓), (∀)
   {B'} := get_new_branches;
   return ABox_LI(B', a, b);
if rule = (⊔)
   individual := get_individual;
   {B', B''} := get_new_branches;
   if individual = b return ABox_LI(B', a, b) ⊓ ABox_LI(B'', a, b);
   else               return ABox_LI(B', a, b) ⊔ ABox_LI(B'', a, b);
if rule = undefined;
   if   inconsistent(B_b) return ⊤;
   if   there is an individual name c ≠ b such that inconsistent(B_c) return ⊥;
   else return propagate_LI(B, a, b);
```

Hollunder shows that for any consistent ABox $\mathcal{A}$ there is a consistent ppc ABox $\mathcal{A}'$ derivable from $\mathcal{A}$. We identify $\mathcal{A}'$ with the set of all branches B build from $\mathcal{A}$. An ABox $\mathcal{A}$ is consistent if there is an open branch B in the ppc ABox $\mathcal{A}'$. For each of these open branches we now construct interpolants as defined in Algorithm 2.3. For this purpose we explicitely collect the information about an object a on a branch B into a concept B_a as defined below.

Definition 2.2 *The set $\mathcal{C}_a^B$ of concepts related to an individual $a \in B$ in a branch B is defined as follows: $C \in \mathcal{C}_a^B$ iff*

- *$a : C \in B$, where C is a literal, i.e. a concept name or its negation,*
- *$C = \exists R.D$ and $a : \exists R.D \in B$ if there is no R successor of a in B or*
- *$C = \forall R.D$ and $a : \forall R.D \in B$.*

For an individual $a \in B$ and a branch B we define a concept $B_a = \bigsqcap_{C \in \mathcal{C}_a^B} C$.

The concept B_a now contains all the conceptual information about an object a in B and an interpolant for two objects a and b on a branch B is simply the concept interpolant for B_a and $\neg B_b$. But there might still be interpolants relating a role assertion $(a, c) : R \in B$ with a universal quantified statement $b : \forall R.C$.[3] But this interpolant is just the R–quantified negated partial interpolant for C and c and to construct "almost" maximal interpolants, all possible partial interpolants of this kind are added disjunctively. To make sure that all interactions between role successors are detected, we have to propagate simultaniously.

Algorithm 2.3 `propagate_LI`(B, a, b): Propagation for ppc ABoxes

Input: A ppc consistent branch B and two individuals a and b.
Output: ABox Lyndon Interpolant for a and b with respect to B.

for all assertions $((b, d_1) : R) \in B$ to $((b, d_n) : R) \in B$
  for the set of all formulas $\{a : \forall R.C_1, \dots, a : \forall R.C_m\} \subseteq B$
            which are universally quantifying over role R for a
    if `partial_LI`$(B, C_1 \sqcap \dots \sqcap C_m, \{d_1, \dots, d_n\})$ exists
      $LI := LI \sqcup \forall R.$`partial_LI`$(B, C_1 \sqcap \dots \sqcap C_m, \{d_1, \dots, d_n\})$;
for all assertions $((a, c_1) : R) \in B$ to $((a, c_n) : R \in B$
  for the set of all formulas $\{b : \forall R.C_1, \dots, b : \forall R.C_m\} \subseteq B$
            which are universally quantifying over role R for b
    if `partial_LI`$(B, C_1 \sqcap \dots \sqcap C_m, \{c, \dots, c_n\})$ exists
      $LI := LI \sqcup \exists R.\neg$`partial_LI`$(B, C_1 \sqcap \dots \sqcap C_m, \{c_1, \dots, c_n\})$;
if `concept_LI`$(B_a, \neg B_b)$ exists
  $LI := LI \sqcup$ `concept_LI`$(B_a, \neg B_b)$;
return LI;

For partial interpolation for a set A of object a_1 to a_n and a concept C we again have to preprocess the branch and to expand the concept C until all possible interactions are made explicit. Algorithm 2.4 describe the preprocessing steps, Algorithm 2.5 the expansion of the concept into sets of concepts.

Note that the expansion of the concepts $C \in S$ can recursively trigger partial interpolation. If all rules are exhaustively applied the partial interpolant is

[3]If e.g. $c : \neg C \in B$, $\exists R.\neg C$ is an ABox interpolant for a and b w.r.t. B.

Algorithm 2.4 `partial_LI`(B, C, A): Disjunctive Partial Interpolation

Input: A consistent Branch B, a concept C and individuals $A = \{a_1, \ldots, a_n\}$
Output: Disjunctive Partial LI L: $C \sqsubseteq L$ and there is an $a_i \in_B \neg L$ for $1 \leq i \leq n$ and $occ(L) \subseteq \tilde{occ}^B(a_i) \cap occ(C)$

apply(rule,B);
if rule $= (\sqcup)$
  $\{B', B''\}$:= get_new_branches;
  return partial_LI$(B', C, \{a_1, \ldots, a_n\})$ $\sqcap$ partial_LI$(B'', C, \{a_1, \ldots, a_n\})$;
if rule $= (\sqcap), (\forall)$
  $\{B'\}$:= get_new_branches;
  return partial_LI$(B', C, \{a_1, \ldots, a_n\})$;
if rule = undefined;
  return exp_concepts$(B, \{C\}, \{a_1, \ldots, a_n\})$;

just the concept interpolant for the conjunction over the concepts in S and the disjunction of the negations of the B_a as defined in Definition 2.2 for all $a \in A$.

Algorithm 2.5 `exp_concepts`(B, S, A): Partial Interpolation

Input: A ppc branch B, a set of concepts S and individuals $A = \{a_1, \ldots, a_n\}$.
Output: Disjunctive partial LI: $\sqcap_{C \in S} C \sqsubseteq L$ and there is an $a_i \in_B \neg L$, $occ(L) \subseteq \tilde{occ}^B(a_i) \cap occ(C)$.

if there is a concept $C_1 \sqcap C_2$ such that $S = S' \cup \{C_1 \sqcap C_2\}$
  return exp_concepts$(B, S' \cup \{C_1\} \cup \{C_2\}, A)$;
else if there is a concept $C_1 \sqcup C_2$ such that $S = S' \cup \{C_1 \sqcup C_2\}$
    return exp_concepts$(B, S' \cup \{C_1\}, A)$ $\sqcup$ exp_concepts$(B, S' \cup \{C_2\}, A)$;
else if for all assertions $(a_i, b_i) : R \in B$ where $a_i \in A$
    for all universally quantified concepts $\forall R.C_1, \ldots, \forall R.C_n \in S$ over R
      return exp_concepts(B, S, A) $\sqcup \forall R.$ partial_LI$(B, C_1 \sqcap \ldots \sqcap C_n, \bigcup_i \{b_i\})$
else return concept_LI$(\sqcap_{C \in S} C, \neg B_{a_1} \sqcap \ldots \sqcap \neg B_{a_n})$;

In [12] soundness and completeness of the algorithms is shown in the sense that whenever there is one, ABox_LI$(\mathcal{A}, a, b)$ returns with an ABox interpolant for a and b w.r.t $\mathcal{A}$.

Let us consider the simple ABox $\mathcal{A} = \{(a, a) : R, a : C, b : \forall R.\forall R.\neg C\}$ which is preprocessing complete. To calculate an ABox interpolant for a and b we call propagate_LI$(\mathcal{A}, a, b)$ which returns $\exists R.\neg$partial_LI$(\mathcal{A}, \forall R.\neg C, \{a\})$. $\mathcal{A}$ is still ppc and exp_concepts$(\mathcal{A}, \{\forall R.\neg C\}, \{a\})$ returns $\forall R.$partial_LI$(\mathcal{A}, \neg C, \{a\})$, which is simply $\forall R.$concept_LI$(\neg C, \neg C)$. But this is $\forall R.\neg C$ and the ABox interpolant for a and b is therefore just $\exists R.\neg\forall R.\neg C$, which is $\exists R.\exists R.C$.

2.3 Assertion Mining and Most Specific Concepts

Our algorithms avoid the construction of a representation for the ABox objects and therefore of the most specific concepts[4]. This is possible due to exclusiveness, because all counterexamples are required to be instances of the negation of all GDCs. This condition is much stronger than the one used for lcs–learning, where counterexamples are only required not to be instances of the learned concepts. ABox interpolation captures this notion of instance checking because it corresponds to a search for possible contradictions between properties of positive and negative examples.[5] But in $\mathcal{ALC}$ the set of possible contradictions in an ABox is finite even if ABox cycles exist. Note however, that the algorithms described above do not terminate for description logics with number restrictions or features because this would allow for dual contradicting infinite chains. In this case non-maximal approximations of GDCs have to be constructed and the method will not be complete.

2.4 Practical Experience

WELLINGTON' KAT implements interpolation based assertion mining as an extension of the hybrid knowledge representation system WELLINGTON, which was developped at King's College London [4]. First tests have been performed on a knowledge base Σ^{arr} for cardiac arrhythmia [5], which contains various information including ECG data (based on 279 attribute values) about 452 patients, which are classified into 16 classes of cardiac arrhythmias. Σ^{arr} has been created by translation from a database available at [2] and has maximal relational depth of three. First results indicate that assertion mining without evaluation procedures over-generalises, i.e. too many previously unknown examples are instances of too many GDCs. Another crucial problem is related to efficiency and complexity. Although the learning procedure itself is reasonably fast, the GDCs which are constructed are too complex for efficient subsequent instance checking of new objects. Both these issues are related and we are currently implementing evaluation procedures to simplify GDCs which include symbolic (e.g. logical decomposition) as well as statistical methods.

3 Conclusion

We have presented algorithms for assertion mining based on interpolation which calculate generalised decision concepts even if the most specific concept for some objects in the ABox do not exist. They allow for the construction of new termi-

[4]The *most specific concept* $msc(a)$ for an ABox instance a is a minimal concept in the set of concepts which instantiate a, i.e. $a \in_\Sigma C$ implies $\Sigma \models msc(a) \sqsubseteq C$ for all $C \in \mathcal{ALC}$.

[5]In an $\mathcal{ALC}$ ABox $\mathcal{A}$, an ABox individual a is an instance of a concept C if and only if $\mathcal{A} \cup \{a : \neg C\}$ is inconsistent.

nological knowledge from classified ABox instances. We are currently extending the DL based WELLINGTON representation and reasoning system with a KAT, a knowledge acquisition tool based on the presented methods.

References

[1] F. Baader and R. Küsters. Least common subsumer computation w.r.t. cyclic $\mathcal{ALN}$-terminologies. In *Proceedings of the 1998 International Workshop on Description Logics (DL'98)*, 1998.

[2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[3] W. Cohen and H. Hirsh. Learning the classic description logic: Theoretical and experimental results. In *KR-94*, pages 121–133, Bonn, Germany, 1994.

[4] U. Endriss. Reasoning in description logic with WELLINGTON 1.0. In *Proceedings of the Automated Reasoning Workshop 2000*, London, UK, 2000.

[5] H.A. Güvenir, B. Acar, G. Demiröz, and A. Cekin. A supervised machine learning algorithm for arrhythmia analysis. In *Computers in Cardiology*, volume 24, pages 433–436, 1997.

[6] B. Hollunder. Consistency checking reduced to satisfiability of concepts in terminological systems. *Annals of Mathematics and Artificial Intelligence*, 18:95–131, 1996.

[7] M. Kracht. *Tools and Techniques in Modal Logic.* North Holland, 1999.

[8] R. Küsters and R. Molitor. Computing most specific concepts in description logics with existential restrictions. LTCS-Report 00-05, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000.

[9] R.C Lyndon. An interpolation theorem in the predicate calculus. *Pacific Journal of Mathematics*, 9:155–164, 1959.

[10] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

[11] Klaus Schild. A correspondence theory for terminological logics: preliminary report. In *Proceedings of IJCAI-91, 12th International Joint Conference on Artificial Intelligence*, pages 466–471, Sidney, AU, 1991.

[12] S. Schlobach. Interpolation methods for assertion mining in hybrid knowledge bases. Technical report, King's College London, 2001.

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