Using Abstraction Network in Complement to DL for Quality Assurance Purposes in Biomedical Terminologies

Student: Duo Wei
Mentor: Olivier Bodenreider

Introduction

Quality assurance is a demanding task for large biomedical terminologies during the development and maintenance process. The Description Logic (DL) serves as the basis of terminology classification and consistency checking for quite a few large terminologies, such as SNOMED CT [1], Galen[2], and NCIT[3]. Without doubt, to some extend DL helps the quality assurance of the terminology system; however, a lot of errors are still hanging around the system, which seems that DL alone is not enough to create an error-free system.

In this paper, we illustrate how our previously defined the Abstraction Network (AN for short) offer a natural solution in complement to DL to detect errors in a biomedical terminology. In previous studies, a number of auditing techniques, including ourselves, have devised: in [Semantic refinement and error correction in large terminological knowledge bases], "semantic refinement" is presented, which helps detection of ambiguity, non-uniform classification and missed synonymy. In [Ontology-based error detection in SNOMED -CT], two algorithms are combined to detect improper assignment of relationships, redundant concepts, and omission of relationships. In [Auditing the semantic completeness of the SNOMED CT using formal concept analysis], the FCA method was used to analysis the anonymous concepts; An analysis has been carried out to determine how well SNOMED's IS-A hierarchy adheres to certain ontological principles in [Investigating subsumption in DL-based terminologies: a case study in SNOMED CT] Ronald Cornet paper two auditing methods were mentioned, one auditing DL system by detecting equivalent concept definitions and the other use DL based method for auditing frame –based medical TS. To summarize, these approaches applied computational method to highlight where errors tend to concentrate and manual interpretation and auditing will be involved.

In general, these approaches are based on lexical, linguistic or ontological methods, whereas the abstraction network is non-lexical, non-linguistic and non-ontological. Various auditing techniques have been proposed based on AN [JBI2007, AMIA2008, AMIA2009]. The specimen hierarchy has been selected as our testbed. We start with introducing the background materials of the DL and the AN; then present the AN based auditing method; the auditing results is verified by the reclassification of the SNOMED with the DL reasoner FaCT++. Finally, we discuss how DL and AN approach can complement with each other.

Background

A. SNOMED CT

SNOMED CT is one of the largest reference terminologies used worldwide. According to the most recent release (Jul. 2009), there are more than 289,000 active concepts and 1.5 million relationships defined in the SNOMED. The concepts are organized into 19 hierarchies, such as Procedure, Clinical Finding, etc.

Due to its large size and complexity, quality assurance tends to be significant during the maintenance process. A large number of errors have been found, such as disobedience of disjoint rules, duplicate definitions, missing parent/children, erroneous target concepts, or even modeling errors. According to the Quality Assurance group of IHTSDO, the current owner and administrator of the SNOMED CT, "From 2002 to 2008 approximately 20,000 concepts were deactivated because they were duplicates, outdated, ambiguous, etc". In virtual of quality assurance, the quality of the descriptions is also improving. For example, more concepts now have sufficient logic definitions – particularly those in the clinical findings and procedures hierarchies. About XX % of errors they found are by DL based tools, while the rest of the errors have to be aided with other auditing techniques.

B. Description Logic

Description logics (DL for short) are a family of knowledge representation languages which provide a language for defining a knowledge base and a tool to carry out inferences out of it.
The basic elements of the DL are atomic concepts and atomic roles. The following table (Table 1) shows concept constructors from the family of AL (Attributive language) which is the minimal language of practical interest. The other languages of this family are extension of AL. Complex descriptions can be built by combining constructors taken from Table 1. In the table, **A** represents atomic concepts and **R** for atomic roles.

Table1. Concept constructor in the basic description language

Name of the construct	**Notation**	**Interpretation**
Universal concept	$\top$	$\top^I = \Delta^I$
Bottom concept	$\bot$	$\bot^I = \emptyset$
Atomic concept	A	$A^I \subseteq \Delta^I$
Atomic negation	$\neg A$	$(\neg A)^I = \Delta^I \setminus A^I$
Intersection	$C \cap D$	$(C \cap D)^I = C^I \cap D^I$
Value restriction	$\forall R.\ C$	$(\forall R.\ C)^I = \{ a \in \Delta^I \mid \forall\ b.\ (a, b) \in R^I \rightarrow b \in C^I \}$
Limited existential qualification	$\exists R.\ T$	$(\exists R.\ T)^I = \{ a \in \Delta^I \mid \exists\ b.\ (a, b) \in R^I \}$

Suppose the concept C has one parent P with a relationship r pointing to another concept Q, and then the concept C can be indicated as $C \subseteq P \cap \exists$ r. Q, More expressive language will be obtained by adding further constructors to AL. For example, Table 2 shows union of concepts, the full existential quantification or number restrictions.

Table2. Extension of AL-language family

Name of the construct		**Notation**	**Interpretation**
Union		$C \cup D$	$(C \cup D)^I = C^I \cup D^I$
Fully existential qualification		$\exists R.\ C$	$(\exists R.\ C)^I = \{ a \in \Delta^I \mid \exists\ b.\ (a, b) \in R^I \wedge b \in C^I \}$
Number restriction	At least	$\geq n\ R$	$(\geq n\ R)^I = \{ a \in \Delta^I \mid \lvert\{b \mid (a, b) \in R^I\}\rvert \geq n \}$
	At most	$\leq n\ R$	$(\leq n\ R)^I = \{ a \in \Delta^I \mid \lvert\{b \mid (a, b) \in R^I\}\rvert \leq n \}$

DL not only provides the facilities for defining concepts and specifying concepts and individuals (refer to the definition of Tbox and Abox), it also provides methods for reasoning about the knowledge being represented. The basic inference on concept expression in DL is subsumption, which is used to check one class (subsumer) is more general than the other (subsumee). Another, inference is concept satisfiability, which, in fact, is a special case of subsumption, with the subsumer being the empty concept, meaning that a concept is not satisfiable. A class is deemed to be inconsistent if it cannot possibly have any instances. Obviously, there are no instances for this class, so it is kind of inconsistent. The subsumption of concepts can be computed by so- called structural subsumption algorithm; while the negation and disjunction can be handled by so called tableau-based algorithm.
For more details on the theoretical aspects and reasoning algorithms of the DL, please refer to [The Description Logic Handbook-Theory, Implementation and Application] or DL homepage at http://dl.kr.org/dl/.

SNOMED is grounded upon the DL model at that DL functions in the SNOMED from the following aspects: (1) it helps to derive implicit knowledge from the one explicitly represented; (2) it classifies SNOMED CT. i.e. compute the concept subsumption hierarchy; (3) it identifies undesired consequences; (4) it explains questionable consequences and debug them.
SNOMED's native view that only defining relationships that an author has explicitly stated to be true is called stated view. The distributed version generated by DL reasoner automatically, which we are using in this work, is called the inferred view. The inferred view is derived from the stated view.
Although DL based reasoning algorithm has been introduced to ensure the compliance of DL rules and check consistencies, still errors can find their ways to the large knowledge base. The reason behind is that errors can be

roughly divided into two categories: Type I errors referred to as the errors that might cause logic contradictory in the system. For example, the classification inconsistencies or the disobedience of certain restrictions, etc. Type II errors are those will not generate logic conflicts of the systems, and DL reasoner seems to be blind to such kind of errors. Therefore, the former errors, given appropriate expression, can be easily detected by DL reasoners while the latter kinds tend to be beyond DL reasoner's grasp.
In this situation, AN comes into play as a supplementary with DL reasoners to help with quality assurance. In the next section, we will introduce AN and its working mechanism, followed by DL based verification.

Method

1. Abstraction Network

In previous work [JBI 2007], we developed structural methodologies for auditing SNOMED based on the associative relationships and their inheritance patterns in the SNOMED hierarchies. Our structural auditing techniques basically consist of two phases: structural abstraction and hierarchical abstraction. Both abstractions, called Abstraction Network (AN), are derived automatically.

AN techniques follow the associative relationship aggregation of a SNOMED hierarchy as captured by the featured groups. The idea is to partition a SNOMED hierarchy to structural uniformity groups (strUG), and then to refine the division into semantic uniformity groups (smtUG). Please note that in this paper, structural uniformity group refers to Area, and semantic uniformity group refers to partial-area in our previous work. A detailed description can be found from our previously published work [JBI 2007, AMIA 2008, AMIA 2009]. Let us start with some definitions.

Structural uniformity group (strUG) is a collection of all concepts with exactly same set of relationships.

The strUG contains all the concepts of the terminology sharing the same structure. Suppose the concept C has one parent P and a relationship r pointing to another concept Q. Then the concept C can be indicated as $\mathbf{C} \subseteq \mathbf{P} \cap \exists\ \mathbf{r.Q}$. Likewise, assuming a set of concepts with the same set of relationships $\{r_1, r_2, r_3 \ldots\}$ pointing to concepts $\{C_{r1}, C_{r2}, C_{r3, \ldots}\}$ in other hierarchies such that each concept has its own set of direct ancestors $\{C_{p1}, C_{p2}, \ldots\}$, and then we can present the strUG$\{ r_1, r_2, r_3 \ldots\}$ in the following DL statement:

$$\mathbf{strUG(\{\bigcap r_t\}) = \{C|\ C \subseteq \bigcap_{i=0}^{n} C_{Pi} \cap (\bigcap_{k=0}^{t} (\exists\, r_k . (\bigcap_{j=0}^{m} Cr_j))\ )\},}$$

where t is # relationships, n is # parents, m is # targets, and r is a relationship. ($0 \le k \le t$, $0 \le i \le n$, $0 \le j \le m$). Because each concept in a SNOMED hierarchy can be defined by a given set of relationships, the strUG can therefore be interpreted as the entire set of concepts that exhibit exactly the same set of relationships. If a concept has no relationship, then its strUG is Ø.
For example, the Specimen hierarchy has five relationships: *specimen substance*, *specimen source morphology*, *specimen procedure*, *specimen source topography*, and *specimen identity (substance, morphology, procedure, topography, and identity, respectively)*. One of the concepts **Surgical excision sample** of the Specimen hierarchy has one direct ancestor **Specimen**, and one relationship *procedure* pointing to the concept **Excision** (of "Procedure" hierarchy). Therefore, concept **Surgical excision sample** $\subseteq$ **Specimen** $\cap\ \exists_{\textit{procedure}}$ **. Excision**, and thus **Surgical excision sample** $\in$ strUG ({*procedure*}). Similarly, another concept **Specimen obtained by amputation** is also in the strUG({procedure}), because it has one direct ancestor **Specimen** and one relationship *procedure* pointing to the concept **Amputation** (of "Procedure" hierarchy). Note that strUG are disjoint, because a concept is expected to have a specific number of relationships and it will belong to one and only one strUG. Therefore, the entire set of strUGs form a partition of a hierarchy's concepts.

Roots The root of a structure uniformity group is a concept, of the structure uniformity group, whose parents all reside in other structure uniformity group.

In other words, a root has no IS-A relationship to another concepts in its strUG. All descendants of a root are subsumed by the root. So the root captures the nature of all its descendants. As a consequence, a strUG may have more than one root. Here is the root representation in DL formalism:

$$\textbf{Roots} = \{C \mid C \subseteq \textbf{strUG}(\bigcap r_t) \bigcap \textbf{Parent}(C) \not\subset \textbf{strUG}(\bigcap r_t)\}$$

Where Parent(C) is a function that indicates all the parents of C. Let's take the previous example, in the strUG({*procedure*}), **Surgical excision sample** has 6 children, such as **Specimen obtained by marginal resection**, **Specimen obtained by radical excision,** etc. All the 6 children are, in essence, kind of surgical excision sample. So we claim that **Surgical excision sample** is a root in strUG({*procedure*}).

We, therefore, further group the single root and all its descendants within the strUG to what we called smtUG.

Semantic uniformity group (smtUG) is a set of concepts comprising a single root and all its descendants.

The smtUGs form a semantic division of the strUG, which provides the semantic uniformity. The DL definition is as follows:

$$\textbf{smtUG(root)} = \{C \mid C \subseteq \textbf{strUG}(\{\bigcap r_t\}) \bigcap f(C) = \textbf{root}\},$$

and the function f(C) is to get the root of a concept C. Here please note that smtUG(root)s within the same strUG({$\bigcap r_t$}) are not necessarily disjoint, since a concept may trace back to more than one roots, and thus smtUGs within a strUG does not form a partition.
As the previous example indicates, smtUG(**Surgical excision sample**) = {**Surgical excision sample, Specimen obtained by marginal resection, Specimen obtained by radical excision,** … }. In addition, **Specimen obtained by amputation** does not have any children in strUG({*procedure*}), so it is a single root and itself forms a singleton smtUG(**Specimen obtained by amputation**) = { **Specimen obtained by amputation**}. In the strUG({procedure}), there are 7 smtUGs all together.
The strUG and smtUG form a graph structure called abstraction network (AN). The AN offer the auditor novel views of the terminology's content and also help to hightlight portions that are ripe for deeper investigation.

2. How AN helps error detection

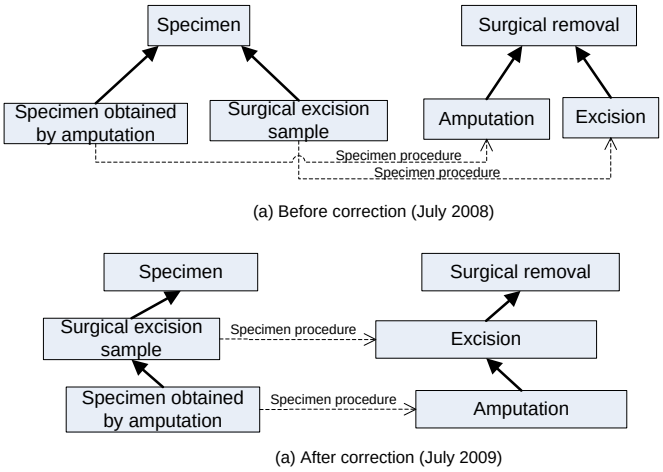
Here we avail ourselves to the inferred view of the SNOMED CT. As we mentioned previously, the inferred view is generated by auto-classification with DL based reasoning algorithms. The structure uniformity group (strUG) and the semantic uniformity group (smtUG) give the auditors a compact view to detect consistencies or irregularities from two facets: disparity and similarity. The disparity among different strUGs or disparity among different smtUGs within the same strUG; the similarity within the same smtUG lead us to the following hypothesis that we wish to investigate. We will illustrate two examples to further support our hypothesis and to demonstrate how strUG and smtUG help to detect errors.

Hypothesis 1 (Disparity) In the same structural uniformity group, different semantic uniformity groups are not hierarchically related and they are referring to different semantic domains.

Hypothesis 1 utilizes the characteristic of the roots which form a natural categorization in a strUG.
Roots within the same strUG are expected to have the same structure (because they are in the same structure uniformity group), but with great difference by nature. That is to say, it is not allowed to have any ancestor-descendent relationship or other genetic relationship among roots. According to this hypothesis many kinds of

errors can be detected, such as incorrect parent/child, incorrect target of relationship, etc. Let us take previous example, both smtUG(**Surgical excision sample**) and smtUG(**Specimen obtained by amputation**) in the strUG({*procedure*}). However, specimen obtained by amputation is, in fact, a kind of surgical excision sample. In this case, there is a parent-child relationship between two smtUGs within the same strUG({*procedure*}) , which is not acceptable according to our hypothesis 1, because it causes structurally inconsistent.
From the concept level, the concept **Specimen obtained by amputation** and **Surgical excision sample** are siblings, and both of them are in the "Specimen" hierarchy under the root concept **Specimen**. Correspondingly, their targets with the relationship *specimen procedure* are **Amputation** and **Excision** in the "Procedure" hierarchy under the parent concept **Surgical removal**. In order to make sure the consistency of the system we proposed to modify both the source "Specimen" hierarchy side as well as the target "Procedure" hierarchy side. Figure 1 shows the comparison before and after audits from a concept-level. As a result of auditing, Specimen obtained by amputation is subsumed by Surgical excision sample, and thus the smtUG(**Surgical excision sample**) is enlarged by the involvement of a new member. So the smtUG(**Specimen obtained by amputation**) disappeared from the strUG({*procedure*}), which keep structurally uniform in this group as well.

Figure 1 parent-child error with "Specimen obtained by amputation" and "Surgical excision sample"



Hypothesis 2 (Similarity) In the same structure uniformity group, concepts within the same semantic uniformity group are hierarchically related.

Concepts in the same smtUG share the similar meanings because they inherit from a common ancestor. This compact view helps the auditor to detect irregularities such as duplicate or missing concepts. Here is an example of concept duplication. In the smtUG(**White blood cell sample**) = {**White blood cell sample**, **Leukocyte specimen**, **Macrophage specimen**, **Basophil specimen**, **Buffy coat**, **Lymphocyte specimen**, **Eosinophil specimen**, **Polymorphonuclear neutrophil specimen**, **Leukocyte specimen from control**}, there are totally 9 concepts. However, it is apparent that **White blood cell sample** and **Leukocyte specimen** are identical concepts, one of which should be synonym of the other. The reason for the duplication is that SNOMED is the integration of SNOMED RT (US) and CTV3 (UK), and as a result duplicate errors occur during the integration process. Since smtUG is a collection of similar concepts, it is efficient for the auditors to detect duplicate errors or missing errors by looking at small portion of a large terminology data set with similar meanings.

3. Auditing Method based on Abstraction Network

Several techniques have been developed based on the Abstraction Network. For instance,

Group-based auditing Previously [JBI 2007], we introduced the notion of "group-based auditing," which was opposed to the "concept-based" approach. That is, concepts of a semantic uniformity group were reviewed at the same time, so that concepts in the same group will provide contexts for each other. As mentioned above, in the same semantic uniformity group, concepts were expected to be both structurally and semantically uniform (Hypothesis 2), and thus semantic uniformity group was an ideal unit for group-based auditing. Errors exposed via "group-based auditing" included redundant concepts, erroneous relationships, incorrect IS-A assignments, and modeling errors.

Error concentration based auditing In Halper, et al. [AMIA 2007], an investigation into the uniformity groups was carried out. Three hypotheses pertaining to the error distributions were put forward. One of the hypotheses was with respect to small semantic uniformity groups. It indicated the expectation that a small group of concepts with similar structure and semantics was less likely to be properly modeled than a similarly constituted large group. This method quantified the correlation between the size of the semantic uniformity group and the error concentration. Such analysis improved the efficiency of auditing by highlighting the most potentially suspicious portions.

Auditing "complex" concepts Wang, et al. reported the technique of auditing "complex" concept, which could aid an auditor by automatically identifying concepts that deserved attention. A "complex" concept could be also viewed as "Overlapping concepts," which was a concept inherited from parents belonging to more than one semantic uniformity group. As was stated in Hypothesis I, different semantic uniformity groups (of the same structure uniformity group) had different semantic domains. A concept in an intersection of semantic uniformity groups resided at a point that elaborates its parent semantic uniformity combination. Therefore, these kinds of concepts tended to be more complex and thus deserve higher auditing priority.

Result

1. Results for AN based auditing techniques

The number of concepts of our testbed - Specimen hierarchy has been changed year by year. For example, in July 2007, there are 1,056 concepts; but in July 2008, the number drop down to 1,044; whereas in July 2009 saw an increase up to 1,073. Part of the reasons of such fluctuation in number is the impact of auditing. In SNOMED editorial group, most of the editing work utilizes the TDE "tree editor", which displays the children of a concept along with its relationships. The quality assurance process applied DL – classifiers that not yet publicly available. The inferred view generated from the DL-classifier was used in our experiment, which means, our AN technique was applied after the DL reasoning procedure was done. Here shows some results by different auditing techniques:

Group-based auditing

Altogether we found 54 errors of different kinds by using the group-based auditing methodologies (July 2004). These errors were reviewed by one of the authors, who were the Chief Terminologist of SNOMED CT. All but four of the errors were confirmed and corrected in the subsequent release of the SNOMED CT.

Error concentration based auditing

The error totals found in the context of the semantic uniformity group of various sizes can be seen in Table 3. The table, in fact, breaks the space of semantic uniformity group into two: those with seven or fewer concepts and those with eight or more. Semantic uniformity group in the former range are deemed to be "small"; those in the latter, large. As can be seen from the table, the number is only 6.83% for large semantic uniformity group. The result also confirmed our hypothesis.

Table 3. Errors across ranges of semantic uniformity group size (# of concepts)

Size of the smtUG	# smtUG	#Concepts	#Errors	%Errors
1-7	427	646	69	10.68
8 or more	25	410	28	6.83
Total	**452**	**1,056**	**97**	**9.19**

Auditing "complex" concepts Table 4 lists eight types of errors and their counts by auditing complex concepts. This table appeared in Wang's paper [AMIA Annu Symp Proc. 2008; 2008: 273-277].

Table 4. Kinds of errors and their counts

Kinds of Errors	#
Ambiguous concept	1
Missing child	48
Missing parent	30
Missing relationship	21
Missing sibling	4
Incorrect child	5
Incorrect parent	44
Incorrect target of relationship	5
Total	**158**

2. Results versification

To evaluate our methodology, we classified the original version of the SNOMED owl. Next, we corrected the proposed errors and transformed the modification to owl format. Then, we reclassified the modified SNOMED owl file with DL based reasoner FaCT++, which was plugged in to the Protégé 4.
We performed the experiments using a standard desktop with 64-bit Microsoft Windows operating system and 4 GB of RAM. Classification of the original owl version of the SNOMED took 3.563s+898s = 901.563s, including loading and classification. The reclassification of the modified version of the SNOMED CT took almost the same amount of time. There was no warning or error reported during the reclassification process, which meant that the modification based on abstraction network did not introduce new errors. It also proved that abstraction network in concert with DL and the two methods have complementary advantages and mutual benefit.

Discussion

It is noticeable that AN approach is based on DL created inferred view and verified by DL based reasoner, but our work extend DL's function by further abstracting the inferred view according to the semantic and structural correlation. So we say that DL and AN can complement with each other.
From the previous illustration, we know that AN rooted from DL *per se*, but it offers a mechanism to group concepts with maximum structural and semantic correlation. DL based reasoning algorithm consider an ontology as a whole without looking at the correlation among concepts. While AN enables navigation into the content and structure of a terminology by providing abstract view of different granularities.
Quality assurance on large medical terminologies like SNOMED CT has attracted more and more attention. However, the modeling and maintenance issues seem to be a never-ending story. DL formalism is a powerful tool for quality assurance, but it is less successful without other auditing methodologies' assistance. The AN technique is among numerous techniques. Although similar topics have been discussed in some papers like [Ontology and Medical Terminology: Why Description Logics Are Not Enough], we are from auditors' point of view instead of ontological or linguistic perspective. As a result of auditing, there are totally 158 errors revealed based on auditing

"complex" concepts and 58 errors based on group auditing. Regardless the fact that a concept might be involved in more than one error, the ratio of erroneous concepts with these two methods constitute approximately 15% and 5% separately of the total concepts population in the Specimen hierarchy. The reclassification of the SNOMED OWL future proved the efficiency and efficacy our auditing methods in dealing with Type II errors.

We have tracked 67 samples of erroneous concepts from the Specimen hierarchy (released in July 31, 2007), and all of which were confirmed by Dr. K.A. Spackman, who is the chief terminologist in IHTSDO. Forty-five of those errors (67.2%) were found corrected in the July 2009 release. Of the 67 concepts, 43 concept definitions have been changed from "Primitive" to "Fully Defined." From which, we also conclude a potential impact of auditing that the percentage of primitive concepts in the entire active concepts becomes smaller. This can be reflected from the following table about the percentage of primitive concepts vs. defined concepts of the whole SNOMED terminology system over three successive years.

Table 5. Primitive concepts vs. defined concepts (%) in three years

Year	IsPrimitve	#Concepts	#Total[1]	Percentage
20070731	Y	234,295	283,131	82.75%
	N	48,836		17.25%
20080731	Y	230,889	289,029	79.88%
	N	58,140		20.12%
20090731	Y	227,681	289,898	78.54%
	N	62,217		21.46%

Footnote: 1 represents the total number of active concepts, that is, concepts with a ConceptStatus = 0. Active concepts are concepts with unique meanings and formal logic-based definitions.

From Table 5 we can see that there is nearly 5% decrement of the primitive concepts from the year 2007 to 2009. On the contrary, the percentage of the defined concepts saw an increment. A possible reason is that quality assurance helped editors or terminologists assign more exact definitions to a concept, which is, in a sense, avoid ambiguous.

As we know, among lots of other supplementary of DL, AN is just one of them. Moreover, AN has its own limitations, for example, manual revisions were involved in the auditing process, so AN based technique is still semi-automatic. Furthermore, it is required that systematic inheritance of relationships for derivation of AN and thus, it is not applicable to hierarchies of concepts without inheritance. As for those hierarchies with few relationship (or no relationship at all), we had developed the converse abstraction network [AMIA 2009]. In practice, it is unavoidable that new errors may be introduced while conduct auditing on portion of the terminologies.

As for the future work, efforts should be made to eliminate Type I and Type II errors. The former can be improved by developing more efficient reasoning algorithms and much more accurate models; while the latter can be enhanced by introducing automatic and systematic auditing methodologies.