CHAPTER 4

SEMANTIC RELATIONSHIP BASED REASONING
(SemPER)

The main objective of this research work is to transform the reasoning methodology with semantic relationships for knowledge discovery. The proposed reasoning algorithm performs consistency and satisfiability tests and derives the novel semantic relationships from the implicitly specified knowledge in order to ameliorate the open world semantics. This chapter commences with the need of the proposed semantic relationship based reasoning (SemPER) algorithm. The components of the reasoning algorithm such as semantic relationships extraction (RelExtract), semantic relationships classification (SeRelaC) and semantic relationship based reasoning algorithm (SemPER) are elaborated in the subsequent sections.

4.1 NEED FOR SEMANTIC RELATIONSHIP BASED REASONING

The semantic relationships are fundamental building blocks to semantics. Thus, it provides enhanced understanding of the domain knowledge. Generally, the represented domain knowledge consists of various concepts, relationships, instances and properties. Lack of semantic relationships in domain knowledge provides diversified perception among users. It also establishes inconsistency among knowledge users. In particular domain knowledge, all the represented concepts are semantically interrelated however semantically inconsistent with the occurrence of the semantic relationships. The concepts are interrelated through various semantic relationships. Hence, it will be very difficult for the user to understand the concepts completely. Thus, the exploitation of the semantic relationships with respect to its functional properties and its semantics will help to improve the open world semantics. In turn, it helps in bringing common understanding among users. This could be achieved by inferring relationships from implicitly specified knowledge. Semantic relationships discovery has both theoretical and practical implications. It could be very useful in knowledge intensive applications such as question generation, information retrieval, ranking, etc.
With this focus, the following section describes the extraction and classification of semantic relationships which helps in transforming reasoning methodology. The algorithm for reasoning about semantic relationships is described in detail and it is proved that the algorithm is complete and sound. Finally, the optimization techniques which are used to improve the performance of the reasoning systems are described.

4.2 COMPONENTS OF REASONING SYSTEM

The semantic relationship based reasoning involves the implication of relationship between the given concepts from the known relationships that exist among other concepts. The various components of the reasoning system are depicted in Figure 4.1. In this work, effort has been made to retrieve the concepts and relationships from

![Fig 4.1. Components of Reasoning System](image-url)
various resources such as ontology and datasets. The classification of the extracted relationships is achieved with the consideration of the semantics of the relationships such as synonym, hypernymy, hyponymy, meronymy.

The reasoning algorithm checks for consistency and satisfiability of the retrieved relations and concepts. The reasoning algorithm tries to discover the existing immediate relationships and/or collateral relationships between the given concepts. The strength of the relationships is derived and the derivation principle is applied to infer the novel relationships. Next section describes the way in which the semantic relationships are extracted from structured information and unstructured information.

4.3 SEMANTIC RELATIONSHIPS EXTRACTION

The goal of extracting semantic relationships from structured and unstructured information is, to learn relations between concepts. In structured information, the information is represented using various representation languages like XML, RDF and OWL. OWL provides facilities to express more semantics as compared to XML and RDF. OWL is conceptually based on DL. DL is widely used to create terminological knowledge base, database schema and queries. The semantic relationships are extracted from TBox and ABox and represent the extracted concepts and semantic relationships in RBox (Relational Box). OWL DL supports the expressiveness of $\text{SHOIN}(\mathcal{D})$. For example, in OWL, the concepts are represented as follows:

```xml
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
xml:base="http://www.semanticweb.org/shanthibala/ontologies/2013/6/untitled-ontology-3"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdf="http://www.w3.org/1999/02/rdf-syntax-ns#"
ontologyIRI="http://www.semanticweb.org/shanthibala/ontologies/2013/6/untitled-ontology-3">
  <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
  <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
  <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
  <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
  <Declaration>
    <Class IRI="#Application_Based_IDS"/>
  </Declaration>
  <Declaration>
```

50
The semantic relationships between the concepts and the properties of the relationships in OWL are represented as follows:

  <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
  <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
  <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
  <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>

  <Declaration>
    <ObjectProperty IRI="#adjacent_to"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="#part_of"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="#affect"/>
  </Declaration>

  <owl:FunctionalProperty rdf:ID="improve">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      This property indicates whether an individual improve the performance.
    </rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdfs:domain rdf:resource="#Computer_Networks"/>
  </owl:FunctionalProperty>
  <SymmetricObjectProperty>
    <ObjectProperty IRI="#adjacent_to"/>
  </SymmetricObjectProperty>
  <SymmetricObjectProperty>
    <ObjectProperty IRI="#part_of"/>
  </SymmetricObjectProperty>
  <TransitiveObjectProperty>
    <ObjectProperty IRI="#affect"/>
  </TransitiveObjectProperty>
  <ReflexiveObjectProperty>
    <ObjectProperty IRI="#is_a"/>
  </ReflexiveObjectProperty>
</Ontology>
OWL DL provides maximum expressiveness without losing computational completeness and decidability of the reasoning systems. It supports all the language constructs with restrictions. For example, the DL representation of the University ontology is given below:

PhDStudent

\[
\text{PhDStudent} \subseteq \forall \text{publication Publication} \\
\text{PhDStudent} \subseteq \forall \text{supervisor AcademicStaff} \\
\text{PhDStudent} \subseteq \forall \text{Graduate} \\
\text{PhDStudent} \subseteq \forall \text{worksAtProject Project}
\]

PhDThesis

\[
\text{PhDThesis} \subseteq \text{Thesis}
\]

Proceedings

\[
\text{Proceedings} \subseteq \forall \text{publisher Organization} \\
\text{Proceedings} \subseteq \forall \text{organization Organization} \\
\text{Proceedings} \subseteq \forall \text{editor Person} \\
\text{Proceedings} \subseteq \text{Publication}
\]

Product

\[
\text{Product} \subseteq \forall \text{developedBy Organization}
\]

Project

\[
\text{Project} \subseteq \forall \text{projectInfo Publication} \\
\text{Project} \subseteq \forall \text{head Employee} \\
\text{Project} \subseteq \forall \text{carriedOutBy Organization} \\
\text{Project} \subseteq \forall \text{isAbout ResearchTopic} \\
\text{Project} \subseteq \forall \text{member Person} \\
\text{Project} \subseteq \forall \text{financedBy Organization}
\]

The concepts and relations are extracted from ontology and stored in RBox for further processing. However, extracting semantic relations between concepts in unstructured information is a crucial task. The turnover of unstructured information into structured information is achieved by annotating semantic information. In this work, the relationships from unstructured information are extracted using semi-supervised approach.
In this approach, the unstructured information is labeled and generated the positive and negative training dataset. The relationships are extracted from the unstructured information and retain only dependable relationships with associated concepts. It makes use of dependency parser (Klein and Manning, 2003) to extract the relations. Figure 4.2 shows the algorithm for relationships extraction. In this, all possible relations and concepts are extracted. In this work, the ontologies for various domain such as biology, chemistry, computer science and tourism domain are used. The dataset such as BioInfer, Reverb, SemEval2010 and Wikipedia are used for extracting semantic relationships and associated concepts.

<table>
<thead>
<tr>
<th>Algorithm for Relationships Extraction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Ontology or Dataset represented as K</td>
</tr>
<tr>
<td><strong>Output:</strong> Concepts with associated relations</td>
</tr>
<tr>
<td><strong>RelExtract (K)</strong></td>
</tr>
<tr>
<td><strong>Begin</strong></td>
</tr>
<tr>
<td>Concept tconcept = main;</td>
</tr>
<tr>
<td>int no_of_relations_found=0;</td>
</tr>
<tr>
<td>While (tconcept! =null &amp;&amp;! tconcept.isleaf() &amp;&amp; no_of_relations_found &lt; K.length) {</td>
</tr>
<tr>
<td>Relation nextrelations= choose relations from tconcept.relations</td>
</tr>
<tr>
<td>If (nextrelation !=null) {</td>
</tr>
<tr>
<td>Concept=nextrelation.possibleconcept</td>
</tr>
<tr>
<td>Retrieve properties of the relations</td>
</tr>
<tr>
<td>No_of_relation_found= nextrelation.length + 1</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>tconcept = null;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>Return (Concept &amp;&amp; no_of_relations_found &amp;&amp; Relations)</td>
</tr>
<tr>
<td><strong>End</strong></td>
</tr>
</tbody>
</table>

Fig 4.2. Algorithm for Relationships Extraction
The extracted relationships and concepts from ontology and various datasets are stored in the form of tuples i.e. \(<C_1, R_1, C_2>\) where \(C_1\) and \(C_2\) denotes the concepts and \(R_1\) denotes the semantic relationship that exist between the concepts \(C_1\) and \(C_2\). The properties of the relations are retrieved with the corresponding matrix representation of the retrieved concepts. The semantic relationships hold the reflexive property if the matrix is in the following form:

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 2 \\
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1
\end{pmatrix}
\]

The semantic relationships hold the irreflexive property if the represented matrix is in the following form:

\[
\begin{pmatrix}
0 & 0 & 1 & 0 & 2 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{pmatrix}
\]

The other relationship properties such as symmetry, asymmetry, transitive and functional properties have been identified with the use of matrix representation. The given relation is symmetry, for every value in the matrix is equal to its transposed position. When the transposed position of the given matrix is not equal with the corresponding value, then the given relation holds asymmetry. The asymmetry relation also satisfies irreflexive property. The transitive relations are identified with every possible transitive values in matrix is 1. In relationship extraction, the semantic relationships and corresponding concepts are extracted from various sources and stored in RBox for classification. The implementation and the experimental results of the semantic relationship extraction algorithm have been dealt in Chapter 7.

This section explained about the method through which the semantic relationship and concepts are extracted. Next section describes the technique which is used to classify the semantic relationships.
4.4 SEMANTIC RELATIONSHIPS CLASSIFICATION (SeRelaC)

The extracted semantic relationships are primitive. It could not be used effectively intrinsically. Thus, there is a need of semantic relationship classification. A wide range of semantic classification techniques are developed which are based on the need and granularity of the various applications. The main issue in relationship classification is the consideration of semantics and properties of the relationships. In this work, lexical resource i.e. WordNet has been used to provide semantics of the relationships.

The semantic relationships have been classified into various categories such as Hypernymy, Hyponymy, Meronymy, Casual, Entailment and Spatial. To enhance the classification accuracy, the inputs are represented properly. It includes categorical

**Algorithm for SeRelaC**

**Input:** Categorical Features as F, Set of relationships as R and concepts as C

**Output:** Classified relationships SR

**SeRelaC (F, R, C)**

**Begin**

Set \( n \) = Number of relationships

Repeat

For \( i=1 \) to \( n \) do

Calculate the depth(\( R_i \), \( R_j \)) with Lexical Resources where \( i=1,2, \ldots, n \)

-1; \( j=2,3, \ldots,n \)

If (depth \( \geq 1 \))

Add it into category SR [ ] ; where SR [ ] = {Hypernymy, Hyponymy, Meronymy, Causal, Entailment, Spatial}

Else Discard

End for

For all relationships

// Find semantics of the relations

**Begin**

Read R and F

Find relatedwords (R), relatedwords (F) // use WordNet

**End**

Run SVM classifier

Add into SR [ ]

Until No relationships available

**End**

**Fig 4.3.Algorithm for Relationships Classification**

The semantic relationships have been classified into various categories such as Hypernymy, Hyponymy, Meronymy, Casual, Entailment and Spatial. To enhance the classification accuracy, the inputs are represented properly. It includes categorical
features, concepts and the relationships between the concepts. Categorical feature includes the semantic relations between the concepts as found in WordNet such as Hypernymy, Hyponym, Meronymy, Cause-Effect, Spatial and Entailment. The relationship is derived using \textit{findRelationshipsDemo}(R1, R2, Pointer Type, Relation Category) function. Support Vector Machine (SVM) is used to classify the semantic relationships that exist between the concepts. It provides more accuracy as compared to other classification algorithms. The SeRelaC algorithm is shown in Figure 4.3.

The classified semantic relationships are used to predict accurately, the relationships for each relationship in the set and also used to enrich the communication among the represented concepts. The classified relationships and extracted concepts are stored in the knowledge base. The implementation and the experimental results of the SeRelaC algorithm have been dealt in Chapter 7. This section elucidated the importance of semantic relationships classification and algorithm used to classify the semantic relationships. Next section reveals the semantic relationship based reasoning approach.

\textbf{4.5 SemPER ALGORITHM}

Semantic relationships are used to strengthen the structure of the concepts. It provides well defined meaning to realize more about concepts. Lack of semantic relationships in domain knowledge provides diversified perception among users. It also establishes inconsistency among knowledge users. This could be adjudicated by inferring relationships between the concepts.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig4.4.png}
\caption{Architecture of SemPER}
\end{figure}
The proposed SemPER algorithm deduces the novel semantic relationships between the concepts. The architecture of SemPER is shown in Figure 4.4.

The semantic relationships can have n-ary relationships. In this work, the binary relations are considered. In DL, it can be represented as follows:

\[ C_1 \subseteq \exists \mathbin{\forall} R. C_2 \]

Here C1 and C2 represent concepts and R represents relationship that exists between the given concepts. Semantic relationship based reasoning is performed on the basis of the following model.

4.5.1 SemPER Formal Model

Definition 1: The Domain knowledge \( D \) is a collection of concepts, relationships, their properties and individuals for a particular domain. Domain interpretation consists of non-empty set \( \Delta^D \) and an interpretation function \( I \).

Definition 2: The concepts \( C \) are collection of concepts for a given domain. It is represented as \{\( C_1, C_2, \ldots, C_n \)\}. Domain interpretation function \( I \) which maps every concepts \( C \) is a subset of \( \Delta^D \), i.e. \( C^D \subseteq \Delta^D \).

Definition 3: Semantic relationships \( R \) are collection of relationships that exists between the given concepts for given domain. The relation \( R \) is a binary relation which assigns every relation to a subset of \( \Delta^D \times \Delta^D \), i.e. \( R^D \subseteq \Delta^D \times \Delta^D \).

The semantic relations that exists between the concepts \( C_i \) and \( C_j \) are represented as

\[ R_{ij}, \text{ where } i = 1, 2, \ldots, n-1, j = i+1; \text{ n – no. of concepts} \]

The relationships that does not exist between the concepts \( C_i \) and \( C_j \) are represented as

\[ R_{ij} = \text{NULL} \]

Definition 4: Properties \( P \) are collection of properties of the relations for a given domain. It holds transitive, symmetric, asymmetric, reflexive and irreflexive properties.
A relation \( R \) is transitive iff
\[
\forall C_1, C_2, C_3: (C_1, C_2) \in R \cap (C_2, C_3) \in R \rightarrow (C_1, C_3) \in R
\]

A relation \( R \) is symmetric iff
\[
\forall C_1, C_2: (C_1, C_2) \in R \Leftrightarrow (C_2, C_1) \in R
\]

A relation \( R \) is asymmetric iff
\[
\forall C_1, C_2: (C_1, C_2) \in R \rightarrow \neg (C_2, C_1) \not\in R
\]

A relation \( R \) is reflexive iff
\[
\forall C_i: (C_i, C_i) \in R
\]

A relation \( R \) is irreflexive iff
\[
\forall C_i: (C_i, C_i) \not\in R
\]

**Definition 5:** Instances \( I \) are collection of individual elements that belongs to each concepts.

By the definition, the domain knowledge can be represented as \{\( C \), \( R \), \( P \), \( I \)\}. The concepts, relations and properties are represented in TBox. The instances corresponding to TBox is represented in ABox. ABox has concept assertion and relation assertion which is represented as \( C(x) \) and \( R(x,y) \). RBox has relations with associated concepts. The algorithm should prove its satisfiability in order to deduct subsumption, consistency and equivalence. The rules used for satisfiability algorithm (Baader and Nutt, 2003), is given in Figure 4.5. To prove the satisfiability of the semantic relations that exists between the given concepts, the following properties have been adopted.

**Property 1:** Any relation \( R \) that belongs to the particular domain iff
\[
R = \{ R_i/R_i \in D_i \cap R_i \in R \}, \text{ where } i = 1, 2, \ldots, n
\]

**Property 2:** The relation that does not exists in the particular domain iff
\[
R = \{ R_i/R_i \not\in D_i \cap R_i \in R \}, \text{ where } i = 1, 2, \ldots, n
\]
**Property 3:** The relations that exist between more than two concepts satisfies

\[ R_i \times R_j = \{(C_i, C_j) / \exists C_k (C_i, C_k) \in R_i \cap (C_i, C_k) \in R_j\} \]

**Property 4:** The relations \( R_i \) is containment of another relation \( R_j \) iff

\[ R_i \subseteq R_j = \{(C_i \in D_i) / \forall C_j (C_i, C_j) \in R_i \rightarrow (C_i, C_j) \in R_j\} \]

**Property 5:** The relations \( R_i \) is equal to another relation \( R_j \) iff

\[ R_i = R_j = \{(C_i \in D_i) / \forall C_j (C_i, C_j) \in R_i \leftrightarrow (C_i, C_j) \in R_j\} \]

\[ \cup \text{ Rule: If } A \text{ contains } C_i(x) \cap C_j(x), \text{ but it does not contain } C_i(x) \text{ and } C_j(x) \]
\[ \text{ then } A = A \cup \{C_i(x), C_j(x)\} \]

\[ \cap \text{ Rule: If } A \text{ contains } C_i(x) \cap C_j(x), \text{ but neither } C_i(x) \text{ nor } C_j(x) \]
\[ \text{ then } A = A \cup \{C_i(x), C_j(x)\} \]

\[ \exists \text{ Rule: If } A \text{ contains } \exists(R.C)(x), \text{ but it does not contain } C_i(x) \text{ and } C_j(x) \]
\[ \text{ then } A = A \cup \{C_i(y), R(x, y)\} \]

**Fig 4.5. Rules for Satisfiability**

These rules have been used to perform consistency check over retrieved relationships. SemPER algorithm employs tableau algorithm for satisfiability checking. This algorithm performs satisfiability checking over the relations i.e. checks whether the relation exists between the given concepts for any instances. It retrieves all possible relations between the concepts even if the indirect relation persists. The algorithm is shown in Figure 4.6. Initially, the algorithm verifies the direct relationships and then the properties of the relationships. The relationships are retrieved from explicitly specified knowledge. The collateral relationships are retrieved and novel semantic relationships are inferred on the basis of semantic relation strength and semantic derivation principle. The strength of the semantic relationship can be obtained with the use of lexical resource(s). For example, the strength for hypernymy semantic relationships is identified with the help of following code:

```java
IndexWord start = WordNetHelper.getWord(POS.VERB, s1);
IndexWord end = WordNetHelper.getWord(POS.VERB, s2);
Relationship rel = WordNetHelper.getRelationship(start, end, PointerType.HYPERNYM);
if (rel != null){
```
Semantic relationships are inferred using semantic derivation principle. The derivation principle is defined by $\varphi$ which infer novel semantic relations from existing semantic relationships. The semantic relationships such as Hypernymy ($HR$),
Hyponym (HO), Meronymy (MR), Cause-effect (CE), Spatial (SP) and Entailment (EN) are considered. The semantic derivation principle is shown in Table 4.1.

<table>
<thead>
<tr>
<th>HR</th>
<th>HO</th>
<th>MR</th>
<th>CE</th>
<th>SP</th>
<th>EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>HR</td>
<td>HO</td>
<td>MR</td>
<td>CE</td>
<td>SP</td>
</tr>
<tr>
<td>R2</td>
<td>HR</td>
<td>HO</td>
<td>MR</td>
<td>CE</td>
<td>SP</td>
</tr>
<tr>
<td>HO</td>
<td>HO</td>
<td>HO</td>
<td>MR</td>
<td>CE</td>
<td>SP</td>
</tr>
<tr>
<td>MR</td>
<td>MR</td>
<td>MR</td>
<td>MR</td>
<td>CE</td>
<td>SP</td>
</tr>
<tr>
<td>CE</td>
<td>CE</td>
<td>CE</td>
<td>CE</td>
<td>CE</td>
<td>SP</td>
</tr>
<tr>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>CE</td>
<td>SP</td>
</tr>
<tr>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>CE</td>
<td>EN</td>
</tr>
</tbody>
</table>

Table 4.1 Semantic Derivation Principle

The semantic relationships are inferred using semantic derivation principle \( \varphi \) and the semantic relation strength is calculated to deduct the relationship between the concepts. In this algorithm, the derived relationships are based solely on the semantics and properties of the relationships. The proposed algorithm is proved to be sound and complete.

4.5.2 Proof for Soundness

**Lemma 1:** The relation exists between the concepts is consistent iff it is satisfiable

**Proof:**

To prove there exist a relation R between the given concepts i.e. \( \exists R (C_1, C_2) \)

To prove it consider,

\[
\exists R (C_1) \land \exists R (C_2) \subseteq \exists R (C_1 \cap C_2)
\]

(1)

\[
\exists R (C_1) \land \exists R (C_2) \land \neg (\exists R (C_1 \cap C_2)) \implies C \subseteq D = C \cap \neg D
\]

By Demorgan’s Theorem,

\[
\exists R (C_1) \land \exists R (C_2) \land (\forall R (\neg C_1 \cap \neg C_2))
\]

(2)
We have to prove that,

\[ \exists R (C_1) \cap \exists R (C_2) \cap (\forall R (\neg C_1 \cap \neg C_2)) \neq \emptyset \]

Consider \( x \) is an instance which belongs to Eq. (2)

It should satisfy,

\[ x \in \exists R (C_1), x \in \exists R (C_2) \text{ and } x \in (\forall R (\neg C_1 \cap \neg C_2)) \]

To deduce \( x \in \exists R (C_1) \),

Consider there exists an another individual element \( y \) such that,

\[ (x, y) \in R \text{ and } y \in C_1 \]

(3)

To deduce \( x \in \exists R (C_2) \),

Consider there exists an another individual element \( z \) such that,

\[ (x, z) \in R \text{ and } z \in C_2 \]

(4)

Assume that \( y \neq z \)

It should satisfy,

\[ x \in (\forall R (\neg C_1 \cap \neg C_2)) \]

To deduce it, consider there exists some individual elements \( y \) and \( z \).

Finally we get,

\[ y \in \neg C_1 \text{ Clashes with Eq. (3)} \]

We must choose,

\[ z \in \neg C_1 \text{ to satisfy the constraint } x \in (\neg C_1 \cap \neg C_2) \text{ without having any contradiction.} \]

Hence, it is proved that it is satisfiable. Hence, it is true that Eq. (1) is consistent.
4.5.3 Proof for Completeness

Lemma 2: For every relation and concepts for a Domain, the SemPER terminates.

Proof:

Let $R_i$ be a semantic relation for a domain. Using property (1), it is verified that it belongs to domain. It can be proved that $\exists R_i \in (C_i, C_j)$ using property (3). The relation subset and relation equality is verified with property (4) and (5). If the relation doesn’t belong to the particular domain, it satisfies property (2).

Thus, the soundness and the completeness of the algorithm is proved. The optimization is necessary for improving the performance of the reasoning system. It helps to reduce the execution time of the reasoning algorithm. Next section explains the optimization techniques that are adopted in our proposed system.

4.5.4 Optimization

The knowledge base (TBox, ABox and RBox) is constructed from the set of ontologies and corpus. It consists of large and complex concepts and relationships. Reasoning over unpruned knowledge base will affect the performance of the system. Hence, there is a necessity of optimization to prune the knowledge base to improve the performance of the reasoning system. The proposed SemPER algorithm performs preprocessing and satisfiability checking optimizations (Tsarkov and Horrocks, 2006). The preprocessing optimization includes normalization, absorption and cycle elimination. The satisfiability checking optimization includes dependency directed backtracking and semantic branching.

4.5.4.1 Preprocessing Optimization

The preprocessing optimization is used to eliminate the redundancy and the general concept axioms which are not necessary. Normalization technique is used to transform the complex concepts into simplified normal form by identifying contradictions and tautologies. The simplified normal form includes the operators such as negation ($\neg$), conjunction ($\cap$) and universal quantification ($\forall$). Normalization is widely used and it can be used to detect subsumption and perform satisfiability check. Absorption can be used to eliminate the axiom in the form of $C \subseteq D$. The
general concept inclusion axiom is absorbed into concept definition axiom and relation definition axioms to increase the performance during satisfiability and subsumption test. The cycle is eliminated by transforming the axioms into its $C \equiv D$ representations. It helps to reduce the processing time for performing satisfiability check over the concepts and relationships.

### 4.5.4.2 Satisfiability Checking Optimization

Satisfiability checking optimization is used to reduce the search space. In dependency directed backtracking, the algorithm will backtrack and find an alternative solution if the branching leads to any clash.

**ALGORITHM : Satisfiability Test**

**Input:** A Knowledge Base $K (T, A, R)$ in negation formal Form

**Output:** Return true if $K$ is satisfiable and false if not

**Begin**

Construct And-Or graph $G$ with root node $N_0$ for $K$

Set unsatnodes = $\phi$

If $G$ contains a node $N_\perp$ with $\{ \perp \}$

$N = \{ N_\perp \}$

While $N$ is $\neg \phi$ do

Get node $n$ from $N$

$\forall$ Ancestor node of $n_1$ of $n$

If $n_1 \notin$ unsatnodes and $n_1$ belongs to either ‘and’ or ‘or’ node and all successor belong to unsatnodes then

Add $n_1$ to unsatnodes and $N$

End

End

Return True if $N \notin$ unsatnodes and false otherwise

**End**

Fig 4.7. Satisfiability Checking Algorithm
It is used to reduce the size of the search space and to perform the heuristic search. Semantic branching is widely used and highly effective to reduce the search space and rewrite the axioms into clash-free. Boolean Constant Propagation (BCP) is used to reduce the search space in semantic branching. The satisfiability checking algorithm is also adopted to perform satisfiability test, which is shown in Figure 4.7.

The implementation and the experimental results of the SemPER algorithm have been dealt in Chapter 7. The SemPER algorithm is very effective in information retrieval and question generation system.

4.6 SUMMARY

This chapter discussed the necessity of semantic relationship based reasoning algorithm. The components of the reasoning algorithms such as semantic relationships extraction, semantic relationships classification and semantic relationship based reasoning algorithm are elaborated in detail. This chapter also discussed about the proof for soundness and completeness of the proposed SemPER algorithm. It also explained about the optimization techniques that are adopted to improve the performance of the SemPER. Next chapter discusses about the information retrieval and ranking system with semantic relationships which incorporates SemPER reasoning algorithm.