Organizing the Semantics of Text with the Concept Relational Tree

Ungku Azmi Ungku Chulan[†], Md. Nasir Sulaiman[†], Ramlan Mahmod[†], Hasan Selamat[†], Jamaliah Abdul Hamid[††]

[†]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia
[††]Faculty of Education, Universiti Putra Malaysia, Malaysia

Summary
In this paper, a new model of representing semantics called the concept relational tree model is proposed. The model adapts the architecture of expression tree in providing a hierarchical organization to semantics. It is motivated by an approach known as the discourse structure tree that organizes semantics based on preposition and relies on rhetorical relations to define the connection between the prepositions. Comparatively, the concept relational tree uses concept and relation as its organizational unit instead. This allows the semantics of text to be defined at a finer level, which consequently enables a more flexible way of controlling the structure of semantics.

Key words:
Semantics, Parsing, Relation Extraction

1. Introduction

One of the common task in relation extraction [15] is finding the related concepts and the corresponding relation that connects them. Each extraction can be defined simplistically as $R(C_1, C_2)$ where R is the relation that connects the two concepts C_1 and C_2. In a sentence that consists of many terms, the objective is to search for the terms that relate to one another and knowing how they relate. For instance, there are four extractions for the sentence below (Figure 1.0).

The color of the toy catches they boy's attention during recess

of(color, toy)
has(boy, attention)
catches(toy, attention)
during(catches(toy, attention),recess)

Figure 1.0 : Relation Extraction

Some extractions like 'of(color,toy)' can be easily identified. However, a relation like 'catches(toy, attention)' might be more difficult to extract since the concepts are further away from each other. This can create a chain reaction that deters extraction accuracy since an extraction can be used as the parameter for another extraction. For example (Figure 1.1), the extraction 'during(catches(toy, attention), recess)' uses the extraction 'catches(toy, attention)' as one of its parameter. In effect, if a particular extraction is erroneous, other extractions that use it will be invalid too.

The **color of** the **toy** catches they boy's attention during recess
of(color, toy)

The **color** of the toy **catches** they boy's **attention** during recess
catches(toy, attention)

catches(toy, attention)
during(catches(toy, attention),recess)

Figure 1.1 : Difficulty in Extraction

In order to assists the effort of finding the related concepts to be extracted, text is usually structured in a hierarchical form [8]. Discourse structure tree [1] can be employed to organize the semantics of text in an incremental manner [6], where by the hierarchy of meaning is augmented iteratively with the progressing levels of the tree. This incremental approach ensures the proper partitioning of semantics. The relationship between concepts can be searched between the segmented regions instead of the whole text. Thus, making the effort of concept identification easier.

For the sake of illustration, consider the semantics of the sentence in the example (Figure 1.3), which is organized hierarchically using the discourse structure tree. Semantics is decomposed into separate regions and a more focused analysis can be done. As a result, it is easier to identify the concepts that are related to one another. To quote an example, the related concepts 'she' and 'John' can be extracted by analyzing the node 'she tricked John' without having to probe the entire text.

Manuscript received September 5, 2008.
Manuscript revised September 20, 2008.

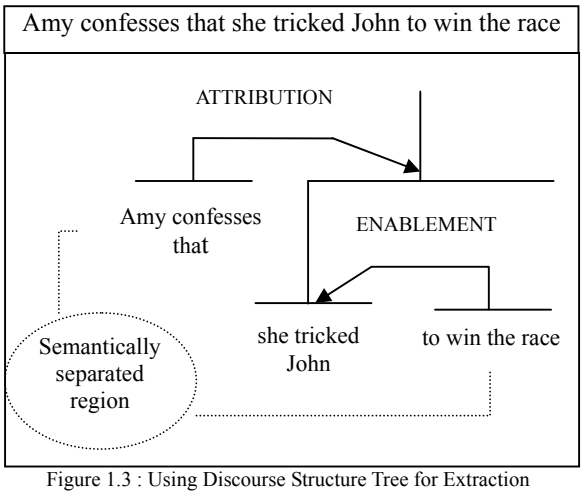

Figure 1.3 : Using Discourse Structure Tree for Extraction

Despite its undeniable potential, discourse structure tree does suffer from some innate setbacks from the perspective of relation extraction. For one, it relies on rhetorical relations to work. This can prove to be detrimental to the coverage of the approach. Not all text uses the markers demanded by rhetorical relations [13]. Due to this, the approach may not be applicable in certain text, paralyzing the entire endeavor of building the tree. For instance, the rhetorical relation for the two text spans below (Figure 1.4) is RESULT. However, since no markers can be identified, they will not be properly analyzed.

Amy made funny expressions to the little girl, she laughed at them.
[Amy made funny expressions to the little girl]
[she laughed at them].

Figure 1.4 : Non-identifiable Rhetorical Relation

Another problem lies in the way the unit of the tree is defined [12]. The organizational unit for discourse structure tree does not necessarily have to be concept and relation. Defining the unit in this manner can sometimes lack the level of granularity required for relation extraction. In the example (Figure 1.5), the notion of 'finer' decomposition is illustrated. Although the connection between the two units 'she tricked John' and 'to win the race' is known, it is hard to determine precisely the connection between the concepts within them. In effect, it is not known for certain which relation is correct from the tree. Which concept from 'she tricked John' is connected to 'race'? Is it 'she' or 'John'.

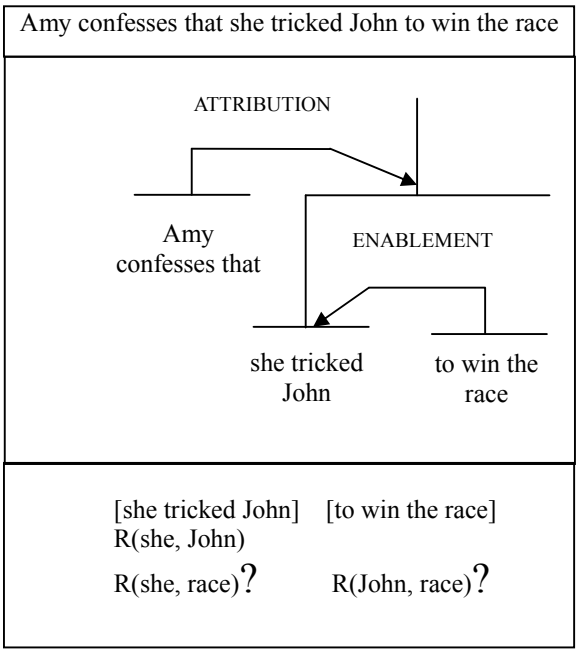

Figure 1.5 : Granularity of Discourse Structure Tree

2. Related Work

In order to assists the effort of finding the related concepts to be extracted, text is usually structured in a hierarchical form [8]. Many kind of structures have been used for this purpose. This includes parse tree, dependency tree and discourse structure tree. Dependency tree is being used widely in text processing application. It is usually employed to depict the relationship between words by organizing them into head and dependents (Figure 2.0). By relying mostly on syntax instead of semantics, the dependency tree is easier to construct as compared to the discourse structure tree. The tree however, does not decompose semantics into separate regions. Instead, it provides a word by word organization that may not necessarily ease the concept identification process.

Dependency tree loosely defines the nodes of the tree as well as the head-dependent relation that connects the nodes together. As such, connected words can imply too many possible relationships, including determiner-noun, subject-verb, verb-object and so on. Finally the dependency tree is non incremental as perceived by the construct of semantics. The higher level meaning of the tree does not build upon lower level ones. In the illustration, the determiner 'an' lies at the same level of the functor 'of'. Rationally, this should not be the case since 'of' defines the connection between the two concepts 'example' and 'virtue' while the determiner 'an' is merely an article that corresponds to a concept in isolation.

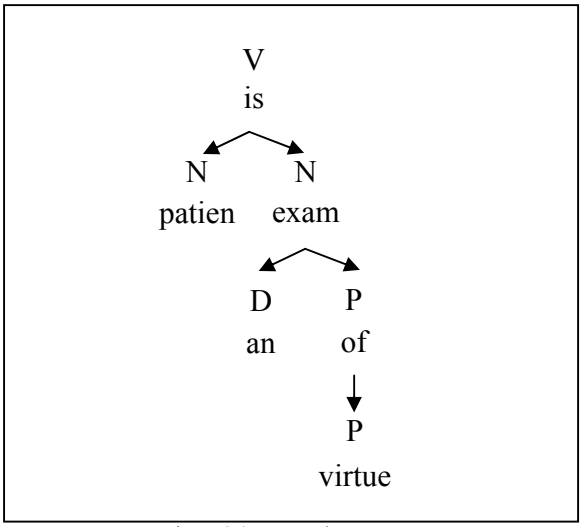

Figure 2.0 : Dependency Tree

3. Concept Relational Tree

To solve the issue of granularity and coverage suffered by discourse structure tree, the concept relational tree is proposed. The concept relational tree is inspired from the expression tree [5], a binary tree used to evaluate a mathematical expression in deciding the proper sequence of operations. For instance, the expression below can be evaluated in two ways.

2+3*6 = 5 * 6 = 30
2+3*6 = 2 + 18 = 20

As we all know, the latter is the correct one. Multiplication should always precede addition. As such, to find the correct manner of evaluation, the mathematical expression is represented in the form of an expression tree. The expression tree is then traversed to find the value of the expression. In the illustration (Figure 3.0), '3 * 6' is traversed first to result to '18'. Then, '2' is added to '18' for the final result which is '20'.

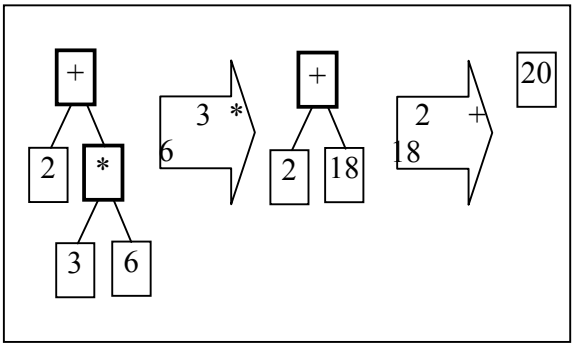

Figure 3.0 : Expression Tree Evaluation

An expression tree is made of two basic units, operand and operator (Figure 3.1). Conceptually, the operand is the unit of information for the expression while the operator signifies the connection between these units of information. In the illustration, '2', '3' and '6' are numerical information within the expression. Thus, they arethe operands. On the other hand, '*' does not carry any information. Instead, it connects '3' and '6'. This makes '*' an operator.

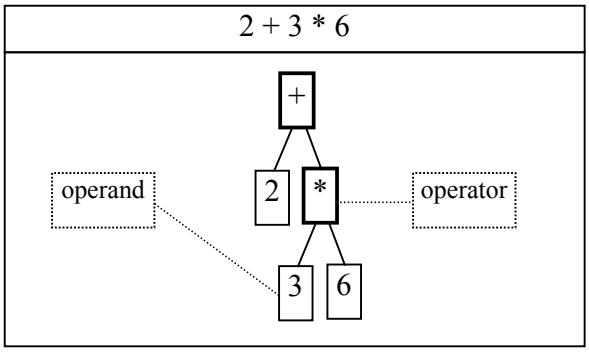

Figure 3.1 : Operand and Operator

For this research, the representation of semantics is assumed to be analogous to the problem of evaluating a mathematical expression. This compels the adoption of the expression tree's architecture. It is assumed that the components of text can be organized into two classes like operands and operators that are called concepts and relations [3] respectively. Concepts function in a similar way as operands in storing information while relations work as the elements that connect these concepts. Rationalizing in this manner, it is possible to materialize the illustration below (Figure 3.2) that works as a catalyst in suggesting how the expression tree may be utilized in representing semantics.

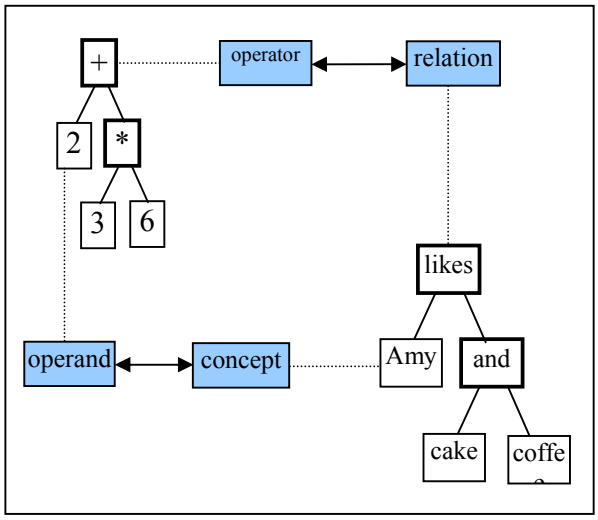

Figure 3.2 : Expression Tree to Concept Relational Tree

3.1 Organizational Unit

To represent text using the concept relational tree, the components of text must first be represented in term of concept and relation (Figure 3.3). This is done by modeling a sentence using the concept relational model [4]. With this model, all the words in the sentence are classified into concepts and relations. Nouns are categorized as concepts [11] while certain functors like prepositions [2] as well as verbs [7] are deemed as relations.

Each organizational unit contains attribute that extends the meaning of the unit [14]. An adjective is an examples of an attribute since it extends the meaning of a concept. For instance, in the phrase 'cute cat', the adjective 'cute' further narrates the feature of the concept 'cat'. Now, when an attribute resides within a concept, it is called concept attribute. On the other hand, when it is contained within a relation, it is defined as relation attribute. The adverb 'slowly' in 'Amy writes slowly' is an example of a relation attribute for the relation 'writes'.

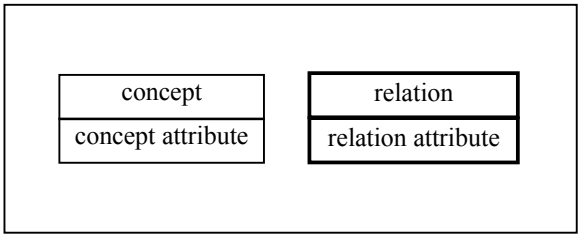

Figure 3.3 : Organizational Unit

For the sake of better illustration (Figure 3.4), consider the sentence below which is made of two concepts and a relation. The two concepts are 'Amy' and 'cake', while the single relation is 'eats'. Now, the term 'cute' is an adjective that explains the concept 'Amy'. Consequently, 'cute' is the concept attribute of 'Amy'. The relation 'eats' connects 'Amy' and 'cake' and is further explained by the adverb 'happily'. Therefore, 'happily' is the relation attribute of 'eats'.

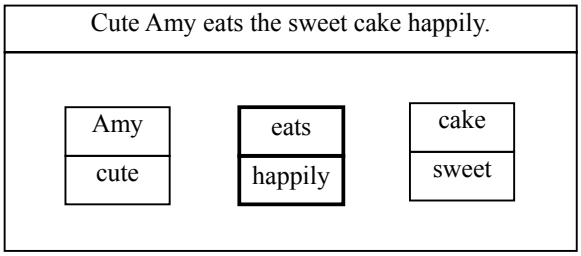

Figure 3.4 : Sentence Organization

It is possible to define the set of terms in a sentence into concept, relation and attribute using the following mapping (Table 5.1). Certain part of speech tags that are not included in the mapping such as determiner is considered trivial for relation extraction and therefore omitted [9].

Table 5.1: Part-of-Speech Mapping to Concept Relational Model

Tag	***Description***	***CR-Tag***
NN | NNP | NNPS | NNS	noun	C
VB | VBD | VBG | VBN | VBP | VBZ	verb	R
JJ | JJR | JJS	adjective	A_C
RB | RBR | RBS	adverb	A_R
PRP | PRP$	pronoun	C
CC	conjunction coordinating	R
IN	preposition or conjunction subordinating	R
CD	numeral cardinal	A_C
POS	genitive marker	R
TO	"to" as preposition or infinitive marker	R

Tag	*Description*	*CR-Tag*
WDT | WP | WP$ | WRB	WH-(determiner | pronoun | adverb), possessive	R
RP	particle	R

3.2 Tree Structure

Once text is decomposed into a set of organizational units, it can be constructed into the concept relational tree. Structurally, the concept relational tree is a binary tree with concept and relation as its nodes. The tree (Figure 3.5) has the following properties:

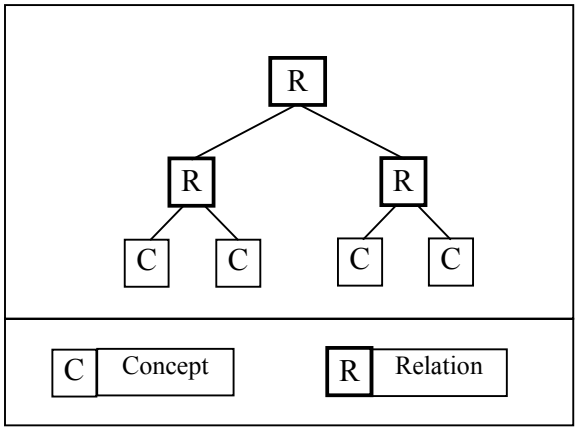

Figure 3.5 : Concept Relational Tree

1. Node is made of either concept or relation.
   This emulates the expression tree where a node can either be an operand or an operator.

2. Internal nodes are all relations.
   Internal nodes can be seen as the branches of the tree. In the expression tree, all non-terminal nodes are operators. With the same reasoning, the non-terminal nodes in the concept relational tree shall all be relations.

3. External nodes are all concepts. They are the leaves of the tree.
   By following the rationale of theexpression tree, terminal nodes in the concept relational tree are all concepts. A profound consistency is made in the analogy of a tree where branches connect the leaves.

4. A tree is made of a node, a combination of nodes or a combination of this trees.
   The basic definition of a tree (Figure 3.6) starts from the simplest form which is a concept (T =C). Recursively, two trees and a relation are also a tree by itself (T = T R T).

$$T = C \mid T\ R\ T$$

Figure 3.6 : Tree

5. A relation 'describes' the connection between the two trees under it.
   If $T = T_1\ R\ T_2$, then R describes the connection between T_1 and T_2. For a concrete example, consider the phrase 'Amy looks at Susan with a smile' (Figure 3.7). Here, the entire tree can be decomposed into two trees. 'Amy looks at Susan' can imply the first tree T_1 while the concept *"smile"* may represent another tree T_2. In light of this, the connection between the two trees is defined by the relation R = 'with'. Therefore, 'with' the connection between T_1 and T_2. This coincides with the semantics of text where T_1 narrates that 'Amy looks at Susan' while T_2 states the manner in which the act is done. That is, with a 'smile'.

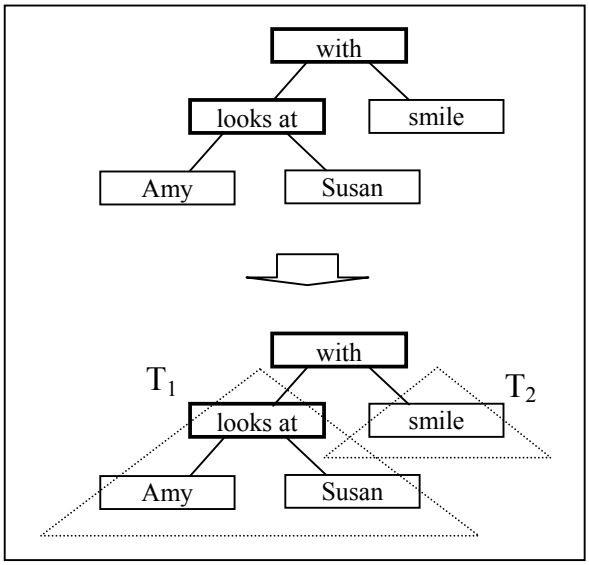

Figure 3.7 : Trees in Concept Relational Tree

3.3 Sentence Nesting

Like the expression tree, the concept relational tree is written using the parenthesis. Each enclosed pair of parenthesis denotes a particular tree. In the illustration (Figure 3.8), since the parenthesis is enclosed for '(Amy sings opera)', it constitutes a tree. When another parenthesis is enclosed in '((Amy sings opera) on stage)', another tree is built. This new tree uses '(Amy sings opera)' as one of its children.

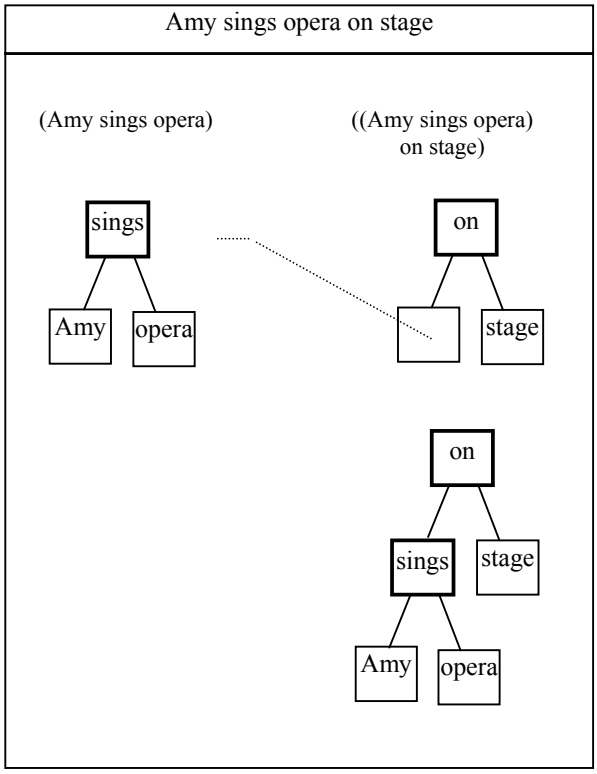

Figure 3.8 : Tree Notation

The entire process of incorporating parenthesis to a sentence is called sentence nesting S_Θ. Sentence nesting (Figure 3.9) stops only when all the components in the sentence are nested. Nesting can also be represented via tree merging where each tree T = C | T R T. Whenever two trees are nested, they are considered to merge. Merging two trees T_{N-1} R_{N-1} T_N within a nesting results to a single tree $T_{(N-1,N)}$. Sentence nesting is complete only when a single tree T_Θ exists. T_Θ is defined as the tree where all the components are properly nested.

$S = C_1 R_1 ... R_{N-1} C_N$
$S_\Theta = (C_1 R_1 ... R_{N-1} C_N)$

$S = T_1 R_1 ... T_{N-1} R_{N-1} T_N$
$S = T_1 R_1 ... (T_{N-1} R_{N-1} T_N) \rightarrow S = T_1 R_1 ... T_{(N-1,N)}$
$S_\Theta = T_\Theta$

Figure 3.9 : Sentence Nesting

3.4 Semantic Modification

The research relies on the idea of defining semantics on the basis of nesting. This impels the notion of semantic modification to surface. It attempts to describe the construction of meaning from the aspect of hierarchical connectivity. Modification is not a new concept in linguistics. Adjective has always been the modifier of nouns. The same goes with an adverb that modifies a verb. These modifications however, only occur at the lexical level. Therefore, it is quite compelling to take the principle further, where meaning is formed by how a word connects to another word. This can be shown from a simple illustration (Figure 3.10) where the concept 'ball' is connected to the concept 'box' in three different ways.

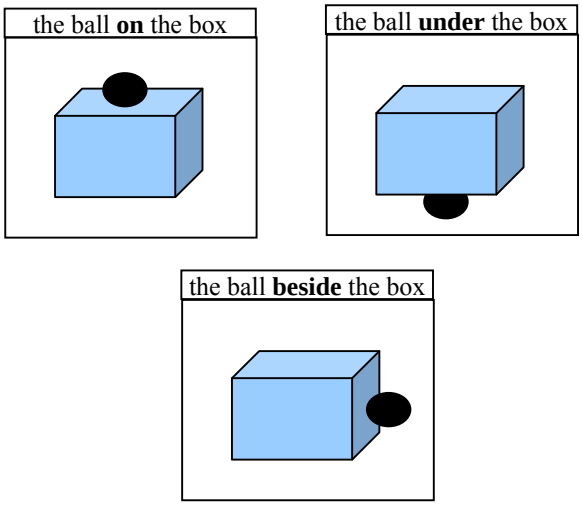

Figure 3.10 : Semantic Modification

As such, when the concept 'ball' and 'box' are mentioned in isolation, they simply exist. However, when a certain relation is introduced between these two concepts, a new meaning emerges. This is the basic principle of semantic

modification where the component of meaning is modified by what and how it is attached to. In the context of the concept relational tree, a component of meaning is the tree itself.

The idea related to semantic modification can also be seen in the prepositional phrase attachment problem [10]. Attachment, which can either be adjectival attachment or adverbial attachment, is basically a dilemma of modification from the paradigm of the this research. To illustrate the point, consider the following example (Figure 3.11) where the simple sentence 'the kid eats the cake' is modified in two ways. Adjectival attachment transpires when modification occurs at the concept 'cake' while adverbial attachment is reflected by a modification that occurs at the relation 'eats'. Modification at 'cake' by 'with cream' implies that the cake contains the cream. On the contrary, the modification at 'eats' by 'with spoon' shows that the eating action is done using the spoon as an instrument.

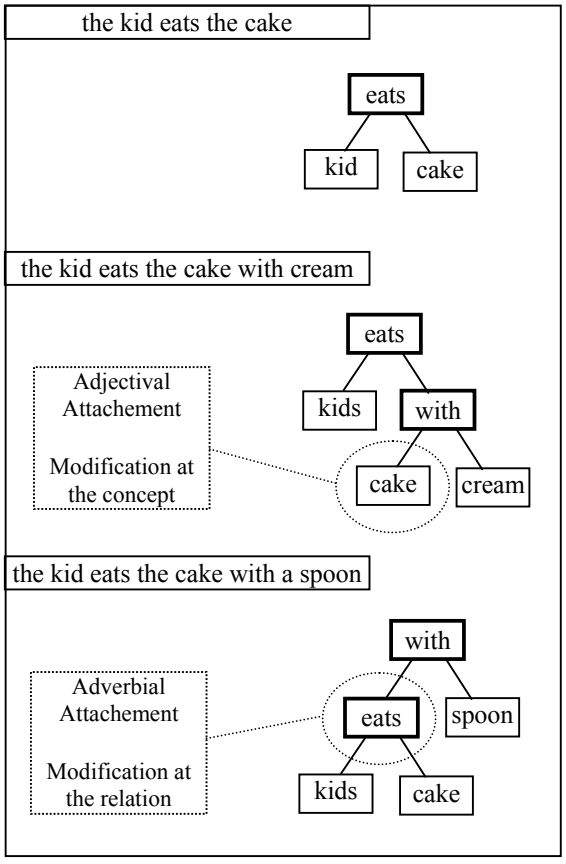

Figure 3.11 : Semantic Modification and the Prepositional Phrase Attachment Problem

For this research, semantic modification is assumed to be hierarchical where higher level trees build upon the semantics of lower level ones. As such, semantics becomes more complex as it goes up the tree. To see this, observe the given example (Figure 3.12). The simplest tree is '(Amy eats soup)' which simply means that Amy consumes the soup. However, when it is attached to 'with Susan', the meaning is modified to include the person with whom Amy eats the soup. Finally, as the tree is further attached with 'on Monday', the meaning no longer tells us who eats soup with whom, but also when it transpires.

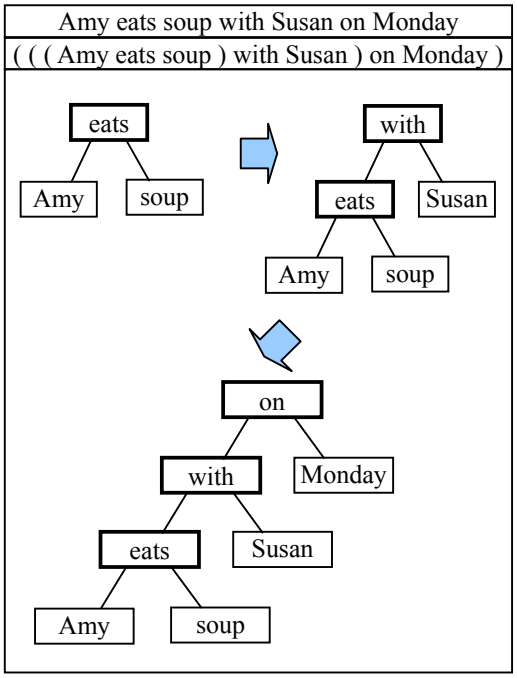

Figure 3.12 : Impact of Tree Levels on Semantics

It would favorable to consider another illustration to see how a particular tree can evolve to reflect the change in semantics (Figure 3.13). What would happen if the sentence is changed into 'Amy eats soup with Susan who loves food, on Monday'? To answer this question, it must first be ascertained which part of the tree would be modified by this new construct. As such, how does 'Amy eats soup with Susan who loves food, on Monday' extends the structure implied by 'Amy eats soup with Susan food on Monday'. Obviously, the new construct merely modifies the concept 'Susan' in the earlier one. The new meaning can be reflected by attaching 'who loves food' to 'Susan'.

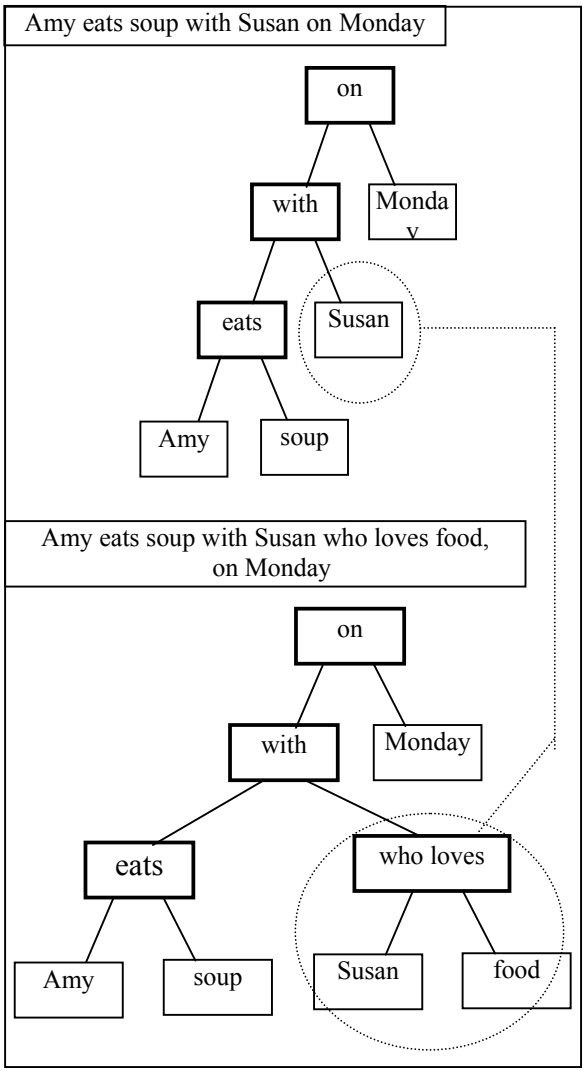

Figure 3.13 : Effect of Semantics on Tree Structure

Semantic modification of a tree T is therefore defined as the change of meaning for T through the relation R and and another tree T^+. In other words, the semantics of T is modified when another tree T^+ attaches itself to T through the relation R. The new tree resulting from this is T' = (T R T+). Since modification is confined by the semantics of R and T^+, it is greatly determined by the location where the attachment occurs. The impact of attachment has been shown in the previous examples.

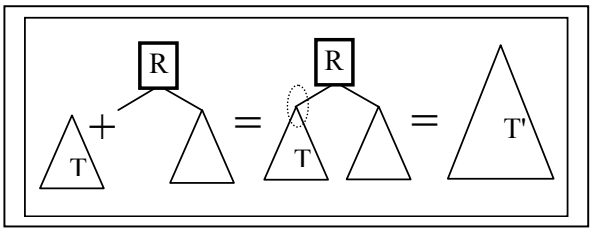

Figure 3.14 : Semantic Modification

3.5 Relational Precedence

Semantic modification determines meaning by the manner of which connections are made between concepts. It is denoted by sentence nesting, a process emulated from the expression tree. Now, the building of the expression tree is dictated by the precedence of operators within the expression. Thus, it is only rational to consider the same principle in building the concept relational tree. For concept relational tree, the relational precedence o(R) is defined as the affinity of a tree to merge with another 'adjacent' tree in the process of semantic modification. It can also be defined via sentence nesting where relational precedence determines the order of which nesting is done.

The example below (Figure 3.15) demonstrates relational precedence. Initially, the sentence in the illustration contains three trees of single concept which are 'Amy', 'cake' and 'spoon'. The shape of the tree is determined by where 'cake' would merge at. If the relational precedence of 'eats' is higher than 'with', then 'cake' would merge with 'Amy'. On the other hand, if the relational precedence of 'with' is higher than 'eats', 'cake' would merge with spoon instead of 'Amy'. The correct shape is the first tree T_1 since 'Amy' is using the 'spoon' to eat the 'cake'. The alternative tree T_2 suggests that 'Amy' is eating the 'cake' and the 'spoon'. This means that the relational precedence of 'eats' should be higher than 'with' for the valid tree T_1 to be constructed instead of T_2.

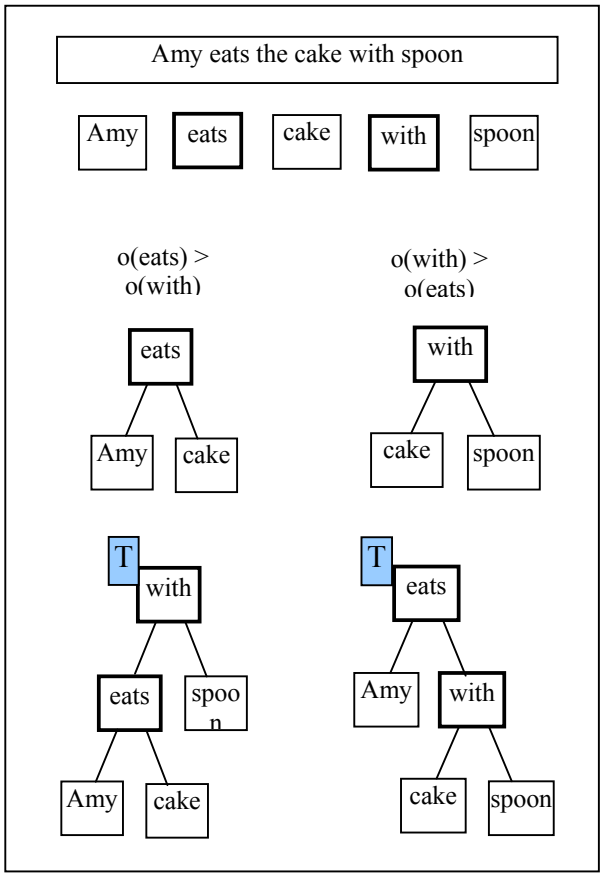

Figure 3.15 : Impact of Relational Precedence on Tree Structure

In an arbitrary sentence S, the merging or nesting of a particular tree T_i depends on the relational precedence of the relations adjacent to it (Figure 3.16). To note, the merging of T_{N-1} is determined by the relative precedence of R_{N-2} and R_{N-1}. If the relational precedence of R_{N-2} is higher than R_{N-1}, then T_{N-1} will merge with the tree T_{N-2} to its left. On the other hand if the relational precedence of R_{N-1} is higher than R_{N-2}, T_{N-1} will merge with the tree to right, which is T_N.

$$S = T_1 R_1 \ldots T_{N-2} R_{N-2} \quad T_{N-1} R_{N-1} T_N$$

$$o(R_{N-2}) > o(R_{N-1})$$
$$S = T_1 R_1 \ldots (T_{N-2} R_{N-2} \quad T_{N-1}) R_{N-1} T_N$$

$$o(R_{N-2}) < o(R_{N-1})$$
$$S = T_1 R_1 \ldots T_{N-2} R_{N-2} \quad (T_{N-1} \quad R_{N-1} T_N)$$

Figure 3.16 : Relational Precedence

From the context of rhetorical structure theory, relational precedence can be seen as the factor that determines whether a particular elementary discourse unit is a nucleus or satellite. Relational precedence with higher priority is considered to be the nucleus while the other one with lower precedence is the satellite. The difference here however is the absence of an exact rhetorical relation to define the nature of association between the nucleus and the satellite. Concept relational tree also differs from discourse structure tree in term of decomposition. The elementary discourse unit in discourse structure tree can be defined indefinitely while the organizational unit of concept relational tree are always decomposed into concepts and relations.

To be able to estimate the relative value of relational precedence, these issues regarding relations must be addressed.

3. Precedence Class
   In a mathematical expression, it is known that multiplication and division belongs to the same class of precedence while addition and subtraction are grouped together (Figure 3.17). Analogically, words are usually classified based on their part-of-speech. It is only rational to begin the effort of classifying the precedence of a particular relation using its part-of-speech. Here, it is assumed that the precedence class can be determined from the relation class.

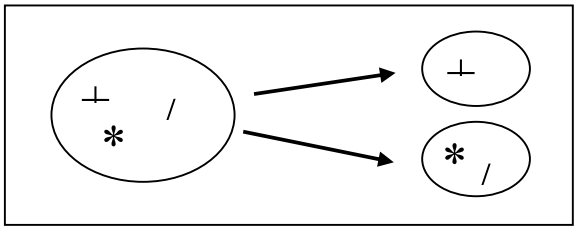

Figure 3.17 : Precedence Class of Mathematical Operators

4. Precedence Order
   Having a set of precedence classes, their order must be defined. The order decides which class would be given priority when compared to another class. In the case of the expression tree, the precedence class for multiplication '*' should be higher than addition '+'. This gives priority to multiplication '*' when nesting is employed. For concept relational tree however, how should the order of the precedence class be determined? It is assumed that the order depends on the relative way, a relation modifies concepts as compared to another.

To resolve the issue of precedence class, the general function of each relation is investigated. This is initially

done by observing a set of simple cases where the structure of the concept relational tree as well as the relational precedence of each relation within the tree is known beforehand. In the example (Figure 3.18), the relational precedence of the relation 'of' is higher than 'amused'. Because of that, 'cat' merges with 'color' and not 'Sally'. The relation 'of' connects 'color' and 'cat' in an attributive way. In other words, 'color' is the attribute of 'cat'. However, the relation 'amused' connects the sub-tree 'color of cat' to 'Sally' in a causative sense. This means that relatively speaking, the attribute relation should have higher relational precedence when compared to causative relation or o(of) > o(amused).

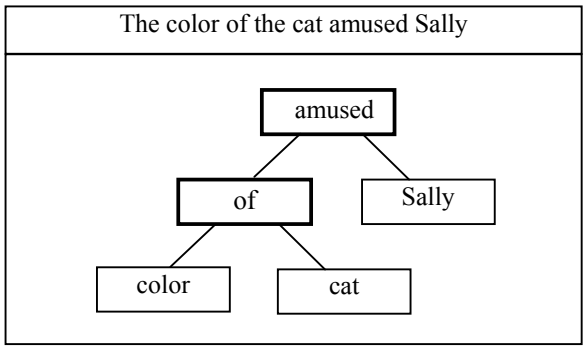

Figure 3.18 : Case Study in Relational Precedence

Continuing inductively with other cases, three classes of precedence as well as their order are proposed. Classes are arranged from the highest to the lowest where for each relation R, the connection type and the relation type determine its precedence. There are three types of connection.

1. C – **R** – C
   The simplest type of connection is when the relation **R** connects two concepts. This is similar to the subject-verb-object template. Examples include 'name **of** singer', 'dress **with** pattern' and 'fish **in** bowl'.

2. R – **R** – C
   The connection type R – **R** – C occurs when the relation **R** connects another relation with a concept. This can normally be seen in situation where a particular relation R is used to narrate the means of an action. For instance, in the text 'Amy solves problem through rationalization', the relation 'solves' is connected to 'rationalization' by the relation 'through'. It basically conveys that the action 'solves' is done by the means of 'rationalization'.

3. R – **R** – R
   The most complex connection type is R – R – R where **R** is utilized to provide the connection between relations. A common example of this would be in the case of temporal relations like 'Amy loved to read novels **before** Susan dominated the library'. The relation '**before**' connects two notions 'Amy loved to read novels' and 'Susan dominated the library'

Each connection type has a set of relation types that narrate the actual function of the relation. Combination of connection and relation types are as given below:

Table 3.2 : Connection Type and Relation Type

	description	**example**
C – **R** – C		
attribution	Property of an entity	color **of** car cake **with** cream cake **that** contains cream Amy **who** **likes** cat
conjunction	Entities that occur together in an event	cake **and** coffee
causation	Entities that affect one another through certain interaction	Amy **likes** cat Amy **closes** the door
R – **R** – C		
means	Entity employed for a certain use	Amy eats soup **with** a spoon
intention	Entity employed to reach a particular goal	Amy eats soup with a fork **to** **annoy** John
R – **R** – R		
temporal	Events related by the notion of time	Amy plays game **when** Susan reads the book

While connection type describes the pattern of connection, relation type defines the semantic of the connection. To illustrate the idea, consider how the relation 'and' is used in the example sentences (Figure 3.19). In the first sentence, the relation 'and' is used to connect the two concepts 'books' and 'pens'. Since it is used to connect two concepts, it is of connection type 'C – R – C'. For the second sentence however, the relation 'and' is not used to connect two concepts. Instead, it is employed to connect two relations 'buys' and 'reads', implying the connection type 'R – R – R'. As such, although the relation 'and' in both sentences belongs to the same relation type 'conjunction',

they differ in term of their connection type.

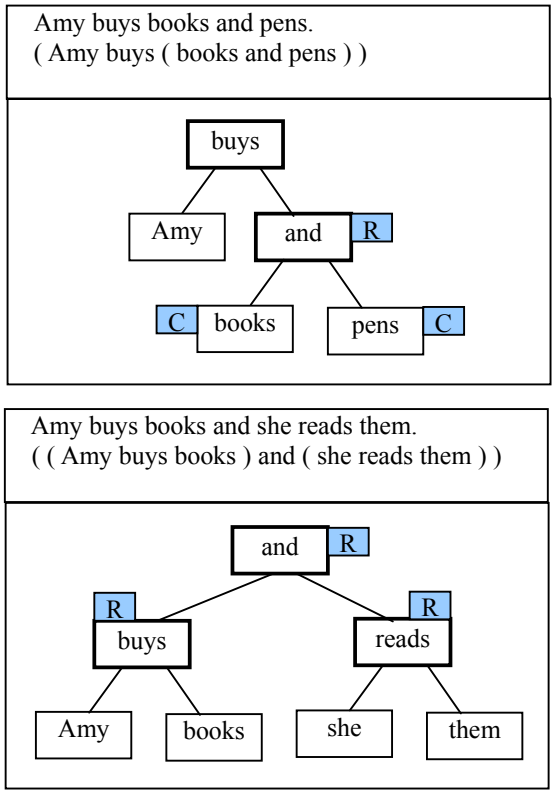

Figure 3.19 : Similar Relation Type (and) with Different Connection Type

The combination of connection type and relation type is used to compute the precedence. Connection type is compared first prior to the relation type. For instance, in the illustration below , both relations '**and**' are relation types within the category of conjunction. In the first case, 'and' has a higher precedence than 'likes' where conjunction precedes action. For the second case however, the opposite is true where action is given higher priority than conjunction. This transpires due to the fact that comparison occurs at the connection type level instead of the relation type level. Observe that The action 'likes' which lies at the connection type level 'C – R – C' precedes the conjunction 'and' that resides at the connection type level 'R – R – R'. Since order is already resolved at the connection type level, the relation type level is not employed for comparison.

Amy likes cat and hamster
(Amy likes (cat and hamster))

o(and) > o(likes)
and : C – R – C : conjunction
likes : C – R – R : action

Amy likes cat but Susan likes hamster
((Amy likes cat) but (Susan likes hamster))
o(likes) > o(and)
likes : C – R – C : action
but : R – R – R : conjunction

The relation type is defined by studying the various kinds of semantic roles and grouping them together. Relational type is more general as compared to semantic role. Instead of emphasizing on the actual semantics of a particular relation, relation type focuses on the way relations differ from one another in term of their possible precedence. Thus, it makes no difference if the function of two relations are not the same semantically. As long as they share the same precedence, they are considered to be of the same type.

For both the sentences in the illustration (Figure 3.20), their relations belong in the same precedence class of attribution. This is true despite the fact that they play different semantic roles. Note that the relation 'on' implies a locative relationship while the relation 'that scratches' denotes an action when defined by semantic roles. Despite their difference from the perception of semantic roles, they are considered to share the same relation type since both relations 'on' and 'are' describes further what the 'cat' is. The relation 'on' defines the cat as an entity that lies on the sofa while the relation 'that scratches' narrates the character of the cat, as the perpetrator that scratched the sofa.

The cat **on** the sofa is looking at Susan
The cat **that scratches** the sofa is looking at Susan

Figure 3.20 : Different Semantic Roles with Similar Relation Type

It must be reminded however that the phrase 'cat that scratches sofa' does not share the same relation type as 'cat scratches sofa'. The former is an attribution while the latter is a form of action or causality. To understand the reasoning, consider the implication of both phrases. Semantically, it can be said that 'cat that scratches sofa' is describing the habit of the cat. On the other hand, 'cat scratches sofa' is pointing out an instance of a particular action performed by the cat, which may not be a habit or attribute of the cat at all.

Performing sentence nesting is a rather difficult process. To assist the task of nesting, the iterative approach is proposed. Generally speaking, the iterative approach works by identifying the simplest components that can be built. Then, it extends and merges these components iteratively until a single complex component is formed.

The approach consists the following steps:

1. Enumerate atomic tree.
   Atomic tree is a tree that consists of a single concept. Initially, a sentence is represented as a set of isolated atomic trees. The atomic trees in the example (Figure 3.21) are 'Mother', 'Amy', 'she', 'people' and 'respect' where all of the are concepts.

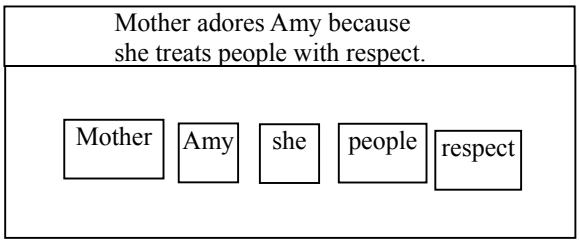

Figure 3.21 : Atomic Tress of Sentence

2. Build basic trees.
   Basic tree is defined as the tree that consists purely of two concepts and a relation or T = C R C. In the iterative approach, the basic tree is identified first before anything else. To do so, it is crucial to identify the simplest form of semantic modification present within the sentence. Examples of basic trees can be seen in the illustration (Figure 3.22). They are depicted by 'Mother adores Amy' and 'she treats people'. This is so, since each tree is made entirely by two concepts and a relation.

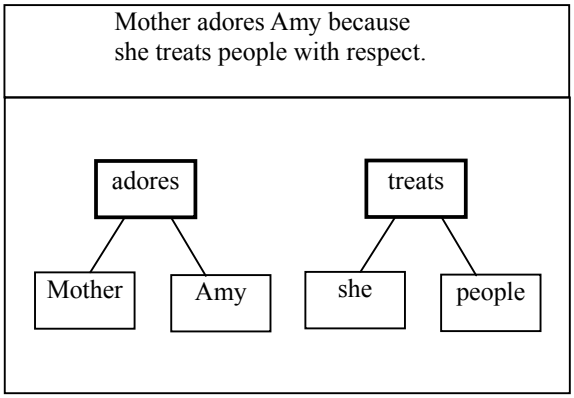

Figure 3.22 : Basic Trees of Sentence

3. Extend trees
   Once the basic tree is found, all the possible ways of extending the tree is considered. A basic tree can be extended by merging it with an adjacent tree. Consider the example (Figure 3.23) to understand the idea. Here, the basic tree '(she treats people)' has two alternatives in term of extension. It can either merge with the other basic tree '(Mother adores Amy)' or the atomic tree 'respect'. The correct extension would be to the right, where 'she treats people' merges with 'respect'. By doing it this way, the process of extending each basic tree is simplified to two alternatives. That is, either by merging to the left or right.

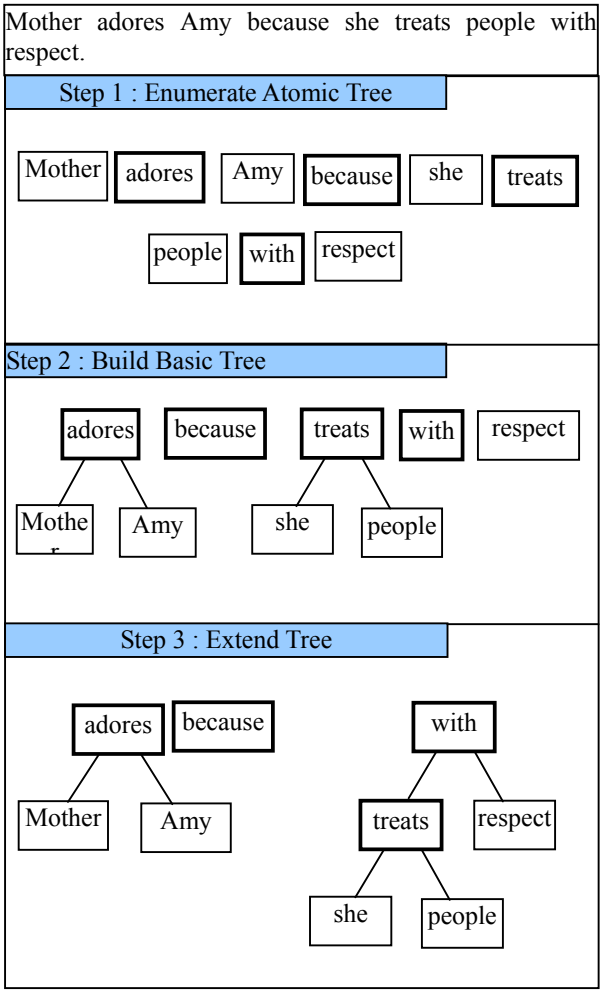

Figure 3.23 : Building the Concept Relational Tree

4. Repeat step 3 until no more tree exists in isolation.
   The goal of building the tree is completed only when a single tree exists. This can be achieved by repeatedly extending each trees. The outcome of the sentence used in the previous step consists of two trees (Figure 3.23). Since '(Mother adores Amy)' exists in isolation from '((she treats people) with respect)', the process is incomplete.

To complete the process, ways of extending the tree is again inspected. In this case however, there are no other alternative for merger except between the two trees. The final outcome of the merger can be seen in the illustration (Figure 3.24).

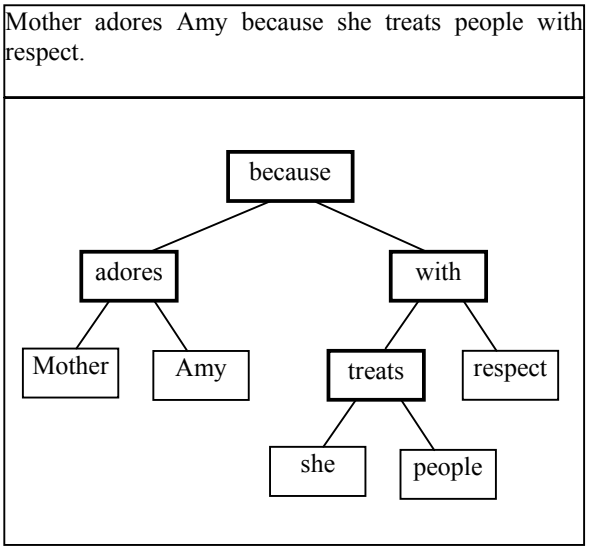

Figure 3.24 : Completing the Building Process

The iterative approach can be used to build the concept relational tree manually. However, automating the process may pose a series of daunting challenges. The biggest problem is measuring the the relational precedence of relations within a sentence. The reason for this is the nature of language that exhibits diversity. For instance, in a mathematical expression, the operators involved are practically atomic. It can either be +, -, .. and so on. The relations in language however, can be composite, such as 'likes', 'may like', 'might undeniably like' and so on. This form of flexible combination can definitely complicate computation.

4. Conclusion

The concept relational tree offers a new way of organizing the semantics of text in a hierarchical form. Its architecture emulates the expression tree where the semantics for the higher level part of the tree is built incrementally from the lower level one through the notion of semantic modification. Concept relation does not define relations specifically as discourse structure tree. This allows better coverage of text despite lacking expressiveness when it comes to defining the relation. However, concept relational tree decomposes text into a set of specific organizational unit known as concepts and relations as opposed to discourse structure tree that denotes elementary discourse unit more loosely. Using concepts and relations promotes a more structured form of organization for the concept relational tree that enables a more consistent representation of semantics.

References

[1] Baldridge, J., Asher, N. and Hunter, J. 2007. Annotation for and Robust Parsing of Discourse Structure on Unrestricted Texts. *Zeitschrift fur Sprachwissenschaft* 26: 213-239.

[2] Berland, M. and Charniak, E. 1999. Finding Parts in Very Large Corpora. In Proceedings of the the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99).

[3] Cañas, A. J., Carff, R., Hill, G., Carvalho, M., Arguedas, M., Eskridge, T. C., Lott and J., Carvajal, R. 2005. Concept Maps: Integrating Knowledge and Information Visualization, in Knowledge and Information Visualization: Searching for Synergies, S.-O. Tergan, and T. Keller, Editors. 2005. Heidelberg / New York: Springer Lecture Notes in Computer Science.

[4] Chulan, U. A. I. U. 2007. Connector Based Extraction with Concept Relational Parsing for Extracting Semantic Relation from Text. PhD Thesis (Unpublished). Universiti Putra Malaysia, Malaysia, 2007.

[5] Cohen, R. F. and Tamassia, R. 1995. Dynamic Expression Trees. Algorithmica, 13:245--265, 1995.

[6] Cristea, D., Postolache, O. and Pistol, I. 2005. Summarization through Discourse Structure. Computational Linguistics and Intelligent Text Processing, 6th International Conference CICLing 2005, LNCS, vol. 3406, ed. by Alexander Gelbukh, pp. 632-644. Springer, Berlin.

[7] Girju, R. and Moldovan, D. I. 2002. Text Mining for Causal Relations. In FLAIRS 2002.

[8] Harabagiu, S., Bejan, C., Morarescu, P. 2005. Shallow Semantics for Relation Extraction. In Proceedings of International Joint Conferences on Artificial Intelligence 2005 (IJCAI-2005)

[9] Hearst, M. A. 1992. Automatic acquisition of hyponyms from large text corpora. In Proceedings of

the Fourteenth International Conference on Computational Linguistics, pages 539-545, Nantes, France, July 1992.

[10] Pantel, P. and Lin, D. 2000. An Unsupervised Approach to Prepositional Phrase Attachment using Contextually Similar Words. In K. VijayShanker and Chang-Ning Huang, editors, Proceedings of the 38th Meeting of the Association for Computational Linguistics, pages 101-108, Hong Kong, October 2000.

[11] Reinberger, M. L., Spyns, P. and Pretorious, A. J. 2004. Automatic Initiation of An Ontology. In Proceedings of ODBase, 2004.

[12] Soricut, R. and Marcu, D. 2003. Sentence Level Discourse Parsing using Syntactic and Lexical Information. Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics.

[13] Sporleder, C. and Lascarides, A. 2006. Using Automatically Labelled Examples to Classify Rhetorical Relations: An Assessment. Natural Language Engineering 1 (1). Cambridge University Press

[14] Tomuro, N., Kanzaki, K. and Isahara, H. 2007. Self-organizing Conceptual Map and Taxonomy of Adjectives. In Proceedings of the 18th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2007).

[15] MUC-7. (1998). *Proceedings of the 7th Message Understanding Conference (MUC-7).* Morgan Kaufmann, San Mateo, CA.Kambhatla N. (2004).

Ungku Azmi Ungku Chulan received his B.Sc. and M.Sc. degrees, from Universiti Putra Malaysia. in 2001 and 2003, respectively. He received his Ph.D from Universiti Putra Malaysia in 2007. After working as a research officer, he is now a freelance consultant specializing in technologies related to text engineering. His research interest includes text mining, ontology, relation extraction and semantic parsing.