Integrating Semantic Analysis into Syntactic Parsing: Combining Categorial Grammar and Frame Semantics

Ke Wang and Rongpei Wang
School of Computer Science and Technology
Dalian University of Technology, Dalian 116024, China
wang.coco.ke@gmail.com

***Abstract*—For a long time, syntacticians and semanticists have been seeking a way to integrate semantic analysis into syntactic parsing, or inverse; because they have realized that understanding cannot be fully explained simply with syntax or semantics. Categorial Grammarians devoted a lot to extend the expressiveness of CG, but they missed a point what behinds the scene is meanings rather than grammar rules. Frame Semantics invented a very good formal representation of meanings; however it lacks an apparatus of syntactic operation like the one of CG. So, in this paper, we integrated them together, so as they could complement each other. This theoretical model can explain why some grammatical sentences are unacceptable in semantics. Besides, we introduced our Parsing System built on this theoretical model, which well supports our argument that if a semantic category is allowed by the frame, then the sentence is acceptable in semantics.**

***Index Terms*—natural language understanding, syntactic parsing, semantic analysis, categorial grammar, frame semantics**

I. Introduction

Grammar is a set of rules that governs the composition of phrases or words to be meaningful and interpretable in a given natural language; so, a grammar should explain why a sentence is acceptable while others are not. In this sense, syntax and semantics shouldn't be opposite to each other. However, due to some reasons, syntacticians and semantists they disagree on the matters how meanings are represented by linguistic symbols.

Two representative syntactic theories are Chomskian TG Grammar and Categorial Grammar. Both of them consider the grammaticality of sentences is universally controlled by some symbolic operations. However, some linguists could not agree the way of TG Grammar in explaining language. For example, TG cannot explain why "colorless green idea sleeps furiously" is syntactically grammatical, but semantically unreasonable. So in the late 1960s, they established a new banner later called Cognitive Linguistics. Cognitive linguists deny that there is an autonomous language faculty in our minds; instead, they understand grammars in terms of conceptualizations. Hence, many cognitive semantic theories do not pursue formalization mechanisms as most of the syntactic do. One of the most famous semantic theories is Frame Semantics.

CG suffer a similar embarrassing situation, though CG grammarians had tried to extend it to more expressive formalisms; for instance the sudden emerged variants of CG in 1970s and 80s, like Generalized CG, Unification CG, Combinatory CG, etc. For more references see [15]. CGs[1] try to explain the deep semantic issues with a set of rules that can just operate on syntactic surface. This kind of method is too mechanical though smart. In short, CGs just like T-G Grammar missed the point that what behind the movement of grammars rules are meanings rather than symbolic operations.

The purpose of this study is to compensate this deficiency of CG with some conceptions of Frame Semantics, and also to complement the shortness of Frame Semantics in syntactic formalization with the power of CG. This study is meaningful; because, only formalized semantic theories can be installed to a computer, and only formalisms with stronger capabilities of understanding semantic issues are useful when facing commands encoded in human's languages, thus there may be hope to achieve the goal of implementing natural language understanding on computers.

The following sentences share the same construction, coordination-reduction that has been intensively studied in CCG [13]. Both of them are grammatical; but (2) is unreasonable.

(1) Mary planted and Harry cooked the beans.

(2) *Harry cooked and Mary planted the beans.

(2) is unreasonable, because the beans "Mary planted" were cooked formerly. According to the common sense, a bean cannot grow anymore after it is cooked. So, it is weird someone planting cooked beans. We hypothesize that there must be an apparatus in our cognitive system, which can handle such kind of complicated issues. We

[1] CGs is the general name of variants of Categorial Grammar. A good introduction of CGs is Mary M. Wood's work [15].

doi:10.4304/jcp.9.9.2239-2245

are wondering if it is possible that such apparatus is a combination of syntactic parsing and semantic analysis working together alternately. Probably this is a reasonable way to explain why some grammatical sentences are not acceptable in semantics, such as (2).

We will introduce our methods in section 2 and show how these methods work in 3.1; and we will give our diagnosis of example (2) in 3.2.

We assume that semantic frames are like functions, and semantic categories[2] (usually labeled with semantic roles) are like arguments. Thus, a semantic frame can apply to semantic categories if these semantic categories are consistent with the semantic frame. In other words, whether a sentence is acceptable depends basically on if the arguments are allowed by the predicate verb. For instance, in *λx.plant*(*x*:=*bean*) if variable "*x*" can be unified or replaced with "*bean*", i.e. if "*bean*" is allowed by "*plant*", then the given sentence is consistent, and thus acceptable in semantics. We consider this problem in this way, because it is widely accepted that predicate verbs play a very important role in assigning constraints to noun phrases that they govern (Levin 1993; Fillmore 1968, 1982). *In section 3.3, we'll discuss about why such a way of resolving semantic issues is computationally feasible* (decidable). Decidability problem is very critical for computing applications in that only "decidable sentences" can actually be "understood" by machines.

To verify our hypothesis, we built a Parsing System, which can judge the syntactic properties and semantic properties of a sentence simultaneously, see section 4. The whole grammar system is composed of a type hierarchy (classification of verbs and taxonomy of utterances), a lexicon and a set of grammar rules. It at present includes about 50 rules, more than 600 verbs, and hundreds of nouns. The limit of the grammar system at present is that it can only process clausal sentences. However, it is very hopeful to extend it to a wider coverage by adding a probability based preprocessor, which is, for example, capable of cutting complex sentences into simple clauses.

II. Methods Used in This Paper

A. Syntactic Rules of CGs

Categorial Grammar originates from the ideas in work of Ajdukiewicz [1] and Bar-Hillel [2] (hence AB-CG). Joachim Lambek [11] introduced a syntactic calculus along with various rules for the combination of functions, which mainly include Application, Associativity, Composition, and Raising. CG is distinguished from other formal grammars by its syntactic categories and inference rules. The syntactic categories **SyC** is defined as follows:

Atomic categories: NP, S, …∈**SyC**

Complex categories: if X, Y∈**SyC**, then X/Y, X\Y∈**SyC**. Complex categories X/Y or X\Y are functors with an argument Y and a result X, for example S\NP/NP in (3). For another, NP/NP would be the type of determiner that it looks forward for a noun to produce a noun phrase; S\S would be the type of adverb that it looks backward for sentence to produce a sentence, as illustrated in (4) and (5):

(3) He sells tomatoes.
NP S\NP/NP NP
_____S\NP_____
_____S_____

(4) I bought a red book yesterday.
NP S/NP\NP NP S\S
____S/NP___
_____ S_____
_____S_____

Application and Composition are the most frequently used rules of CG. "The rule of forward application states that if a constituent with category X/Y is immediately followed by a constituent with category Y, they can be combined to form a constituent with category X. Analogously, backward application allows a constituent X\Y that is immediately preceded by a constituent with category Y to combine with this to form a new constituent of category X" [8].

- Forward application
  X/Y Y →X
- Backward application
  X\Y Y →X

"Composition allows two functor categories to combine into another functor" (ibid).

- Forward composition
  X/Y Y/Z →X/Z
- Backward composition
  Y\Z X\Y →X\Z

For example, in (5), the article "a" asks for a noun phrase to be its argument, so does the adjective "red"; therefore they are composed together. Some more sophisticated examples can be found in Mark Steedman's work [14].

(5) a red book
NP/NP NP/NP NP
____NP/NP___
_____NP_____

B. Semantic Representation of Frame Semantics

Frame semantics is the development of Charles Fillmore's Case Grammar [6]. The basic idea is that one cannot understand meaning without world knowledge. A semantic frame is defined as a structure describing the relationships among concepts evoked by words (mostly, by verbs). For example, in an exchange frame, the

[2] It is hard to give a formal definition of semantic category, like the definition of syntactic categories given in CGs, see section2.1; because semantic categories could be intertwined with syntactic components, see section 4.2.

concepts of SELLER, BUYER, and GOODS would be evoked by verbs, e.g. *sell, buy,* etc. In a sentence, semantic structures that are composed of these concepts are usually represented by the syntactic relations of semantic roles over predicates, as in (6) and (7), compare them to (3) and (5) respectively:

(6) He sells tomatoes.
SELLER <exchange> GOODS

(7) I bought a red book yesterday.
BUYER <exchange> GOODS TIME

The assignment of semantic roles to categories depends on the meanings of both verbs and the categories. For example, in (8), "tomatoes" is assigned PATIENT, instead of GOODS; because, the predicate "cooked" evoked frame <cook> and all the concepts related to <cook>, such as Agent and Patient. In (9), "tomatoes" is assigned THEME, because its state does not change after it is moved to the "truck"[3].

(8) He cooked tomatoes.
AGENT <cook> PATIENT

(9) He loaded truck with tomatoes.
AGENT <fill> GOAL THEME

C. Combining CG Rules and Semantic Frames

It is widely agreed that it is mainly the predicate verb of a sentence that bears the responsibility of evoking semantic frames. Thus, the predicates in (3) and (6) can be ***rewritten*** as (10), which means frame <exchange> has two arguments, SELLER, and GOODS

(10) <exchange>\SELLER/GOODS

Through applying (10) to (6), we can extract semantic frame <exchange>, as shown in (11):

(11) He sells tomatoes.
SELLER <exchange>\SELLER/GOODS GOODS
_____<exchange>\SELLER_____
_____<exchange>_____

Semantic frames can also be composed to form a complex frame.

(12) John said [he sold tomatoes]^{CONTENT}
INFORMER X SELLER Y GOOD
X' Y'

Here, we replace the verb's meaning with X and Y. X' represents the semantic frame of the main clause, and Y' represents the semantic frame of the complement clause:

• X'=X\INFORMER/CONTENT
• Y'=Y\SELLER/GOODS

The two semantic frames are ***composed*** in the way of (13):

(13) X'/Y' Y' →X'

Where, X'/Y' means, the semantic frame X' asks for Y' to be its argument. We write it in a more conventional form, X'(Y'). Note that the composition of two semantic frames into a complex frame needs to ***convert*** each semantic frame into a more complex form. In the following of this paper, we will continue to use X and Y to represent the lexical meanings of verbs, and X' and Y' to represent the frames.

III. EXAMPLES

A. Insertion

In Bar-Hillel's paper [2], there is a tough problem that CGs cannot resolve neatly until today, as shown in (14):

(14) Paul, strangely enough, refuses to talk.
Z X Y W

Literally, it means "*it is strangely enough that Paul refuses to talk"*. Apparently, (14) is a complex construction composed of two semantic frames. We just need to make them go back to their places. In Curry's work [5], he presented a combinator C, which is used for switching the first two arguments of a function. Here, we use it to ***reorder*** the disturbed arguments' position. Example (14) can be transformed into (15) without causing any changes of meanings.

(15) Strangely enough, Paul refuses to talk.
reorder: X Z Y W
rewrite: X' (X'=X) Y'=Y\Z/W
convert: X'/Y' [Y\Z/W]$^{Y'}$
compose: X'/Y' [Y\Z/W]$^{Y'}$ → X'/[Y\Z/W]

where, *rewrite, convert,* and *compose* are the operators that have been introduced in section 2. (16) and (17) are similar examples:

(16) He, *frankly speaking,* is not good enough.

(17) He, *I guess*, must be angry.

B. Coordination-reductions

The examples (1) and (2) mentioned at the beginning of this paper share the same construction, coordination-reduction. The omitted parts […] did not disappear;

[3] The main difference between PATIENT and THEME is that PATIENT undergoes the action and its state changes, whereas THEME does not. For more explanations about semantic role labeling, please refer to Fillmore's paper about Case Grammar [6]

actually they exist in deep semantic layers, as shown in (1)' and (2)':

(1)'Mary planted […] and Harry cooked the beans.

(2)'* Harry cooked […] and Mary planted the beans.

To give a further explanation on why (2) is not acceptable in semantics, we need to evoke our world knowledge. We may share the common sense that a cooked bean cannot grow anymore, so it's meaningless to plant cooked beans. The reasoning processes with such world knowledge can be represented as (18):

(18) Harry cooked and Mary planted the beans.

	Y	X
	Mary planted the beans	Harry cooked […].
reorder:	X	Y
rewrite:	X\AGENT$_1$/THEME	Y\AGENT$_2$/PATIENT
convert:	X'/Y'	[Y\AGENT$_2$/PATIENT]$^{Y'}$
compose:	X'/Y' [Y\AGENT$_2$/PATIENT]$^{Y'}$ → X'/[Y\AGENT$_2$ /PATIENT]	

Technically, this composition can be reduced to the unification of […]^{PATIENT} and THEME, see (19). If […]^{PATIENT} is consistent with X', then, it can replace the variable THEME in X'. Apparently, it is not, so […]^{PATIENT} cannot replace THEME. Hence, the sentence is not acceptable in semantics.

(19) $\lambda PATIENT.Y'(PATIENT:=[...])=[...]^{PATIENT}$

$\lambda THEME.X'(THEME:=[...]^{PATIENT})$

C. Discussion

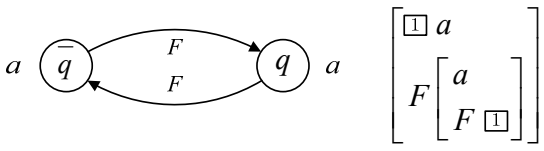
How do we know that such way talked above is resolvable? Or how do we decide that THEME can be replaced with […]^{PATIENT}, in (19)? In Kfoury's work [9], he proved that an instance Δ of unification problem U (β-unification) has a solution iff M is *β*-strongly normalizable, (where M is a lambda term, from which Δ can be transformed); and that M is *β*-strongly normalizable iff M is typable in the lean fragment of the system of intersection types. We hypothesize that semantic frames are the lean fragments of the system of intersection types, and if verbs that bear the meanings of semantic frames are typable in such system, then semantic consistency problems like the one of (19) is decidable. Linguistically, being typable in the system of semantic frame means that verbs, such as "cook" and "plant" in (1) and (2), are of completely different types. That is to say, the arguments that <cook> and <plant> ask for are of completely different types. So, verb types will help explain why the change of "the beans" caused by "cook" is unacceptable in frame <plant>.

IV. Parsing System

The grammar system is implemented in ALE [3] a programming language based on the logic of typed feature structure [4]. During constructing this grammar system we encountered three difficulties. The first is that the structures of syntactic and semantic components are both independent of each other and intertwined together. On one hand, the independency will guarantee the generality of the grammar system. On the other hand this is quite a dilemma, for there is not yet such a parsing method that can skip one while parsing the other. This means we must find a method that can parse the syntactic and semantic structures simultaneously. The second is labeling categories with types. This is not as easy as it looks; because a semantic category could be as complex as a clause; though sometimes they are just bare words. As for the simple categories, we defined a set of lexical rules that can change them directly into intended forms. As for complex categories, we employed Application Rules of CG in the way of like (28). The third is to describe the four operators with typed feature structures (we omitted *reorder* for it is very easy). For the rewrite operator, we defined a set of structures (they are called Macros in ALE) to separate the inflections from the verb; as in (21) and (25). The four operators are like steps, by which we dig gradually from surface into the core. In this sense, the whole parsing process is basically bottom up.

A. Typed Feature Structure

A typed feature structure is a directed graph possibly with cycles. The nodes on the graph are labeled with types; and the edges between nodes are labeled with features, see Figure 1. A typed feature structure could be represented either by a graph, like (a), or by an AVM, like (b). *The squares* with numbers in (b) mean that the types are same. This is very useful in describing some linguistic phenomena, like anaphora and reduction. It will appear again in section 4.4.



(a) (b)

Figure 1. The graphic and AVM representation of typed feature structure

More intensive studies about typed feature structures can be found in Carpenter's book [4] and Penn's doctoral thesis [12]. Typed feature structure is widely used in many grammar representations, for example Lexical Functional Grammar and Head-driven Phrase Structure Grammar.

The only operation in the logic of typed feature structure is Unification. Whether two feature structures can be unified together depends on if their types are consistent to each other. Carpenter in his book named "the logic of typed feature structures" gives a definition of **Type Consistency**: "A set *T* of types is said to be consistent or bounded if they share a common subtype or upper bound σ such that $\sigma \sqsubseteq \tau$ for every $\tau \sqsubseteq T$ [4].

In the above, we used λ-calculus to explain some semantic issues. Typed feature structure can take its place

in that the type is like a function and the features are like arguments; and the replacement of variables with values in λ-calculus can be represented by the unification of features values and some other types. The constraints of the argument assigned by the predicate are also well described in the logic of typed feature structures. Different types are discriminated by their features, and all the features must be valued by some other types, i.e., types constrain their features' values.

B. Intertwined Syntactic and Semantic Structures

The structure of a sentence is like a double helix, as shown in Figure 2. Syntactic and semantic components are intertwined together. This addresses ourselves to a question: how do we define syntactic and semantic components? Generally speaking, firstly semantic components are more independent. Even the dependents (the thicken parts) were omitted, the meaning of those sentences would not change much. Secondly, semantic components are more stationary. As shown in (a) and (b), when the sentence is transformed, the semantic components do not need to move; while, the dependents have to move with their "governors".

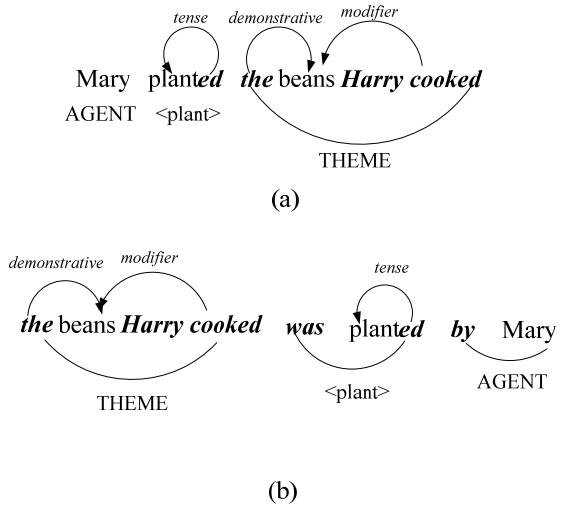


Figure 2. An example of intertwined syntactic and semantic structures

For example, in (a) of Figure 2, AGENT and THEME are dependent on <plant>, so they are the syntactic components or dependents of <plant>; while "plant" is the semantic component of <plant> or the governor of AGENT and THEME; so it is more stationary. When the sentence is changed into a passive form, as in (b), AGENT and THEME are moved, "plant" is still in the middle. For another example, in THEME "the" and "Harry cooked" are dependent on "beans"; so they are the syntactic components of THEME and "beans" is the semantic component. When THEME is moved to the front, as shown in (b), "the" and "Harry cooked" have to move, but the position of "beans" relative to "the" and "Harry cooked" is not changed.

C. Labeling Categories with "types"

(20), (22), (23) and (24) are to label categories with semantic roles. It is assumed that what semantic role a category is to have depends on the category's lexical meaning and on the category's syntactic environment. In (20), as long as "marry" is predefined as a subtype of "agent_element", it is unifiable with the "node" [agent_element].

(20)

$$mary \rightarrow [agent\_element] \rightarrow \begin{bmatrix} agent\_word \\ MEANING : mary \end{bmatrix} ①$$

(21)

$$planted \rightarrow \begin{bmatrix} plant\_word \\ SYNTAX \begin{bmatrix} tense\_modality \\ TENSE\ did \end{bmatrix} \\ MEANING\ plant \end{bmatrix} \xrightarrow{rewrite} \begin{bmatrix} plant \\ SYN: \begin{bmatrix} plant\_frame \\ AGENT : agent \\ THEME : theme \end{bmatrix} \\ SEM: \begin{bmatrix} plant\_word \\ SYNTAX: \begin{bmatrix} tense\_modality \\ TENSE : did \\ MEANING : plant \end{bmatrix} \end{bmatrix} \end{bmatrix} ②$$

(22)

$$beans \rightarrow [theme\_element] \rightarrow \begin{bmatrix} theme\_word \\ SYNTAX : s \\ MEANING : bean \end{bmatrix} ③$$

(23)

$$beans \rightarrow [patient\_element] \rightarrow \begin{bmatrix} theme\_word \\ SYNTAX : s \\ MEANING : bean \end{bmatrix} ④$$

(24)

$$harry \rightarrow [agent\_element] \rightarrow \begin{bmatrix} agent\_word \\ SYNTAX : modality \\ MEANING : harry \end{bmatrix} ⑤$$

(25)

$$cooked \rightarrow \begin{bmatrix} cooke\_word \\ SYNTAX : \begin{bmatrix} tense\_modality \\ TENSE : did \end{bmatrix} \\ MEANING : cook \end{bmatrix} \xrightarrow{rewrite} \begin{bmatrix} cook \\ SYN: \begin{bmatrix} cook\_frame \\ AGENT: agent \\ PATIENT: patient \end{bmatrix} \\ SEM: \begin{bmatrix} cooke\_word \\ SYNTAX: \begin{bmatrix} tense\_modality \\ TENSE: did \end{bmatrix} \\ MEANING\ cook \end{bmatrix} \end{bmatrix} ⑥$$

(26)

$$the \rightarrow [demonstrative] ⑦$$

(21) and (25) are approximations of the operator *rewrite*, as introduced in section 2. Because "semantic frame" is a very abstract concept, so the modalities and tenses that have been attached to a verb should be interpreted in some other ways. After a predicate verb becomes an abstract "frame_word", i.e. it will be able to bear the responsibility for evoking some semantic frame. (26) is to label "the" with a type called "demonstrative" in linguistics. The purpose of doing this is to make a referent clearer. But in the coming-up stage, as shown in (28), they will be composed together so that they will be treated as a whole.

D. Semantic and Syntactic Parsing

The only one operation in the logic of typed feature structure is **Unification**. This is very similar to the unification operation in PROLOG except that it is constrained by "type consistency". In order to parse the semantic structures of utterances, we need to translate the **Application** rules of Categorial Grammar into Unification operation. We replace "/" and "\"in categorial grammar with "⊔" the unification operation in the logic of typed feature structure[4]. In order to make it tidier, in the following examples, we don't copy the whole structure; instead, we use the number on the upper right corner of each product, as shown in the followings.

(27)

$$⑤ \sqcup ⑥ \xrightarrow{convert} \begin{bmatrix} cook \\ SYN: \begin{bmatrix} cook\_frame \\ AGENT: \begin{bmatrix} agent\_word \\ SYNTAX: modality \\ MEANING: harry \end{bmatrix} \\ PATIENT: \begin{bmatrix} patient\_word \\ MEANING: patient\_element \end{bmatrix} \end{bmatrix} \\ SEM: \begin{bmatrix} cook\_word \\ SYNTAX: \begin{bmatrix} tense\_modality \\ TENSE: did \end{bmatrix} \\ MEANING: cook \end{bmatrix} \end{bmatrix}^{⑧}$$

(28)

$$⑦ \sqcup ③ \Rightarrow \begin{bmatrix} theme \\ SYN: \begin{bmatrix} modifier \\ DEMON: the \\ CLUE: fact \end{bmatrix} \\ SEM: \begin{bmatrix} theme\_word \\ MEANING: bean \end{bmatrix} \end{bmatrix}^{⑨}$$

(29)

$$① \sqcup ② \sqcup ⑨ \xrightarrow{convert} \begin{bmatrix} plant \\ SYN: \begin{bmatrix} plant\_frame \\ AGENT: \begin{bmatrix} agent \\ SYN: modifier \\ SEM: \begin{bmatrix} agent\_word \\ MEANING: mary \end{bmatrix} \end{bmatrix} \\ THEME: \begin{bmatrix} theme \\ SYN: \begin{bmatrix} modifier \\ DEMON: the \\ CLUE: fact \end{bmatrix} \\ SEM: \begin{bmatrix} theme\_word \\ SYNTAX: s \\ MEANING: bean \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ SEM: \begin{bmatrix} plant\_word \\ SYNTAX: \begin{bmatrix} tense\_modality \\ TENSE: did \end{bmatrix} \\ MEANING: plant \end{bmatrix} \end{bmatrix}^{⑩}$$

(27) and (29) are to approximate the operator "convert", see section 2. Convert is to change a frame word into a more complex frame according to its surroundings. (27) is constructed by unifying "harry" and "cooked". (29) is constructed by unifying "mary", "planted" and "the beans". (28) is to make the noun phrase as a whole category. The features "SYN" and "SEM" in the above examples are to describe the intertwined syntactic and semantic structures.

(30)

$$⑩ \sqcup ⑧ \xrightarrow{compose}$$

$$\begin{bmatrix} plant \\ SYN: \begin{bmatrix} plant\_frame \\ AGENT: \begin{bmatrix} agent \\ SYN: modifier \\ SEM: \begin{bmatrix} agent\_word \\ MEANING: mary \end{bmatrix} \end{bmatrix} \\ THEME: \begin{bmatrix} theme \\ SYN: \begin{bmatrix} modifier \\ DEMON: the \end{bmatrix} \\ CLUE: \begin{bmatrix} cook \\ SYN: \begin{bmatrix} cook\_frame \\ AGENT: \begin{bmatrix} agent\_word \\ SYNTAX: modality \\ MEANING: harry \end{bmatrix} \\ PATIENT: \begin{bmatrix} patient\_word \\ MEANING: bean\ \boxed{1} \end{bmatrix} \end{bmatrix} \\ SEM: \begin{bmatrix} cook\_word \\ SYNTAX: \begin{bmatrix} tense\_modality \\ TENSE: did \end{bmatrix} \\ MEANING: cook \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ SEM: \begin{bmatrix} theme\_word \\ SYNTAX: s \\ MEANING: bean\ \boxed{1} \end{bmatrix} \end{bmatrix} \\ SEM: \begin{bmatrix} plant\_word \\ SYNTAX: \begin{bmatrix} tense\_modality \\ TENSE: did \end{bmatrix} \\ MEANING: plant \end{bmatrix} \end{bmatrix}$$

[4] The operations of Categorial Grammar are intrinsically different from Unification of the logic of Typed Feature Structures [4]. However, from an angle both of them could be considered as apparatuses to combine small parts into bigger ones.

(30) is to approximate operator "compose". Whether they can be composed together depends on if the two frames are compatible to each other; or more exactly if <cook> is compatible to <plant>. Recall we reduced this problem to the replacement of "the beans" in <plant> with the "beans" in <cook>. In here, we use an equation (the squares with numbers) to describe the replacement (refer to Figure 1). Apparently, the beans are of two types, so they are not compatible. Therefore, "the beans" in <plant> cannot be replaced with the "beans" in <cook>; i.e. the whole sentence is semantically unacceptable.

V. Further Discussion

In this paper, we demonstrated the deficiencies of CG and FS, and how to complement them with each other.

One of the deficiencies of CG is that the syntactic categories are too superficial. For example, all noun phrases are represented as NP, no matter if their meanings have undergone changes. That's why "the beans" Mary cooked cannot be distinguished from the "beans" Harry planted, as shown in (10) and (11). For another example, as shown in (5), both determiner and adjective are represented as NP/NP. But, FS distinguishes categories meticulously. Therefore, we replace the part of speech with semantic roles so that nuance of meanings can be represented.

Besides, CG is order-dependent. Some very common linguistic phenomena are still challenges to CGs; for example, omission and insertion. But these are not problems to FS. Unlike CG, in FS, a semantic frame is a set of concepts that are related in a way of "gestalt". For example, in (1)' and (2)', omitted constituents are regained and they are represented as […].

FS does not pursue formal mechanism; so it cannot explain how smaller frames are composed together. But this is not a problem to CG, for syntactic categories can be composed through syntactic rules, such as Application and Composition introduced in section II. Being inspired by this notion of CG, we proposed four operators: *reorder, rewrite, convert* and *compose*. *Reorder* makes a disturbed sentence change back to its order. *Rewrite* defines the evoking process as shown in (18), (21) and (25). *Convert* defines how simple frames change into composite forms. *Compose* defines how a simple frame is integrated into a composite one.

From some aspects, semantic frames can be considered as conceptualized experiences. Since we can describe such conceptual experiences with linguistic symbols, there must be a mapping between them. Such mapping might be meaningful in that what a machine can do is simply operating on symbols, but what we expect it to do is far more than that.

Acknowledgment

I am grateful to Prof. Penn for his helps during constructing the parsing system. I am also thankful to many anonymous reviewers for their valuable advises.

References

[1] Kazimierz Ajdukiewicz. 'Syntactic Connecxion' in S. McCall (ed.), Polish Logic, Oxford, 1976. [Die Syntaktische Konnexitat. Studia Philosophica Vol. 1, pp. 1-27, 1935].
[2] Yehoshua Bar-Hillel. A Quasi-arithmetical Notation for Syntactic Description. Language, Vol. 29, 1953.
[3] Bob Carpenter and Gerald Penn. The Attribute Logic Engine: User's Guide. It can be obtained from www.cs.toronto.edu/~gpenn/ale.html
[4] Bob Carpenter. The Logic of Typed Feature Structures, Cambridge University Press, 1992.
[5] Haskell Curry. The Combinatory Foundations of Mathematical Logic. The Journal of Symbolic Logic, Vol. 7, Number 2, 1942.
[6] Charles J. Fillmore. The Case for Case. In Bach and Harms (ed.): Universals in Linguistic Theory. New York: Holt, Rinehart, and Winston, 1968.
[7] Charles J. Fillmore. Frame semantics. In The Linguistic Society of Korea, eds. Linguistics in the Morning Calm. Seoul: Hanshin, 1982.
[8] Julia Hockenmaier and Mark Steedman. CCGbank: User's Manual. Technical Report MS-CIS-05-09. Department of Computer and Information Science. University of Pennsylvania, Philadelphia, 2005.
[9] A.J. Kfoury. Beta-Reduction as Unification. It can be obtained from http://types.bu.edu/index.php?p=reports".
[10] Beth Levin. English Verb Classes and Alternations: A Preliminary Investigation. The University of Chicago Press, Chicago and London, 1993.
[11] Joachim Lambek. The Mathematics of Sentence Structure. The American Mathematical Monthly, Vol. 65, No. 3, 1958.
[12] Gerald Penn. The Algebraic Structure of Attributed Type Signatures. Doctoral Thesis, Carnegie Mellon University. Pittsburgh PA, 2000.
[13] Mark Steedman. Combinatory Grammars and Parasitic Gaps. Natural Language and Linguistic Theory, Vol. 5, pp. 403-439, 1987.
[14] Mark Steedman. The Syntactic Process. The MIT Press, Cambridge, Massachusetts, London, England, 2000.
[15] Mary M. Wood. Categorial Grammars. Routledge, USA and Canada, 1993.

Ke Wang received her BA and MA in computational linguistics from Dalian Maritime University, China, in July of 1999 and April of 2005 respectively. She now is a Ph.D Candidate of Dalian University of Technology, working in the area of Intentions of Speech. Her research interest is seeking the interface between natural language and artificial language with methods of mathematics, formal logics, and so on. She is also an enthusiast of cognitive science, psychology and neurobiology, anything that potentially may uncover the secret of human's understanding mechanism.