International Journal of Knowledge and Language Processing
Volume 4, Number 1, **2013**

www.ijklp.org
KLP International ©2013 ISSN 2191-2734
pp. 26-**34**

Implementation of Chinese Resource Grammar in Grammatical Framework

Chen Peng
Beijing Language and Culture University
No.15, Xueyuan Road, Haidian District, Beijing, China
chenpeng@blcu.edu.cn

Received February 2013; revised February 2013

ABSTRACT. *This paper describes a Chinese resource grammar implemented in Grammatical Framework (GF), a programming language for multilingual grammar applications. GF differentiates between concrete grammars and abstract grammars: translation among concrete languages is provided via abstract syntax trees. Thus the same concrete grammar is effectively used for both language analysis and language generation. Furthermore, GF differentiates between general-purpose resource grammars and domain-specific application grammars that are built on top of the resource grammars. The GF resource grammar library (RGL) currently supports nearly 30 languages that implement a common API. We briefly describe the grammatical features of Chinese and illustrate how those features are handled in the multilingual framework of GF, especially noun phrase, verb phrase, adjective phrase, clause and question clause. We also test Chinese resource grammar, and briefly discuss the limitations of the Chinese resource grammar and potential recent and long-term improvements.*

Keywords: computational grammar; Chinese language generation; Grammatical Framework

1. **Introduction.** The application background of this paper is automatic translation between natural language and artificial language, it is aimed at semantic parsing of Chinese and natural language generation in Chinese. We describe an implementation of Chinese resource grammar in Grammatical Framework (GF)[1], which is applied in language analysis and language generation.

The Chinese Language which is one of the branches of Sino-Tibetan family of languages is one of wildly used languages. Our implementation of Chinese resource grammar is different from other corpus like Comprehensive Language Knowledge Base developed by Institute of Computational Linguistics at Peking University[2] or HowNet developed by Dong Zhendong[3]. It focuses on grammar, especially at syntax level and belongs to syntax rules library. Since GF is grammar formalism based on dependent type theory[4], Our implementation of Chinese resource grammar library can be used as a research platform of

Chinese grammar formalism.

2. **Grammatical Framework and resource grammar.** GF is a grammar development tool and a grammar formal theory based on type theory, which implements a kind of program language and a kind of software platform for writing grammar. GF differentiates between abstract syntax and concrete syntax. The abstract syntax is language independent and is common to all languages in GF grammar library. It is based on common syntactic or semantic constructions, which work for all the involved languages on an appropriate level of abstraction. The concrete syntax is language dependent and defines a mapping from abstract to actual textual representation in a specific language. GF uses the term 'category' to model different parts of speech (e.g verbs, nouns adjectives etc.). An abstract syntax defines a set of categories, as well as a set of tree building functions. Concrete syntax contains rules telling how these trees are linearized. Separating the tree building rules (abstract syntax) from linearization rules (concrete syntax) makes it possible to have multiple concrete syntaxes for one abstract syntax, which makes it possible to parse text in one language and translate it to multiple languages.

As a functional programming language for writing grammars, GF provides some data types and data structures.

TABLE 1. major syntax elements of GF

Keywords	*Functions*	*Examples*
cat	*Defining a category in abstract syntax*	*cat* *Quality, Kind*
fun	*Defining function signature in abstract syntax*	*fun* *Mod : Quality -> Kind -> Kind*
lincat	*Linearizing a category in concrete syntax, corresponding with cat in abstract syntax*	*lincat* *Kind = {s : Number => Str} ;*
lin	*Linearizing a function in concrete syntax, corresponding with fun in abstract syntax*	*lin* *Mod quality kind =* *{s = ¥¥n => quality.s ++ kind.s ! n} ;*

GF has some forms of judgements for defining parameters, tables and records.

1. GF has a form of judgement for defining parameters. For example, we define a parameter type, Number, including singular and plural:

   param Number = Sg | Pl

2. GF has a form of judgement for defining tables. For example, we define a table type: Number=>Str. Object of table type is table, for example, table {Sg => "Pizza" ; Pl =>"Pizze"} is an object of table type Number=>Str. The value assigned to a parameter can be selected by the operator(!). as in the example below which GF computes into the value "Pizza"

   table {Sg => "Pizza" ; Pl =>"Pizze"}! Sg

3. GF has a form of judgement for defining records. For example, {s:Str; n:Number}, for operating a field of record, we need the operator (.),

   {s = "theses" ; n = Pl}.n

Figure 1 is a demonstration of abstract syntax and English & Chinese concrete syntax of "Foods".

```
abstract Foods = {
   flags startcat = Comment ;
   cat
      Comment ; Item ; Kind ; Quality ;
   fun
      Pred : Item -> Quality -> Comment ;
      This, That, These, Those : Kind -> Item ;
      Mod : Quality -> Kind -> Kind ;
      Wine, Cheese, Fish, Pizza : Kind ;
      Very : Quality -> Quality ;
      Fresh, Warm, Italian,
      Expensive, Delicious, Boring : Quality ;
}
```

(a) abstract syntax of Foods

```
concrete FoodsEng of Foods = {
   lincat
      Comment, Quality = {s : Str} ;
      Kind = {s : Number => Str} ;
      Item = {s : Str ; n : Number} ;
   lin
      Pred item quality =
         {s = item.s ++ copula ! item.n ++ quality.s} ;
      This    = det Sg "this" ;
      That    = det Sg "that" ;
      These = det Pl "these" ;
      Those = det Pl "those" ;
      Mod quality kind =
         {s = ¥¥n => quality.s ++ kind.s ! n} ;
      Wine = regNoun "wine" ;
      Cheese = regNoun "cheese" ;
      Fish = noun "fish" "fish" ;
      Pizza = regNoun "pizza" ;
      Very a = {s = "very" ++ a.s} ;
      Fresh = adj "fresh" ;
      Warm = adj "warm" ;
      Italian = adj "Italian" ;
      Expensive = adj "expensive" ;
      Delicious = adj "delicious" ;
      Boring = adj "boring" ;
   param
      Number = Sg | Pl ;
   oper
      det : Number -> Str ->
         {s : Number => Str} -> {s : Str ; n : Number} =
            ¥n,det,noun -> {s = det ++ noun.s ! n ; n = n} ;
      noun : Str -> Str -> {s : Number => Str} =
         ¥man,men -> {s = table {Sg => man ; Pl => men}} ;
      regNoun : Str -> {s : Number => Str} =
         ¥car -> noun car (car + "s") ;
      adj : Str -> {s : Str} =
         ¥cold -> {s = cold} ;
      copula : Number => Str =
         table {Sg => "is" ; Pl => "are"} ;
}
```

```
concrete FoodsChi of Foods = {
   lincat
      Comment, Quality = {s : Str} ;
      Kind = {s : Number => Str} ;
      Item = {s : Str ; n : Number} ;
   lin
      Pred item quality =
         {s = item.s ++ copula ! item.n ++ quality.s} ;
      This    = det Sg "这" ;
      That    = det Sg "那" ;
      These = det Pl "这些" ;
      Those = det Pl "那些" ;
      Mod quality kind =
         {s = ¥¥n => quality.s ++ kind.s ! n} ;
      Wine = regNoun "酒" ;
      Cheese = regNoun "奶酪" ;
      Fish = noun "fish" "鱼" ;
      Pizza = regNoun "披萨" ;
      Very a = {s = "非常" ++ a.s} ;
      Fresh = adj "新鲜的" ;
      Warm = adj "温暖的" ;
      Italian = adj "意大利式的" ;
      Expensive = adj "昂贵的" ;
      Delicious = adj "美味的" ;
      Boring = adj "难吃的" ;
   param
      Number = Sg | Pl ;
   oper
      det : Number -> Str ->
         {s : Number => Str} -> {s : Str ; n : Number} =
            ¥n,det,noun -> {s = det ++ noun.s ! n ; n = n} ;
      noun : Str -> Str -> {s : Number => Str} =
         ¥man,men -> {s = table {Sg => man ; Pl => men}} ;
      regNoun : Str -> {s : Number => Str} =
         ¥car -> noun car (car + "s") ;
      adj : Str -> {s : Str} =
         ¥cold -> {s = cold} ;
      copula : Number => Str =
         table {Sg => "is" ; Pl => "are"} ;
}
```

(b) English and Chinese concrete syntax of Foods

FIGURE 1. An example of GF grammar

GF differentiates not only between abstract syntax and concrete syntax but also between general-purpose resource grammars and domain-specific application grammars. Resource grammars are general purpose grammars[5] that try to cover the general aspects of a language linguistically and whose abstract syntax encodes syntactic structures. Application grammars, on the other hand, encode semantic structures, but in order to be accurate they are typically limited to specific domains. However, they are not written from scratch for each domain, but they use resource grammars as libraries [6].

The GF Resource Grammar Library (RGL) is a set of natural language grammars implemented in GF. These grammars are parallel in a strong sense: they are built upon a common abstract syntax, i.e. a common tree structure. Individual languages are obtained via compositional mappings from abstract syntax trees to feature structures specific to each language. The grammar defines, for each language, a complete set of morphological paradigms and a syntax fragment comparable to CLE (Core Language Engine)[7]. The current coverage is twenty-nine languages: Afrikaans, Amharic, Arabic, Bulgarian, Catalan, Danish, Dutch, English, Finnish, French, German, Hindi, Interlingua, Japanese, Italian, Latin, Latvian, Nepali, Norwegian bokmål, Persian, Polish, Punjabi, Romanian, Russian, Sindhi, Spanish, Swedish, Thai, Turkish, Urdu.

3. **Syntax.** Chinese is a typical isolated language and lack inflections, so when we build Chinese resource grammar library, syntax fragment is taken into account, especially noun phrase, verb phrase, adjective phrase, clause and question clause.

3.1. **Noun phrases (NP).** According to internal combination of noun phrase in modern Chinese, noun phrase can be divided into three types of classical combination, including dozens of kinds of combination schemas[8-9]. For simplicity, noun phrase is linearized as a string in GF:

```
lincat NP = {s : Str}
```

There are more than twenty generation rules of generating NP in RGL[10], combining quantifier, determiner, pronoun with noun to generate NP.

In generation of NP in Chinese RGL, involving determiner is more complex, type of determiner is as follows:

```
param DetType = DTFull Number | DTNum | DTPoss
```

DetType represents type of determiner, including full name determiner, quantity determiner and pronoun determiner. According to the type of determiner, noun phrases can be generated as follows:

```
DetCN det cn = case det.detType of {
        DTFull Sg => {s = det.s ++ cn.c    ++ cn.s} ;    -- this house
        DTFull Pl => {s = det.s ++ xie_s ++ cn.s} ;    -- these houses
        DTNum     => {s = det.s ++ cn.c    ++ cn.s} ;    -- (these) five houses
        DTPoss    => {s = det.s                ++ cn.s}      -- our (five) houses
  }
```

3.2. **Verb phrases (VP).** In Chinese RGL, verb phrases are represented as a record with three fields.

```
VP = {verb : Verb ; compl : Str ; prePart : Str}
```

where verb is verb category, compl represents complement of VP, prePart represents auxiliary verb of VP.

At the level of verb, there are five aspects of verb, including progressive aspect, continuous aspect, perfect aspect, future and past. In GF representation, we define aspects of verb as parameter:

```
Aspect = APlain | APerf | ADurStat | ADurProg | AExper
```

When we generate verb, aspects of verb are taken into account, and some relevant adverb or auxiliary verb are added.

```
regVerb : (walk : Str) -> Verb = \v -> mkVerb v "了" "着" "在" "过" "没"
```

There are more than thirty generation rules of generating NP in RGL[10], covering subject-predicate phrase, verb-object phrase, verb-complement phrase, adverbial-verb phrase, coordinative phrase, etc[9]. For example, through combination VP with adverb, it can generate verb phrase modified by the adverb.

```
AdvVP vp adv = case adv.advType of {
        ATManner => insertObj (ss (deVAdv_s ++ adv.s)) vp ;
            _ => insertAdv (ss (zai_V.s ++ adv.s)) vp
}
AdvType = ATPlace | ATTime | ATManner
```

where AdvType represents type of adverb, ATPlace is adverb of position, ATTime is time adverb and ATManner is manner adverb.

In generation of VP in Chinese RGL, “ba” phrase is special, it’s generation rules are as follows:

```
Slash2V3 v np = insertAdv (mkNP (ba_s ++ np.s)) (predV v) ** {c2 = v.c3 ; isPre = False}
Slash3V3 v np = insertObj (mkNP (appPrep v.c3 np.s)) (predV v) ** {c2 = v.c2 ; isPre =
True}
```

3.3. **Adjective phrases (AP).** In Chinese RGL , adjective phrases are represented as a record with two fields.

```
AP = {s : Str; monoSyl: Bool}
```

where s is kernel word and monoSyl represents AP is monosyllabic or not.

There are more than ten generation rules of generating AP in RGL[10], covering adverbial-verb structure, predicate-complement structure, etc.

Cite generation of comparative adjective phrase for example, it’s signature at abstract level is as follow:

```
fun ComparA : A -> NP -> AP
```

it’s corresponding implementation at concrete level is:

```
ComparA a np = complexAP (than_s ++ np.s ++ a.s)
```

3.4. **Clauses (Cl).** In Chinese RGL , clauses are represented as a record with three fields.

```
Clause : Type = {
        s   : Polarity => Aspect => Str ;
        np : Str;
        vp : Polarity => Aspect => Str
}
```

where np and vp represent noun phrase and verb phrase respectively, clause has variable polarity and aspect. Keep np and vp separate is to insert interrogative adverb as needed. Polarity represents polarity of clauses and Aspect represents aspect of clauses (Chinese is a language without tense, aspect is not displayed directly to the verb form and is expressed by time Adverbs [11]).

There are more than thirty generation rules of generating Cl in RGL[10].

Cite most simple generation of clause for example, it’s signature at abstract level is as

follow:

fun UseCl : Temp -> Pol -> Cl -> S

It's corresponding implementation at concrete level is:

UseCl t p cl = {s = cl.s ! p.p ! t.t}

3.5. **Question Clauses (QCl).** In Chinese RGL , question clauses are represented as a table type.

QCl = {s : Polarity => Aspect => Str}

where Polarity represents polarity of question clause and Aspect represents aspect of question clause.

There are almost thirty generation rules of generating QCl in RGL[10].

Cite most simple generation of clause for example, it's signature at abstract level is as follow:

fun QuestCl: Cl –> QCl

it's corresponding implementation at concrete level is adding a "ma" at the end of question clause:

QuestCl cl = {s = \\p,a => cl.s ! p ! a ++ question_s}

However this clause still has variable tense and polarity which is fixed at sentence level e.g

fun UseQCl : Temp -> Pol -> QCl -> QS

Other forms of question clauses include clauses made with interrogative pronouns (IP), interrogative adverbs (IAdv), and interrogative determiners (IDet), categories. Some of the functions for creating question clauses are:

QuestVP : IP -> VP -> QCl

QuestIAdv : IAdv -> Cl -> QCl

3.6. **Miscellaneous.** There are big differences between the Indo-European languages and Chinese in structure words, numeral words, etc. For example, the conjunction "and" implemented in Chinese RGL need distinguish parts of the connection is "phrases" or "sentence". According to the parts of the connection, different conjunctions are selected.

```
and_Conj = {s = table {
                CPhr CNPhrase => mkConjForm "和" ;
                CPhr CAPhrase => mkConjForm "而" ;
                CPhr CVPhrase => mkConjForm "又" ;
                CSent => mkConjForm "并且"
    }
            } ;
```

where

ConjForm = CPhr CPosType | CSent;

CPosType = CAPhrase | CNPhrase | CVPhrase ;

ConjForm represents type of connection: "Phrases" or "Sentences". CPosType represents subtype of "Phrases": AP or VP or NP.

4. **Example.** As an example consider the translation of following sentence from English to

Chinese, to see how our implementation works.

“if the man is old then the woman is old”

Figure 2 shows the parse tree for this sentence (Figure 2 is automatically produced by running command “p -lang=Eng “if the man is old then the woman is old” | vt –view=open”in GF).

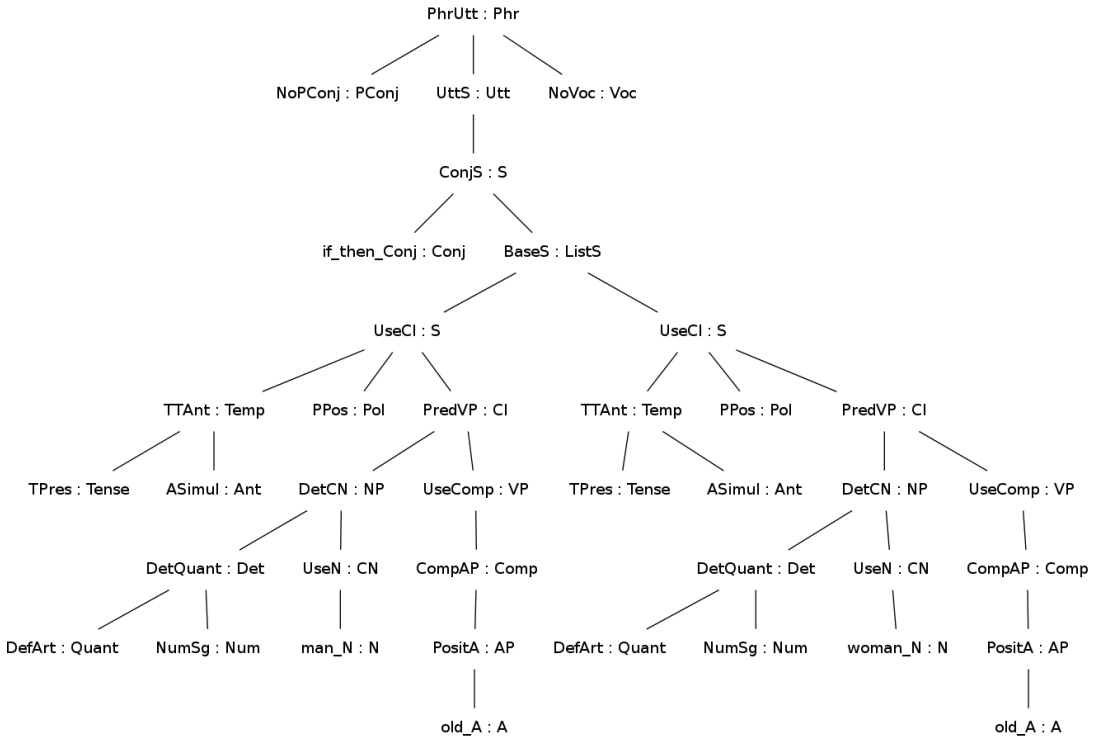

FIGURE 2. The parse tree of “if the man is old then the woman is old”

The nodes in this tree represent different categories and its branching shows how a particular category is built from other categories and/or leaves (words from lexicon). In GF notation these are the syntactic rules which are declared at abstract level.

Table 2 illustrates main category building of ”if the man is old then the woman is old”.

TABLE 2. Category building type signature

Category	*Type signature*	*Remark*
Phr	*fun PhrUtt: PConj -> Utt -> Voc -> Phr*	*Phrase is builded from conjunction ,utterance.*
Utt	*fun UttS S -> Utt*	*Utterance is builded from Sentence*
S	*fun ConjS: Conj -> [S] -> S*	*Sentence is builded from conjunction of some sentences*
S	*fun UseCl: Temp -> Pol -> Cl -> S*	*Sentence is builded from polarity, clause*
Cl	*fun PredVP: NP -> VP -> Cl*	*Clause is builded from noun phrase and verb phrase*
NP	*fun DetCN: Det -> CN -> NP*	*Noun Phrase is builded from determiner, proper noun*
VP	*fun UseComp: Comp -> VP*	*Verb Phrase is builded from copula*

In GF, we linearize the parse tree to Chinese implementation of RGL. The result of

linearization is the translation of ” if the man is old then the woman is old” to Chinese (in GF , by running command “i english/LangEng.gf chinese/LangChi.gf”, we imports English and Chinese RGL, and then run command “ p -lang=Eng “if the man is old then the woman is old” | l ”):

“如果男人是老的那么女人是老的”

The translation procedure can be demonstrated as figure 3.

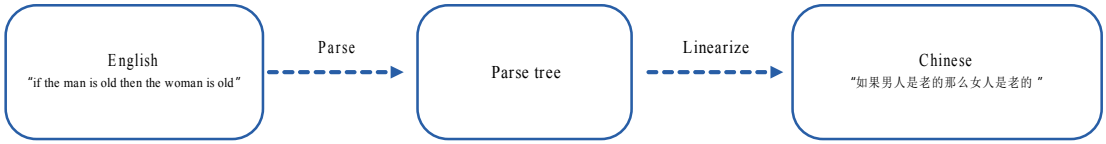

FIGURE 3. The illustration of translation procedure

Additionally, we can show the word alignment of the two sentences. (In GF, running command “p -lang=Eng “if the man is old then the woman is old” | aw –view=”eog””)

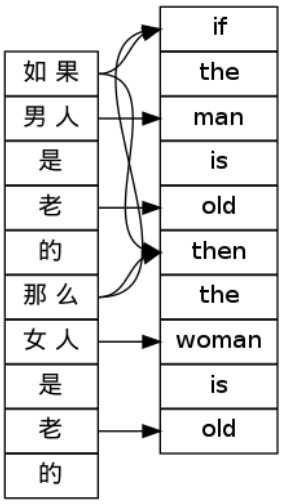

FIGURE 4. Word alignment

5. **Experiment and Test.** We has implemented around 80 categories and 200 construction functions in Chinese resource grammar. For test our implementation, we experiment as follow procedure:

1. Run GF
2. Import English and Chinese resource grammar: i english/LangEng.gf chinese/LangChi.gf
3. Parse English sample sentences, and translate them to Chinese sentences, for example, p -lang=Eng “if the man is old then the woman is old” | l
4. Evaluate the automatically translated Chinese sentences manually and subjectively, if the Chinese sentence is grammatically correct and agree with Chinese practice well, then the translation is evaluated as “good”. If the Chinese sentence is grammatically correct and agree with Chinese practice basically, then the translation is evaluated as “soso”. If the Chinese sentence is not grammatically correct, then the translation is evaluated as “bad”.

We select 450 English sample sentences covering almost all categories for testing. The result is as table 3.

TABLE 3. evaluating result

Number of Samples	*Number of "good"*	*Number of "soso"*	*Number of "bad"*
450	*338*	*80*	*32*

According to the test, we find that the percentage of "good" is just 75%, and he percentage of "bad" is 7%. The result is not ideal, major reason is that we don't handle Chinese characteristics very well, just like word order, functional word and all kinds of complex and flexible phrase structure.

6. **Conclusions.** We have implemented Chinese resource grammar in Grammatical Framework, which is applied in language analysis and language generation. We have applied it to automatic translation system between Object Constraint Language and Chinese. However, as we mentioned earlier, the implementation does not handle Chinese characteristics very well, which will be our recent work. In addition, automatic translation between natural language and artificial language, improving GF and abstract syntax in RGL, integrating semantic checking based on montague grammar are our future work direction.

GF is a grammar formalism based on type theory and a software library and software development platform. Our implementation of Chinese resource grammar in RGL can promote multilingual translation, and more importantly can become a research platform and comparative study platform of Chinese grammar characteristics.

Acknowledgment. This work has been supported by the Fundamental Research Funds for the Central Universities (Approval number: 11JBB036 , 12YBG04 , XK201203).The author would like to thank Aarne Ranta and Zhuo Lin Qiqige, our work is based on their work, and Our implementation is instructed by Aarne Ranta. Thank the anonymous reviewers for their suggestions on how to improve this paper.

REFERENCES

[1] Ranta, A. (2011). Grammatical Framework: Programming with Multilingual Grammars. Stanford: CSLI Publications.

[2] Yu Shiwen, Xue-Feng Zhu. Detailed Explanation of Modern Chinese Grammar Information Dictionary (Second Edition) [M] Beijing: Tsinghua University Press, 2003.

[3] Dong Zhendong Dong Qiang. HowNet see website: http://www.keenage.com/ the, 2013.2.

[4] Peter Ljunglöf. Expressivity and complexity of the Grammatical Framework. GÖTEBORG university. 2004.

[5] Ranta A. The GF Resource Grammar Library A systematic presentation of the library from the linguistic point of view. to appear in the on-line journal Linguistics in Language Technology, 2009.

[6] Ranta A. Grammars as Software Libraries. From Semantics to Computer Science, Cambridge University Press, Cambridge, pp. 281-308, 2009.

[7] Alshawi, H. 1992. The Core Language Engine. Cambridge, Ma: MIT Press.

[8] Zhan Weidong, the study of modern Chinese phrase structure rules for Chinese information processing. Peking University doctoral dissertation in 1999.

[9] Zhu Dexi grammar handouts Commercial Press in 1982.

[10] GF Resource Grammar Library.http://www.grammaticalframework.org/lib/doc/synopsis.html.2013.2

[11] Qu Chengxi, the Chinese cognitive function syntax, Harbin: Heilongjiang People's Publishing House, 2005.