Machine Translation Based on Constraint-Based Synchronous Grammar

Fai Wong[1], Dong-Cheng Hu[1], Yu-Hang Mao[1],
Ming-Chui Dong[2], and Yi-Ping Li[2]

[1] Speech and Language Processing Research Center,
Department of Automation, Tsinghua University, 100084 Beijing
huangh01@mails.tsinghua.edu.cn
{hudc, myh-dau}@mail.tsinghua.edu.cn
[2] Faculty of Science and Technology of University of Macao,
Av. Padre Tomás Pereira S.J., Taipa, Macao
{dmc, ypli}@umac.mo

Abstract. This paper proposes a variation of synchronous grammar based on the formalism of context-free grammar by generalizing the first component of productions that models the source text, named Constraint-based Synchronous Grammar (CSG). Unlike other synchronous grammars, CSG allows multiple target productions to be associated to a single source production rule, which can be used to guide a parser to infer different possible translational equivalences for a recognized input string according to the feature constraints of symbols in the pattern. Furthermore, CSG is augmented with independent rewriting that allows expressing discontinuous constituents in the inference rules. It turns out that such grammar is more expressive to model the translational equivalences of parallel texts for machine translation, and in this paper, we propose the use of CSG as a basis for building a machine translation (MT) system for Portuguese to Chinese translation.

1 Introduction

In machine translation, to analyze the structure deviations of languages pair hence to carry out the transformation from one language into another as the target translation is the kernel part in a translation system, and this requires a large amount of structural transformations in both grammatical and concept level. The problems of syntactic complexity and word sense ambiguity have been the major obstacles to produce promising quality of translation. In order to overcome the obstacles and hence to improve the quality of translation systems, several alternative approaches have been proposed.

As stated in [1], much of the theoretical linguistics can be formulated in a very natural manner as stating correspondences between layers of representations. In similar, many problems in natural language processing, in particular language translation and grammar rewriting systems, can be expressed as transduction through the use of synchronous formalisms [2,3,4,5,6]. Recently, synchronous grammars are becoming more and more popular for the formal description of parallel texts representing translations for the same document. The underlying idea of such formalisms is to combine two generative devices through a pairing of their productions in such a way that right hand side non-terminal symbols in the paired productions are linked. However, such formalisms are less expressive and unable to express mutual translations that have different lengths and crossing dependencies. Moreover, synchronous formalisms do not deal with unification and feature structures, as in unification-based formalisms, that give patterns additional power for describing constraints on features. For examples, Multiple Context-Free Grammar [4], where functions are engaged to the non-terminal symbols in the productions to further interpreting the symbols in target generation. In [7], Inversion Transduction Grammar (ITG) has been proposed for simultaneously bracketing parallel corpora as a variant of Syntax Directed translation schema [8]. But these formalisms are lacked of expressive to describe discontinuous constituents in linguistic expression. Generalized Multitext Grammar (GMTG) proposed by [5,9] is constructed by maintaining two sets of productions as components, one for each language, for modeling parallel texts. Although GMTG is more expressive and can be used to express as independent rewriting, the lack of flexibility in the way to describe constraints on the features associated with a non-terminal makes it difficult to the development of practical MT system.

In this paper, a variation of synchronous grammar, Constraint-Based Synchronous Grammar (CSG), is proposed based on the formalism of context-free grammar. Through the use of feature structures as that in unification-based grammar, the first component of productions in CSG, that describes the sentential patterns for source text, is generalized while the corresponding target rewriting rules for each production are grouped in a vector representing the possible translation patterns for source production. The choice of rule for target generation is based on the constraints on features of non-terminal symbols in pattern. Our motivation is three-fold. First, synchronous formalisms have been proposed for modeling of parallel text, and such algorithms can infer the synchronous structures of texts for two different languages through the grammar representation of their syntactic deviations. That is quite suitable for use in the analysis of languages pair in the development of MT system. Secondly, by augmented the synchronous models with feature structures can enhance the pattern with additional power in describing gender, number, agreement, etc. Since the descriptive power of unification-based grammars is considerably greater than that of classical CFG [10,11]. Finally, by retaining the notational and intuitive simplicity of CFG, we can enjoy both a grammar formalism with better descriptive power than CFG and more efficient parsing and generation algorithm controlled by the feature constraints of symbols hence to achieve the purposes of word sense and syntax disambiguation.

2 Constraint-Based Synchronous Grammars

Constraint-Based Synchronous Grammars (CSG) is defined by means of the syntax of context-free grammar (CFG) to the case of synchronous. The formalism consists of a set of generative productions and each production is constructed by a pair of CFG rules with zero and more syntactic head and link constraints for the non-terminal symbols in patterns. In a similar way, the first component (in right hand side of productions) represents the sentential patterns of source language, while the second component represents the translation patterns in target language, called source and target component respectively in CSG. Unlike other synchronous formalisms, the target component of production consists of one or more generative rules associated with zero or more controlled conditions based on the features of non-terminal symbols of source rule for describing the possible generation correspondences in target translation. In such a way, the source components in CSG are generalized by leaving the task of handling constraints on features in target component, so this also helps to reduce the grammar size. For example, following is one of the productions used in the MT system for Portuguese to Chinese translation:

$$\begin{aligned} S \rightarrow NP_1\ VP^*\ NP_2\ PP\ NP_3\ \{ & [NP_1\ VP^1\ NP_3\ VP^2\ NP_2; VP_{cate}=vb1, \\ & VP_{s:sem} = NP_{1sem}, VP_{io:sem}=NP_2 sem, VP_{o:sem}=NP_{3sem}], \\ & [NP_1\ VP\ NP_3\ NP_2\ ; VP=vb0, VP_{s:sem}=NP_{1sem}, \\ & VP_{io:sem}=NP_{2sem}]\} \end{aligned} \qquad (1)$$

The production has two components beside the reduced syntactic symbol on left hand side, the first modeling Portuguese and the second Chinese. The target component in this production consists of two generative rules maintained in vector, and each of which is engaged with control conditions based on the features of symbols from the source component, and this is used as the selectional preferences in parsing. These constraints, in the parsing/generation algorithm, are used for inferring, not only, the structure of input to dedicate what structures are possible or probable, but also the structure of output text for target translation. For example, the condition expression: $VP_{cate}=vb1$, $VP_{s:sem}=NP_{1sem}$, $VP_{io:sem}=NP_{2sem}$, $VP_{o:sem}=NP_{3sem}$, specifies if the senses of the first, second and the third nouns (*NP*s) in the input strings matched to that of the subject, direct and indirect objects governed by the verb, *VP*, with the category type of *vb1*. Once the condition gets satisfied, the source structure is successfully recognized and the corresponding structure of target language, $NP_1\ VP^1\ NP_3\ VP^2\ NP_2$, is determined also.

Non-terminal symbols in source and target rules are linked if they are given the same index "*subscripts*" for case of multiple occurrences, such as *NP*s in the production: $S \rightarrow NP_1\ VP\ NP_2\ PP\ NP_3\ [NP_1\ VP^*\ NP_3\ NP_2]$, otherwise symbols that appear only once in both the source and target rules, such as *VP*s, are implicitly linked to give the synchronous rewriting. Linked non-terminal must be derived from a sequence of synchronized pairs. Consider the production: $S \rightarrow NP_1\ VP\ NP_2\ PP\ NP_3\ [NP_1\ VP^*\ NP_3\ NP_2]$, the second *NP* ($NP_2$) in the source rule corresponds to the third *NP* (NP_2) in the target rule, the third *NP* (NP_3) in source rule corresponds to the second *NP* (NP_3) in target pattern, while the first *NP* (NP_1) and *VP* correspond to each other in both source and target rules. The symbol marked by an "*" is designated as *head* element in pattern, this allows the features of designated head symbol propagate to the reduced non-terminal symbol in the left hand side of production rule, hence to achieve the property of features inheritance in CSG formalism. The use of features structures associated to non-terminal symbols will be discussed in the later section in this paper.

In modeling of natural language, in particular for the process of languages-pair, the treatment for non-standard linguistic phenomena, i.e. crossing dependencies, discontinuous constituents, etc., is very important due to the structure deviations of two different languages, in particular for languages from different families such as Portuguese and Chinese [12,13]. Linguistic expressions can vanish and appear in translation. For example, the preposition (*PP*) in the source rule does not show up in any of the target rules in Production (1). In contrast, Production (2) allows the Chinese characters of "本" and "輛" to appear in the target rules for purpose to modify the noun (*NP*) together with the quantifier (*num*) as the proper translation for the source text. This explicitly relaxes the synchronization constraint, so that the two components can be rewritten independently.

$$\text{NP} \rightarrow \text{num NP* } \{[\text{num 本 NP; NP}_{sem}\text{=SEM}_{\_book}], [\text{num 輛 NP; NP}_{sem}\text{=SEM}_{\_automobile}]\} \quad (2)$$

A remarkable strength of CSG is its expressive power to the description of discontinuous constituents. In Chinese, the use of combination words that discontinuously distributed in a sentence is very common. For example, take the sentences pair ["*Ele vendeu-me todas as uvas.* (He sell me all the grapes.)", "他把所有的葡萄賣了給我 "]. The Chinese preposition "把" and the verb "賣了給" should be paired with the Portuguese verb "*vendeu*", and this causes a *fan-out*[1] and discontinuous constituent in the Chinese component. The following fragment of CSG productions represents such relationships.

$$\text{S} \rightarrow \text{NP}_1 \text{ VP* NP}_2 \text{ NP}_3 \ \{[\text{NP}_1 \text{ VP}^1 \text{ NP}_3 \text{ VP}^2 \text{ NP}_2 \text{ ; VP}_{cate}\text{=vb0,...}],..\} \quad (3)$$

$$\text{VP} \rightarrow vendeu^* \ \{[\text{把, 賣了給 };\varnothing]\} \quad (4)$$

In Production (3), the corresponding discontinuous constituents of *VP* (from source rule) are represented by *VP*^{1} and *VP*^{2} respectively in the target rule, where the "*superscripts*" are added to indicate the pairing of the *VP* in target component. The corresponding translation constituents in the lexicalized production are separated by commas representing the discontinuity between constituents "把" and "賣了給" in target translation. During the rewriting phase, the corresponding constituents will be used to replace the syntactic symbols in pattern rule.

3 Definitions

Let *L* be a context-free language defined over *terminal* symbol V_T and generated by a context-free grammar *G* using non-terminal symbol V_N disjointed with V_T, starting symbol *S*, and productions of the form $A \rightarrow w$ where *A* is in V_N and *w* in $(V_N \cup V_T)^*$. Let *Z* as a set of integers, each non-terminal symbol in V_N is assigned with an integer, $\Gamma(V_N) = \{W_\omega \mid W \in V_N, \omega \in Z\}$. The elements of $\Gamma(V_N)$ are indexed non-terminal symbols. Now, we extend to include the set of *terminal* symbols $V_{T'}$ as the translation in target language, disjoint from V_T, $(V_T \bullet V_{T'} = \varnothing)$. Let $R = \{r_1, \ldots, r_n \mid r_i \in (\Gamma(V_N) \cup V_{T'}), 1 \leq i \leq n\}$ be a finite set of rules, and $C = \{c_1, \ldots, c_m\}$ be a finite set of constraints over the associated features of $(\Gamma(V_N) \cup V_T)$, where the features of non-terminal $\Gamma(V_N)$, the syntactic symbols, are inherited from the designated head element during rule reduction. A *target rule* is defined as pair $[r \in R^*, c \in C^*]$ in γ, where $\gamma = R^* \times C^*$ in form of $[r, c]$. Now, we define $\psi(\gamma_i)$ to denote the number of conjunct features being considered in

[1] We use this term for describing a word where its translation is paired of discontinuous words in target language, e.g. "*vendeu*[*-pro*] [*NP*]" in Portuguese gives similar English translation as "*sell* [*NP*] *to* [*pro*]", so "*vendeu*", in this case, is corresponding to "*sell*" and "*to*".

the associated constraint, hence to determine the degree of generalization for a constraint. Therefore, the rules, γ_i and γ_j, are orderable, $\gamma_i \prec \gamma_j$, if $\psi(\gamma_i) \geq \psi(\gamma_j)$ (or $\gamma_i \succ \gamma_j$, if $\psi(\gamma_i) < \psi(\gamma_j)$). For $\gamma_i \prec \gamma_j$ ($\psi(\gamma_i) \geq \psi(\gamma_j)$), we say, the constraint of the rule, γ_i, is more specific, while the constraint of γ_j is more general. In what follows, we consider a set of related target rules working over the symbols, *w'*, on the RHS of production $A \rightarrow w'$, the source rule, where $w' \in \Pi(V_N) \cup V_T$. All of these non-terminals are co-indexed as *link*.

Definition 1: A *target component* is defined as a ordered vector of *target rules* in γ having the form $\sigma = \{\gamma_1, \ldots, \gamma_q\}$, where $1 \leq i \leq q$ to denote the *i*-th tuple of σ. The rules are being arranged in the order of $\gamma_1 \prec \gamma_2 \prec \ldots \prec \gamma_q$.

In rule reduction, the association conditions of the target rules are used for investigating the features of corresponding symbols in source rules, similar to that of feature unification, to determine if the active reduction successes or not. At the mean while, this helps in determining the proper structure as the target correspondence.

Definition 2: A *Constraint-Based Synchronous Grammar* (*CSG*) is defined to be 5-tuple $G = (V_N, V_T, P, C_T, S)$ which satisfies the following conditions:

- V_N is a finite set of *non-terminal* symbols;
- V_T is a finite set of *terminal* symbols which is disjoint with V_N;
- C_T is a finite set of *target components*;
- P is a finite set of *productions* of the form $A \rightarrow \alpha\, \beta$, where $\alpha \in (\Pi(V_N) \cup V_T)^*$ and, $\beta \in C_T$, the non-terminal symbols that occur from both the source and target rules are *linked* under the index given by $\Pi(V_N)$[2].
- $S \in V_N$ is the initial symbol.

For example, the following CSG productions can generate both of the parallel texts [“*Ele deu um livro ao José*. (He gave a book to José)”, “*他給了若澤一本書*”] and [“*Ele comprou um livro ao José*. (He bought a book from José)”, “*他向若澤買了一本書*”]:

$$\begin{aligned} S \rightarrow NP_1\ VP^*\ NP_2\ PP\ NP_3\ \{[&NP_1\ VP^1\ NP_3\ VP^2\ NP_2; VP_{cate}=vb1,\\ &VP_{s:sem} = NP_{1sem}, P_{io:sem}=NP_2 sem, VP_{o:sem}=NP_{3sem}],\\ &[NP_1\ VP\ NP_3\ NP_2\ ; VP=vb0, VP_{s:sem}=NP_{1sem},\\ &VP_{io:sem}=NP_{2sem}]\} \end{aligned} \tag{5}$$

$$VP \rightarrow v^3\ \{[v\ ; \varnothing]\} \tag{6}$$

$$NP \rightarrow det\ NP^*\ \{[NP\ ; \varnothing]\} \tag{7}$$

$$NP \rightarrow num\ NP^*\ \{[num\ \text{本}NP; NP_{sem}=SEM_{\_book}]\} \tag{8}$$

[2] Link constraints are dedicated by the symbols indices, which is trivially for connecting the corresponding symbols between the source and target rules. Hence, we assume, without loss of generality, that index is only given to the non-terminal symbols that have multiple occurrences in the production rules. It is assumed that “$S \rightarrow NP_1\ VP_2\ PP_3\ NP_4\ \{NP_1\ VP_2^1\ NP_4\ VP_2^2\}$” implies “$S \rightarrow NP_1\ VP\ PP\ NP_2\ \{NP_1\ VP^1\ NP_2\ VP^2\}$”.

[3] Similar for the designation of head element in productions, the only symbol from the RHS of production will inherently be the head element. Thus, no head mark “*” is given for such rules, and we assume that “$VP \rightarrow v^*$” implies “$VP \rightarrow v$”.

NP → n {[n ; ∅]} (9)

NP → pro {[pro ; ∅]} (10)

PP → p {[p ; ∅]} (11)

n → *José* {[若澤 ; ∅]}| *livro* {[書 ; ∅]} (12)

pro → *ele* {[他 ; ∅]} (13)

v → *deu*{[給了 ; ∅]} | *comprou* {[向, 買了 ;∅]} (14)

num → *um* {[一 ; ∅]} (15)

p → *a* {∅} (16)

det → *o* {∅} (17)

A set *P* of productions is said to *accept* an input string *s* iff there is a derivation sequence *Q* for *s* using source rules of *P*, and any of the constraint associated with every *target component* in *Q* is satisfied[4]. Similarly, *P* is said to *translate s* iff there is a synchronized derivation sequence *Q* for *s* such that *P* accepts *s*, and the link constraints of associated *target rules* in *Q* is satisfied. The derivation *Q* then produces a translation *t* as the resulting sequence of terminal symbols included in the determined target rules in *Q*. The translation of an input string *s* essentially consists of three steps. First, the input string is parsed by using the source rules of productions. Secondly, the link constraints are propagated from source rule to target component to determine and build a target derivation sequence. Finally, translation of input string is generated from the target derivation sequence.

3.1 Feature Representation

In CSG, linguistic entities are modeled as feature structures which give patterns additional power for describing gender, number, semantic, attributes and number of the arguments required by a verb, and so on. These information are encoded in the commonly used attribute value matrices (AVMs), attached to each of the lexical and syntactic symbols in CSG. This allows us to specify such as syntactic dependencies as agreement and sub-categorization in patterns. Unlike other unification-based grammars [11,14], we do not carry out the unification in full, only interested conditions that are explicitly expressed in the rule constraints are tested and unified. Such unification process can perform in constant time. The use of feature constraints has to be restricted to maintain the efficiency of parsing and generating algorithms, especially to the prevention from generating a large number of ambiguous structure candidates. The word selection in the target language can also be achieved by checking features. In the parsing and generating algorithm, the features information are propagated to the reduced symbol from the designated head element in pattern, hence to realize the mechanism of features inheritance. Features can either be put in lexical dictionary isolated from the formalism to make the work simpler to the construction of analytical grammar, or explicitly encoded in the pre-terminal rules as:

[4] If there is no any constraint associated to a target rule, during the parsing phase, the reduction of the source rule is assumed to be valid all the time.

Pro → *José*:[CAT:*pro*;NUM:*sg*;GEN:*masc*,SEM:*hum*] {[若澤 ; ∅]} (18)

n → *livro*:[CAT:*n*;NUM:*sg*;GEN:*masc*;SEM:*artifact+book*] {[書 ; ∅]} (19)

Where the features set is being bracketed, and separated by a semi-colon, the name and the value of a feature are delimited by a colon to represent the feature pair. Another way to enhance the CSG formalism is to apply the soft preferences other than hard constraints in the process of features unification. Our consideration is two-fold: first, we found that more than one combination of feature values engaged to a single lexical item is very common in the process of natural language, i.e. one word may have several translations according to the different senses and the pragmatic uses of the word, and this has been the problem of word senses disambiguation [15]. Secondly, the conventional feature unification method can only tell us if the process successes or not. In case of a minor part of conditions get failed during the unification, all the related candidates are rejected without any flexibility to choosing the next preferable or probable candidate. In order to resolve these problems, each feature structure is associates with a weight. It is then possible to rank the matching features according to the linear ordering of the weights rather than the order of lexical items expressed in grammars or dictionary. In our prototyping system, each symbol has its original weight, and according to preference measurement at the time in checking the feature constraints, a penalty is used to reduce from the weight to give the effective weight of associated features in a particular context. Features with the largest weight are to be chosen as the most preferable content.

4 Application to Portuguese-Chinese MT

CSG formalism can be parsed by any known CFG parsing algorithm including the Earley [16] and generalized LR algorithms [17] augmented by taking into account the features constraints and the inference of target structure. In the prototyping system, the parsing algorithm for our formalism is based on the generalized LR algorithm that we have development for MT system, since the method uses a parse table, it achieves a considerable efficiency over the Earley's non-complied method which has to compute a set of LR items at each stage of parsing [17]. Generalized LR algorithm was first introduced by Tomita for parsing the augmented Context-Free grammar that can ingeniously handle non-determinism and ambiguity through the use of graph-structured stack while retaining much of the advantages of standard LR parsing[5]. It takes a shift-reduce approach using an extended LR parse table to guide its actions by allowing the multiple actions entries such as shift/reduce and reduce/reduce hence to handle the nondeterministic parse with pseudo-parallelism. In order to adapt to our formalism, we further extend the parse table by engaging with the features constraints and the target rules into the actions table. Our strategy is thus to parse the source rules of CSG productions through the normal shift actions proposed by the parsing table, while at the time reduce action to be fired, the associated conditions are checked to determine if the active reduction is a valid action or not depending on if the working symbols of patterns fulfill the constraints on features.

[5] Especially when the grammar is close to the LR grammars.

4.1 The CSG Parse Table

Fig. 1 shows an extended LR(1) parsing table for Productions (5)-(17)[6] as constructed using the LR table construction method described in [18] extended to consider the rule components of productions by associating the corresponding target rules with constraints, which are explicitly expressed in table. The parsing table consists of two parts: a compact ACTION-GOTO table[7] and CSONTRAINT-RULE table. The ACTION-GOTO table s indexed by a state symbol s (row) and a symbols $x \in V_N \cup V_T$, including the end marker "$\perp$". The entry ACTION[s, x] can be one of the following: *s n*, *r m*, *acc* or blank. *s n* denotes a shift action representing GOTO[s, x]=n, defining the next state the parser should go to; *r m* means a reduction by the m^{th} production located in the entry of CONSTRAINT-RULE in state s, and *acc* denotes the accept action and blank indicates a parsing error. The CONSTRAINT-RULE table is indexed by state symbol s (row) and the number of productions m that may be applied for reduction in state s. The entry CONSTRAINT-RULE[s, m] consists of a set of involved productions together with the target rules and features constraints that are used for validating if the active parsing node can be reduced or not, then try to identify the corresponding target generative rule for reduced production.

4.2 The CSG Parser

In the parsing process, the algorithm operates by maintaining a number of parsing processes in parallel, each of which represents an individual parsed result, hence to handle the case of non-deterministic. In general, there are two major components in the process, *shift*(i) and *reduce*(i), which are called at each position i=0, 1, …, n in an input string $I = x_1x_2\ldots x_n$. The *shift*(i) process with top of stack vertex v shifts on x_i from its current state s to some successor state s' by creating a new leaf v'; establishing edge from v' to the top of stack v; and making v' as the new top of stack vertex.

The *reduce*(i) executes a reduce action on a production p by following the chain ofparent links down from the top of stack vertex v to the ancestor vertex from which the process began scanning for p earlier, then popping intervening vertices off the stack. Now, for every reduction action in *reduce*(i), there exists a set C of ordered constraints, $c_1 \prec \ldots \prec c_m$, with the production, each of which is associated with a target rule that may be the probable corresponding target structure for the production, depending on whether the paired constraint gets satisfied or not according to the features of the parsed string p. Before reduction takes place, the constraints c_j ($1 \leq j \leq m$) are tested in order started from the most specific one, the evaluation process stops once a positive result is obtained from evaluation. The corresponding target rule for the parsed string is determined and attached to the reduced syntactic symbol, which will be used for rewriting the target translation in phase of generation. At the mean while, the features information will be inherited from the designated head element of production. The parsing algorithm for CSG formalism is given in Fig. 2.

[6] For simplicity, the productions used for building the parse table are deterministic, so no conflict actions such as shift/reduce and reduce/reduce appear in the parse table in Fig.1.

[7] Original version introduced in [17] maintains two tables, ACTION and GOTO.

Ste	ACTIONs/GOTOs																			Reduced Rules Constraints/Target Rules
	pro	*num*	*n*	*v*	*det*	*p*	*NP*	*VP*	*PP*	*S*	⊣	*o*	*a*	*um*	*ele*	*José*	*livro*	*deu*	*comprou*	
0	s8	s9	s10		s11		s7			s6		s5		s2	s1	s4	s3			
1															r1					(1) pro → *ele* {[*他* ; ∅]}
2														r1						(1) num → *um*
3																	r1			(1) n → *livro* {[書 ; ∅]}
4																r1				(1) n → *José* {[*若澤* ; ∅]}
5												r1								(1) det → *o*
6											acc									
7				s14				s15										s12	s13	
8	r1																			(1) NP → pro
9	s8	s9	s10		s11		s16					s5		s2	s1	s4	s3			
10			r1																	(1) NP → n
11	s8	s9	s10		s11		s17					s5		s2	s1	s4	s3			
12																		r1		(1) v → *deu* {[*給了* ; ∅]}
13																			r1	(1) v → *comprou* {[*向*, *買了* ;∅]}
14				r1																(1) VP → v
15	s8	s9	s10		s11		s18					s5		s2	s1	s4	s3			
16							r1													(1) NP → *num* NP* {[num 本NP; NP_{sem}=$SEM_{\_book}$]}
17							r1													(1) NP → det NP* {[NP ; ∅]}
18						s21			s20				s19							
19													r1							(1) p → *a*
20	s8	s9	s10		s11		s22					s5		s2	s1	s4	s3			
21						r1														(1) PP → p
22										r1										(1) S → NP_1 VP* NP_2 PP NP_3 {[...]}

Fig. 1. Extended LR(1) parse table

```
PARSE(grammar, $x_1$ … $x_n$)
  $x_{n+1} \Leftarrow \perp$
  $U_i \Leftarrow \varnothing$ $(0 \le i \le n)$
  $U_0 \Leftarrow v_0$
  for each terminal symbol $x_i$ $(1 \le i \le n)$
    $P \Leftarrow \varnothing$
    for each node $v \in U_{i-1}$
      $P \Leftarrow P \cup v$
      if ACTION[STATE($v$),$x_i$] = "shift $s'$", SHIFT($v$,$s'$)
      for each "reduce $p$" $\in$ ACTION[STATE($v$),$x_i$], REDUCE($v$,$p$)
      if "acc" $\in$ ACTION[STATE($v$),$x_i$], accept
    if $U_i = \varnothing$, reject

SHIFT($v$,$s$)
  if $v' \in U_i$ s.t. STATE($v'$)=$s$ and ANCESTOR($v'$,1)=$v$ and state
      transition $\delta(v,x)=v'$
    do nothing
```

```
 else
   create a new node $v'$
   s.t. STATE($v'$)=$s$ and ANCESTOR($v'$,1)=$v$ and state tran-
       sition $\delta(v,x)=v'$
   $U_i \Leftarrow U_i \cup v'$

REDUCE($v$,$p$)
 for each possible reduced parent $v_1' \in$ANCESTOR($v$,RHS($p$))
   if UNIFY($v$,$p$)="success"
     $s'' \Leftarrow$ GOTO($v_1'$,LHS($p$))
     if node $v'' \in U_{i-1}$ s.t. STATE($v''$)=$s''$
       if $\delta(v_1'$, LHS($p$))=$v''$
         do nothing
       else
       if node $v_2' \in$ANCESTOR($v''$,1)
         let $v_c''$ s.t. ANCESTOR($v_c''$,1)=$v_1'$ and STATE($v_c''$)=$s''$
         for each "reduce $p$" $\in$ ACTION[STATE($v_c''$),$x_i$]
            REDUCE($v_c''$,$p$)
       else
         if $v'' \in P$
           let $v_c''$ st. ANCESTOR($v_c''$,1)=$v_1'$ and STATE($v_c''$)=$s''$
           for each "reduce $p$" $\in$ ACTION[STATE($v_c''$),$x_i$]
              REDUCE($v_c''$,$p$)
         else
           create a new node $v_n$
           s.t. STATE($v_n$)=$s''$ and ANCESTOR($v_n$,1)=$v_1'$ and
           state transition $\delta(v_n,x)=v_1'$
           $U_{i-1} \Leftarrow U_{i-1} \cup v_n$
   else current reduction failed

UNIFY($v$,$p$)
 for "constraint $c_j$" $\in$ CONSTRAINT(STATE($v$)) ($1 \le j \le m$,
 $c_1 \prec \ldots \prec c_m$)
   if $\xi(c_j,p)$="true"              ($\xi(\varnothing,p)$="true")
     TARGET($v$)$\Leftarrow j$
     return "success"
```

Fig. 2. Modified generalized LR Parsing algorithm

The parser is a function of two arguments PARSE(*grammar*, $x_1 \ldots x_n$), where the grammar is provided in form of parsing table. It calls upon the functions SHIFT(v, s) and REDUCE(v, p) to process the shifting and rule reduction as described. The UNIFY(v, p) function is called for every possible reduction in REDUCE(v, p) to verify the legal reduction and select the target rule for the source structure for synchronization. The function TARGET(v) after unification passed is to dedicate the jth target rule as correspondence.

4.3 Translation as Parsing

Our Portuguese-to-Chinese translation (PCT) system is a transfer-based translation system by using the formalism of Constraint-Based Synchronous Grammar (CSG) as its analytical grammar. Unlike other transfer-based MT systems that the major components: analysis, transfer and generation are carried out individually in pipeline by using different sets of representation rules to achieve the tasks of structure analysis and transformation [19], in PCT, only a single set of CS grammar is used to dominate the translation task. Since the structures of parallel languages are synchronized in formalism, as well as the deviations of their structures are also captured and described by the grammar. Hence, to the translation of an input text, it essentially consists of three steps. First, for an input sentence *s*, the structure of string is analyzed by using the rules of source components from the CSG productions; by using the augmented generalized LR parsing algorithm as described. Secondly, the link constraints that are determined during the rule reduction process are propagated to the corresponding target rules *R* (as selection of target rules) to construct a target derivation sequence *Q*. And finally, based on the derivation sequence *Q*, translation of the input sentence *s* is generated by referencing the set of generative rules *R* that attached to the corresponding constituent nodes in the parsed tree, hence to realize the translation in target language.

5 Conclusion

In this paper, we have proposed a variation of synchronous grammar based on the syntax of context-free grammar, called Constraint-based Synchronous Grammar (CSG). The source components of CSG are being generalized for representing the common structure of language. Different from other synchronous grammars, each source rule is associated with a set of target productions, where each of the target rules is connected with a constraint over the features of source patterns. The set of target rules are grouped and maintained in a vector ordered by the specificity of constraints. The objective of this formalism is to allow parsing and generating algorithms to inference different possible translation equivalences for an input sentence being analyzed according to the linguistic features. We have presented a modified generalized LR parsing algorithm that has been adapted to the parsing our formalism that we have developed for analyzing the syntactic structure of Portuguese in the machine translation system.

References

1. Rambow, O., Satta, G.: Synchronous Models of Language. In Proceedings of 34th Annual Meeting of the Association for Computational Linguistics, University of California, Santa Cruz, California, USA, Morgan Kaufmann (1996) 116-123.
2. Lewis, P.M., Stearns, R.E.: Syntax-directed transduction. Journal of the Association for Computing Machinery, 15(3), (1968) 465-488.
3. Shieber, S.M., Schabes, Y.: Synchronous Tree Adjoining Grammar. Proceedings of the 13th International Conference on Computational Linguistic, Helsinki (1990)
4. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science, 88(2) (1991) 191-229

5. Melamed, I.D.: Multitext Grammars and Synchronous Parsers. In Proceedings of NAACL/HLT 2003, Edmonton, (2003) 79-86
6. Wong, F., Hu, D.C., Mao, Y.H., Dong, M.C. A Flexible Example Annotation Schema: Translation Corresponding Tree Representation. In Proceedings of the 20th International Conference on Computational Linguistics, Switzerland, Geneva (2004) 1079-1085
7. Wu, D.: Grammarless extraction of phrasal translation examples from parallel texts. In Proceedings of TMI-95, Sixth International Conference on Theoretical and Methodological Issues in Machine Translation, v2, Leuven Belgium (1995) 354-372
8. Aho, A.V., Ullman, J.D.: Syntax Directed Translations and the Pushdown Assembler. Journal of Computer and System Sciences, 3, (1969) 37-56
9. Melamed, I.D., Satta. G., Wellington, B.: Generalized Multitext Grammars. In Proceedings of 42th Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain (2004) 661-668
10. Kaplan, R.M., Bresnan, J.: Lexical-Functional Grammar: A Formal System for Grammatical Representation. In Joan Bresnan, The Mental Representation of Grammatical Relations, Cambridge, Mass, MIT Press, (1982) 173-281
11. Kaplan, R.M.: The Formal Architecture of Lexical-Functional Grammar. Information Science and Engineering, 5, (1989) 30-322
12. Wong, F., Mao, Y.H.: Framework of Electronic Dictionary System for Chinese and Romance Languages. Automatique des Langues (TAL), 44(2), (2003) 225-245
13. Wong, F., Mao, Y.H., Dong, Q.F., Qi, Y.H.: Automatic Translation: Overcome the Barriers between European and Chinese Languages. In Proceedings (CD Version) of First International UNL Open Conference, SuZhou China (2001)
14. Pollard, C., Sag, I.: Head-Driven Phrase Structure Grammar. University of Chicago Press, (1994)
15. Ide, N., Veronis, J.: Word Sense Disambiguation: The State of the Art. Computational Linguistics, 24, (1), (1998) 1-41
16. Earley, J.: An Efficient Context-Free Parsing Algorithm. CACM, 13(2), (1970) 94-102
17. Tomita, M.: Computational Linguistics, 13(1-2), (1987) 31-46
18. Aho, A.V., Sethi, R., Ullman, J.D.: Compiler: Principles, Techniques and Tools. Addison-Wesley, (1986)
19. Hutchins, W.J., Somers, H.L.: An Introduction to Machine Translation. Academic Press, (1992)