CDG: AN ALTERNATIVE FORMALISM FOR PARSING WRITTEN AND SPOKEN THAI

Siripong Potisuk

Department of Electrical Engineering, Academic Division, Chulachomklao Royal Military Academy, Nakorn-nayok 26000, THAILAND

and

Mary P. Harper

School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA

1. Introduction

The impetus for this research arose during an investigation of the language modelling aspects of an automatic speech understanding system of Thai. A good language model not only improves the accuracy of low-level acoustic models of speech, but also reduces task perplexity (the average number of choices at any decision point) by making better use of high-level knowledge sources including prosodic, syntactic, semantic, and pragmatic knowledge sources. A language model often consists of a grammar written using some formalism which is applied to a sentence by utilizing some sort of parsing algorithm. For example, a set of context-free grammar (CFG) production rules can be used to parse sentences in the language defined by that grammar. CFG is a phrase-structure representation of syntax (Chomsky, 1963). Another example is constraint dependency grammar (CDG). CDG parsers rule out ungrammatical sentences by propagating constraints. Constraints are developed based on a dependency-based representation of syntax. Some parsing algorithms combine phrase-structure and dependency grammars. In this paper, a CDG parsing approach is adopted. In the next section, we present a contrastive description of the two major approaches to representing the syntax of natural languages in order to motivate our choice of dependency grammar for parsing Thai sentences.

1.1 Dependency *vs.* phrase-structure grammar

In the theory of the syntax of natural languages, there are currently two major methods of representing the syntactic structure of natural sentences: dependency grammar and phrase-structure (constituency) grammar. There has been no third approach developed although combinations of the two major methods above have been used, e.g., lexical-functional grammar, case grammar, relational grammar, word grammar, etc. (Mel' čuk, 1988).

As a formal syntactic representation, dependencies have been studied and explored for centuries by traditional syntacticians of European, Classical, and Slavic languages. Lucien Tesnière (1959) was credited as the first syntactician who formalized and laid the groundwork for subsequent investigations of the theory. Unfortunately, the dependency formalism has not gained great popularity among today's syntacticians, who generally favor constituency grammar. Constituency or phrase-structure grammar was formulated in North America in the early 1930's by Leonard Bloomfield (1933), primarily for describing the syntax of English. The theory was seriously advanced by Noam Chomsky, and his transformational-generative approach has been accepted throughout the world. Phrase-structure syntax gradually forced dependency syntax into relative obscurity. Nevertheless, there have been some attempts to defend the use of dependency syntax, and several linguists have contributed to this cause. For example, Mel' čuk (1988) presented an argument for the case of dependency formalism and bravely claimed that "dependencies are much better suited to the description of syntactic structure (of whatever nature) than constituency is." His contrastive description of the two methods is summarized as follows.

A dependency grammar describes the syntactic structure of a sentence by using a dependency tree (D-tree) to establish dependencies among words in terms of head and dependents. A D-tree shows a relational characteristic of the syntactic representation in the form of hierarchical links between items, i.e., which items are related to which other items and in which way. On the other hand, a phrase-structure grammar uses a phrase-structure tree (PS-tree) to describe the groupings of words into the so-called *constituents* at different levels of sentence construction. A PS-tree shows which items go together with other items to form tight units of a higher order, a distributional characteristic of a grouping within a larger grouping.

A *tree* is a network consisting of nodes which are linked in a tree-like structure (i.e., with a stem and branches). In a syntactic tree, a node which represents a word or lexical item, the smallest syntactic unit, is called a terminal node; a node which represents an abstract syntactic grouping or phrase, such as noun phrase (NP), verb phrase (VP), prepositional phrase (PP), etc., is called a non-terminal node. A D-tree contains only terminal nodes; no abstract representation of syntactic groupings is used. On the contrary, a PS-tree contains both terminal and non-terminal nodes; most nodes are, however, non-terminal. This hierarchical representation in terms of terminals and non-terminals in a PS-tree leads to the notion of syntactic class membership of an item (i.e., categorization as belonging to an NP, VP, etc.). Syntactic class membership is a way of labelling syntactic roles in a PS-tree because a PS-tree does not and cannot specify the types of syntactic links existing between two items in a natural and explicit way. On the other hand, class membership is not specified in a D-tree. Instead, a D-tree puts a particular emphasis on specifying in detail the type of any syntactic relation between two related items. Such syntactic relations are, for example, predicative, determinative, coordinative relations, etc. In addition, in terms of the ordering of nodes, nodes must be ordered linearly in a PS-tree. In a D-tree, however, nodes are not necessarily in a linear order.

From the above, one can draw the following conclusion. The PS-representation is suitable for languages like English which have a rigid word order and a near absence of syntactically driven morphology. On the other hand, the D-representation is suitable for languages like Latin or Russian which feature an incredibly flexible (but far from arbitrary) word order and very rich systems of morphological markings. In these languages, word arrangements and inflectional affixes are obviously contingent upon relations between words rather than upon constituencies.

1.2 The difficulty of parsing Thai.

Research on the syntactic analysis of Thai sentences by computer has been carried out for over a decade. Thai grammars have been developed utilizing various grammar formalisms based on the above two theories of syntax or their combination. However, the most popular formalism has been the phrase-structure representation of syntax. Vorasucha (1986) was one of the first researchers to use Gazdar's Generalized Phrase-Structure Grammar (GPSG) in his research. Syntactic rules were written in ID and LP rule formats. Pornprasertsakul, et al. (1990) later employed additional FSD constraints of GSPG to describe three types of sentence structures including verb and noun phrases. Aroonmanakul (1990) developed a nondeterministic parser called CUPARSE based on a dependency representation of syntax. The parser uses a chart as its central data structure. As part of a project on machine translation of Asian languages, Sornlertlamvanich and Phantachart (1992) used a combination of phrase-structure and dependency grammars. Phrase-structure grammar rules were used to identify locally well-formed phrase patterns, and thus reduce lexical ambiguities, based on the relatively fixed relation of the positions of Thai words and their syntactic roles. Then, syntactic dependency structures among the words were generated based on verb subcategorization information. Finally, the syntactic dependency structure was mapped to a semantic one by utilizing lexical functional grammar. Wuwongse and Pornprasertsakul (1993) introduced a probabilistic approach using a *least exception logic* (LEL) model of default reasoning to resolve ambiguities. Despite increasing concerted efforts among Thai universities and government agencies, a satisfactory approach to analyzing Thai sentences has not been obtained. Difficulties in parsing Thai sentences arise for the following reasons.

First, written Thai sentences do not contain dilimiters or blanks between words. Unlike English, Thai words are not flanked by a blank space. Words are concatenated to form a phrase or sentence without explicit word dilimiters. This creates a problem for the syntactic analysis of Thai sentences because most parsers operate on words as the smallest syntactic unit in a sentence. To overcome this problem, a word segmentation module must be added to the front end of most Thai parsers. It may seem, on the surface, that the problem has been solved. But, on the contrary, a new problem has been created. Instead of analyzing a single sentence, a parser must now analyze multiple sentence hypotheses comprising a combination of all possible words generated by the word segmentation algorithm. For example, given the following string of Thai characters (Luksaneeyanawin, 1993), ภาพรอออกฉาก, two possible sentence hypotheses are generated given dictionary lookup. Note that the string พรอ is not a legitimate Thai word based on a Thai-English dictionary (McFarland, 1960).

a. ภาพ รอ ออก ฉาก

b. ภา พร ออ อก ฉาก

Secondly, Thai words lack inflectional and derivational affixes. Since words in Thai do not inflect to indicate their syntactic function, the position of a word in a sentence alone shows its syntactic function. Hence, syntactic relationships are primarily determined by word order, and structural ambiguity often arises. For example, without a subject-verb agreement feature or disambiguating context, there is no way of differentiating a 2-syllable noun-verb sequence from a 2-syllable compound noun comprising the same sequence of words. The following example illustrates the problem.

Compound:	เขาเป็นคนเจ้าชู้ มี คนรัก มาก
	'He is a flirting kind. He has a lot of girlfriends.'
Sentence:	เขาเป็นดารานิสัยดี มี คนรัก มาก
	'He is not a stuck-up movie star. Many fans love him.'

Thirdly, inconsistent ordering relations within and across phrasal categories characterize Thai sentences. Based on word order typology, the majority of world's languages usually exhibit consistent ordering relations across phrasal categories. The head (i.e., the central, obligatory member) of the phrase is usually placed either consistently before its modifiers and complements as in the so-called head-initial languages or after its modifiers and complements as in head-final languages. Such classification of a particular language is made according to the majority of its ordering relations within phrases, and thus, the distinction is based on tendencies, not exclusivity. In Thai, a noun, the head of a noun phrase, always precedes its modifying adjectives and determiners. The verb phrase, however, exhibits less consistency. Although a verb, the head of the verb phrase, always precedes its object, its modifying auxiliaries can either precede or follow it. In addition, constituents which optionally occur with the head in both noun and verb phrases, such as determiners and quantifiers, tend to be less consistent in their ordering as well. The following example contains three syntactically correct versions of the same sentence, *'He often invites his friends to have dinner at his house'*, each with a different ordering of constituents.

เขาชวนเพื่อน ๆ มากิน กัน ที่บ้าน บ่อย
เขาชวนเพื่อน ๆ มากิน กัน บ่อย ที่บ้าน
เขาชวนเพื่อน ๆ มากิน ที่บ้าน กัน บ่อย

Lastly, Thai sentences sometimes contain discontinuous sentence constituents in their construction. In grammatical analysis, discontinuity refers to the splitting of a construction by the insertion of another grammatical unit. In other words, discontinuity occurs when the elements which make up the constituents are interrupted by elements of another constituent in a sentence. Consider the following example in which the object noun phrase (NP-obj) is interrupted by the auxiliary (aux.).

เพื่อน	ยืม	หนังสือ	ไป	เล่ม	หนึ่ง
friend	borrowed	book	(aux.)	(classifier)	a
NP-subj.	V.	NP-obj			

In this paper, we introduce a different approach to Thai syntactic parsing based on a constraint dependency grammar. Our decision to use CDG parsing for handling Thai sentences rather than CFG parsing is based on several things. Since we are developing an automatic speech understanding system, vantage points from both speech and natural language processing aspects of the system are taken into consideration.

When choosing between the syntactic formalisms described in section 1.1, it appears that Thai syntax might be better described by a dependency than a phrase-structure representation in syntax. Also, from past work, CFGs have not produced satisfactory results for Thai grammar development.

Given the aforementioned properties of Thai sentences, a CDG parser appears to be an attractive choice for parsing Thai sentences. A CDG parser has many advantages over traditional CFG parsers. The set of languages accepted by a CDG grammar is a superset of the set of languages accepted by CFGs. Also, CDG can accept languages that CFGs accept as well as some they cannot, for example, $a^nb^nc^n$ and *ww* (where *w* is some string of terminal symbols). In addition, CDG is capable of efficiently analyzing free-order languages because order between constituents is not a requirement of the grammatical formalism. Since Thai exhibits significant word order variation, using CFG to describe Thai is cumbersome because numerous rules would be needed to cover all possible configurations of a constituent. Although GSPG, which has previously been used for parsing Thai, can easily and concisely describe free-order languages[1], how to combine the separate GSPG constraints together in parsing has proven to be a problem (Tanaka, 1993). Thus, CDG represents an alternative parsing formalism that is well worth considering.

In order to overcome the difficulties of analyzing Thai sentences, Wuwongse and Pornprasertsakul (1993) suggested that an information-intensive approach might be effective. In other words, a coordinating scheme must be devised to allow the parser to simultaneously or hierarchically utilize lexical, syntactic, semantic, and pragmatic information in resolving ambiguities. A CFG parser does not provide a good coordinating scheme because it is incapable of selectively invoking different knowledge sources. The CDG approach, on the other hand, provides a uniform mechanism of constraint propagation for each knowledge source during parsing. Furthermore, the constraints for each knowledge source can be developed independently, and it is not difficult to add an additional knowledge source if desired. Harper and Helzerman (1994) have successfully constructed and incorporated syntactic, semantic, and pragmatic constraints into their system. In fact, semantic constraints were added to the grammar without having to modify a single syntactic constraint. Moreover, the rules for each of the information sources can be independently applied to the constraint network of word nodes. If ambiguity remains, additional constraints can be used. This property allows decisions about structural ambiguities to be postponed until the constraints settle on a single structure, eliminating the need for backtracking.

From the speech processing point of view, CDG parsing is useful for building a spoken language understanding system, which must be able to tolerate repeated and aborted phrases as well as disfluencies. There is no notion of left-to-right parsing

[1]GPSG is more expressive than a CFG as well.

because it does not matter where in the sentence the word is (unless the constraint needs to relate the order of two words in the sentence). Also, incorporating prosodic information into a CFG requires a tight coupling of prosodic rules with syntactic rules. Not only does tight coupling increase the size and complexity of the grammar and reduce its understandability, it also makes syntactic rules and prosodic rules inseparable. On the other hand, preliminary work with prosodic constraints indicates that prosodic constraints should be easily added to a CDG grammar (Zoltowski, et al., 1992).

Another important consideration in natural language processing is the issue of parallel parsing to achieve real-time performance. A parallel parser using a collection of processors can achieve substantial speed-up over a traditional parser. Although CDG has a relatively slow serial running time, the parsing algorithm is much more parallelizable than CFG parsing (Helzerman, 1993). The best serial running time for a practical CFG parser operating with an ambiguous grammar is $O(G\star\, n^3)$ (Graham, et al., 1980), where n is the number of words in a sentence and G is the size of the CFG grammar. The serial running time for a CDG parser is $O(k\star\, n^4)$, where k is the number of constraints in the grammar and n is the number of word nodes (Harper and Helzerman, 1995). In practice, k is comparable in size to G for grammars with the same coverage, and $G >> n$. In contrast, a parallel CFG parser by Kosaraju's method (Kosaraju, 1975) can parse in $O(n)$ time using $O(n^2)$ processors. A parallelization for the single sentence CDG parser (Helzerman and Harper, 1992; Helzerman, 1993) can parse in $O(k)$ time using $O(n^4)$ processors.

Concerning the need to analyze multiple sentence hypotheses in Thai parsing, an extension to the CDG parsing by Harper and Helzerman (1995) allows efficient processing of multiple sentence hypotheses in the form of an augmented word graph. Usually, a speech recognizer unit provides a list of N-best sentence hypotheses. Likewise, a word segmentation algorithm for written Thai often generates multiple sentence hypotheses. Processing all of the sentence hypotheses individually can be inefficient since many hypotheses are quite similar. Instead of operating on single sentences, multiple sentence hypotheses can be efficiently processed using a data structure called a *Language Constraint Network* (LCN), which is a directed acyclic word graph augmented with parse-related information. The LCN is pruned by propagating various constraints including syntactic, lexical, semantic, prosodic, and pragmatic constraints. It provides a much better representation than a list of sentence hypotheses because it reduces redundancy and compactly represents the set of sentence hypotheses, thereby reducing the storage requirement. Clearly, this represents a more efficient and less redundant means of passing sentence hypotheses to the parser than a list of sentence hypotheses (Harper and Helzerman, 1994). Fig.1 depicts an example of a word graph, which can be converted to an LCN, constructed from a list of three sentence hypotheses.

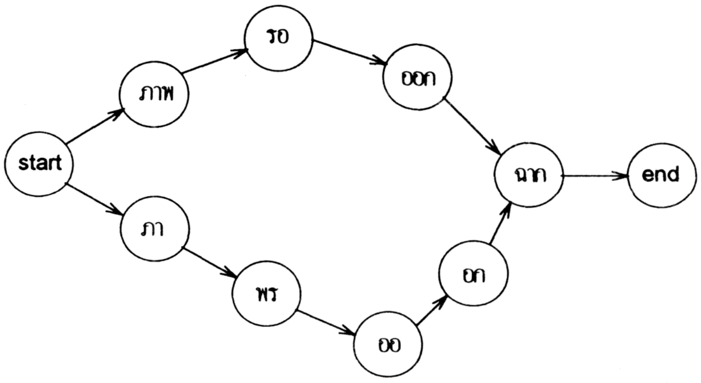

Fig. 1. A word graph constructed from a list of two sentence hypotheses generated from the Thai character string, ภาพรออกฉาก, by a word segmentation algorithm.

2. A Description of CDG Parsing

This section describes the basics of CDG parsing, originally defined by Maruyama (1990a, b) and later extended to handle lexical ambiguity, feature analysis, and multiple sentence hypotheses by Harper and Helzerman (1995). A step-by-step process of parsing a simple Thai sentence is also provided to illustrate constraint parsing following Maruyama's original approach. Interested readers are referred to Harper and Helzerman (1995) for a discussion on the aforementioned extensions to CDG parsing.

2.1 Elements of CDG

Maruyama defines a CDG as a 4-tuple, $G = \langle \Sigma, R, L, C \rangle$, where:

Σ = a finite set of terminal symbols or lexical categories,
L = a finite set of labels, $\{l_1, \ldots\ldots, l_q\}$,
R = a finite set of uniquely named roles (or role-ids), $\{r_1, \ldots\ldots, r_p\}$,
C = a constraint set of unary and binary constraints.

Within this grammar, a sentence $s = w_1w_2w_3w_4......w_n$ is defined as a word string of finite length n and is an element of Σ^*. The elements of Σ are the parts of speech of the words in a sentence. Associated with every word i in a sentence s are all of the p roles in R. Hence, every sentence contains $n \star p$ roles. A role can be thought of as a variable which is assigned a role value. A role value is a tuple $\langle l,m \rangle$, where l is a label from L and m is an element of the set of modifiees, $\{1,2,.....,n,\text{nil}\}$. A modifiee is a pointer to another word in the sentence (or to no word if nil). Role values will be denoted in the examples as *label-modifiee*.

Two types of roles (role-ids) per word are used; *governor* and *need* roles. The governor role indicates the function a word fills in a sentence when it is governed by its head word (e.g., a subject is governed by the main verb). Several need roles (i.e., need1 and need2) may be used to make certain that a head word has all of the constituents it needs to be complete. Each of the need roles keeps track of an item that its word needs in order to be complete (e.g., a verb generally needs a subject for the sentence to be complete).

The function that a word plays within the sentence is indicated by assigning a role value to a role. The label in the role value indicates the function the word fills when it is pointing at the word indexed by its modifiee (e.g., a *subj* label). When a role value is assigned to the governor role of a word, it indicates the function of that word when it modifies its head word. Likewise, when a role value is assigned to a need role of a word, it indicates how that need is being filled.

To parse a sentence using CDG, the constraints (members of C) must be specified. A constraint set, C, is a logical formula of the form: (and C_1 C_2 ...... C_t). Each C_i is a constraint represented in the form: (if *Antecedent Consequent*), where *Antecedent* and *Consequent* are either single predicates or a group of predicates joined by the logical connectives (a conjunction or disjunction of predicates). The possible components of each C_i are variables, constants, access functions, predicate symbols and logical connectives. They are described next.

Variables (i.e., x, y, etc.) used in the constraints stand for the role values. A constraint involving only one variable is called a unary constraint; two variables, a binary constraint. A maximum of two variables in a constraint allows for sufficient expressivity. The use of more than two would unnecessarily increase the running time of the parsing algorithm.

Constants are elements and subsets of $\Sigma \cup R \cup L \cup \{\text{nil}, 1,2,......,n\}$, where n corresponds to the number of words in a sentence.

The definition of the allowable access functions for constructing constraints are given below:

(pos x)	returns the position of the word for the role value x.
(rid x)	returns the role-id for the role value x.
(lab x)	returns the label for the role value x.
(mod x)	returns the position of the modifiee for the role value x.
(cat i)	returns the lexical category for the word in position i.

The predicate symbols allowable in constraints are:

(eq x y)	returns true if x = y, false otherwise.
(gt x y)	returns true if x > y, x,y ∈ I, false otherwise.
(lt x y)	returns true if x < y, x,y ∈ I, false otherwise.
(elt x y)	returns true if x ∈ y, false otherwise.

The logical connectives are:

(∧ p q)	returns true if p and q are true, false otherwise.
(∨ p q)	returns true if p or q is true, false otherwise.
(~ p)	returns true if p is false, false otherwise.

Using the above predicate symbols and access functions, unary and binary constraints for a CDG grammar can be constructed. Unary constraints are often used to restrict the role values allowed by a role given its part of speech. For example, the following unary constraint indicates that if a word is a verb, it must have the label *root* and be ungoverned:

```
(if (∧ (eq (cat (pos x)) verb)
       (eq (rid x) governor))
    (∧ (eq (lab x) root)
       (eq (mod x) nil))).
```

Binary constraints are constructed to describe how the role values assigned to two different roles are interrelated. For example, the following binary constraint is used to indicates that a subject must be governed by a root to its right:

```
(if (∧ (eq (lab x) subj)
       (eq (mod x) (pos y)))
    (∧ (eq (lab y) root)
       (lt (pos x) (pos y)))).
```

In CDG, a sentence *s* is said to be generated by the grammar if there exists an assignment **A** given a set of constraints *C*. An assignment **A** for the sentence *s* is a function that maps role values to each of the $n\star p$ roles such that the constraint set *C* is satisfied. There may be more than one assignment which satisfies *C*, in which case there is more than one parse for the sentence. L(G) is the language generated by a grammar G if and only if L(G) is the set of all sentences generated by G. Note that the null or empty string ϵ has no roles and is always generated by any grammar according to the definition.

A CDG grammar is characterized by two parameters: *degree* and *arity*. The degree is indicated by the size of R. The maximum number of variables used in constructing the constraints indicates the arity or the grammar. Maruyama has proven that a CDG grammar with a degree and arity of two is required to obtain the expressive power of a CFG (and exceed it).

To illustrate the use of CDG grammars, consider a simple example grammar used for parsing a Thai sentence, เสือนี้สวย, given in Fig. 2. The grammar has a degree of one and an arity of two. The constraints in this grammar were chosen for simplicity, not to exemplify constraints for a wide coverage grammar. Note that U-1, U-2, U-3 are unary constraints because they involve a single variable, and B-1 is a binary constraint because it contains two variables. In this paper, we adopt the lexical categorization of Thai words proposed by Panupong (1970).

For G' to generate the sentence, there must be an assignment of a role value to the governor role of each word, and that assignment must simultaneously satisfy each of the constraints in C'. Note that each word is assumed in this example to have a single lexical category, which is determined by dictionary look-up. Table 1 depicts an assignment for the sentence which satisfies C'.

```
Σ' = {det, noun, verb}
R' = {governor}
L' = {det, root, subj}
C' = ∀ x, y (∧
          ; ; [U-1]  A noun receives the label 'subj' and modifies a word to its right.
          (if (eq (cat (pos x)) noun)
              (∧ (eq (lab x) subj)
                  (lt (pos x) (mod x))))
          ; ; [U-2]  A determiner receives the label 'det' and modifies a word to its left.
          (if (eq (cat (pos x)) det)
              (∧ (eq (lab x) det)
                  (lt (mod x) (pos x))))
          ; ; [U-3]  A verb receives the label 'root' and modifies no word.
          (if (eq (cat (pos x)) verb)
               (∧ (eq (lab x) root)
                   (eq (mod x) nil)))
          ; ; [B-1]  A subj is governed by a verb.
          (if (∧ (eq (lab x) subj)
                  (eq (mod x) (pos y)))
               (eq (cat (pos y)) verb)))
```

Fig. 2. The grammar $G' = \langle \Sigma', R', L', C' \rangle$.

Table 1. An assignment for the sentence เสื้อนี้สวย.

Pos	Word	Cat	Governor role's value
1	เสื้อ	noun	subj-3
2	นี้	det	det-1
3	สวย	verb	root-nil

2.2. Parsing a sentence with CDG grammar.

To determine whether a sentence is generated by a grammar, the CDG parser must be able to assign at least one role value which satisfies the grammar constraints to each of the $n \star p$ roles, where n is the sentence length, and p is the number of role-ids. Because the role values for the role are selected from the finite set $L \times \{\text{nil}, 1, 2, \ldots, n\}$, CDG parsing can be viewed as a constraint satisfaction problem over a finite domain. Hence, constraint propagation can be used to develop the parse of a sentence. Enumeration of the individual parses for a highly ambiguous sentence is intractable. Therefore, a CDG parser generates all parses for a sentence in a compact form. The steps required for parsing a single sentence เสื้อนี้สวย are provided to illustrate both the process of parsing with constraint propagation and the running time of the algorithm.

To develop a syntactic analysis for a sentence using CDG, a *constraint network* (CN) of words is created. Each of the n words in a sentence is represented as a node in a CN. Fig. 3 illustrates the initial configuration of nodes in the CN for the example, เสื้อนี้สวย. Notice that associated with each node is its word, category, sentence position, and roles (only one role for this example). Each of the roles is initialized to the set of all possible role values (i.e., the domain). Given G', the domain for the example is $L' \times \{\text{nil}, 1, 2, 3\}$ = {det-nil, det-1, det-2, det-3, subj-nil, subj-1, subj-2, subj-3, root-nil, root-1, root-2, root-3}. Since there are $q \star (n+1) = O(n)$ possible role values for each of the $n \star p$ roles for a sentence (where p, the number of roles per word, and q, the number of different labels, are grammatical constants, and n is the number of words in the sentence), there are $n \star p \star q \star (n+1) = O(n^2)$ role values which must be initially generated for the CN, requiring $O(n^2)$ time. Note that each role value has a direct access to its role-id, label, modifiee, and the position of its word. A role value must access the word node to determine its lexical category (this was changed in Harper and Helzerman, 1995).

To parse the sentence using G', the unary and binary constraints in C' are applied to the CN to eliminate those role values from the roles of each word which are incompatible with C'. For a sentence to be grammatical, each role in each word node must contain at least one role value after constraint propagation.

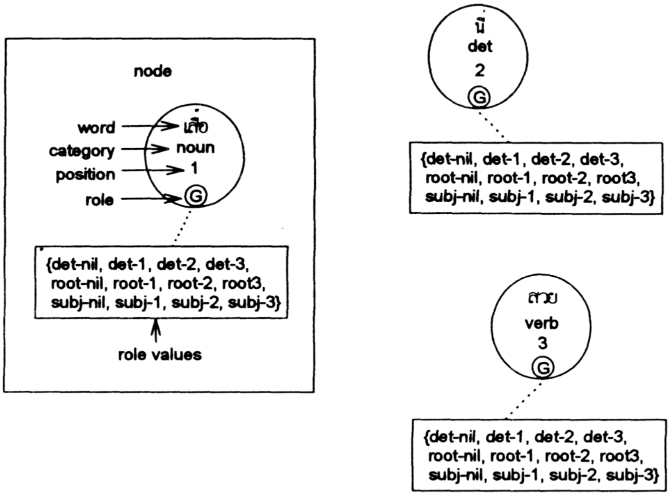
node
word
category
position
role
เสือ
noun
1
G
{det-nil, det-1, det-2, det-3,
root-nil, root-1, root-2, root3,
subj-nil, subj-1, subj-2, subj-3}
role values
นี้
det
2
G
{det-nil, det-1, det-2, det-3,
root-nil, root-1, root-2, root3,
subj-nil, subj-1, subj-2, subj-3}
สวย
verb
3
G
{det-nil, det-1, det-2, det-3,
root-nil, root-1, root-2, root3,
subj-nil, subj-1, subj-2, subj-3}

Fig. 3. Initialization of roles for the sentence เสือนี้สวย.

The unary constraints are applied to each of the roles in the sentence to eliminate the role values incompatible with each word's role in isolation. To apply the first unary constraint (i.e., U-1, shown below) to the network in Fig. 3, each role value for every role is examined to ensure that it obeys the constraint:

```
;; [U-1] A noun receives the label 'subj' and modifies a word to its right.
(if (eq (cat (pos x)) noun)
    (and (eq (lab x) subj)
         (lt (pos x) (mod x))))
```

If a role value causes the antecedent of the constraint to evaluate to TRUE and the consequent to evaluate FALSE, then the role value is eliminated. Fig. 4 shows the remaining role values after U-1 has been applied to the CN in Fig. 3. Since each constraint can only contain access functions and predicates that operate in constant time (i.e., like those defined in section 2.1), the propagation of the unary constraint U-1 to $O(n^2)$ role values requires $O(n^2)$ time.

To further eliminate role values which are incompatible with the word categories in the example, the remaining unary constraints (i.e., U-2 and U-3) are applied to the CN in Fig. 4, producing the network in Fig. 5. Given that the time to apply the unary constraints to a single role value is a grammatical constant denoted as k_u, the time required to apply the unary constraints in a grammar to all role values is $O(k_u \star n^2)$.

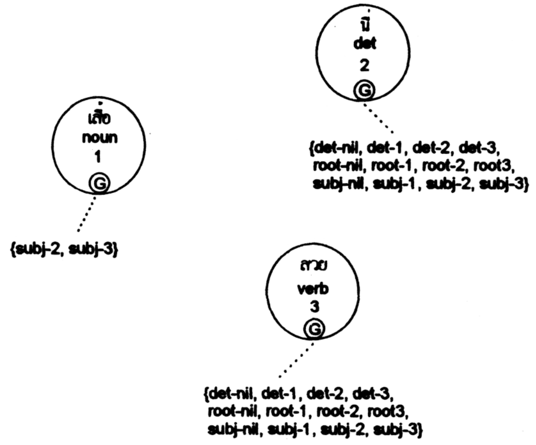
เสือ
noun
1
G
{subj-2, subj-3}
นี
det
2
G
{det-nil, det-1, det-2, det-3, root-nil, root-1, root-2, root3, subj-nil, subj-1, subj-2, subj-3}
สวย
verb
3
G
{det-nil, det-1, det-2, det-3, root-nil, root-1, root-2, root3, subj-nil, subj-1, subj-2, subj-3}

Fig. 4. The constraint network after the propagation of U-1 for the sentence เสือนีสวย.

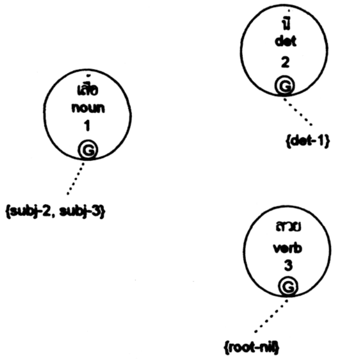

Fig. 5. The constraint network after the propagation of all unary constraints.

The binary constraints determine which pairs of role values can legally coexist. To keep track of pairs of role values, *arcs* connect each role to all other roles in the network, and each arc has an associated *arc matrix*, whose row and column indices are the roles value associated with the two roles. The elements of an arc matrix can either be a 1 (indicating that the two role values which index the element are compatible) or a 0 (indicating that the role values cannot simultaneously exist). Initially, all elements in each matrix are set to 1, indicating that the two role values are initially compatible. Since there are $O(n^2)$ arcs required in the CN, and each arc contains a matrix with $(q\star (n+1))^2 = O(n^2)$ elements, the time to construct the arcs and initialize the matrices is $O(n^4)$. Fig. 6 shows the matrices associated with the arcs before any binary constraints are propagated. Unary constraints are usually propagated before preparing the CN for binary constraints because they eliminate impossible role values, and hence reduce the size of the arc matrices.

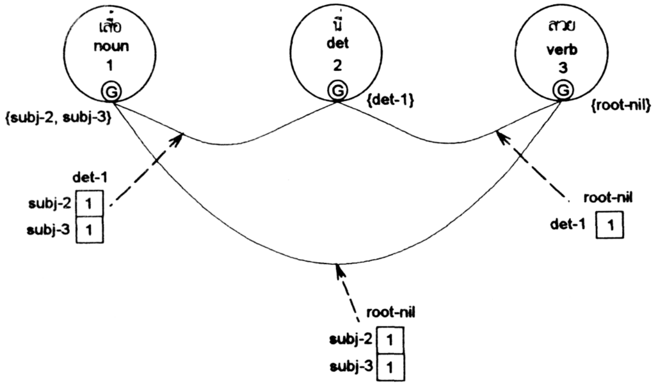

Fig. 6. The constraint network after unary constraint propagation and before binary constraint propagation.

Binary constraints are applied to the pairs of roles values indexing each of the arc matrix elements. When a binary constraint is violated by a pair of role values, the entry in the matrix indexed by those role values is set to zero. The binary constraint, B-1, ensures that a subj is governed by a verb.

```
;; [B-1] A subj is governed by a verb.
(if (∧ (eq (lab x) subj)
       (eq (mod x) (pos y)))
   (eq (cat (pos y)) verb)))
```

After the application of this constraint to the network in Fig. 6, the element indexed by the role values x = subj-2 and y = det-1 for the matrix on the arc connecting the governor roles for เสือ and นี้ is set to zero, as shown in Fig. 7. This is because เสือ must be governed by a verb, not a det. Since the constraint must be applied to $O(n^4)$ pairs of role values, the time to apply the constraint is $O(n^2)$. Given that the time to apply the binary constraints to a pair of role values is a grammatical constant denoted as k_b, the time required to apply binary constraints in a grammar to all pairs of role values is $O(k_b \star n^4)$.

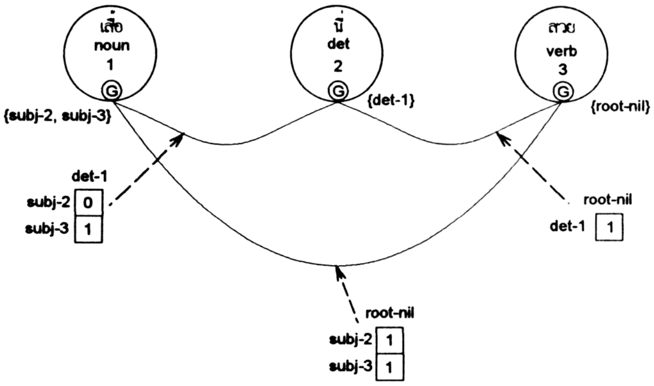

Fig. 7. The constraint network after B-1 is propagated.

Following the propagation of binary constraints, the roles of the CN could still contain role values which are incompatible with the parse for the sentence. To determine whether a role value is still supported for a role, each of the matrices on the arcs incident to the role must be checked to ensure that the row (or column) indexed by the role value contains at least a single 1. If any arc matrix contains a row (or column) of 0s, then the corresponding role value cannot coexist with any of the role values for the second role and so is removed from the list of legal role values for the first role. Additionally, the rows (or columns) associated with the eliminated role value can be removed from the arc matrices attached to the role. The process of removing any rows or columns containing all zeros from arc matrices and eliminating the associated role values from their roles is called *filtering*. Filtering a constraint network is known as arc consistency to constraint satisfaction researchers. Following binary constraint propagation, any of the $O(n^2)$ role values may be filtered immediately. However, filtering must also be applied iteratively since the elimination of one role value could lead to the elimination of another role value.

Consider how filtering is applied to the CN in Fig. 7. The matrix associated with the arc connecting the governor roles of เสือ and นี้ contains a row with a single element which is zero. Because subj-2 cannot coexist with the only possible role value for the governor role of นี้, it cannot be a legal member of the governor role of เสือ, and so subj-2 is eliminated as a role value for node 1's governor role. When the role value is eliminated from all arcs associated with the role, filtering is complete. The resulting CN is depicted in Fig. 8.

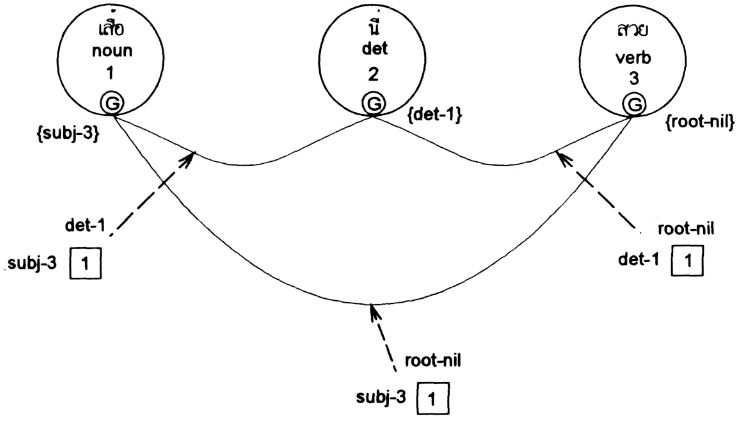

Fig. 8. The constraint network after filtering.

After all the constraints are propagated across the CN and filtering is completed, the CN provides a compact representation for all possible parses. Syntactic ambiguity is easy to spot in the CN since some of the roles in an ambiguous sentence contain more than a single role value. If multiple parses exist, we can propagate additional constraints to further refine the analysis of the ambiguous sentence. The parse trees in a CN, which are called *parse graphs*, consist of a compatible set of role values for each of the roles in the CN. The modifiees of the role values, which point to the words they modify, form the edges of the parse graph. Our example sentence has an unambiguous parse given G_1, shown in Fig. 9.

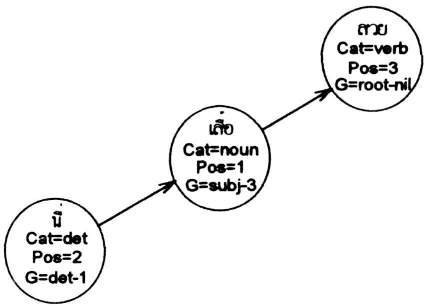

Fig. 9. The parse graph for the constraint network in Fig. 8.

3. Conclusion

The basic framework of a constraint-based parser for parsing written and spoken Thai sentences has been described. The parser uses constraint dependency grammar, originally defined by Maruyama. Our framework for Thai is based on an extension to CDG proposed by Harper. Details of the system and its advantages for parsing written and spoken Thai have also been discussed. A summary of the steps in the CDG parsing algorithm and their associated running times is presented below:

1. Constraint network construction prior to unary constraint propagation: $O(n^2)$.
2. Unary constraint propagation: $O(k_u \star n^2)$.
3. Constraint network construction prior to binary constraint propagation: $O(n^4)$.
4. Unary constraint propagation: $O(k_b \star n^4)$.
5. Filtering (arc consistency): $O(n^4)$.

Notice that the time required to propagate binary constraints is the slowest part of the algorithm.

Due to the scope of the paper, we cannot describe in detail the enhancements made by Harper to CDG parsing algorithm to increase its usefulness for both text-based and spoken natural language processing. However, we will briefly mention them, as they are useful in our implementation of the parser for Thai as well. First, the algorithm has been modified to parse sentences with lexically ambiguous words by allowing role values within the same node to have their own parts of speech. Another enhancement added to the CDG parser is a lexical feature analysis. For

English and in other languages, lexical features are used to enforce subject-verb agreement, determiner-head noun agreement, and case requirements for pronouns. This information can be very useful for disambiguating parses for sentences or for eliminating impossible sentence hypotheses. With respect to the efficiency issue of the algorithm, a fifth parameter, *T*, where *T* is a *table* which restricts the possible labels for each role according to the category of the word and its role id, has been added to the CDG grammar tuple. *T* makes the analysis of a sentence more efficient because the roles are initialized to smaller domains, and many of the unary constraints (i.e., those which restrict labels or role values to lexically appropriate values) can be omitted. Finally, the filtering algorithm of the CDG parser was modified to support the parsing of LCNs. These extensions have been useful for processing spoken English and are currently being used to investigate the impact of prosodic constraints on Thai spoken sentences.

4. References

Aroonmanakul, W. 1990 A Dependency analysis of Thai sentences for a computerized parsing system. Unpublished Master's thesis, Chulalongkorn University.

Bloomfield, L. 1933 Language. New York: Holt, Rinehart and Winston.

Chomsky, N. and Schutzenberger, M. P. 1963 The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg (Eds.) Computer Programming and Formal Systems, Studies in Logic Series. Amsterdam: North-Holland: 119-161.

Graham, S. L., Harrison, M. A., and Russo, W. L. 1980 An improved context-free recognizer. ACM Transactions on Programming Languages and Systems. 2: 415-462.

Harper, M. P. and Helzerman, R. A. 1994 Managing multiple knowledge sources in constraint-based parsing of spoken language. Fundamental Informaticae. 23 (2-4): 303-353.

Harper, M. P. and Helzerman, R. A. 1995 Extensions to constraint dependency parsing for spoken language processing. Computer Speech and Language. 9:187-234.

Helzerman, R. A. and Harper, M. P. 1992 Log time parsing on the MasPar MP-1. In Proceedings of the Sixth International Conference on Parallel Processing. 2: 209-217.

Helzerman, R. A. 1993 PARSEC: A Framework for Parallel Natural Language Understanding. Unpublished Master's thesis, Purdue University.

Kosaraju, S. R. 1975 Speed of recognition of context-free languages by array automata. SIAM Journal of Computing. 4(3):331-340.

Luksaneeyanawin, S. 1993 Speech computing and speech technology in Thailand. Proceedings of the Symposium on Natural Language Processing in Thailand. Chulalongkorn University.

McFarland, G. B. 1960 Thai-English Dictionary. Stanford: Stanford University Press.

Maruyama, H. 1990a Constraint dependency grammar. Tech. Rep. RT0044. IBM, Japan.

Maruyama, H. 1990b Constraint dependency grammar and its weak generative capacity. Computer Software.

Mel' čuk, I. A. 1988 Dependency Syntax: Theory and Practice. Albany: State University of New York Press.

Panupong, V. 1970 Inter-sentence Relations in Modern Conversational Thai. Bangkok: The Siam Society Press.

Pornprasertsakul, A., Wuwongse, V., and Chansaenwilai, K. 1990 A Thai generalized phrase structure grammar and its parser. Proc. Int. Conf. Computer Processing of Chinese and Oriental Languages. Hunan.

Sornlertlamvanich, V. and Phantachat, W. 1992 Information-based language analysis for Thai. Proc. 3rd Int. Symposium Language and Linguistics. Bangkok: 497-511.

Tesnière, L. 1959 Éléments de syntaxe structurale. Paris: Klincksieck (2nd edition, revised and corrected, 1969).

Tanaka, H. 1993 Current trends on parsing. Proceedings of the Symposium on Natural Language Processing in Thailand. Chulalongkorn University.

Vorasucha, V. 1986 Thai syntax analysis based on GSPG. Regional Symposium on Computer Science and Its Application (with Emphasis on Artificial Intelligence), KMITL.

Wuwongse, V. and Pornprasertsakul, A. 1993 Thai syntax parsing. Proceedings of the Symposium on Natural Language Processing in Thailand. Chulalongkorn University.

Zoltowski, C. B., Harper, M. P., Jamieson, L. H., and Helzerman, R. A. 1992 PARSEC: A constraint-based framework for spoken language understanding. In International Conference on Spoken Language Processing.