A Computational Syntax and Its Application to Parsing

Hsin-I Hsieh

Department of East Asian Languages & Literatures
University of Hawaii
Honolulu, Hawaii, U.S.A. 96822
e-mail: hhsieh@uhunix.uhcc.hawaii.edu

Abstract

We propose a theory of syntax in which grammatical sentences are generated by binary composition from lexical frames and syntactic types are computed from lexical types. This theory of computational syntax has an immediate application to the parsing of grammatical sentences. We discuss in detail our theory and briefly demonstrate its applicability to parsing, using Mandarin sentences as examples.

0. Introduction

We propose a theory of syntax in which grammatical sentences are generated by repeatedly composing two elements, each of which belongs to a specific type. At every stage of the binary composition, the type of an element is either specified in the lexicon or is computed by rules based on the types of its two co-composing elements. Thus the syntax is not merely compositional but computational. Grammaticality is precisely characterized: a sentence is grammatical if it belongs to a specific type and also fulfills additional grammaticality conditions.

Our theory of syntax can be applied to the parsing of sentences, especially grammatical sentences. The linguistic part of a parsing system using our syntactic theory can be relatively uncomplicated owing to the internal computation of syntax.

The bulk of this paper is an explication of this computational syntax. We only briefly discuss the applicability of our theory to parsing without actually proposing a full parsing system.

1. Compositional Cognitive Grammar

Our computational syntax is part of a more comprehensive theory of grammar called Compositional Cognitive Grammar (CCG) (Hsieh 1992b). This grammar 'starts' with a component called Imagery Structure (IS) whose elements are called Imagery Structure representations (ISrr). IS maps onto Semantic Structure (SS) whose elements are called Semantic Structure representations (SSrr). SS maps onto Thematic Structure (TS) which contains as its elements Thematic Structure representations (TSrr). TS maps onto Functional Structure (FS) which contains as its members Functional Structure representations (FSrr). FS maps onto Constituent Structure (CS) whose members are Constituent Structure representations (CSrr). Thus the chain of interconnecting mappings is: IS --> SS --> TS --> FS --> CS. At the present stage of our research we are only able to fully articulate the SS and CS components. The TS and FS are lacking but would be similar in purpose to those proposed in LFG (Bresnan 1982). The IS is necessary if the meaning of a sentence is ultimately rooted in its related image, as the cognitive grammarians (Langacker 1987,

Talmy 1985, Jackendoff 1990, Tai 1985, 1989) have suggested or implied. SS is the component where syntax and semantics enter into a systematic interface or interaction (Hsieh 1992a, Chang 1991, Her 1991, M. Hsieh 1992). CS is similar to standard phrase structure. Lacking TS and FS, we map SS onto CS directly. We generate an SSr by repeatedly composing two elements, each one of which is either (i) drawn from a finite set of basic elements listed in the lexicon, or (ii) is a 'persistent' binary composition whose ultimate composing elements are all basic elements. In this sense an SSr is compositional. Each element, basic or non-basic, has a specific pattern or type. If an element is basic, its type is determined in the lexicon, and if non-basic its type is determined by a set of rules of computation applied to its two co-composing elements. In this sense SSr is computational. Hence, as a theory of syntax, SS is both compositional and computational.

1.1. The Lexicon

A sentence or clause in conventional grammar is represented as an Action (AC) in its SSr. An AC may be either a simple Action (sim-AC) or a complex Action (com-AC). Simple Actions compose in a binary way into complex Actions of increasing complexity.

Those simple Actions that are 'legitimate' or well-formed are called Action Frames (ACFs). Each ACF is composed of an Initiator (I) and a complex Act (A'), which in turn is composed of an Act (A) and a Receiver (R). The A is represented by an Abstract Verb (AV) and the I and R may be represented by an indexed variable v_k or a simple constant c or a complex constant f(c), or by an empty symbol 0, which may be indexed by k to express a syntactically empty but pragmatically inferrable NP_k. We say that an ACF 'accepts' a particular AV and an AV 'selects' a particular ACF. An ACF with a particular AV is a particular ACF, or, PACF. Thus an ACF is a type and its associated PACFs are its tokens. For convenience, we sometimes disregard this distinction in our discussion unless it is necessary to maintain it.

The lexicon contains two types of entries: logical-type entries and grammatical-type entries. Logical(-type) entries are divided into three sub-types: (i) indexed variables v_k's; (ii) simple constants c's; (iii) complex constants f(c)'s; (iv) empty symbols without indices, 0's, or empty symbols with indices, 0_k's. An indexed variable v_k is a place-holder for a co-indexed NP_k, which, when suitably attached to the SSr tree, will 'instantiate' v_k by replacing all copies of it. The instantiation operation is based primarily on the concept of instantiation proposed by McCawley (1971) and the notion of controlled empty categories suggested by Huang (1992). A simple constant c indicates the actual site of embedding for an exterior AC indexed by c. A complex constant f(c) refers to a special semantic feature of the AC indexed by c, such as 'aspect', 'tense', 'degree', 'manner', etc. A simple constant c and a complex constant f(c) differ in meaning but behave in the same way syntactically. For convenience, we sometimes write 'c' for both c and f(c) whenever it is convenient to do so. A constant c or f(c) in an ACF is 'saturated' if there is an exterior AC indexed by c which composes with this ACF to supply the content of c. Otherwise, the constant is 'unsaturated'.

Grammatical entries in the lexicon are either abstract verbs (AVs) or nouns (Ns). AVs are divided into three subtypes: (i) full verb (FV); (ii) half verb (HV); and (iii)

grammatical verb (GV). A word (more precisely a morpheme) may function as one or more of these three subtypes. Each AV has a concrete shape in CSr. Roughly, a full verb corresponds to a verb or an adjective; a half verb results in a preposition, conjunction, adverb, auxiliary, tense (marker), aspect (marker), negation (word); and a grammatical verb ends up as a demonstrative, determiner, or a grammatical particle such as the infinitive particle to or gerund affix -ing in English. Exactly which one of these various concrete forms will an AV produce is determined generally by the ACF which the AV selects and occasionally by considering both this ACF and the crucial ACF in its co-composing AC.

A noun may serve as an NP_k to immediately instantiate a co-indexed variable v_k, or may combine with an NP-generating ACF to become the 'core' of an NP_k that instantiates v_k.

ACFs may be classified in two complementary ways: by considering what kind of A (Act) they contain and what kind of I (Initiator) and R (Receiver) they possess. The former consideration yields three v-types and the latter three n-types. The three v-types are: (i) Full Verb AC (FAC); (ii) Half Verb AC (HAC); (iii) Grammatical Verb AC (GAC). An FAC has a full verb for its Act, an HAC a half verb, and a GAC a grammatical verb. The three n-types are: (i) Solitary AC (SAC); (ii) Receptive AC (RAC); and (iii) Warm AC (WAC). An SAC has no unsaturated constants; an RAC contains only one unsaturated constant; and a WAC possesses two unsaturated constants.

As shown in Table I, there are a total of 27 (twenty-seven) ACFs which fall into 9 (nine) compound types. Within each type, the ACFs are further distinguished

Table I. Action Frames Classified.

	<I, <A, R>>				<I, <A, R>>				<I, <A, R>>			
	(1)	x_i	FV	y_j	(10)	x_i	HV	y_j	(19)	x_i	GV	y_j
	(2)	x_i	FV	0	(11)	x_i	HV	0	(20)	x_i	GV	0
SAC	(3)	0	FV	y_j	(12)	0	HV	y_j	(21)	0	GV	y_j
	(4)	0	FV	0	(13)	0	HV	0	(22)	0	GV	0
	(5)	h	FV	y_j	(14)	h	HV	y_j	(23)	h	GV	y_j
	(6)	h	FV	0	(15)	h	HV	0	(24)	h	GV	0
RAC	(7)	x_i	FV	k	(16)	x_i	HV	k	(25)	x_i	GV	k
	(8)	0	FV	k	(17)	0	HV	k	(26)	0	GV	k
WAC	(9)	h	FV	k	(18)	h	HV	k	(27)	h	GV	k
		FAC				HAC				GAC		

FV = Full Verb (verb, adjective).
HV = Half Verb (preposition, conjunction, adverb, auxiliary, aspect, tense, negation).
GV = Grammatical Verb (demonstrative, determiner and grammatical particles).

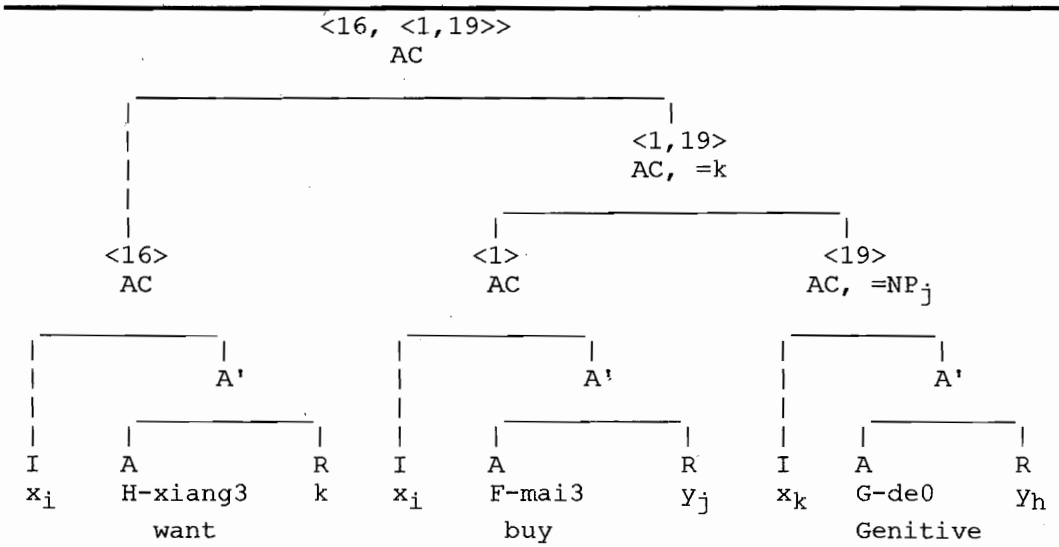
SAC = Solitary AC, containing no unsaturated constants.
RAC = Receptive AC, containing one unsaturated constant.
WAC = Warm AC, containing two unsaturated constants.

k = k or f(k); h = h or f(h).

according to the kind of logical symbols that represent their I and R, namely, v_k, 0, c. For convenience, a v_k is written x_k if it represents I and y_k if it represents R. Also for convenience, a constant c is written h if it represents I and k if it represents R.

A lexical entry is then just a variable v_k, a constant c or f(c), an empty symbol 0, an N, or an AV. This treatment is a technical realization of an insight originated with James H-Y. Tai, who has on many occasions told me that we need to recognize only nouns and verbs in the 'deep' structure of a Chinese sentence. If an entry is an AV, it will select one or more of the 27 ACFs. Each selected ACF, sometimes together with a co-occurring ACF, will determine the concrete form of this AV in that ACF. Each AV in an ACF thus has an abstract category as full, half, or grammatical verb, and a matching concrete category as verb, preposition, demonstrative, etc.

Figure 1 illustrates the composition of ACFs into a com-AC. There we see that ACF <1> and ACF <19> combine to form a com-AC denoted as AC <1,19>, which in turn combines with ACF <16> to form a com-AC denoted as AC <16, <1,19>>. AC <19> is an NP-generating ACF and here it has the effect of creating the NP_j, Li3-si4 de0 che1, which instantiates the y_j in ACF <1>. AC <1,19>, marked as equal to the constant k by the sign '=k', saturates the constant k, which lexicalizes the R in ACF <16>. Within each ACF, the AV is prefixed by F-, H-, or G- to indicate its status in terms of being full, half, or grammatical. In the CSr the variables x_i, y_j, x_k and y_h will be properly instantiated by co-indexed NPs, the AVs given their concrete forms, and the tree will be compressed into a conventional phrase structure by a host of transformations. The result of these operations will yield the CSr for sentence (1).



(1) Zhang1-san1 xiang3 mai3 Li3-si4 de0 che1.
'Zhang-san wants to buy Li-si's car.'

(Note: The prefix F-, H-, or G- to an AV indicates the full, half, or grammatical status of the AV.)

Figure 1. (partial) SSr for (1).

AC <16, <1,19>> illustrates the notion that in the SSr every com-AC is ultimately composed of a finite number of ACFs. In other words, every sentence is compositionally derived and not 'generatively' derived via a set of rewriting rules whose initial symbol is S, as is practiced in most current theories including GB.

1.2. Computation

Although we are free to compose ACFs of various kinds to form a complex AC, not all ACs so formed will have a CSr that yields a grammatical sentence. In other words, only some ACs are well-formed. There are syntactic, semantic, and pragmatic types of well-formedness conditions for an AC. An AC satisfying all these types of well-formedness conditions would be a 'maximally well- formed' AC. Such maximally well-formed ACs are beyond the scope of our discussion, since our focus here is on syntax. Nevertheless, we can identify a syntactic well-formedness condition, based on compound types, which a 'minimally well-formed AC' must meet. A grammatical sentence in a language that requires a compulsory marking for both aspect and tense can be analyzed into three parts: a 'core' content, an aspect, and a tense. To translate this arrangement into a composition in terms of ACs, we can first combine the core content with the aspect, and then combine the result with the tense. Suppose that the core is of the type FAC-SAC, and the aspect is of the type HAC-RAC. Then the composition of the core and the aspect could have the type FAC-SAC. Now suppose that the tense is also of the type HAC-RAC, then composing the core-aspect combination with tense would yield also the type FAC-SAC, to which all grammatical sentences could belong. Thus we have a clear idea of what the syntactic well-formedness condition for a minimally well-formed AC would be. It would be this: every minimally well- formed AC must have the type structure stated in (wf): <<FAC-SAC (core), HAC- RAC (aspect)>, HAC-RAC (tense)>. For example, sentence (2) John has seen Mary will have the SSr in (2'): <<<x_i, <F-see, y_j>> =k, <aspect (k), <H-perfect, 0>>> =h, <tense (h), <H-present, 0>>>. The 'compound-type structure' of (2') is (2'(a)) <<FAC-SAC, HAC-RAC>, HAC-RAC> and its 'compositional structure' in terms of ACFs, or, 'ACFs structure' is (2'(b)) <<1,15>, 15>. (2'(a)) as a compound-type structure can be computed to yield the type FAC-SAC, and for this reason it is the computational part of (2'). (2'(b)) as an ACFs structure is a composition not subject to any computation and so it is the compositional part of (2'). We combine (2'(a)) and (2'(b)) into (2'(ab)) <<FAC-SAC (1), HAC-RAC (15)>, HAC-RAC (15)>, and we obtain the 'compositional- computational complex' (ccc) of (2') and eventually of (2).

Clearly, we want to set up our computational scheme in such a way as to ensure that every grammatical sentence will end up having the (wf) as its compound-type structure. To achieve this, we make sure that when FAC-SAC is combined with a regular compound-type, it will yield FAC-SAC, but when combined with an exceptional type, it will yield the same exceptional type. In other words, we need two guiding principles: (i) FAC-SAC combined with x will yield FAC-SAC, and (ii) FAC-SAC combined with y will yield y. Apart from ensuring that every grammatical sentence has the (wf) as its type structure, we also want to prohibit incompatible ACFs from entering into composition. Specifically, (iii) we need to rule out the composition of two RACs or two WACs or one RAC with one WAC. By considering all these three needs, we decide that the computation of v-types should follow rules as

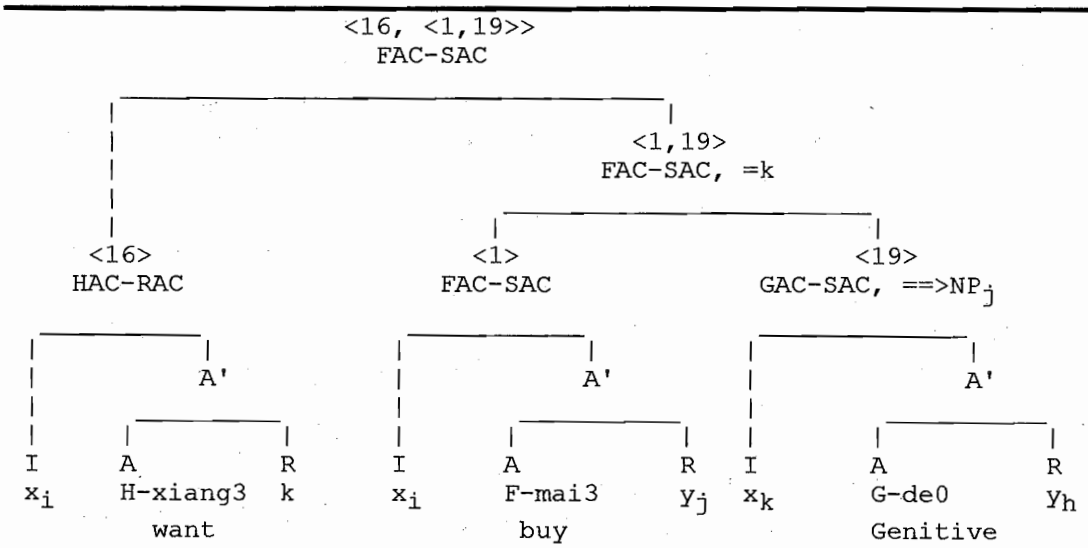
Table II. Computation of v-types.

	FAC	HAC	GAC
FAC	(i) FAC	(ii) FAC	(iii) (a) FAC, if GAC is an SAC (b) GAC, if GAC is an RAC or WAC
HAC		(iv) HAC	(v) (a) HAC, if GAC is an SAC (b) GAC, if GAC is an RAC or WAC
GAC			(vi) GAC

Table III. Computation of n-types.

	SAC	RAC	WAC
SAC	(i) SAC	(ii) SAC	(iii) RAC
RAC		(iv) Goof	(v) Goof
WAC			(vi) Goof

Goof: A complex AC which is not allowed to combine with any AC of the type SAC, RAC, WAC.



(1) Zhang1-san1 xiang3 mai3 Li3-si4 de0 che1.
'Zhang-san wants/intends to buy Li-si's car.'

Figure 2. Computation of Compound Types for (1).

stipulated in Table II and the computation of n-types should obey rules as prescribed in Table III. These two tables are self-explanatory and require no comments except that the computation is commutative as is obvious from the fact that only the upper- right halves of the tables are shown. The two tables work jointly to assign to a composition of any degree of complexity its combination of v-type and n- type, that is, its compound type.

In Figure 2 we illustrate this operation by showing how to derive the compound type for the SSr of sentence (1). Here we see that FAC in (ACF) <1> combines with GAC in <19> to become FAC in <1,19>, following rule (iii)(a) (since the GAC is also an SAC) in Table II. On the other hand, SAC in <1> combines with SAC in <19> to become SAC in <1,19>, following rule (i) in Table III. Combining these two results, we obtain the compound type FAC-SAC for <1,19>. Subsequently, the HAC of <16> combines with the FAC of <1,19> to form FAC, according to rule (ii) in Table II. Simultaneously, RAC of <16> combines with SAC of <1,19> to form SAC, according to rule (ii) in Table III. Putting these two results together, we obtain FAC-SAC for <16, <1,19>>.

1.3. Word Order

The SSr for sentence (1) shown in Figure 2 is explicit about the hierarchical order of the elements (of various complexity) in composition but it still needs information that would determine the linear order of these elements. Linear orders are determined by the 'primacy relation' which holds between a 'primary' ('p') ranked element and a 'secondary' ('s') ranked element in a composition. Each element alone has a 'p' rank, but when two elements are composed, one of them retains its 'p' rank and the other is demoted to an 's' rank. Within an ACF the primacy relation is predetermined and has the configuration AC(p) = <I(s), <A(p), R(s)> = A'(p)>. Across two ACs, the 'p' and 's' ranks are assigned by comparing their degrees of primacy. If one has a higher degree than the other, it is ranked 'p' and the other is ranked 's'. If both have the same degrees of primacy, then semantic and other non-syntactic criteria apply to determine their 'p' and 's' ranks.

The generally distinct degrees of primacy for a pair of co-composing elements are obtained by combining the result of computation based on the v- types with that based on the n-types. The laws of computation for the v-types are set forth in Table IV and those for the n-types are stated in Table V. These two tables are self-explanatory. Two compound types under comparison may be represented as the ordered pair $<v_1n_1, v_2n_2>$ (e.g. <FAC-SAC, GAC-SAC>). The v-type computation will yield an ordered pair $<d(v_1), d(v_2)>$, with $d(v_1)$ and $d(v_2)$ representing the degrees of primacy for v_1 and v_2, respectively. Similarly, the n-type computation will produce an ordered pair $<d(n_1), d(n_2)>$, with $d(n_1)$ and $d(n_2)$ denoting the degrees of primacy for n_1 and n_2, respectively. We then compute the sum of the two ordered pairs by adding up their coordinates and we obtain the sum $<d(v_1) + d(n_1), d(v_2) + d(n_2)>$, which is simply the ordered pair <m,n>, with m and n being positive integers. This pair <m,n> indicates the degrees of primacy for $<v_1n_1, v_2n_2>$, the two compound types under comparison. In other words, $<m,n> = <d(v_1n_1), d(v_2n_2)>$, where $d(v_1n_1)$ is the primacy degree of type v_1n_1, and $d(v_2n_2)$ is the primacy degree of type v_2n_2. If $m > n$, then m is

interpreted as 'p' ('primary') and n as 's' ('secondary'). If m < n, then m is interpreted as 's' ('secondary') and n as 'p' ('primary'). If m = n, then other criteria must apply to interpret one of m and n as 'p' and the other as 's'. Figure 3 illustrates this method of primacy ranking using sentence (1) as example. Here we see that elements within each ACF are assigned their 'p' and 's' through predetermination. Across ACF <1> and ACF <19>, a computation shows that the pair <m,n> representing the primacy degrees of ACF <1> and ACF <19> has the actual value of <2 + 1 = 3, 0 + 1 = 1>, with 3 > 1, hence ACF <1> is ranked 'p' and ACF <19> is ranked 's'. Across ACF <16> and AC <1,19>, the value for <m,n> is <0 + 1 = 1, 2 + 0 = 2>, with 1 < 2, hence ACF <16> is 's' and AC <1,19> is 'p'. Finally, <16, <1,19>> is assigned 'p' since it stands alone. Were it to enter into a composition, its primacy rank may be adjusted.

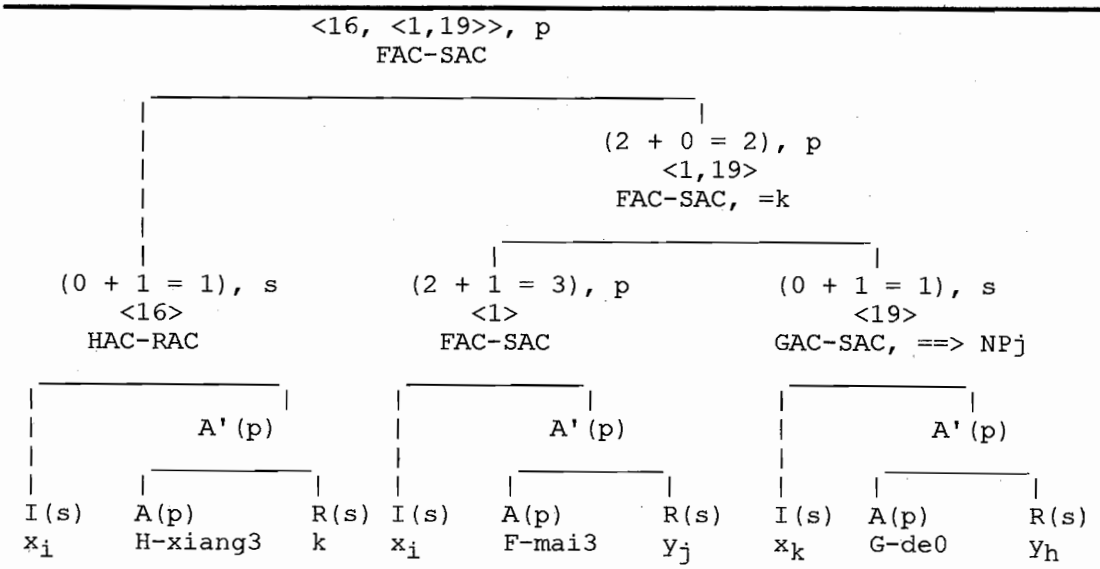
Once we have obtained the 'p' and 's' ranks for the pair of composing elements in each composition, we can interpret the 'p'-to-'s' relation in a particular language as

Table IV. Computing the Primacy Degrees in v-types.

	FAC	HAC	GAC
FAC	(i) <2,2>	(ii) <2,0>	(iii) <2,0>
HAC	(iv) <0,2>	(v) <2,2>	(vi) <2,0>
GAC	(vii) <0,2>	(viii) <0,2>	(ix) <2,2>

Table V. Computing the Primacy Degrees in n-types.

	SAC	RAC	WAC
SAC	(i) <1,1>	(ii) <0,1>	(iii) <0,1>
RAC	(iv) <1,0>	(v) none	(vi) none
WAC	(vii) <1,0>	(viii) none	(ix) none



(1) Zhang1-san1 xiang3 mai3 Li3-si4 de0 che1
'Zhang-san wants to buy Li-si's car.'

Figure 3. Computation of Primacy Degrees for (1).

either a 'p'-preceding-'s' or an 's'-preceding-'p' relation in word order. Although the picture is somewhat complicated, in Mandarin Chinese the general rule of word order is an 's'-preceding-'p' order, as Huang (1982, 1993), Li (1985, 1990), and Tai (1973) have shown, using a head versus modifier distinction, which is roughly parallel to our 'p' versus 's' distinction. Notable exceptions include VO order and prepositions.

The SSr for (1) as given in Figure 3 contains all the specifications needed for an SSr and as such it is a 'complete' SSr, ready to turn into a CSr through a series of transformations involving movement, concretization, instantiation, pruning, and rebuilding. We now describe these transformations using as example sentence (3) whose SSr is shown in Figure (4a).

2. Application to Parsing

By making use of the SSr, we can parse a grammatical sentence with relative ease. For each semantic interpretation of a surface grammatical sentence, there is exactly one SSr. This SSr is determined by its ultimate composing ACFs, or, more precisely PACFs. Once we successfully identify these ACFs and their compositional structures, the generation of the final SSr is automatic, since the compound-type formation rules and the primacy-rank determination rules will apply in a computational manner. After the grammatical sentence under parsing is successfully divided into a number of k segments, each containing (the concrete form of) an AV, our task is to search for the ACF that accepts each such AV, and to determine how these ACFs are composed together. Although each AV can select from a small range of ACFs, we can stipulate a set of ACF acceptance rules (ACFARs) which will assist us to decide on a unique correct ACF. We need also a set of ACFs composition rules (ACFCRs) that determine the composition of ACFs based on their compound types and perhaps also on the AVs they accept. The central linguistic problem in parsing with SSrr then is reduced to the formulation of the ACFARs and ACFCRs. These rules are specific to the language under parsing and are finite in number and so in principle can be exhaustively discovered. Thus the potential application of CCG, with its SSrr, to the parsing of grammatical sentences is in principle an uncomplicated and accomplishable job.

References

Bresnan, Joan. (ed.) 1982. The mental representation of grammatical relations. Cambridge, MA: MIT Press.

Chang, Claire Hsun-huei. 1990. Complex verb and argument structure: interaction between syntax and morphology. Paper presented at the Second Northeast Conference on Chinese Linguistics, The University of Pennsylvania, Philadelphia, May 4-6, 1990.

_____. 1991. Verb copying: Towards a balance between formalism and functionalism. Journal of Chinese Language Teachers Association 26.1:1- 32.

Her, One-soon. 1991. Topic as a grammatical function in Chinese. Lingua 84.1-23.

Hsieh, Hsin-I. 1992a. In search of a grammatical foundation for dialect subgrouping. Symposium series of the Institute of History and Philology, Academia Sinica, no. 2: Chinese languages and linguistics, vol. 1: Chinese dialects, 333-377. Taipei: Academia Sinica.

_____. 1992b. Lexicon and morphology in a compositional cognitive grammar. Proceedings of IsCLL-3, 38-61.

Hsieh, Miao-ling. 1992. Analogy as a type of interaction. Journal of Chinese Language Teachers Association 28.3:75-92.

Huang, C.-T. James. 1982. Logical relations in Chinese and the theory of grammar. Doctoral dissertation, MIT.

_____. 1992. Complex predicates in control. Control and grammar, ed. by R. K. Larson, S. Iatridou, U. Lahiri and J. Higginbotham, 109-147. Dodrecht: Kluwer Academic Publishers.

_____. 1993. More on Chinese word order and parametric theory. MS.

Jackendoff, Ray. 1990. Semantic structure. Cambridge, MA: MIT Press.

Langacker, Ronald W. 1987. Foundation of cognitive grammar, vol. 1: Theoretical prerequisites. Stanford: Stanford University Press.

Li, Y.-H. Audrey. 1990. Order and constituency in Mandarin Chinese. Dordrecht: Kluwer Academic Publishers.

McCawley, James. 1971. Where do noun phrases come from? Semantics, ed. by D. D. Steinberg and L. A. Jakobovits, 217-231. Cambridge: Cambridge University Press.

Tai, James H-Y. 1973. Chinese as a SOV language. Papers from the 9th Chicago Linguistic Society meeting, 659-71. Chicago: Chicago Linguistic Society.

_____. 1985. Temporal sequence and Chinese word order. Iconicity in syntax, ed. by John Haiman, 49-72. Amsterdam: John Benjamins Publishing Co.

_____. 1989. Toward a cognition-based functional grammar in Chinese. Functionalism and Chinese grammar, ed. by James H-Y. Tai and Frank F. S. Hsueh, 187-226. Chinese Language Teachers Association Monograph Series No. 1.

Talmy, Leonard. 1985. Lexicalization patterns: semantic structure in lexical forms. Language typology and syntactic description, vol. 3: grammatical categories and the lexicon, ed. by Timothy Shopen, 57-149. Cambridge: Cambridge University Press.