[Proceedings of the *Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Colgate University, Hamilton, New York, August 14-16, 1985]

Machine Translation in the SDCG Formalism

Xiuming Huang

Institute of Linguistics
Chinese Academy of Social Sciences
Beijing, China*

Abstract

The paper describes the SDCG (Semantic Definite Clause Grammars), a formalism for Natural Language Processing (NLP), and the XTRA (English Chinese Sentence TRAnslator) machine translation (MT) system based on it. The system translates general domain English sentences into grammatical Chinese sentences in a fully automatic manner. It is written in Prolog and implemented on the DEC-10, the GEC, and the SUN workstation.

SDCG is an augmentation of (Pereira et al 80)'s DCG (Definite Clause Grammars) which in turn is based on CFG (Context Free Grammars). Implemented in Prolog, the SDCG is highly suitable for NLP in general, and MT in particular.

A wide range of linguistic phenomena is covered by the XTRA system, including multiple word senses, coordinate constructions, and prepositional phrase attachment, among others.

Index Terms

Machine translation, AI, DCG, SDCG, CFG, Case grammar, Preference semantics, Prolog

1. Introduction

Machine translation systems are traditionally bulky, incomprehensible for the outsiders, and lacking consistent formalisms. Nobody seems to really know what's happening in systems like SYSTRAN, for instance; most of the existing MT systems (if we don't count machine aided translation systems like LOGOS, ALPS or WEIDNER as MT systems in the strict sense) are not portable, and written in non-NLP languages like Fortran, Algol, Basic or Cobol which make high efficiency impossible.

* Mailing address: Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003

The work described here is meant to be an attempt on MT from a new perspective, taking advantage of such grammatical theories as case grammar, preference semantics and definite clause grammar, and of AI-oriented programming languages (Prolog in this case). The results gained show us that portable, cost-efficient, easy-to-understand and easy-to-modify fully automatic MT systems are feasible.

2. The Semantic Definite Clause Grammars

The DCG is developed by (Pereira & Warren 80) for NLP. It is a top-down, depth-first non-deterministic parsing formalism. By contrasting the following extracts of a CFG and a DCG, the reader will see the relationship between the DCG and the CFG.

A CFG

```
sentence --> noun_phrase, verb_phrase.

noun_phrase --> determiner, noun, relative.

noun_phrase --> proper_noun.

verb_phrase --> trans_verb, noun_phrase.

verb_phrase --> intrans_verb.

relative --> [].

relative --> [that], verb_phrase.

determiner --> [every].

determiner --> [a].

determiner --> [the].

noun --> [man].

noun --> [woman].

proper_noun --> [John].

proper_noun --> [Mary].

trans_verb --> [loves].

intrans_verb --> [lives].
```

A DCG

```
sentence( s(NP, VP) ) --> noun_phrase( NP ), verb_phrase( VP ).*
```

* The uppercase arguments (NP,VP,etc) are variables in Prolog notation.

```
% [Comments:] This statement, consisting of a right-hand side (a 'goal'), an arrow,
% and a left-hand side (zero or more 'sub-goals') finished with a period, is called
% a 'clause' in PROLOG. The above 'sentence' clause can be read declaratively
% as "an input string is a sentence with the structure s(NP,VP) if it consists of
% a noun phrase NP and a verb phrase VP", or procedurally as "to prove that an
% input string is a sentence with the structure s(NP,VP), prove it consists of
% a noun phrase NP followd by a verb phrase VP".
noun_phrase( np(Det, Noun, Rel) ) -->        determiner(Det), noun(Noun), relative(Rel).
noun_phrase( np(Name) ) -->        proper_noun(Name).
verb_phrase( vp(TV, NP) ) -->        trans_verb(TV), noun_phrase(NP).
verb_phrase( vp(IV) ) --> intrans_verb(IV).
relative( rel(that, VP) ) --> [that], verb_phrase(VP).
relative( rel(nil) ) -->  [].
determiner( det(every) ) --> [every].
determiner( det(a) ) --> [a].
determiner( det(the) ) --> [the].
noun( n(man) ) --> [man].
noun( n(woman) ) --> [woman].
proper_noun(John) --> [John].
proper_noun(Mary) --> [Mary].
trans_verb( tv(loves) ) --> [loves].
intrans_verb( iv(lives) ) --> [lives].
```

Using the above CFG, we can prove that the following string

Every man that lives loves a woman

a legal sentence; if we use the DCG instead, we can, besides proving the same, get a representa-
n for the string as follows:

```
s(np(det(every),n(man),rel(that,vp(iv(lives)))),vp(tv(loves),np(det(a),n(woman),rel([])))).
```

The DCG is a purely syntactic formalism which is not adequate for a serious NLP system (it cannot resolve most cases of word sense ambiguities, for instance), hence the addition of predicates* which execute certain semantic functions in our system. The formalism thus gained, the SDCG, features

the integration of syntax and semantics, and is capable of handling various kinds of linguistic phenomena.

To augment the above DCG to an SDCG, we can, for instance, rewrite it to get the following sub-grammar:

```
sentence(s(Subj_Np,vp(v(Verb_sense),Obj_Np)) -->
        noun_phrase(Subj_Np),
        [Word],   % '[]' indicates the consuming of an input unit (a word or a punctuation
              mark).
        {is_verb(Word, Verb, Tense)},   % '{}' indicates a test. 'Verb' is the base form of
              'Word'.
        subject_verb_match(Subj_Np,Verb,Verb_sense),
        noun_phrase(Obj_Np),
        verb_object_match(Verb_sense,Obj_Np).

noun_phrase(np(det(Det), adj(Adj_sense),n(Head_Noun))) -->
    determiner(Det),
    adjective(Adjective),
    noun(Noun),
    adj_noun_match(Adjective, Noun, Adj_sense, Head_noun).
```

The semantic matches (along the lines of (Katz & Fodor 63)'s selectional restrictions) are carried out in predicates "subj_verb_match", "verb_object_match", and "adj_noun_match", whose codings we will omit here to save space.

As for the lexicon, we can have

```
determiner(the).
```

* A predicate in Prolog is the head of a goal or sub-goal. It is similar to a statement in Pascal or a function in Lisp.

noun(coach,[coach1,coach2]).

noun(star,[star1,star2]).

adjective(tough,[tough1,tough2,tough3,tough4]).

verb(marry,[marry1,marry2]).

And the following are semantic codings for word-senses:

sem(coach1,[head(thing)])*. % 'a passenger coach'.

sem(coach2,[head(man)]). % a trainer.

sem(star1,[head(thing)]). % a celestial object.

sem(star2,[head(man)]). % 'a singing star', etc.

sem(tough1,[poss(thing),head(kind)],preps([])). % 'a tough materail'.

sem(tough2,[poss(man),head(kind)],preps([])). % 'a tough mountaineer'.

sem(marry1,[subj(man),obj(man),head(do)],preps([])).

sem(marry2,[subj(man),obj(thing),head(do)],preps([])). % 'he married money'.

Now let us use this SDCG to parse

(2) The tough coach married a star.

We start from "noun_phrase". After we have the variables *Det, Adjective* and *Noun* instantiated to "the", "tough" and "coach", respectively, we are at "adjective_noun_match", where we match the different senses of "tough" to those of "coach", producing ONE plausible reading as Subj_Np; here first we have 'tough1 + coach1', because the semantic 'head' of "coach1" ('thing') fits into the 'poss(thing)' slot of the semantic coding for "tough1". The slot here specifies the preferred semantic category ('thing') of what is being modified by the particular adjective sense ("tough1"). Then, after the verb is found, we try to match Subj_Np with a sense of the verb, and fail because both "marry1" and "marry2" prefer the subject to have the semantic head 'man', which "coach1" cannot satisfy; we backtrack, producing ANOTHER plausible Subj_Np ('tough2 + coach2'), and try subj_verb_match again, this time succeed, with 'marry1' chosen as the verb sense. We proceed to analyse the rest of the sentence, employing "noun_phrase" and "verb_obj_match", and get one plausible Obj_Np ("star2"). ("Star1" is first tried and fails because it doesn't fit the preferred object slot ('man' for "marry1")). Thus one plausible reading of the sentence is gained (see the next section for another example with a representation given).

* The semantic primitives such as 'thing', 'man', etc, come from (Wilks 75)'s 'Preference semantics'. It should be noted, though, that the notion 'preference' is applied differently by Wilks and me: I don't compare the competing structures, while he does.

3. Outline of the XTRA System

The XTRA system is composed of two phases: PARSE which takes the input sentence, parses it, and produces an intermediate structure for it; and GENERATE, which takes the intermediate structure and produces the output Chinese sentence.

The top level clause of the system is to the effect of the following:

```
translate(English_Stn, Chinese_Stn) -->
    parse(English_Stn, Tree),
    generate(Tree, Chinese_Stn).
```

"Tree", the output of the "parse" predicate, is a semantico-syntactic representation for the input sentence (English_Stn). The format is borrowed from (Boguraev 79), though the approaches for getting the representations are entirely different. There are no ambiguities in this representation; in each case slot (Fillmore 68) underneath the verb-sense we have a word senses instead of the word in the original sentence; and all the syntactic information necessary for generating the Chinese sentence (Chinese_Stn) is present. The following is an input English sentence and the intermediate structure produced by the predicate "parse":

(3) John struck the girl on the head on the bank with a club with Fred yesterday.

```
s
  type(dcl)
  tense(past)
  aspect([])
  modality([])
  neg([])
   v
    strike1
    agent(np(det([]),pre_mods([]),n(John),post_mods([]), rel([])))
    object(np(det(the),pre_mods([]),n(girl1),participle([]),rel([])))
    pre_verb_mods([])
    verb_mods
      post_mods(pp(prep(on3),
            prep_obj(np(det(the),pre_mods([]),n(head1),participle([]),rel([]))),
            case(loc_dynamic)))
      post_mods(pp(prep(on4),
```

```
        prep_obj(np(det(the),pre_mods([]),n(bank1),participle([]),rel([]))),
        case(loc_static)))
    post_mods(pp(prep(with5),
        prep_obj(np(det(a),pre_mods([]),n(club2),participle([]),rel([]))),
        case(instrument)))
    post_mods(pp(prep(with12),
        prep_obj(np(det([]),pre_mods([]),n(Fred),rel([]))),
        case(accompaniment)))
    adv_mod
      adv(yesterday)
      case(time)
```

The output of the predicate "generate" is as follows, without any post-editing:

(4) Yuehanh zuoxtian gen Fuxleixde yihqii yongh banghzi zai anh shangh zhaoh
John yesterday with Fred together using club on bank above toward

toux shangh daa le guniangx*.
head above strike tense participle girl

The interested reader is referred to (Huang 84a) and (Huang 84b) for more details.

4. The Performance of the XTRA system

The XTRA is a prototype system, with a rather small vocabulary (about one thousand lexical entries, some of them having more than ten senses). Its coverage of linguistic phenomena, however, is wide. Part of the successfully translated test sentences listed below can serve to illustrate the system's performance.

4.1 Coordinate constructions

(5) John drove his car through and completely demolished a plate glass window .

(This is a classical sentence in the computational linguistics literature. See (Winograd 78).)

(6) Some indicators are known to and their corresponding values used by the Lisp system .

* The four tones of the Chinese characters in the Pinyin form are indicated by the following scheme: 1st tone, nil; 2nd, an additional 'x'; 3rd, repetition of the vowel and 4th, an additional 'h'.

(7) Practical systems for natural language analysis are necessarily large and complex and, for the time being at least, writing them is very much an experimental activity .

(8) John begged Mary to write a novel and Fred Joe a play.

(9) John begged Mary to get married and Fred Joe .

(10) She gave them a bowl and I gave them a spoon .

(11) She gave them a bowl and took nothing in return .

(12) She gave them a bowl and a spoon .

(13) She gave them a bowl and spoon .

(10 - 13 represent another set of classical sentences. See (Wilks 82) and (Winograd 78).)

4.2 Relative clauses

(14) I know the man who saw Mary and shot Dave .

(15) I know the man Mary saw and Dave shot .

(16) I know the man who Mary saw and who shot Dave .

(Here "the man" is the object in the first conjunct of the relative clause, and the subject in the second.)

(17) I know the man who saw Mary and who Dave shot .

(The reverse of the above.)

(18) The term 'phrase' is here used deliberately in a sense which does not necessarily imply that it is a specific element within a clause .

4.3 Participle clauses

(19) I knew the girl bitten by the dog and the cat .

(A choice has to be made between "I knew the girl and the cat" and "I knew the girl bitten by (the dog and the cat)".)

(20) The man shooting John drove to the park .

(21) The man shooting John was shot by Fred .

(22) The man shot by Fred had shot John .

(23) The man shot had shot John .

(24) The man shot shot John.

4.4 Possessives

25) John drove the big man's old sister's car to the park to meet the girl.

4.5 PP attachment

26) John met the girl he worked with at a dance .

("John met the girl at a dance".)

27) John liked the girl he worked with at a dance .

("The girl worked with John at a dance").

28) John met the tall slim auburn haired girl from Montreal that he worked with at a dance .

(Same pp attachment as (26), despite the long distance between "met" and the pp).

29) John bought the book that I have been trying to obtain for Sue .

("for Sue" is attached to "obtain" rather than to "bought".)

30) The woman wanted the dress on the rack .

("the dress was on the rack".)

31) The woman positioned the dress on the rack .

("position on the rack".)

32) The woman wanted the dress for her sister .

33) Joe brought the book that I bought for Mary for Sue .

("for Mary" was attached to "bought", "for Sue" to "brought".)

34) Joe lost the ticket to Paris .

("ticket to Paris".)

35) John lost the game to Fred .

("lost to Fred".)

For more details of the pp attachment machanism, see (Wilks et al 85).

4.6 Concluding remarks

The XTRA is concise: without counting the vocabularies, the whole system takes only 114k bytes storage. The actual time for translating one English sentence into a Chinese sentence is somewhere between a couple of seconds and a couple of minutes. The system is presently running under a C-PROLOG interpreter; if compiled (which is to happen shortly), the speed will increase by about twenty times.

The XTRA system works on an sentence-by-sentence basis; inevitably, its power is limited in certain aspects. For instance, it would be difficult for the system to deal with problems involving anaphora. Multiple sentence processing and inferencing will be the direction for the future work.

Acknowledgements

I would like to thank Dr. Yorick Wilks for commenting on the paper. Any errors are mine.

Bibliography

Boguraev, R.C. (1979) *Automatic Resolution of Linguistic Ambiguities.* Technical Report No. 11, University of Cambridge Computer Lab, Cambridge.

Fillmore,C.J. (1968) "The case for case," Bach & Harms (eds), *Universals in Linguistic Theory.* Holt, Reinhart & Winston.

Huang, X-M. (1984a) "Generating Chinese sentences from the representations of the input English sentences," *Proceedings of the 1984 International Conference on Machine Translation,* February 84, Cranfield Institute of Technology, Bradford.

Huang, X-M. (1984b) "A computational treatment of Gapping, Right Node Raising and Reduced Conjunction", *Proceedings of COLING84,* Stanford University, Palo Alto.

Katz, J. & Fodor, J. (1963) "The structure of a semantic theory," *Language* 39, pp.170-210.

Pereira, F. & Warren, D. (1980) "Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks," *Artificial Intelligence,* 13:231-278.

Wilks, Y.A. (1975) "Preference semantics," Keenan (ed), *Formal Semantics of Natural Language,* Cambridge University Press, London.

Wilks, Y.A. (1982) "Notes on ATNs, charts and the Marcus parser," mimeo.

Wilks, Y.A., Huang, X-M & Fass, D. (1985) "Syntax, preference and right attachment," to appear in *Proceedings of the 9th IJCAI (International Joint Conference on Artificial Intelligence),* August 85, UCLA.

Winograd, T. (1978) "Parsing natural language via a recursive transition net," mimeo.