Technologies for Machine Translation

R.F. SIMMONS [1]
Dept. Computer Science, University of Texas, Austin, TX 78712, USA

Advances in hardware have made available micro-coded LISP and PROLOG workstations, supported by text editing and formatting software. Some of these have been augmented with linguistic technology including large bilingual dictionaries, parsers, generators, and translators to make them powerful tools for research and development of automated translation. Some techniques of linguistic engineering for accomplishing translation are described, and it is suggested that the present barely satisfactory approach involving sentence-by-sentence translation will eventually be improved by incorporating the results of research on analyzing discourse.

1. Overview of Translation Technologies

As the technology of Machine-Assisted and Automated Language Translation emerges from basic research laboratories into industrial development, [2] the question arises of just what technologies are involved. The area of science concerning automated translation is *computational linguistics*, a discipline that is partially included in Artificial Intelligence, but one depending on the best classical linguistic theories concerning the structure and function of natural human languages. The engineering disciplines include computer architecture, software design, human factors design, and a new area which I shall call *linguistic engineering*.

Translation workstations incorporate all these engineering disciplines. They are micro or mini computers with specialized software that includes language editing, access to multi-lingual dictionaries, computer grammars for parsing and generating natural language sentences in its various languages, spelling correctors, and special technical terminology vocabularies. The most powerful workstations are LISP or PROLOG machines that may be networked to communicate with each other and with gigantic file servers that may provide thousands of megabytes of file storage. These machines are hardware that is specially designed to process non-numeric symbolic logic computations in such languages as LISP and PROLOG. Fujitsu's FACOM Alpha with its Atlas II, and the prototypes of Prolog workstations currently under development are two Japanese examples of such specially designed machine architecture for application to translation and AI.

Advanced software and human factors engineering are exemplified in systems such as SCRIBE, EMACS, TEX, WRITER'S WORKBENCH, and EPISTLE that provide technological prototypes for the development of tools oriented toward sophisticated text editing and printed text production. Workstations achieve much of their value by using software that approximates these tools to a greater or lesser extent. The translation workstation uses this class of tools, but must also provide parsing, translation, and generation software which is yet available only in first approximations to their eventual development as fully understood programs, well engineered for human use.

The availability of such advanced computer architecture, miniaturized, packaged attractively, relatively inexpensive, and available to individual translators, can be expected to contribute to rapid advances in the accumulation of the vast linguistic databases necessary to describe human languages in enough detail to enable routine high quality, automated translation of technical documentation. Just as ordinary business applications require thousands of clerks using computer terminals to input and process terabytes [3] of business data,

[1] This research was supported by NSF Grant IST-8403028 and the U.S. Army Research Office under contract DAAG29-84-K-0060.

[2] a process that began twenty years ago with several eager companies of whom one or two yet survive.

[3] giga for billion, tera- for trillion

North-Holland
Future Generations Computer Systems 2 (1986) 83–94

translation systems require comparable legions working on specialized workstations to accumulate and process at least megabytes of linguistic data that will support translation.

While hardware, software, and human factors engineering contribute most heavily to the development of practicable translation systems, the essential hidden technology is linguistic. The formal representation of natural languages in grammars, lexicons, and conceptual paraphrase rules, enables algorithms called parsers, translators, and generators to be used for accomplishing translation. The function of a *parser* is to accept natural language sentences as input and to translate these strings of words into formal representations of meaning – usually in some variant of predicate calculus. A *translator* can use paraphrase rules to transform the resulting logical expressions in a truth-preserving manner to equivalent expressions that can then be input to a generator to form new natural language strings. Where the paraphrase rules translate from Japanese conceptual structures to English ones and the generator takes the result to English sentences, the system is a Japanese-to-English translator. If instead, the transformed Japanese conceptual structures are fed to a Japanese generator, the result will be a Japanese paraphrase of the original Japanese strings. Systems used for preparing summaries and for answering questions employ a single grammar ignoring translation to another language.

Classical linguistic theory underlies the construction of the lexicon and the grammar, informing us how strings of natural language can be translated into constituents of logical representation. Most notable among current linguistic theories are Lexical Functional Grammar, LFG; Generalized Phrase Structure Grammar, GPSG; Extraposition Grammar, EG; and Government Binding Theory, GB. There is fortunately a convergence of these different theories toward the use of augmented phrase structure rules that translate natural language into its constituent structures and those constituent structures into logic. The linguistic content of these theories is largely concerned with stating complex constraints on how one constituent may relate to another. The computational side of these theories requires that the constraints of the theory be incorporated into a computational grammar in such a manner as to minimize the number of possible interpretations for each sentence.

The technologies outlined so far provide at least minimally adequate science for understanding and translating single sentences between languages. They are moderately successful in providing medium to high quality translation – sentence by sentence – of technical documents. One current outstanding achievement is the Metal system, realized on a Symbolics LISP machine, that has translated over 2000 pages of German technical documentation to English with quality rated medium to high, measured in comparison to human translators [4]. The present encouraging situation has led to considerable commercial effort to market usable systems – an effort which in Japan is represented by at least eighteen research projects, more than the rest of the world combined!

Yet the current status of translation requires considerable improvement in order that such simple exchanges as those of a tourist at a train station can be understood. Pronouns and elisions abound in ordinary speech and in non-technical writing. These and repeated references can only be dealt with by linguistic analyses that relate a current utterance to an ongoing context. These analyses require an appreciation of how human communications involve understanding the situation in which the utterance is made, and using that knowledge to comprehend what is meant rather than what is merely stated. Unfortunately, classical linguistic theories have not progressed much beyond the sentence level; excellent theories of sentence structure abound, but in the literature of discourse analysis there are as yet no satisfying theories of pronominalization and coreference among terms occurring in different sentences. In computational linguistics a dozen or so important experimental programs have studied these phenomena, and an optimist can believe that the current limitation of translation aids for application to single sentences will shortly yield to techniques of discourse translation – the translation of sentences with specific reference to preceding and following context.

Most of this paper is concerned with linguistic engineering: parsing and parsers, the forms of lexical, syntactic, and paraphrase rules, semantic representation, linguistic knowledge acquisition systems, and current research in discourse analy-

[4] Personal communication with Linguistic Research Center, Univ. of Texas, Austin. See [14].

Technologies for Automated and Semi-automated Translation

Computer Architecture
- Workstations – Prolog and Lisp Machines micro-coded to achieve high speed symbolic computation
- Specialized Microcomputers

Software Engineering
- Editors, Text formatters, Spelling Checkers, Style Editing (Writer's Workbench, Epistle)
- Office Automation (networking, mail and memo, services, central fileserver)

Linguistic Software
- Semantic Relation Theory,
- Parsers, Translators, Generators
- Lexicon, grammar, paraphrase rule systems
- Rule Acquisition Systems

Linguistic Theories
- Lexical Functional Grammar
- Generalized Phrase Structure Grammar
- Extraposition Grammar
- Government Binding Theory

Discourse Theories
- Theories of Topic and Focus
- Rhetorical Relations
- Coherence Relations

Fig. 1. Technologies for NL Translation.

sis. It attempts to communicate the basic computational techniques for accomplishing these technologies.

2. The Translation Paradigm

From the perspective of the mid-1980s it is quite difficult to realize that pioneers in mechanical translation, using first generation hardware, coding programs in assembly language before the invention of compilers, nevertheless accomplished computational experiments in translating between pairs of natural languages thirty-five years ago. The hypotheses they tested at first treated each language simply as an assemblage of dictionary entries and a translator as the automatic application of a bilingual dictionary. This hypothesis failed decisively and was succeeded by considering the place of grammar in a translation system. By 1965 a couple of large, clumsy translation systems had succeeded in moderate quality translation for several paragraphs of text but, more importantly, the linguists Satterthwaite [12] and Tosh [15] had discovered and published the basic translation procedure for sentences. It remains valid to this day.

The flow of this procedure can be appreciated as the following mapping from Source Language SL, to Source Language Semantic Representation, SLSR, to Target Language Semantic Representation, TLSR, and finally to Target Language, TL.

SL → SLSR → TLSR → TL

At the time of its first publication the "semantic representations" were simply phrase structure trees from the sentence parse, and transfer rules mapped the source language tree into a tree representing a corresponding phrase structure in the target language. This syntactic formulation was sufficient for translating among related European languages but rules for mapping phrase structure trees into other phrase structure trees are complex and cumbersome when the two languages are as different as English and Japanese. What was needed was a better representation of the "meaning" of a sentence.

3. Representation of Meaning

By the late 1970s two lines of research on sentence representation – the study of semantic networks and the use of predicate logic – had matured sufficiently for the realization to grow that semantic nets were a computational variation of ordinary predicate logic. A *semantic network* represents a sentence as a head node and a set of arc-node pairs, where the arc is the name of a deep case relation and the nodes are either atoms or other network nodes, representing concepts. A primitive *semantic relation* is a case arc relating two terms, a triple. A semantic relation may be represented as a head term and a list of pairs or as a set of triples. For example, the sentence, *The old samurai ate fish* is represented as:

(EAT TNS PAST AGT (SAMURAI MOD OLD DET THE) AE(FISH DET GEN NBR_FREE))[5]

[5] _FREE is a variable that matches either singular or plural.

translation systems require comparable legions working on specialized workstations to accumulate and process at least megabytes of linguistic data that will support translation.

While hardware, software, and human factors engineering contribute most heavily to the development of practicable translation systems, the essential hidden technology is linguistic. The formal representation of natural languages in grammars, lexicons, and conceptual paraphrase rules, enables algorithms called parsers, translators, and generators to be used for accomplishing translation. The function of a *parser* is to accept natural language sentences as input and to translate these strings of words into formal representations of meaning – usually in some variant of predicate calculus. A *translator* can use paraphrase rules to transform the resulting logical expressions in a truth-preserving manner to equivalent expressions that can then be input to a generator to form new natural language strings. Where the paraphrase rules translate from Japanese conceptual structures to English ones and the generator takes the result to English sentences, the system is a Japanese-to-English translator. If instead, the transformed Japanese conceptual structures are fed to a Japanese generator, the result will be a Japanese paraphrase of the original Japanese strings. Systems used for preparing summaries and for answering questions employ a single grammar ignoring translation to another language.

Classical linguistic theory underlies the construction of the lexicon and the grammar, informing us how strings of natural language can be translated into constituents of logical representation. Most notable among current linguistic theories are Lexical Functional Grammar, LFG; Generalized Phrase Structure Grammar, GPSG; Extraposition Grammar, EG; and Government Binding Theory, GB. There is fortunately a convergence of these different theories toward the use of augmented phrase structure rules that translate natural language into its constituent structures and those constituent structures into logic. The linguistic content of these theories is largely concerned with stating complex constraints on how one constituent may relate to another. The computational side of these theories requires that the constraints of the theory be incorporated into a computational grammar in such a manner as to minimize the number of possible interpretations for each sentence.

The technologies outlined so far provide at least minimally adequate science for understanding and translating single sentences between languages. They are moderately successful in providing medium to high quality translation – sentence by sentence – of technical documents. One current outstanding achievement is the Metal system, realized on a Symbolics LISP machine, that has translated over 2000 pages of German technical documentation to English with quality rated medium to high, measured in comparison to human translators [4]. The present encouraging situation has led to considerable commercial effort to market usable systems – an effort which in Japan is represented by at least eighteen research projects, more than the rest of the world combined!

Yet the current status of translation requires considerable improvement in order that such simple exchanges as those of a tourist at a train station can be understood. Pronouns and elisions abound in ordinary speech and in non-technical writing. These and repeated references can only be dealt with by linguistic analyses that relate a current utterance to an ongoing context. These analyses require an appreciation of how human communications involve understanding the situation in which the utterance is made, and using that knowledge to comprehend what is meant rather than what is merely stated. Unfortunately, classical linguistic theories have not progressed much beyond the sentence level; excellent theories of sentence structure abound, but in the literature of discourse analysis there are as yet no satisfying theories of pronominalization and coreference among terms occurring in different sentences. In computational linguistics a dozen or so important experimental programs have studied these phenomena, and an optimist can believe that the current limitation of translation aids for application to single sentences will shortly yield to techniques of discourse translation – the translation of sentences with specific reference to preceding and following context.

Most of this paper is concerned with linguistic engineering: parsing and parsers, the forms of lexical, syntactic, and paraphrase rules, semantic representation, linguistic knowledge acquisition systems, and current research in discourse analy-

[4] Personal communication with Linguistic Research Center, Univ. of Texas, Austin. See [14].

Technologies for Automated and Semi-automated Translation

Computer Architecture
- Workstations – Prolog and Lisp Machines micro-coded to achieve high speed symbolic computation
- Specialized Microcomputers

Software Engineering
- Editors, Text formatters, Spelling Checkers, Style Editing (Writer's Workbench, Epistle)
- Office Automation (networking, mail and memo, services, central fileserver)

Linguistic Software
- Semantic Relation Theory,
- Parsers, Translators, Generators
- Lexicon, grammar, paraphrase rule systems
- Rule Acquisition Systems

Linguistic Theories
- Lexical Functional Grammar
- Generalized Phrase Structure Grammar
- Extraposition Grammar
- Government Binding Theory

Discourse Theories
- Theories of Topic and Focus
- Rhetorical Relations
- Coherence Relations

Fig. 1. Technologies for NL Translation.

sis. It attempts to communicate the basic computational techniques for accomplishing these technologies.

2. The Translation Paradigm

From the perspective of the mid-1980s it is quite difficult to realize that pioneers in mechanical translation, using first generation hardware, coding programs in assembly language before the invention of compilers, nevertheless accomplished computational experiments in translating between pairs of natural languages thirty-five years ago. The hypotheses they tested at first treated each language simply as an assemblage of dictionary entries and a translator as the automatic application of a bilingual dictionary. This hypothesis failed decisively and was succeeded by considering the place of grammar in a translation system. By 1965 a couple of large, clumsy translation systems had succeeded in moderate quality translation for several paragraphs of text but, more importantly, the linguists Satterthwaite [12] and Tosh [15] had discovered and published the basic translation procedure for sentences. It remains valid to this day.

The flow of this procedure can be appreciated as the following mapping from Source Language SL, to Source Language Semantic Representation, SLSR, to Target Language Semantic Representation, TLSR, and finally to Target Language, TL.

SL → SLSR → TLSR → TL

At the time of its first publication the "semantic representations" were simply phrase structure trees from the sentence parse, and transfer rules mapped the source language tree into a tree representing a corresponding phrase structure in the target language. This syntactic formulation was sufficient for translating among related European languages but rules for mapping phrase structure trees into other phrase structure trees are complex and cumbersome when the two languages are as different as English and Japanese. What was needed was a better representation of the "meaning" of a sentence.

3. Representation of Meaning

By the late 1970s two lines of research on sentence representation – the study of semantic networks and the use of predicate logic – had matured sufficiently for the realization to grow that semantic nets were a computational variation of ordinary predicate logic. A *semantic network* represents a sentence as a head node and a set of arc-node pairs, where the arc is the name of a deep case relation and the nodes are either atoms or other network nodes, representing concepts. A primitive *semantic relation* is a case arc relating two terms, a triple. A semantic relation may be represented as a head term and a list of pairs or as a set of triples. For example, the sentence, *The old samurai ate fish* is represented as:

(EAT TNS PAST AGT (SAMURAI MOD OLD DET THE) AE(FISH DET GEN NBR_FREE))[5]

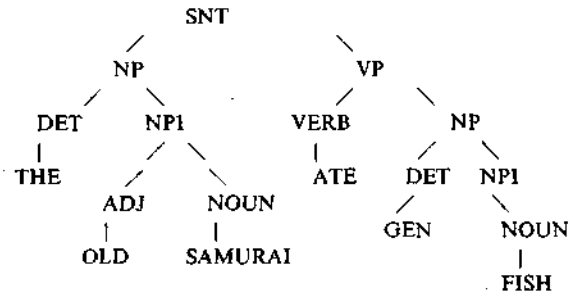
[5] _FREE is a variable that matches either singular or plural.

It is drawn as the following network:

```
EAT-TNS-PAST
|
|-AGT-SAMURAI-MOD-OLD
|    |-NBR-SING
|    |-DET-THE
|
|-AE-FISH-DET-GEN
  |-NBR-_FREE
```

Some case arcs in semantic relations have meaning at both the syntactic and logical levels. At the syntactic level the arc labels designate classes of surface constituents that may be used to realize the concept; at the logical level they are relations that identify various applicable inference rules for paraphrase. The arcs agent (AGT), instrument (INSTR), affected entity (AE), source (*FROM), and goal (*TO) are most common within sentences and reflect linguistic case theory. Between sentences relations of sequence, taxonomy, and causality are used.

A phrase structure tree for this sentence appears as follows:



As each constituent in the tree is recognized by a grammar rule, the rule immediately transforms it into an appropriate case relation. The syntax of a sentence guides the flow of control through a grammar program in the process of translating the constituents to SR form.

Semantic relations are actually a variant on the notation of predicate calculus. This can be observed by comparing the predicate logic description with the SR described as a set of triples. Instead of using the usual variables, *X*, *Y*, *Z*, etc, subscripted words are used as variables and unsubscripted words as constants.

```
(EXIST OLD2 SAMURAI1 EAT1 FISH3
   (SAMUARAI SAMURAI1)
   (OLD OLD2)
   (EAT EAT1)
   (FISH FISH3)
   (AGT EAT1 SAMURAI1)
   (AE EAT1 FISH3)
   (TENSE EAT1 PAST)
   (NBR SAMURAI1 SING)
   (DET SAMURAI1 THE)
   (MOD SAMURAI1 OLD2)
   (NBR FISH3 _FREE)
   (DET FISH3 GEN))
```

EXIST declares the subscripted words to be existentially quantified variables, and terms such as (SAMURAI SAMURAI1) state that the variable inherits the properties of the concept SAMURAI. Predicate names such as AGT, AE, etc. are the case relations and assert that the relation holds between two variables such as EAT1 and SAMURAI1. Following the LISP tradition, parentheses are moved to surround each predicate.

The SR notation differs from the logic in minor ways.

```
((EAT TNS PAST)(EAT AGT SAMUARI)
(SAMURAI MOD OLD)(SAMURAI DET THE)
(SAMURAI NBR SING)(EAT AE FISH)
(FISH DET GEN)(FISH NBR_FREE))
```

For ease in indexing the predicate name is moved to an infix position. Because each SR is a unique set of triples, the words can at once do the double duty of representing a term and a variable, and under the convention that all terms are existentially quantified unless otherwise noted, the existential quantifiers can be discarded. (For more technical descriptions of the relations between predicate calculus and semantic nets see [13] and [3])

The important observation is that the semantic network representation *is* a predicate logic representation that grounds the linguistic structure of meaning in one of the most powerful tools of rational thought, developed over many centuries independently in Eastern and Western cultures. Predicate logic offers not only a consistent, complete, representation for sentences, it provides also a method for deducing new relations from those given, by using rules of inference. It is a primary basis for cognitive models of human rational thought processes.

Logical inference rules are what accomplish the transfer phase in the translation procedure. We can think of them as paraphrase rules that map from an SR in one language to an SR of equivalent meaning in another. The rules must be sensitive to context so that such a phrase as *English man* can map to *man from England* while *English computer* might go to *computer made in England* or for some later date, *computer using English*.

Today's version of the control flow in a translator is hardly changed from that of the original discovery,

Parse Input Language → Logic Representation, LRIL
Paraphrase LRIL → Logic Representation, LRTL
Generate LRTL → Target Language translation

but twenty years of linguistic and computational research have provided much improved understanding of the structure of language, effective forms of grammars, and the nature of parsers, generators, and inference systems.

4. Parsers, Generators, and Inference Engines

It is PROLOG that synthesizes all of these procedures in a single algorithm, resolution theorem proving. Traditionally a parser transforms language strings into a data structure such as a phrase structure tree or a semantic representation. It uses a set of rules to examine each substring of the input string to derive the desired structure. A generator accomplishes the opposite procedure, transforming the data structure into output strings in a language. An inference engine, i.e. a theorem prover, uses a set of axioms including rules for transforming axioms into alternate forms, to determine if a statement given it in a formal language is consistent with its axioms. Prolog is such a system and it shows us in concrete fashion that parsing and generation are actually specialized forms of theorem proving.

In Prolog, [6] grammar rules for parsing are programs interpreted as theorems to be proved. The grammar rules are written and loaded as axioms in the system. If the theorem:

```
(SENTENCE (THE OLD SAMURAI EATS FISH)
  -V NIL)
```

is successfully proved to be consistent with the grammar the variable V becomes bound to the interpretation, as (abbreviated) below:

```
(SENTENCE (THE OLD SAMURAI EATS FISH)
  (EAT TNS PAST
    AGT(SAMURAI MOD OLD)
    AE (FISH))NIL)
```

A few grammar rules for a noun phrase appear as follows:

```
((NP1(_X._Y)(_V MOD_X1._V1)_R)
← (ADJ_X_X1)(NP1_Y(_V._V1)_R))
((NP1(_X._Y)_V1_Y) ← (NOUNX_V1))
((NOUN SAMURAI(SAMURAI NBR_FREE)))
((ADJ OLD (OLD)))
```

If we try the theorem,

```
(?-(NP1(OLD SAMURAI EATS FISH)_V_R))
```

the system returns,

```
V = (SAMURAI MOD OLD NBR_FREE)
R = (EATS FISH)
```

So Prolog itself is the parser. Now if the theorem:

```
(?-(NP1_STRING(SAMURAI MOD OLD NBR_FREE)_R))
```

is tried, the system replies:

```
STRING = (OLD SAMUARI) R = NIL
```

Aha! Prolog is also a generator. The grammar axioms describe the relation between string constituents of a language and their translation into some structure. If the relation is symmetric both parsing and generation are described by the same grammar.

Now we can write a paraphrase grammar to translate *old samurai* into *elderly knight*. We shall use rules of the form:

```
((PARAPHRASE_STR1_STR2)
← (CONSTRAINT1...)
...(CONSTRAINTn...))
```

STR1 and STR2 are not strings, but structures and the constraints are predicates that prove certain properties hold on STR1 in order to produce

[6] The Prolog illustrated here is Lisp Prolog for the Symbolics 3600 Lisp machine. Variables are recognized only by underscore, e.g. _X, upper and lower case are not distinguished, the optional separator between a consequent and its antecedents is the symbol ←, and parentheses surround the relations.

the paraphrased structure, STR2. The following rules accomplish the paraphrase.

```
((PARAPHRASE(SAMURAI._REST)(KNIGHT._REST)))
((PARAPHRASE(OLD._REST)(ELDERLY._REST))
← (CONTEXT(_HD._PRS))(FEATURE_HD PERSON))
((FEATURE SAMURAI PERSON))
((CONTEXT(SAMURAI MOD OLD NBR_FREE)))
```

The first paraphrase rule is context free; in any environment *samurai* may be paraphrased as *knight* by this rule. The second shows one way context may be consulted to limit the paraphrase of *old* to *elderly* just in case the term it modifies is a person. If we read the above axioms into the Prolog system and present the theorem:

(?-(PARAPHRASE(SAMURAI)_V))

the response is V = KNIGHT, but we need additional axioms to paraphrase an entire SR structure:

```
((PARA(_HD._REST)(_V._W))
← (RETRACT(CONTEXT_))
   (ASSERT(CONTEXT(_HD._REST)))
   (PARAPHRASE(_HD._REST)_V)(PARA1_REST_W))
((PARA1(_ARC_VAL._REST)(_ARC_V._W))
← (NOT(ATOM_VAL))(PARA_VAL_V)(PARA1_REST_W))
((PARA1(_ARC_VAL._REST)(_ARC_VAL._W))
← (ATOM_VAL)(PARA1_REST_W))
((PARA1 NIL NIL))
```

Without delving into these axioms, it is worth noting that they describe flow of control during the process of examining each node and arc of a structure; the PARAPHRASE predicates contain the knowledge of when and how to paraphrase terms.

So Prolog is a translator? Not quite. It needs more than paraphrase rules to accomplish the translation; it must have program axioms to recursively translate the portions of structure it is given. We need to call

(?-(PARA(SAMURAI MOD OLD NBR_FREE)_V))

to get:

V = (KNIGHT MOD ELEDERLY NBR_FREE)

But there may be rule forms, as in the grammar, that would more fully utilize Prolog as a translator.

So what happens to forty years of research on parsing and generation algorithms? Prolog was invented by Colmerauer with Roussel in the early 1970s as part of an effort to translate natural languages. They incorporated much of their understanding of grammars and parsing into the system. Since Colmerauer [1978] is also a logician he could see that a theorem prover was the most general way to accomplish these tasks. David Warren [16] took the next step and incorporated a deep understanding of compilers and AI to make this general resolution theorem prover fast enough to be an effective programming language. The result is that Prolog needs mainly linguistic knowledge to accomplish translation – and the place of parsers and generators is to augment other programming languages with some of the capabilities integral to Prolog.

5. Technology of Grammar Rules

We have seen that a grammar is a rule system for transforming strings of a language into semantic representations. To a parser and to Prolog it is a program – a non-deterministic program when developed to encompass any large sample of natural language. From the current AI viewpoint, a grammar can also be viewed as a rulebased *expert system* whose expertise concerns the patterns occurring in language strings and how to translate them into semantic representations. A grammar, including lexical rules, is a most difficult program to write and debug. Let us see why.

A grammar is composed of syntactic and lexical rules that are interdependent. The syntactic rules recognize patterns of syntactic classes and features; the lexical rules recognize patterns of morphemes, mapping them into related patterns of wordclasses and features. A syntactic rule for a simple sentence in English has the following form using an augmented phrase structure grammar expressed in Prolog:

```
(SNT_STRING(_W_ARC_V._W1))
← (NP_STRING_V_REMDR)(VP_REMDR(_W._W1)NIL)
   (MEMPR(SUBJ_ARC)_W1).
```

The variable STRING is the input sentence and the result of applying the rule is constructed during the parse, as the second argument in the rule. The variable W is the head of the verb phrase, W1 is the list of pairs associated with the head; ARC is the arcname associated with the feature, SUBJ is the verb entry; and V is the structure returned by the NP rule. The result is obtained by com-

puting the NP, then the VP, then by using MEMPR, a function to return the values of elements in a list of pairs in a semantic relation, to find an arcname associated with the subject of this class of verb. Assembling these elements into an SR, (_W_ARC_V._W1) in the case of our example sentence gives:

W = EAT
ARC = AGT
V = (SAMURAI MOD OLD...)
W1 = (TNS PAST AE(FISH...))

i.e.

(EAT AGT (SAMURAI MOD(OLD)...)
TNS PAST AE (FISH...))

The lexical entry for ATE appears as follows:

(ATE VERB(EAT TNS PAST
SUBJ AGT OBJ AE TY ACT))

The SUBJ and OBJ features provide the arc names associated with those elements of a clause using some morphological form of EAT. A verb like *give* that takes an indirect object as in:
John gave the library books,
or
John gave books to the library.
has a lexical entry coded to recognize those forms.

(GAVE VERB(GIVE TNS PAST SUBJ AGT OBJ AE
I-OBJ*TO OPT(TO FOR)TY TRANSFER))

The I-OBJ or indirect object feature informs that the arcname for an indirect object is *TO. In contrast, the verb *get* which also takes an indirect object selects a different arc, *FOR.

(GOT VERB(GET TNS PAST SUBJ AGT OBJ AE
I-OBJ*FOR OPT(FROM TY TRANSFER))

So a sentence like,
John got Mary a book,
gets the same interpretation as,
John got a book for Mary.
A pair of rules for recognizing noun complements of verbs uses these features to decide upon arc names for objects and indirect objects.

(VERB_COMPL_VERB_X(_ARC_SR._VAL)_REMDR)
← (NP_X_SR_R)(VERB_COMPL_VERB_R_VAL_REMDR)
MEMPR(OBJ_ARC)_VAL)(NOT(MEMPR(_ARC_W)_VAL))

This rule assigns the arc associated with the OBJ feature to an NP complement just in case the OBJ arc has not been assigned to a following NP. Notice that because of the recursive flow of control, the second NP, if it exists, is parsed by VERB-COMPL before the earlier one assigns its arc. The second rule for VERB-COMPL applies if the one above *fails*.

(VERB-COMPL_VERB_X(_ARC_SR._VAL)_REMDR)
← (NP_X_SR_R)(VERB-COMPL_VERB_R_VAL_REMDR)
(MEMPR(I-OBJ_ARC)_VAL)

This rule simply assigns the arcname associated with the I-OBJ feature.

It should be emphasized that in these rules, the phrase structure components, SNT, NP, VP, etc. are used to find syntactic constituents of the sentence, but no phrase structure tree is constructed.[7] Instead, as each constituent is discovered, it is immediately translated into a constituent of the semantic relation which is the goal of the computation. In the recursive top-down flow of control, a tree of syntactic constituents is discovered in the **down path** and the translations of each syntactic constituent to SR constituents is transmitted back up in the **up path** of the recursion.

We can now appreciate that each lexical entry and grammar rule may become a very complex combination of data structure and code. We notice that the way to recognize an indirect object in English is first to determine that a direct object occurs later in the sentence; if it does not then the NP in question must be the direct object. Thus the application of one rule often depends on the fact that some other rule has already succeeded or failed, earlier or later in the computation. Such logic is extremely difficult to work out in the first place, and may be of great difficulty to program in the second.

It is well known that obtaining and programming the expertise for an expert system is a very difficult, time-consuming task that requires first finding a cooperative expert, then translating his/her understanding into rules, then debugging the rules to reduce inconsistencies and ambiguities to tolerable levels. The experts needed in constructing grammars are linguistically trained and usually not skilled in the use of computer technology, so the first problem is to make available to

[7] Although if there were some use for it, it could be built along with the translation.

them easily used tools for constructing lexicon and grammar. Unfortunately, there aren't many. Lexical acquisition programs have been constructed to provide menu questioning of the user to obtain the features associated with each word to be entered into the dictionary, but these are still in early development stages and are dependent on the form of the grammar to be constructed. Rule acquisition systems have been studied in AI Expert systems research, but are still in the most primitive stages of development. I know of no general approaches to developing a grammar rule acquisition system beyond the use of compiler techniques to translate a linguist's general notations into the set of particular rules they imply.

Part of the solution is to have a team of linguists, some of whom are reasonably good grammar programmers, supported by a strong systems programmer who can solve the computational "mysteries" that the linguists will encounter.

But the problem also involves the fact that a grammar is not a program that can be completed – it grows and grows. Every time a new grammar rule is added, there is some risk that sentences previously translated successfully will now translate differently. Every time a grammar rule is changed the risk is greater; previously successful sentences may then fail. A standard sentence sample must be maintained so that at periodic intervals – presumably when the linguists feel happiest about their recent successes in modifying the grammar to deal successfully with new material – it must be tested against the standard to determine how badly the changes may have affected overall performance.

6. Linguistic Theories

The linguists who prepare the grammar may be of any theoretical persuasion; making it work makes pragmatic eclectics of us all. Each of the current theories accounts for the ways in which immediate constituent phrase structure analysis of a natural language fails in certain cases. Each theory accounts for the way in which relative pronouns and elisions of certain constituents signal dependence on more distant constituents. Each theory is concerned to develop lexical and rule systems that account for psychological facts about how people can learn their individual languages. Each seeks *linguistic universals* that hold across all human languages. Most of the current theories include a notion of a meaning structure underlying the surface utterances and most of these represent that meaning structure in predicate calculus.

The differences among the theories are of profound significance to classical linguists and indeed may in the eventual development of linguistic theory be of crucial importance in accounting for the less obvious facts of language. But for the level at which sentence translation can be obtained, the similarities among the theories is of most importance.

Currently, Lexical Functional Theory and Generalized Phrase Structure Theory are most frankly concerned with computational development and testing of grammars. At least two efforts, one in Britain and one in California are concerned to produce grammars encompassing English written in forms of GPSG. Government Binding theorists appear to be more concerned with accounting for critical problems concerning how a language can be acquired by children. A form of Lexical Function Theory has recently been used in a dissertation by Yukiko Alam [1985] that develops translation linguistics for English and Japanese in LFG using a case structured analysis. In recent years excellent descriptions of these theories have been published [8]

7. Discourse Research

In this area there exist only the beginnings of theory and initial experimental explorations of computational technology. Yet this is the cutting edge of computational linguistic research, and this is the area in which great advances are foreshadowed by current experiments. Without delving into theory, [9] let us look at some of the current work and its problems.

Earlier it was mentioned that some very simple human dialogs are beyond current sentence-by-sentence translators. Some typical dialogs are in the form of questioning and response, as by a traveler and an information clerk. Eventual applications of translation technology might include

[8] See refs. [5,6,9,17].

[9] There is a linguistic literature; see refs. [8,10].

multi-lingual information clerks. A problem for the travel clerk is that in order to provide an answer to a question he/she must estimate the users' intentions in asking the question and so respond appropriately. Allen [1979] used a train station example like the following:
Query: Is there a train to Nara?
Response: Track 15 at 1100.
The travel clerk might have responded directly to the question with the answer, "Yes", but then the user would almost certainly have asked about time and location. The clerk, anticipating the questioner's intention to take or meet the train, returned a cooperative response providing the additional information. If there were no train to Nara, the clerk might have responded,
"No, but an express bus leaves the bus station at noon daily."
In this event the clerk again is cooperating by anticipating the questioner's probable intention. Computational methods for dealing with intentions at this level are only beginning to be understood.

Let us now see how current experimental technology might be used to account for the travel clerk's behavior when asked, "Is there a train to Nara?" We can assume that the clerk is able to access a Transportation Table that contains entries of the following form:

(TRANSPORT_VEHICLE_FROM-CITY
_TO-CITY_TIME_GATE)

with examples such as:

(TRANSPORT TRAIN TOKYO NARA
1100 TRACK 15)
(TRANSPORT BUS TOKYO NARA
1200 BUSGATE 2)

When an inquirer asks the clerk a question about a vehicle, the clerk can assume that the inquirer doesn't know the answer, does have a need to meet the vehicle for some purpose, and therefore needs to know the time and gate. It the query is about a vehicle *from* a city, the clerk may wish to state the destination as well to allow the inquirer additional information that might be used to correct an assumption. (In fact for this overly-simple transport table, the clerk might well have the strategy of reporting all the information just to be safe.)

The question is first translated to the logical form of a transport entry:

(TRANSPORT TRAIN_FROM-CITY NARA
_TIME_GATE).

Each underscore signifies an unbound variable. The clerk's problem solver has the simple task of proving this TRANSPORT hypothesis by matching it with type entries in the TRANSPORT list. The hypothesis unifies with:

(TRANSPORT TRAIN TOKYO
NARA 1100 TRACK 15)

and under the assumptions given above the clerk generates the response, "The train to NARA arrives at 1100 on Track 15". If several trains to NARA existed on the schedule, the clerk might have generated, "The next train to NARA arrives..." and waited for additional questions.

If unification of the question failed in the event there were no train to NARA, the clerk would then relax the constraint of the vehicle, thus allowing for buses and planes.

(TRANSPORT_VEHICLE_FROM-CITY
NARA_TIME_GATE)

which unifies with:

(TRANSPORT BUS TOKYO
NARA 1200 BUSGATE-2)

In this event, the clerk responds, "No, but there is a bus at 1200 leaving from bus-gate 2.

All this for such a simple dialog? Yes, and a deep theory of how people leave their intentions implied rather than expressed in speech. Computer systems to translate dialogs intelligently must not only translate the statements to meaning representation but also must infer what is meant from what is said, attending to the situational context in order to respond cooperatively.

Translation technology has benefited from the fact that pronouns may not have to be understood to be translated correctly. But it is well known that different languages place different agreement constraints on pronouns and they really must be resolved in order to obtain good quality translation. The example sentences below show how the resolution of pronouns may depend on an understanding of the events being described.
John had a bike. Bill wanted it.
a) He give it to him.

b) He stole it from him.
In a *giving* situation, the donor must *have* the object to be given and the recipient might want it. But for a *stealing* situation, the thief must want the object and the victim must have it. Alterman [1982] developed a small dictionary of coherence relations and a system of rules for organizing sequences of clauses describing events into a discourse structure. His system resolves such pronominal references quite naturally as a by-product of its organizing the text. It uses rules such as the following:

```
1. (RULE (WANT ANTE HAVE)
         (MUST = (AE) NOT = (AGT)))
2. (RULE (GIVE ANTE HAVE)
         (MUST = (AGT AE)))
3. (RULE (GIVE ANTE WANT)
         (PAIR = (*TO AE)))
4. (RULE STEAL ANTE HAVE)
         (PAIR = (AE AE)(*FROM AGT)))
5. (RULE (STEAL ANTE WANT)
         (MUST = (AGT AE)))
```

These rules form a large network of how events can relate under the constraints that their arguments must not violate certain correspondence relations. The first rule states that a *wanting* can follow a *having* provided the AE argument, i.e. the object wanted, is the same as the AE argument in the *having*, and provided the Agents of the two events are not equal. Rule 2. shows that the giver must have the object given; rule 3. requires (optionally) that the object to be given should be wanted. In contrast rules 4. and 5. require that the victim initially have the object to be stolen and that the thief should want the booty.

Alterman demonstrated his approach to be effective on a dozen example discourses, including several that had previously been analyzed in the AI literature. Current experimentation in our UT AI Laboratory is confirming that the technique is generally useful and can be incorporated into a coming generation of machine translation technology.

Hobbs in a series of papers from as early as 1976 to the present [10] has developed a different computational theory of rhetorical relations that organize successive sentences in discourse and make it coherent to readers. He demonstrates that a method he calls selective inferencing is central to our understanding of the meaning of text descriptions. In an example such as,

John opened the safe. He knew the combination.
an inference rule such as:

```
((OPEN_X_Y)
←(PERSON_X)(SAFE_Y)(COMBINATION_Y_Z)
  (KNOW_X_Z)))
```

i.e. *X can open y, If X is a person, and Y is a safe, and the combination of Y is Z, and X knows Z.*
can both resolve the pronoun *he* and relate *combination* to *safe*, thus organizing the two sentences into a coherent discourse under the rhetorical relation of **expansion**. Notice that Hobbs expects extremely detailed knowledge to be associated with each lexical item in the form of logical assertions and inference rules. The research problem, as always, is to determine what knowledge is associated with each word, how to acquire it, and how to organize it to avoid exponential explosions of attempted irrelevant inferences.

Such techniques as these, and several others concerning the topic and focus of discourse, are active areas of research in many AI and NL laboratories. An oncoming generation of translation technology will surely be improved by their results.

8. Summing Up the State of Translation Technology

Machine-aided and automated translation of technical text in a sentence by sentence mode is now flowing from the basic research laboratories into the product development labs of industry. One great impetus to this flow comes from the general realization that AI expert systems can be of great use in many industrial domains, one of which is translation of technical documention. Another is that computational linguistic research has resulted in a technology that supports natural language interfacing to database and operating systems, and has demonstrated good quality automatic translation for large numbers of sentences. Neither of these factors would of themselves be sufficient to arouse industrial participation if it

[10] See ref. [11] for a good overview.

were not for the concomitant development of advanced symbol-processing workstations supported by equally advanced text editing and formatting software. Finally, the flow is part of the great international tide of technological advance initiated by Japan's thrust to introduce a fifth generation of computer technology.

Should we still have reservations about the feasibility of automatic translation? Certainly. We need have no doubts that the technology is mature enough to be useful, but we should accept no illusions that is it fully adequate. The Meteo system in Montreal has been in use for years supporting the translation of Canadian weather reports. A first-generation Chinese system is used routinely to translate a mathematical journal. Metal is in regular use for translating German documentation to English. [11] In every case, some post-editing is required, but post-editing is essential for insuring that human translations meet certain levels of quality as well. The Metal experience showed that the cost of post-editing for machine-produced material might be less than that for trained translators. The technology is useful and cost effective in that application.

Still, we are discussing minimally adequate technology: a translation system for sentences when much of the meaning of texts lies in the inter-relations of sentences, a translation of static text when much of the desirability for such systems will be found in dynamic dialog applications, a translation without adequate treatment of pronouns, ellipsis, or definite NP reference, and a translation almost utterly lacking considerations of style.

In the basic research laboratories the corrections to these shortcomings command a sharp focus of attention. The study of methods for automatically analyzing discourse, resulting in larger contextual structures that have resolved pronominal references in on the verge of applications. Research on the structure and understanding of human dialogs will soon result in experimental expert systems that can answer tourist questions and thereafter become translation technology.

The fact that translation is in a developmental stage will encourage dozens of applied laboratories to produce acquisition systems for accumulating linguistic knowledge in computer readable forms. Lexicons, grammars, and world knowledge inference systems – all on a large scale – can be expected to be primary contributions from industry. Parsers, grammars, morphological analyzers, dictionaries, general inference systems, and translators, are all examples of profitable software for the near future. Competition to develop and market these systems can be expected to maintain and even increase the level of resources devoted to translation technology.

The market for translation of technical documents is large. The documentation departments of the great international corporations are yearly turning out hundreds of thousands of pages of technical documentation, most of which needs to be translated to several languages for international customers. The challenge is to so improve our technologies of translation that using the forthcoming fifth generation computers, we can write technical documentation once and simultaneously translate it to the major languages of the world. The task will not be easy. The next generation of parallel computers, the 16K to 64K processor machines that we barely know how to build and certainly don't know how to program, will eventually lead to a complete revision of our still primitive, single-processor translation technology.

The new commitment of industrial resources into NL and AI applications comes thirty-five years after the beginnings of these sciences at a time when they have demonstrated definite and valuable applications; it is an excellent bet that those resources can stimulate automated translation technology into rapid advances.

[11] See ref. [14] for detail on these examples.

References

[1] Alam, Yukiko, *A Study of Japanese and English Verbs and its Application in Computational Linguistics*. PhD Diss., Dept. Linguistics, Univ. of Texas, Austin, 1985, (draft mss.).

[2] Allen, James F., *A Plan-based Approach to Speech Act Recognition*. PhD. Diss., Univ. of Toronto, 1979.

[3] Allen, James F. and Frisch, A.M., "What's in a semantic net?" *Proc. Assoc. Comp. Ling.*, pp. 19–27, Toronto, 1982.

[4] Alterman, Richard, *A System of Seven Coherence Relations for Organizing Event Concepts*. PhD Diss., Dept. Comp. Sci., Univ. of Texas, Austin, 1982.

[5] Berwick, Robert and Weinberg, Amy, *The Grammatical Basis of Linguistic Performance*. The MIT Press, 1984.

[6] Bresnan, Joan, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, Mass. 1982.

[7] Colmerauer, Alain, "Metamorphosis Grammars" in Bolc, L. (ed.) *Natural Language Communication with Computers*. Springer Verlag, New York, 1978.

[8] Grimes, Joseph, *The Thread of Discourse*. Mouton, The Hague, 1975.

[9] Gazdar, Gerald, "Phrase Structure Grammar," in P. Jacobson and and G.K. Pullum (eds.) *The Nature of Syntactic Representation*. D. Reidel, Dordrecht, 1981.

[10] Halliday, M.A., and Hasan, R., *Cohesion in English*. Longman Press, London, 1976.

[11] Hobbs, Jerry, "Selective Inferencing." *Proc. 3rd Conf. Canad. Soc. for Comp. Studies of Intell*. CSCSI, 1980.

[12] Satterthwaite, A.C., "Sentence-for-sentence translation: an example." *Mechanical Translation and Computational Linguistics*. Vol. 8, No. 2, pp. 14-38, Feb. 1965.

[13] Simmons, Robert F. *Computations from the English*. Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[14] Slocum, Jonathan, "Machine translation: its history, current status and future prospects." *Computational Linguistics* Vol 11, No. 1, 1985.

[15] Tosh, Wayne, *Syntactic Translation*. Mouton, The Hague, 1965.

[16] Warren, David H.D., "Implementing PROLOG - compiling logic programs." *Research Reports* 39, 40, University of Edinburgh, 1977.

[17] Winograd, Terry, *Language as a Cognitive Process*. Addison-Wesley, Menlo Park, Calif., 1983.