[From: Makoto Nagao (ed.) *Machine translation summit*. Tokyo: Ohmsha, 1989]

3 • 7 CMU Project

Masaru Tomita and Jaime G. Carbonell
Carnegie-Mellon University, U.S.A.

[1] Introduction

Accurate translation requires a degree of comprehension, and several projects have developed prototype systems to demonstrate the feasibility of knowledge-based machine translation (Carbonell 1981, Nirenburg 1986, Lytinen l984). These approaches combine syntactic and semantic information to produce an intermediate knowledge representation of the source text which is then generated in the target language. This paper does not attempt to revisit the ample rationale for the knowledge-based machine translation concept - such discussion may be found in the literature (Carbonell 1986, 1981, Nirenburg 1986) - but rather discusses new advances in computational methods for combining syntactic, semantic and lexical knowledge that promise to make systematic large-scale KBMT a practical reality. These methods, based on static separation and dynamic integration (via precompilation) of linguistic knowledge sources, are brought together in the *universal parser* architecture, a radical improvement over ad-hoc manual methods for integrating semantics into syntactic parsing methods.

This intermediate semantic representation is often called the *interlingua,* though that is a misnomer. The semantic representation is encoded in a completely formal, canonical and unambiguous notation such as first-order logic or frame-based representation. Once the meaning representation is extracted it may be re-generated in multiple languages, paraphrased, summarized, stored, fleshed out by an inference procedure, or otherwise processed. Thus, the so-called *interlingua approach* unifies machine translation back into the mainstream of natural language processing research in AI.

First, however, let us review the set of performance objectives that contributed to the design of the universal parser:

- *Semantic accuracy* — The translation should maintain *semantic invariance* above all else. Paralleling syntactic form, maintaining equivalent length of text, and other such criteria are considered of secondary importance. Thus, the knowledge-based approach was the only logical choice.

- *Multi-lingual generality* - The system should be able to handle any natural language and any semantic domain, with the addition of a new declarative grammar for the former, and a new declarative knowledge base for the latter. Moreover, the addition of any new language should enable immediate translation to and from all the previous languages, without requiring explicit hand-built transfer grammars for all pairs of languages (as is the case with the best present systems (Boitet 1976, Nagao 1984, Kittredge l976, Slocum 1984)).

- *Interactive Translation* — Translation should occur in real time, interacting with the user as required. Most existing practical machine translation systems are designed to translate off-line large documents, such as technical papers and manuals. However, there is a growing need for interpreting personal communication, such as telexes, business letters, conversations with telephone directory assistance (or at a foreign hospital, hotel, or airport counter). Such interactive usage adds the following demands:

  - *No post-editing* should be required, as one cannot carry along a personal post-editor in case he or she

is needed, just as one cannot always have a professional translator at one's side.

- *Real-time performance* is an absolute requirement, as participants in a dialog will not wait minutes or hours for a response. Nor can the person who composed a letter or telex wait long to see if the system will pose any clarificational questions.

- *Speech compatibility* is an equally strong requirement, as the utility for real-time KBMT translation systems increases dramatically when coupled with speech recognition and synthesis. Speech recognition imposes the requirement to handle unsegmented input, with multiple word candidates present at any point in the input stream, (i.e., the input is typically a lattice rather than a linear string (Hayes 1986).)

- *Linguistic Generally* — Linguistic information (syntactic, semantic, and lexical) should be expressed in elegant, theoretically — motivated formalisms — ones that linguists can use to develop and modify grammars and knowledge bases rapidly (such as LFG).

- *Discourse Phenomena* — Extra-sentential phenomena such as anaphora, ellipsis, metalanguage, and speech acts, should be handled within the framework, as should inference required to support cross-linguistic variation (such as politeness levels, inference of missing constituents — e.g., subjects in Japanese — and finer grain lexical selection required in some target languages).

- *Multiple Utility* - In addition to machine translation proper, the methods developed should be applicable to multi-lingual natural language interfaces (to data bases, expert systems, etc.), automated - skimming and indexing of texts, and to the development of a computational linguistics workbench, where different linguistic theories can be subjected to comparative empirical testing across multiple languages and linguistic phenomena.

We have achieved the majority of these objectives in an experimental system at CMU's center for machine translation, and we are actively working on developing the other capabilities. The system, consisting of the *universal parser* and *universal generator,* is an open-architecture approach to knowledge-based machine translation, integrating multiple off-line knowledge sources into a fast on-line run-time system (Tomita, Carbonell 1986). The rest of this paper discusses that architecture in some detail. We have chosen English and Japanese as our initial languages, and simple doctor-patient communications as our initial test domain, and have produced real-time, semantically-accurate, bi-directional translations at the sentential level.

[2] The Universal Parser: a New Architecture for Multi-lingual Parsing

Multi-lingual systems require parsing multiple source languages, and thus a universal parser, which can take a language grammar as input (rather than building the grammar into the interpreter proper) is much preferred for reasons of extensibility and generality. When dealing with multiple languages, the linguistic structure is no longer a universal invariant that transfers across all applications (as it was for pure English language parsers), but rather is another dimension of parameterization and extensibility. However, semantic information can remain invariant across languages (though, of course, not across domains). Therefore, it is crucial to keep semantic knowledge sources separate from syntactic ones, so that if new linguistic information is added it will apply across all semantic domains, and if new semantic information is added it will apply to all relevant languages. The question, of course, is how to accomplish this factoring, and how to accomplish it without making major concessions to either run-time efficiency or semantic accuracy.

The idea of the Universal Parser is depicted in figure 3-1. There are two kinds of knowledge sources: one containing syntactic grammars for different languages and the other containing semantic knowledge bases for different domains. Each of the syntactic grammars is

totally independent from any specific domain, and likewise, each of the semantic knowledge bases is totally independent from any specific language. Syntactic grammars and domain knowledge bases are written in a highly abstract, human-readable manner. This organization makes them easy to extend or modify, but possibly machine-inefficient for a run-time parser. The grammar compiler takes one of the syntactic grammars (say Language L_i) and one of the domain knowledge bases (say Domain D_j, along with mapping rules (that determine which semantic concept is expressed by what word and what structure), and produces one large grammar which contains both syntactic and semantic information. Such compilation proceeds off-line, producing a compiled grammar that need not be human-readable, but must be machine-efficient in terms of on-line run-time parsing speed. The pre-compiled grammar, in essence, consists of the legal subset of the cross product of L_i and D_j, cross-indexed and optimized for efficient machine access and computation. When the user inputs sentences in Language L_i (and Domain D_j), the run-time parser parses the sentences very efficiently, referencing only the compiled grammar, and producing semantic representations of the sentences.

[3] The System Architecture

Figure 3-2 shows the architecture of the universal parser. We adopt *semantic case frames* for domain knowledge representation and the *functional grammar formalism* for syntactic grammar representation. The run-time grammar produced by the multi-phase compiler is an *augmented context-free grammar* (ACFG) which is further compiled into an *augmented LR table* to be used by a run-time parser based on the Tomita parsing algorithm, the fastest CFG algorithm known for natural languages in practice (Tomita 1985). These components are described in detail in the following subsections.

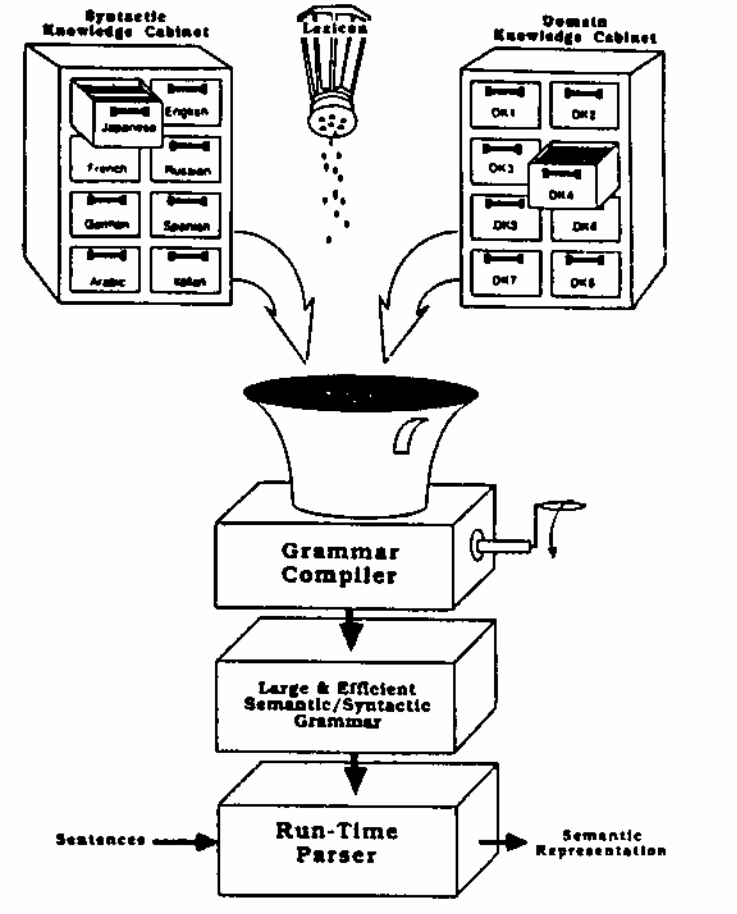

Fig. 3-1 Universal parser concept

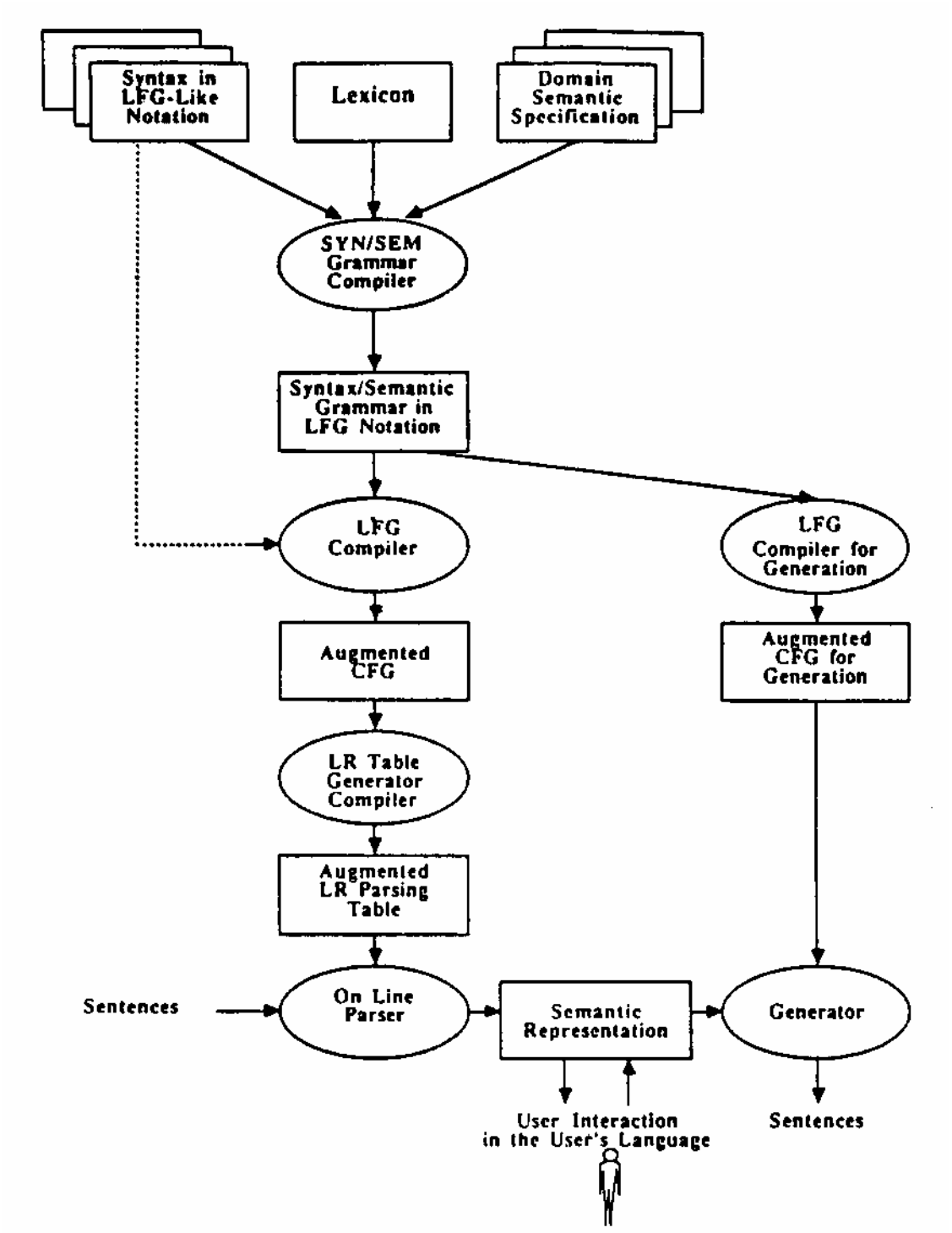

Fig. 3-2 System architecture

(a) Semantic Frame Representation

We use FrameKit (Carbonell 1985) as our knowledge representation language to encode domain semantic knowledge. FrameKit is a compact and fairly efficient general-purpose frame-representation language with multiple-inheritance, procedural attachment, and default semantics. Domain knowledge consists of a set of frames organized into an inheritance hierarchy. Each frame represents a concept such as object, event, state, etc. In the domain with appropriate semantic links to other related frames in the hierarchy. Frames encode typing information, functional dependencies and express compositional constraints used in the parser to block non-productive computations.

Let us consider the domain of simple doctor-patient conversations, in particular the patient's initial complaint about some ailment. Entities in this domain include an event frame *HAVE-A-SYMPTOM and object frames *SYMPTOM, *PAIN, *BODY-PART and so on. Example frame definitions are shown in figure 3-3. Sentences with different surface forms that should be recognized as instantiations of these frames include the following four examples.

```
(*HAVE-A-SYMPTOM
    (is-a (value *SENTENTIAL))
    (:agent (sem *PATIENT))
    (:symptom (sem *SYMPTOM))
    (:associated-action (sem *PATIENT-ACTION))
    (:when (sem *TIME))
    (:start (sem *TIME))
    (:end (sem *TIME))
    (:freq (sem *FREQUENCY))
    (:duration (sem *DURATION))
   )

(*SYMPTOM
    (is-a (value *NOMINAL))
    (:severity (sem *SEVERITY))
    (:location (sem *BODY-PART))
    (:pain-spec (sem *PAIN-TYPE))
    (:name (sem *SYMPTOM-NAME))
   )

(*PAIN
    (is-a (value *SYMPTOM))
   )
```

Fig. 3-3 Fragment of domain semantics specification

I have a headache.
I have a burning pain in the chest.
I have no pain.
Do you have a dull ache in your head.

As a more direct example, the final semantic representation of the sentence
"I have a dull ache in my chest"
produced by instantiating frames is shown in figure 3-4.

```
(*HAVE-A-SYMPTOM
   (:object (*PAIN
                (:location (*BODY-PART
                             (:name *CHEST))
                (:pain-spec (*DULL))))
  (:agent (*HUMAN
            (:person 1)
            (:number SG)))
)
```

Fig. 3-4 Sample semantic representation:
instantiated entities for
"I have a dull ache pain in my chest."

The result of parsing a sentence (see figure 3-4), will be a composition of the instantiated individual frames. This knowledge structure may then be given to any back-end process, whether it be a language generator (to translate into the target language), a paraphraser, a data-base query system, or any expert system.

(b) The Functional Grammar Formalism

We adopt the functional grammar formalism (Kay 1979) for syntactic knowledge representation of each particular language. In essence, this formalism defines syntax in a functional manner based on syntactic roles, rather than by strict positions of constituents in the surface string. The functional framework has clear advantages for languages such as Japanese, where word order is of much less significance than in English, and where case markings take up the role of providing many of the surface cues for assigning syntactic and semantic roles to noun phrase constituents. Moreover, functional structures integrate far more coherently into frame based semantic structures.

Two well-functional grammar formalisms are Functional Unification Grammar (UG) (Kay 1984) and Lexical Functional Grammar (LFG) (Bresnan 1982).

Figure 3-5 is a fragment of LFG written in a notation similar to PATR-II (Pereira 1985, Shieber 1985). The last rule is generated automatically from the dictionary and general morphological rules.

There are two main advantages of using the functional grammar formalism in multi-lingual NLP systems over more traditional linguistic theories:

- A grammar in this formalism can be used for both parsing and generation. Thus, we do not need to write and maintain separate grammars with equivalent coverage for parsing and generation.

- Functional grammar formalisms such as UG and LFG are well-known among computational linguists, and therefore they need not be trained (with some justifiable resistance) to write grammars in arcane system-specific formalisms.

The general problem in parsing with functional grammars is their implementation inefficiency for any practical application. Although much work has been done to enhance efficiency (Shieber 1985, Pereira

```
(<DEC <==>  (<NP> <VP>)
      (((xl case) = nom)
       ((x2 form) =c finite)
       (*OR*
        (((x2 :time) = present)
         ((xl agr) = (x2 agr)))
        (((x2 :time) = past)))
       ((x0) = (x2))
       ((x0 :mood) = dec)
       ((x0 subj) = (xl))))

(<VP> <==> (<V> <NP>)
       (((x2 case) = acc)
        ((x0) = (xl))
        ((x0 obj) = (x2))))

(<V> <--> (feels)
       (((x0 root) = FEEL)
        ((x0 form) = finite)
        ((x0 :time) = present)
        ((x0 agr) = 3sg)))
```

Fig. 3-5 Fragment of English LFG in the PATR-like notation

1985), the functional grammar formalisms are considered far less efficient than formalisms like ATNs (Woods 1970) or (especially) context-free phrase structure grammars. Moreover, pure functional grammars do not provide the semantic information that is required to eliminate nonsensical parses and to construct the output meaning representation. We address both problems by precompiling a syntactic LFG together with a separate domain semantics specification into an augmented context-free grammar, as described in the following section.

(c) Grammar Compilation and Efficient On-Line Parsing

The previous two sections have described how to represent domain semantics and language syntax. The universal parser unifies both knowledge sources and optimizes the grammar for run-time performance in a series of off-line precompilation phases. The first compiler named *syn/sem grammar compiler* compiles the syntactic and semantic knowledge, as well as morphological rules and dictionary, into a single large LFG called *syn/sem grammar*. The compiled the *syn/sem grammar* is in exactly the same form as its original syntactic grammar except that it acquired many additional semantic equations generated automatically by the compiler. The semantic equations check semantic constraints and build semantic representation rather than syntactic f-structures.

This syn/sem grammar is further compiled into an *augmented context-free grammar* (ACFG) by the second compiler named the *LFG compiler*. This ACFG grammar is represented by a set of context-free phrase structure rules, each of which is augmented with Lisp programs for its *test and action* as in ATNs. All the Lisp functions are generated automatically by the compiler from the constraint equations in the syn/sem grammar. Also note that those Lisp functions can be further compiled down to machine code (by the standard LISP compiler).

Once we have a grammar in this form, we can apply efficient context-free parsing algorithms. In fact, we subject this grammar to a final round of compilation, where the context-free rules are compiled into a

large augmented LR table for a generalized shift-reduce parser (Aho 1972) based on the Tomita algorithm (Tomita 1985). Whenever the parser reduces constituents into a higher-level nonterminal using a phrase structure rule, the Lisp program associated with the rule is simply evaluated. The Lisp program handles such tasks as:

- blocking partial parses that violate syntactic or semantic constraints (thus enforcing subject-verb agreement, type checking on the arguments to a proposed semantic relation, etc.),

- constructing a semantic representation of the input sentence from its constituent parts (an instantiated frame or causally related set of frames), and

- passing attribute values among constituents at different levels in order to have the information that is needed to perform the constraint-checking and frame-instantiation tasks.

The Tomita algorithm has two major advantages for real-time parsing over other methods:

- The algorithm is fast, due to the LR table precompilation; in several tests it has proven faster than any other general context-free parsing algorithms presently in practice. For instance, timings indicate a 5 to 10 fold speed advantage over Earley's algorithm (Earley 1970) in several experiments with different English grammars and various sample sets of sentences.[2]

- The efficiency of the algorithm is not affected by the size of its grammar, once the LR parsing table has been precomputed. This characteristic is especially important for our system, because the size of the syn/sem grammar may be very large in practical applications. Parsing time increases with the length and local ambiguity of the input. Fortunately, the semantic constraints precompiled into the grammar prevent ambiguities from propagating and thus long sentences may be parsed fairly efficiently.

The algorithm parses a sentence *on-line,* i.e., strictly from left to right and it starts the moment the user types the first word, without waiting for completion of the sentence. There are two main benefits from on-line parsing in interactive applications:

- The parser's response time can be reduced significantly. By the time the user finishes typing an entire sentence, most of the input has been already processed by the parser, as parsing speed surpasses normal typing speed.

- Any errors, such as mis-typing and ungrammatical usages, can be detected almost as soon as they occur, and the parser can warn the user immediately, and save much retyping and frustration when a severe typographical error at the start of a long sentence prevents further interpretation.

Applications such as real-time parsing, immediate translation of telex messages, and eventual integration with speech recognition and synthesis systems benefit substantially from on-line parsing, which is transparent when operating in batch-processing mode for long texts. A more detailed discussion of on-line parsing can be found in Chapter 7 of Tomita (Tomita 1985).

Because both the syntactic grammar and the syn/sem grammar are written in the same PATR-like notation, it is possible to test the system with a purely syntactic grammar without semantics (as illustrated by the dotted line in figure 3-2). Of course, in this configuration the system produces syntactic parse trees only, and cannot resolve ambiguities requiring semantic constraint unification.

[2] These timings were measured empirically over significant samples for average case performance. All context free parsers require $0(n^3)$ in the worst-case analysis, but in practice, for the type of grammars written for natural language, the Tomita algorithm is only slightly worse than linear.

(d) The Universal Generator

So far, we have mentioned only parsing, i.e. mapping a sentence into a semantic representation. However, generating a sentence from a semantic representation also requires the very same syntactic knowledge and domain semantic knowledge. The same philosophy of the universal parser applies to sentence generation: precompilation of elegant high level knowledge structures into low-level machine-efficient integrated grammars. We call the corresponding concept the *universal generator.* Both syntactic grammars and domain semantics in the universal parser are written in such a way that they are independent of their use, and therefore we can use the same knowledge base for both parsing and generation; grammar writers do not have to write two separate grammars, one for each task. To implement this concept, we require primarily another LFG compiler to produce an ACFG for the generator, as illustrated in figure 3-2. If all syntactic grammars have equivalent coverage, and the parser and the generator use the same domain semantics, then it is guaranteed that whenever the parser produces a semantic representation, the generator, will be able to render it in the target language. This is a good illustration of how the universal parser architecture can ensure that NLP systems will meet specification criteria.

[4] Implementation Notes

The first pilot integrated implementation of the universal parser was completed in September 1986 — demonstrating the computational feasibility of the concept. We are working on a second much more robust implementation, that incorporates additional capabilities and should yield better performance. The discussion here is based on results obtained from the first comprehensive implementation.

We have written a fairly comprehensive English syntactic grammar and Japanese syntactic grammar in LFG, each containing somewhat under 1,000 rules of grammar and regular morphology. The English grammar handles declaratives, imperatives, yes-no questions, "wh"-questions and other gapped constructions, auxiliary complexes and related phenomena. Additionally we built grammar rules for specialized constructions such as times and dates. The Japanese grammar corresponds roughly in coverage to the English grammar, in addition to having far more comprehensive morphological analysis rules in the same notation required for Japanese. Although these are perhaps the largest LFG-style grammars developed to date, they are still being refined and extended to achieve full syntactic and morphological coverage. For instance, we are currently improving the coverage of our grammars with respect to generalized subordinate and coordinate structures.

We have started grammar development for a third language, French, to make our system tri-lingual, and we expect to start development of similar grammars for at least one more language soon (Italian, Spanish, German, Arabic, or Russian). Note that with each additional grammar the universal parser architecture produces bidirectional translators between the new language and all previous ones. And we intend to use these syntactic grammars for multiple purposes in addition to real-time machine translation; natural language interfaces, text processing, speech recognition, etc.

We also developed a non-trivial domain semantic knowledge in FrameKit for certain classes of doctor-patient conversations, plus mapping rules and a corresponding lexicon including over 500 disease names. This domain was chosen as our test bed for developing the universal parser and generator architectures, and we are currently starting on a second domain. In the future we intend to tackle larger scale domains and complete terminological dictionaries.

All modules are programmed in Common Lisp and running on Symbolics 3600s, HP bobcats, and IBM RTs – the entire system should be portable to any other workstations running Common Lisp (Explorer, Micro Vax, Sun, etc.). The (source) Japanese grammar is about 90K bytes and the English grammar requires about 75K bytes at present. It takes a symbolics about an hour to compile grammars, mapping rules and domain knowledge into a fast run-time grammar (an

LR table and all the augmentation Lisp functions), producing a run-time grammar of about 1 Megabyte.

Given a compiled grammar, the run-time parser functions in a character-based mode, rather than word-based, making it possible to parse unsegmented sentences (sentences without any blank spaces between words, as typical in Japanese, and as required for any spoken language). The parser is quite fast; with the 1 Megabyte LR table (about 2000+ states), it takes on average only 20 to 30 milliseconds per character on a Symbolics 3600. This speed does not seem to be affected by the length of the sentence very much; 80 character long unsegmented Japanese sentences are still parsed in about 25 milliseconds per character. Thus, parse times of 1 to 3 seconds per sentence are typical.[3] Moreover, for fully segmented languages (such as English) where words rather than characters are the atomic units, parse times should be faster still. (We currently input English in character-based mode). Recalling the on-line parsing capability, where the system parses input as the user types interactively, all parsing appears instantaneous to the user.

The compiled run-time grammar for the generator requires about 700K bytes, although it could be represented in a more compact form. It takes a symbolics about a half hour to compile a run-time grammar for generator. The run-time generator takes about 1 to 3 seconds to generate a sentence from a semantic representation produced by the parser. The Japanese generator *is* capable of printing out Japanese characters on the screen. The Japanese parser, on the other hand, handles only a Roman-alphabet version of Japanese, for the time being. We are developing, jointly with Intelligent Technology Incorporation, a fully automatic Roman-alphabet to Japanese Kana-Kanji character converter based on syntactic and semantic knowledge (in the compiled grammar) in order to generate full Kanji characters unambiguously from romanized syllabic input typed in ordinary keyboards.

[3] Recall that by "parsing" we include full semantic interpretation and structural disambiguation, as well as syntactic analysis.

[5] Discussion and Future Work

Reiterating our results thus far, we have designed and tested the universal parser (and universal generator) architecture for real-time knowledge-based machine translation across multiple languages and domains. Tests thus far have proven encouraging with respect to the generality of the approach, and its ability through pre-compilation to produce real-time translation systems requiring no post-editing. Thus interactive, multilingual, semantically-accurate translation has been demonstrated feasible. With respect to syntactic coverage, we have developed comprehensive English and Japanese LFG grammars, handling the bulk of all desired syntactic and morphological constructions. These will be extended, but we have already written well over half of the total expected number of grammar rules for English and Japanese.

However, although we have every reason to believe in the extensibility of the approach to handle multiple languages, and to handle much larger domains and lexicons, we have yet to demonstrate scaling up beyond two languages and beyond a thousand or so word lexicon. This is very much work in progress. In order to facilitate the scaling up process, we are developing dictionary, grammar and knowledge-base tools to facilitate development and maintenance, and to ensure a significant degree of internal consistency and uniformity. Additionally, we are re-implementing and extending parts of our universal parser architecture in order to make it sufficiently robust to distribute to other research teams, and eventually to end users.

The two initial design objectives we have not yet addressed are handling discourse phenomena and integration with speech recognition and synthesis. Our main research activity at present lies in the area of discourse, as our initial system operates only on a sentential basis. First, we intend to borrow the successful case-frame ellipsis resolution methods developed recently in XCALIBUR (Carbonell 1985), Language

Craft (Carnegie 1985), and PSLI-3 (Frederking 1987), and integrate them into the universal parser architecture. These methods rely primarily on case-frame semantics and on functional properties of the syntax. Second, we expect to work on extending and applying the embryonic work on practical anaphora resolution in XCALIBUR and work on handling metalinguistic utterances (Carbonell 1982). Third, we will focus attention on default inference processes to fill in implicit information lacking in the source text, but required for accurate translation. Such information includes subjects in Japanese, which are optional when inferable from context, but which must be stated explicitly in translating to English. At present we utilize a handful of ad-hoc rules to supply default subjects, levels of politeness, etc., but a more principled and systematic approach is required. Fortunately, the universal parser architecture provides an ideal computational framework into which new knowledge sources may be introduced. And, the knowledge-based translation task provides copious and severe empirical tests for our theoretically-inspired ideas and methods.

6. References

Aho, A. V. and Ullman, J. D. : *The Theory of Parsing, Translation and Compiling.* Prentice-Hall, Englewood Cliffs, N.J., 1972.

Boitet, C. : Problèmes actuels en TA: Un essai de réponse. In *Proc. 6th International Conference on Computational Linguistics.* Ottawa, Canada, 1976.

Bresnan, J. and Kaplan, R. : Lexical-Functional Grammar: A Formal System for Grammatical Representation. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, Massachusetts, 1982,pages 173-281

Carbonell, J. G., Cullingford, R. E. and Gershman A. G.: Steps Towards Knowledge-Based Machine Translation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-3(4), July, 1981

Carbonell, J. G. and Joseph, R. : The FrameKit+ Reference Manual.1985. CMU Computer Science Department internal paper.

Carbonell, J. G. and Tomita, M. : Knowledge-Based Machine Translation, The CMU Approach. In Nirenburg, S. (editor), *Machine Translation: Theoretical and Methodological Issues.* Cambridge, U. Press, 1986.

Carbonell, J. G. Boggs, W. M., Mauldin, M. L. and Anick, P. G.: The XCALIBUR Project, A Natural Language Interface to Expert Systems and Data Bases. In S. Andriole (editor), *Applications in Artificial Intelligence.* Petrocelli Books Inc., 1985.

Carbonell, J. G.: *Meta-Language Utterances in Purposive Discourse.* Technical Report, Carnegie-Mellon University, Computer Science Department, 1982.

Earley J.: An Efficient Context-free Parsing Algorithm. *Communication of ACM* 6(8): 94-102, February, 1970

Frederking, R. E : *Natural Language Dialog in an Integrated Computational Model.* PhD thesis, Carnegie-Mellon University, Computer Science Department, 1987.

Hayes, P. J., Hauptmann, A. G., Carbonell, J. G. and Tomita, M. : Parsing Spoken Language: A Semantic Caseframe Approach. In *11th International Conference on Computational Linguistics (COLING86).* Bonn, West Germany, August, 1986.

Kay, M. : Functional Grammar. In *Fifth Annual Meeting of the Berkeley Linguistic-Society,* pages pp. 142-158. Berkeley Linguistic Society, MIT Press, Berkeley, California, February, 1979.

Kay, M. : Functional Unification Grammar: A Formalism for Machine Translation. In *10th International Conference on Computational Linguistics,* pages 75-78. Stanford, July, 1984.

Kittredge, R., Bourbeau, L. and Isabelle, P.: Design and Implementation of an English-French Transfer Grammar. In *Proceedings of the 6th International Conference on Computational Linguistics.* Ottawa, Canada, 1976.

Carnegie Group Inc.: *The Language Craft Reference Manual.* Pittsburgh, PA, 1985.

Lytinen, S. : *The Organization of Knowledge in a Multi-lingual, Integrated Parser.* PhD thesis, Yale University, 1984.

Nagao, M., Nishida, T. and Tsujii, J. : Dealing with Incompleteness of Linguistic Knowledge in Language Translation - Transfer and Generation Stage of the MU Machine Translation Project. In *Proceedings of the 10th International Conference on Computational Linguistics.* Stanford, CA, 1984.

Nirenburg, S., Raskin, V and Tucker, A. : On Knowledge-based Machine Translation. In *Proceeding of the 11th International Conference on Computational Linguistics,* pages 39-51. COLING86, Bonn, West Germany, August, 1986.

Pereira, F. C. N.: A Structure-Sharing Representation for Unification-Based Grammar Formalisms. In *23rd Annual Meeting of the Association for Computational Linguistics, pages* 137-144. Chicago, July, 1985.

Roesner, D. : The generation system of the SEMSYn project: Towards a task-independent generator for German. In *1st European Workshop on Language Generation.* Paris, January, 1987.

Shieber, S. M. : Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms. In *23rd Annual Meeting of the Association for Computational Linguistics,* pages 145-152. Chicago, July, 1985

Slocum,J.: Machine Translation: its History, Current Status, and Future Prospects. In *Proceedings of the 10th International Conference on Computational Linguistics.* Stanford, CA, 1984.

Tomita, M. : *Efficient Parsing for Natural Language: A Fast algorithm for Practical Systems.* Kluwer Academic Publishers, Boston, MA, 1985.

Tomita, M. : An Efficient Context-free Parsing Algorithm for Natural Languages. In *9th International Joint Conference on Artificial Intelligence (IJCAI85).* August, 1985.

Tomita, M. and Carbonell, J. G. : Another Stride Towards Knowledge Based Machine Translation. In *11th International Conference on Computational Linguistics (COLING86).* Bonn, West Germany, August, 1986.

Woods, W. A. : Transition Network Grammars for Natural Language Analysis. *CACM* 13: pp. 591-606, 1970.