

The Logos Model: Principles and Motivations Underlying the OpenLogos MT System

Bernard (Bud) Scott
AMDG
Chief Architect, Logos System

Keywords: rule-based MT, semantico-syntactic abstraction language (SAL), associative semantics, mental model, pipeline architecture, human sentence processing, FAHQT

Abstract.

The Logos Model underlying OpenLogos is described. The Logos Model is characterized with respect to four fundamental issues: (1) how natural language is to be represented; (2) how linguistic knowledge is to be stored, (3) how this knowledge store is to be applied to the input stream, (4) how complexity effects are to be dealt with as the knowledge store grows, year after year, in the quest for fully automatic, high-quality translation (FAHQT). The Model reflects principles derived from assumptions about human sentence processing, which are described. Using the metaphor of a biological neural net, a complex, 57-word sentence is tracked as it proceeds along a pipeline architecture, simulating an hypothesized human model. Limitations of the Logos Model are also discussed.

1.0 HISTORICAL BACKGROUND

Logos Corporation and the Logos Machine Translation System came into being in 1970, in response to a sudden national requirement to translate massive quantities of U.S. military manuals into Vietnamese. This requirement was triggered by a presidential decision to turn the materiel and conduct of the war over to the South Vietnamese, and was compounded by the critical lack of human translation resources to implement this new policy. This all took place only a few years after ALPAC, and officials of the US Government were understandably skeptical of help from machine translation. But when our newly formed Logos Corporation insisted it could build a machine translation system able to address this requirement, we were given a chance to prove ourselves. Feasibility was to entail an ability to machine translate some twenty pages of a previously unseen helicopter manual. We were given three months to prepare for this.

The results of the trial were judged sufficiently promising and the Company was awarded an emergency contract to develop a full-scale, production system. This effort proved quite successful. In his annual report for 1972, Dr. John Foster, Director of Defense Research and Engineering (DDR&E), stated that the Logos System had now “established the feasibility of large-scale machine translation”¹. This was the first positive word accorded MT since the advent of the ALPAC winter six years earlier.

1.0 Design Objectives of the Logos Model

From the outset, Logos developers elected to build a general-purpose system that could be used for any language combination. Major components of the Model are shown in Fig. 1. Fig. 2 shows the model’s pipeline architecture.

2.1 ARCHITECTURAL DESIGN

Logos Model implementation was guided by the following design objectives:

- Language-Neutral Software: A physically common, language-neutral body of software serves all language combinations (except for language-specific I/O functions). Apart from I/O, all language-specific operations (morphological, syntactic, semantic) are accomplished by means of tables (for morphology) and rules (for everything else), all in the form of data.

- ❑ Software Modularity: The system is highly modular. A fully implemented, operational version of the model, running in megabytes of main memory, has also been implemented in memory environments as small as 64K (1981 implementation on a Wang OIS microchip).
- ❑ Declarativeness: Semantico-syntactic representation of an input sentence is passed down a pipeline (Fig. 2) where the symbolic input string itself now drives rule base interaction. (Input stream and rules are expressed in the same symbolic representation language). How input stream and rule base interact is a fundamental, characterizing aspect of the Logos Model (See 4.3).
- ❑ Multi-target Functionality: New targets (any number) are added to an existing source by linking target data modules (morphological tables, lexicon, rule bases). No new programming is required (except for target-specific I/O).

Extendibility and Improvability: The system is open-ended, designed to absorb endless extensions and improvements. Two considerations were paramount: (1) there should be no inherent risk of logic saturation as the knowledge base grows in size, leading to developmental stasis; (2) nothing in the design should preclude fully automatic, high quality translation (FAHQT) long-term. In sum, translation quality shortfall should be attributable to knowledge base deficiencies, never to design decisions.

2.2 FAHQT

The long-term objective of FAHQT, of course, can only apply to *discursive* texts, i.e., texts intended for information transfer. And such texts must be reasonably well written. Texts written for edification, where style is paramount, thus are ruled out. Fig. 3 illustrates the areas of written language where the Logos Model (and MT in general) can expect cost-effective application. (Quantitative indications are pure estimates.)

FAHQT remains a far-off goal for Logos developers, and the fact that FAHQT has not been reached after years of effort indicates its remoteness, although not its theoretical unattainability (for discursive texts). True, where source text is especially well written and preparatory terminology work is complete, output has sometimes already approached near-human quality. But generally, the number of linguistic situations that require attention to achieve such quality, and that have yet to be attended to, seems endless. A key question then poses itself, viz., whether an MT system could be designed to accommodate the endless growth in knowledge implied in any quest for FAHQT.

(See Figures on following pages)

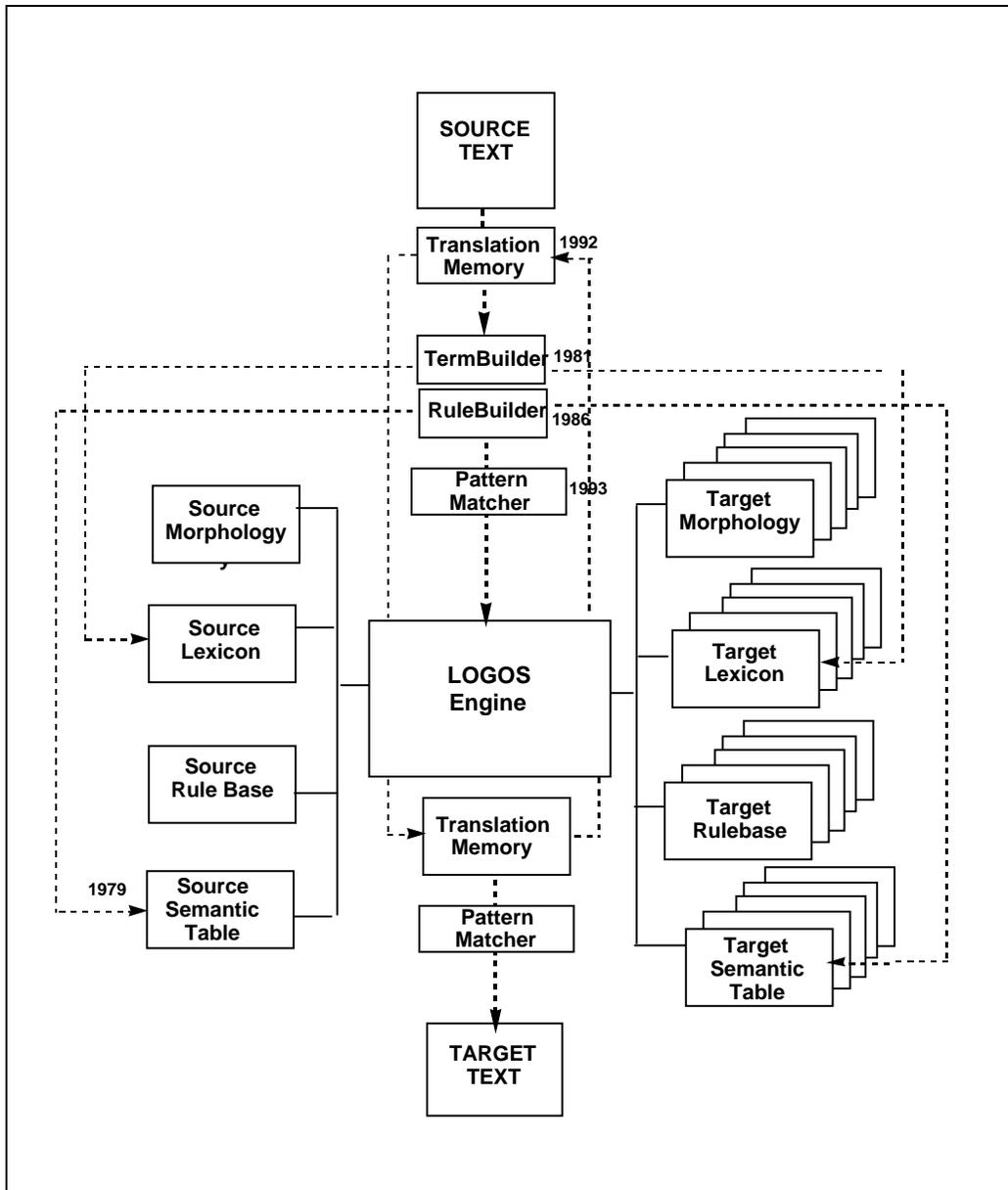


Fig. 1 - Major components of the complete Logos Model. (Undated elements were part of the original 1971 system.) Translation process is as follows: (1) Language-neutral software Engine presents NL input stream to lexicon, converting NL string into a symbolic string. (2) Engine then seeks to match symbolic string with symbolic data patterns in rule bases. (Symbols are semantico-syntactic.) (3) Upon match, software engine interprets action portion of fired rule, driving progressive source analysis in bottom-up manner. Notations pertinent to target equivalences are recorded as analysis of each source constituent is completed, in contrastive linguistic (tree-to-tree) fashion. (4) Target is generated upon completion of source analysis. Steps 2 through 4 are accomplished *incrementally* over a cascade of modules called a pipeline (See Fig. 2). Source and target lexicons are actually integrated, as are morphology tables. **TermBuilder** and **RuleBuilder** are developer/user tools. **Semantic Tables** comprise deep-structure rules that (i) support deterministic parsing and (ii) effect *context-sensitive* target transfers. **Pattern Matcher** allows users to effect global string edits in source input or target output via Regular Expressions. System has been interfaced with several leading TM products. New targets can be added to an existing source language by linking target data modules (for morphology, lexicon and semantico-syntactic rule bases). (N.B. Not all elements, e.g., **RuleBuilder**, are available in the current version of **OpenLogos**).

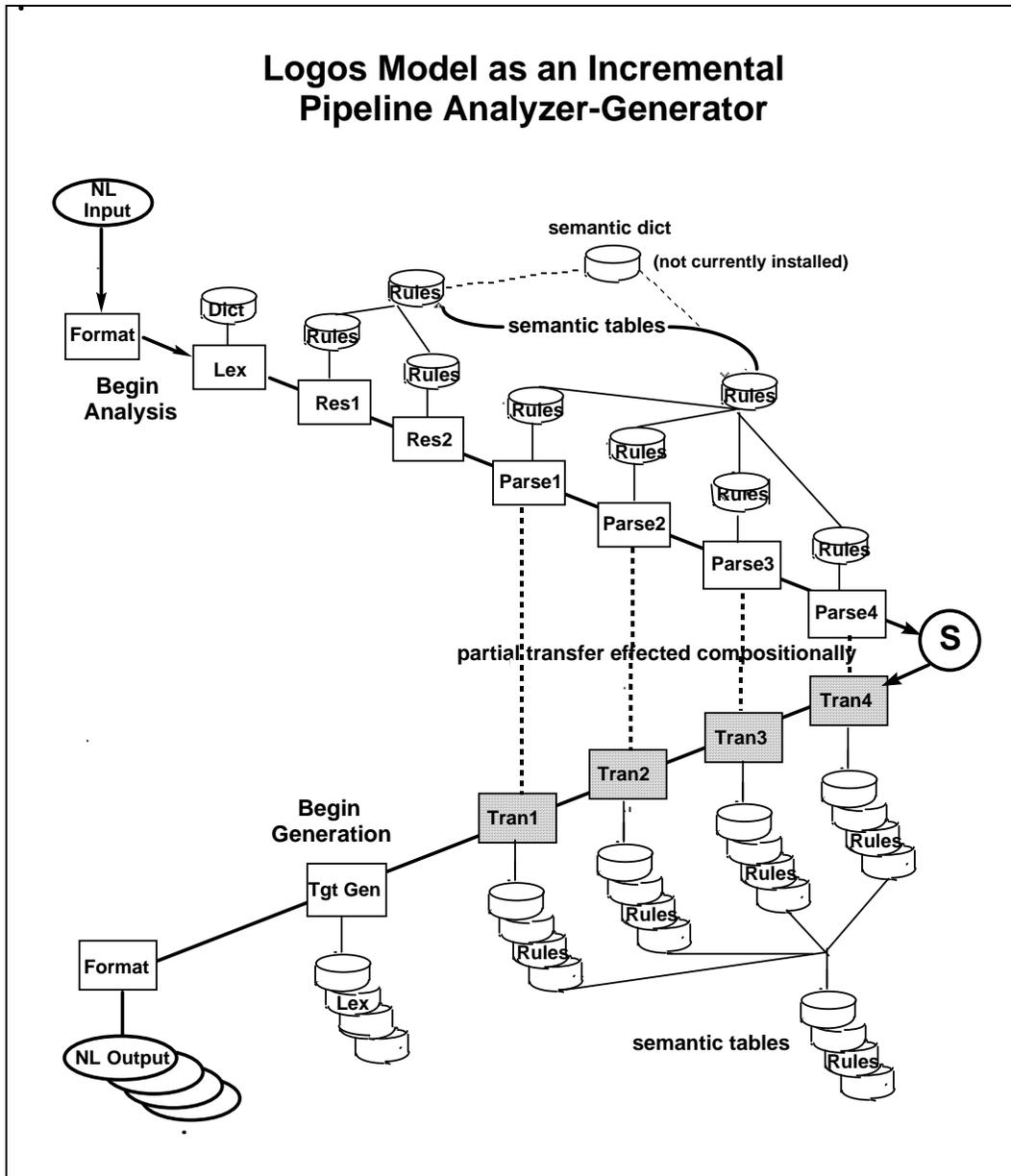
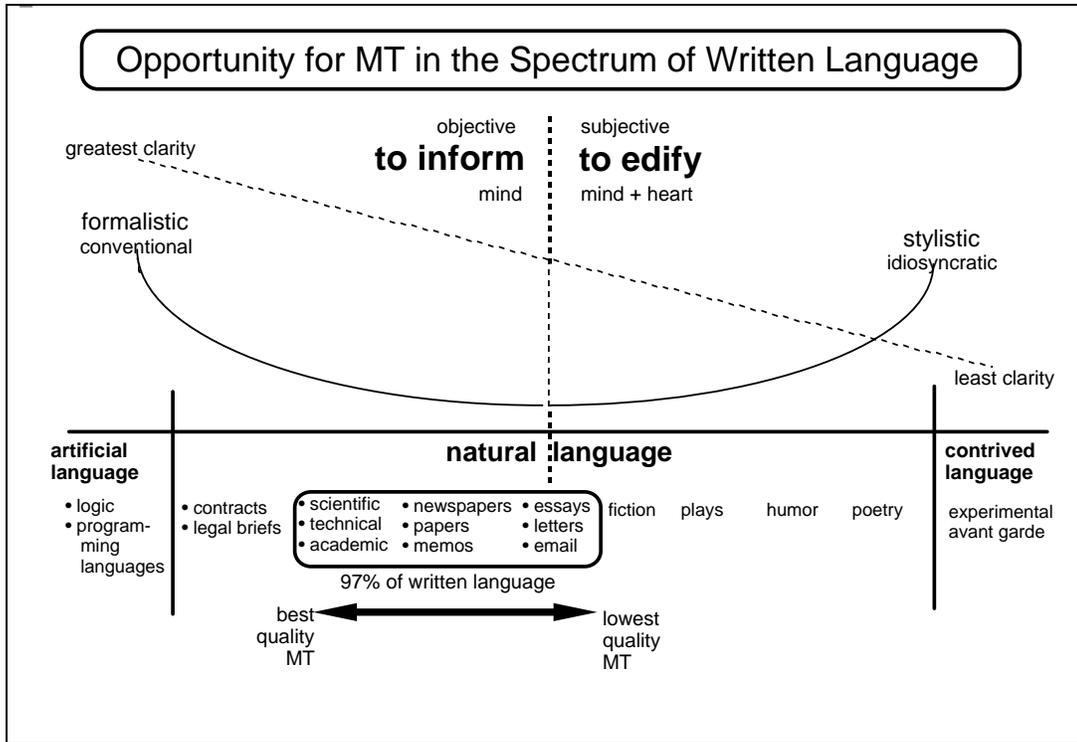


Fig. 2 – Incremental, Pipeline Architecture of the Logos Model. Architecture of the Logos Model resembles a pipeline. (The 1971 system had only **RES1** and the first two **Parse/Tran** modules) NL text enters at the top where formatting is analyzed and stripped out, and sentence boundaries identified. For each sentence, NL string is next converted to a symbolic, semantico-syntactic string via lexical substitution. Symbolic string passes down pipeline and interacts with rule bases, effecting a single, bottom-up parse. Rules consist of semantico-syntactic patterns which, when matching some portion of symbolic input stream, become active and compete for right to fire. Target transfer is accomplished as a tree-to-tree equivalencing at four parse-tree levels, reflecting an incremental, compositional approach. Level 1: **Parse1/Tran1** analyzes and transfers simple **NPs**. Level 2: compound **NP's** and **NP** complementation. Level 3: intra-clausal structures, esp. predicate/argument analysis. Level 4: inter-clausal relationships. Process is not purely deterministic in that early parsing/transfer decisions can be modified at higher levels (within strict limits).



© 2002 Bernard E. Scott

Fig. 3 - Written Language Bandwidth Suitable for Effective MT. MT potentially applies to a broad spectrum of written language as suggested in the above figure. Translation quality will always tend to be better in texts with a natural but conventionalized writing style and will become progressively less so as idiosyncratic content increases. Successful MT necessarily presupposes well-written text and diligent user maintenance of the lexicon

3.0 Addressing the Problems

Someone has aptly said that natural language is the most complex event in the universe. What makes for this complexity of course is NL's fuzzy richness, an immense, open-ended composite of elements much of which have no univocal signification or grammatical function except as provisionally granted by the context, which context itself shares these same characteristics. NL's complexity and ambiguity therefore pose a daunting challenge for a binary, yes/no machine, as ALPAC rightly saw and every developer knows full well. Moreover, in a computer, the complexity and ambiguity of NL interact in such a way that any attempt to cope with the one normally exacerbates the other. For example, add information to cope with ambiguity and you increase complexity. Relieve complexity by lessening this information load and you weaken the power to disambiguate. This circumstance forms the horns of a true computational dilemma for the developer.

3.1 THE COMPLEXITY PROBLEM

Despite the fundamental importance of linguistics, the computational problem posed by complexity was deemed to be the more critical, the more telling issue--how to cope with complexity effects that must inevitably arise in unconstrained, general-purpose MT. And by complexity here is meant not so much the classic difficulties of complexity theory relating to compute time and space requirements, but rather what one may call *cognitive complexity*, complexity relating to the difficulties humans experience in maintaining logic in maturing systems as that logic becomes increasingly more complex.

Cognitive complexity will not be obvious at the prototype stage, and many MT projects have been undertaken without providing for complexity. But complexity effects become a major headache when scaling up to a working system, where one must now deal not just with the regularities of a language, but with its countless exceptions, and exceptions to exceptions, all in endless quantity. The typical experience of the developer of an English parser, for example, goes like this: generalized logic introduced to resolve a N/V homograph in one context backfires in another, requiring refinements to the knowledge base (logic). This refinement in turn now unwittingly undoes a resolution of an entirely unforeseen kind elsewhere, something that used to work and now does not. After a great deal more of this, the logic and/or strategies involved may become so complex as to become virtually unmanageable, i.e. unimprovable. The old adage about the developer who bends over to pick up one marble and finds that in the process he or she has dropped two well illustrates cognitive complexity and the developmental *stasis* to which it tends. All seasoned developers recognize this problem.

Foreseeing this in some dim way, our approach to machine translation would be defensive from the very beginning, driven not so much by a desire to instantiate theory as to avoid a developmental *cul-de-sac* by whatever principled means it took. In sum, what was needed was a computational approach that would effectively minimize complexity effects. As we saw it, the attainability of industrial strength MT hinged precisely on this.

3.2 THE LACK OF PROVEN MODELS

There were no proven translation models to guide our efforts when we began this work some thirty years ago. The only proven translation system in sight was the human brain, about which little was understood. Chomsky's view of human language processing was well known but his "syntactocentrism" eschewed semantics (Jackendorf, 2002) and it was abundantly clear, after ALPAC, that without semantics, MT had little prospect of success (Hutchins, 1986). It is easy to understand why language was initially approached in terms of syntax alone: language *qua* syntax was amenable to systematic analysis in ways that semantics was not, as people like Harris (1968) and Chomsky (1965) well understood in those days. Yet, as Chomsky (1957) famously noted, children easily master language, and (to stress what Chomsky more or less elected to ignore) children do so despite language's tangled web of irregularities and ambiguities. It seemed to us then that we would do well to examine whether something from human sentence processing could be inferred and then imitated in the computer.

3.3 MENTAL MODEL

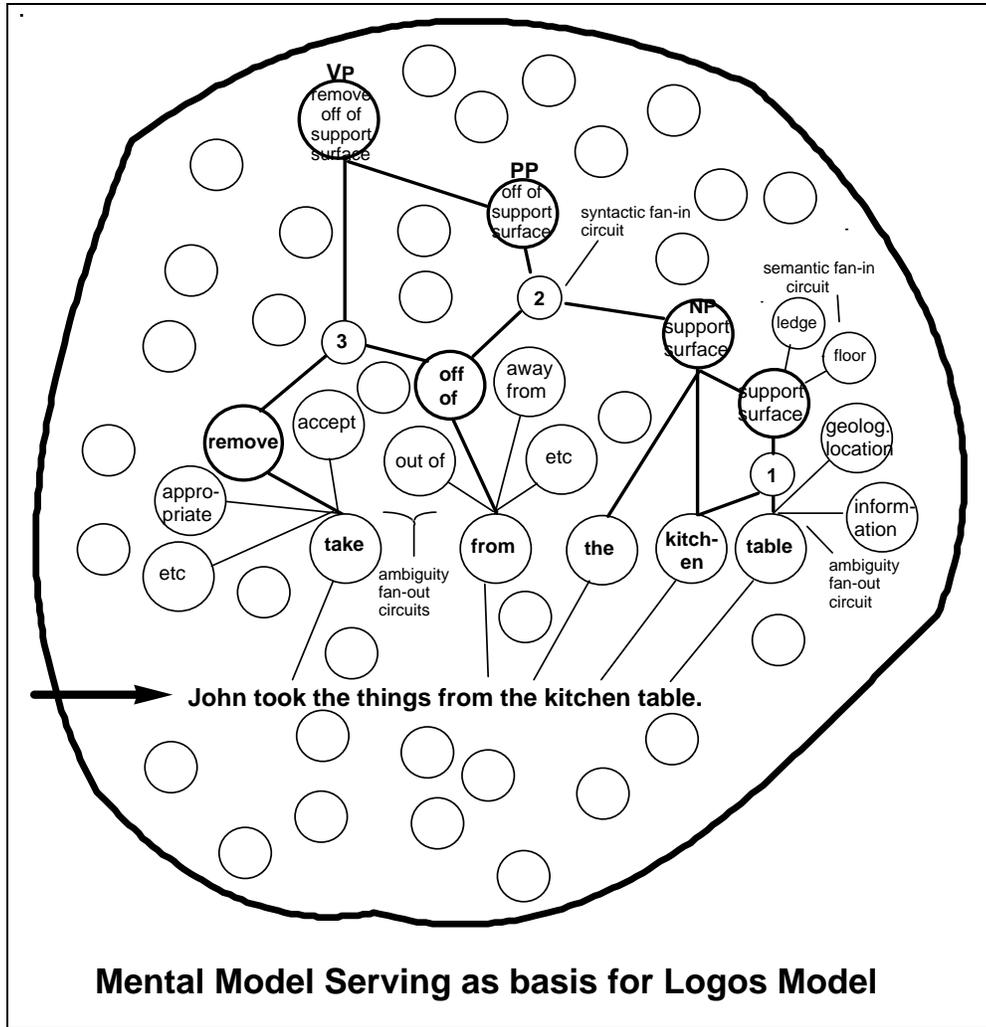
We proceeded to make certain assumptions about human sentence processing and, in particular, how the brain manages to cope so well with natural language complexity. These assumptions were purely intuitive and can be considered pre-scientific, but they nevertheless proved fundamental to the design of the Logos Model with beneficial effect.

Key properties of the hypothesized mental model are summarized here, along with their implications for the Logos Model (See Fig. 4 for graphic depiction):

- Opportunistic, non-algorithmic processing. The language processor in the brain is an algorithm-free, associative memory network. The process does not rely on some sort of homunculus or supervisory logic controlling processes and decisions along the way. Nor is this mental process anything like the analysis performed by transition networks designed to prove well-formedness, and therefore where only explicitly specified decision branches are permitted. Rather, analysis simply emerges in unpredictable ways from stored memory associations reacting opportunistically to input signals, well-formed or otherwise. The process is controlled by the input stream itself, as the brain reacts to input and seeks to assimilate its import. In effect, then, sentences themselves constitute the controlling algorithm. The principal implication for the Logos Model is that associative memory networks, being non-algorithmic in this sense, would be relatively

immune to logic saturation as developers expand language coverage. (See Sections 4.2 and 4.3 and Section 7 for fuller discussion.)

- Incremental processing. Sentence processing (analysis) is done in incremental stages across a series of modules, very loosely analogous to the brain's visual pathway. This assumption translated into a pipeline architecture whereby compositional parsing decisions can be made "as early as possible and as late as necessary," depending on the circumstances (Scott, 1989). (See Fig. 2.)
- Integration of syntax and semantics. Syntax and semantics constitute a representational *continuum*. Because human sentence analysis, we felt, had to be deterministic, producing a single analysis (even if sometimes initially a wrong one), both syntactic and semantic information had to be available at every decision point along the linguistic pathway, contrary to the "syntactocentrism" of the generative school (and their resultant parse forests). For the Logos Model this assumption led to the creation of an ontology-based, semantico-syntactic representation language exhibiting such integration. (See Section 4.1)
- Use of abstraction. Abstraction is a principal means employed for controlling complexity, i.e., for keeping things simple. It is arguable, we felt, that abstraction is fundamental to brain function. This at least is suggested by the very structure of the brain cell--a neuron with many dendrites for input and typically a single axon (with collaterals) for output. The prevalence of fan-in circuitry in the brain further suggests a structure designed for abstraction. For the Logos Model, this required that the semantico-syntactic representation language we created consist of *second-order* abstractions (natural language being deemed to comprise first-order abstractions.) (See Figs. 5-7.)



© 2003 Bernard E. Scott

Fig. 4 - Mental Model Assumptions, as Basis for the Logos Model.

Key assumptions about human sentence processing (from a neurophysiological perspective) underlay the design of the Logos Model. Chief among these are:

- processor is associative memory itself, its organization and interconnections. Analysis is effected by memory associations (rather than controlling decision logic), responding to input signals in opportunistic fashion.
- syntactic and semantic aspects of language are integrated (as in a continuum).
- language is treated as *second-order abstractions* (e.g., *table* qua 'support surface'). Abstraction (via fan-in circuits) was thought to be the principal means used by the brain to reduce complexity, illustrating nature's law of least effort.
- memory is *content-addressable* (blank cells suggest the vast number of cells that are not made active by input signals shown). This is understood to explain why increase of stored knowledge does not have complexity effects in humans.

Stage 1 employs associative semantics to resolve the meaning of *table* (collocation with *kitchen.*). In Stage 2, *table* qua 'support surface,' in turn, allows process to resolve meaning of the preposition *from* to 'off of.' In Stage 3, the association of the verb *take* with the preposition 'off of' provides for that verb's resolution to *remove*. Concatenation via fan-in circuitry produces a more abstract (i.e. simpler) representation of the expression. Many operations, of course, are not accounted for here (morphological analysis, homograph resolution, etc.). The prevalence of fan-in and fan-out circuitry in the brain was deemed to corroborate these analysis (fan-out) and abstraction (fan-in) functions, as illustrated.

- Content-addressable memory. Information stored in memory is *content-addressable*. This well-established property of neurophysiology (McClelland, Rumelhart, and Hinton, 1986) explains why, out of the brain's billions of cells, relatively few cells are ever activated by any given stimulus. Such specificity invited new thinking as to how linguistic knowledge is to be stored and applied in an information-rich MT system, and this new thinking became a key factor in shaping the design of the Logos Model. (See Section 4.3.2)

4.0 Fundamental Design Decisions

Translating this hypothesized mental model into a model for machine translation entailed decisions regarding four fundamental design questions:

- How do you *represent* natural language internally to the computer?
- How do you *store* that knowledge?
- How do you *apply* that knowledge to the input stream
- How do you deal with *complexity* issues?

4.1 HOW DO YOU REPRESENT NATURAL LANGUAGE?

Semantics for MT and for NLP in general was as yet largely uncharted territory and it was far from clear in those days how to characterize the meaning of words to a machine. But the ability to deal with meaning was imperative: ALPAC attributed the failure of MT to the "semantic barrier" that all MT systems had thus far run up against (Yngve, 1964). Two routes of exploration lay open: one could devise a list of semantic primitives and apply them to words as appropriate, or one could develop a semantic taxonomy (ontology), in which case NL words are mapped into abstract semantic entities. We elected the latter option.

Logos developers chose to build a taxonomy but one that was *semantico-syntactic* rather than purely semantic. What we were looking for in words were semantic features that had syntactic implications, i.e., the point in words where semantics and syntax seemed to intersect. This decision was motivated in part by our intention to build a *deterministic* parser (where a single parse is produced). To do this, one had somehow to integrate semantics and syntax so that both would be available to essentially irreversible parsing decisions at every stage of analysis.

The resulting semantico-syntactic representation language, developed in the 1970's, was built inductively from analysis of countless output errors and consideration of what the computer needed to know in order to avoid such errors. We dubbed the language SAL (semantico-syntactic abstraction language), an acronym actually suggested to the author by Jaime Carbonell upon his review of the effort. The language is open-ended but has continued to evolve in only minor ways since the 70's. (For examples of the SAL taxonomy, see Figs. 5-7 and Appendix A.)

4.1.1 Semantico-Syntactic Abstraction Language (SAL)

A number of special considerations motivated SAL.

- SAL was to look and function like a natural language (at a more abstract level) such that any NL string could be readily expressed by an equivalent SAL string, thus enabling developers and users alike to map easily from NL to SAL in lexical work. For example, the noun *highchair* maps to the SAL subset 'support surface.' This subset, of course, inherits its superordinate classifications ('functional device' for set, and 'concrete noun' for superset).
- Semantics and syntax were to be seen as a continuum from literal string to word class. For example, the input string *highchair* could be dealt with during pipeline analysis at any of the following representational levels:
 - Literal level: *highchair*
 - Head morpheme: *chair*
 - SAL subset: **COsupp** ('concrete noun', 'support surface')

- All SAL elements are characterized by the triplet: WC(Type; Form), where Type is expressed as one of three levels (semantico-syntactic superset, set, and in many cases, subset). Figs. 5-7 show Type codes for several of the noun supersets. Appendix A shows Type codes for the entire adjective word class.
- SAL verbs, deverbal nouns, and certain of the SAL adjectives (see Appendix A) have set or subset codes that denote governance. For example, the NL deverbals *egress*, *deliverance*, *rescue*, *absence* are all subsumed under a SAL deverbal element that signifies *from* governance. Similarly, NL verbs like *furnish*, *supply*, *provide*, are represented by a SAL di-transitive verb element that governs the set of argument structures: *furnish x with y = furnish y to x = furnish x y*. Although developed purely inductively, SAL bears obvious kinship here to Valency and Case grammars.

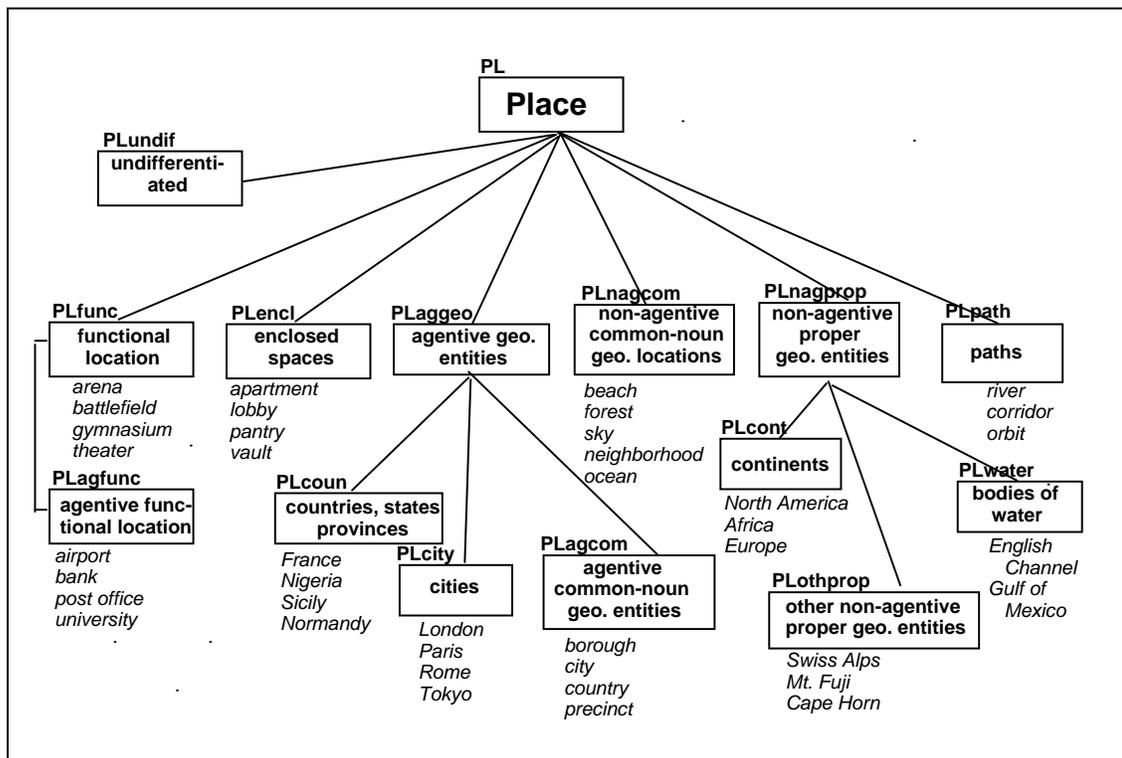


Fig. 6 – SAL Noun Superset: Place. Note the role of *agentiveness* here. Agentiveness can sometimes have a bearing on resolution of noun/verb homographs in English. For example, the string *the beach plans*, because *beach* is non-agentive, would bias analysis away from *plans* as a verb, unlike the string *the city plans*. In the expression *from the pantry*, the SAL code for *pantry* as ‘enclosed space’ allows sense resolution of the preposition *from* to ‘out of.’ SAL differentiation between ‘countries’ and ‘cities’ supports needs of some targets, e.g., French, when transferring the English preposition *in* before such ‘place’ nouns:
in N(PLcity) → à N(PLcity), as in à Paris; *in* N(PLcoun) → en N(PLcoun), as in en France.

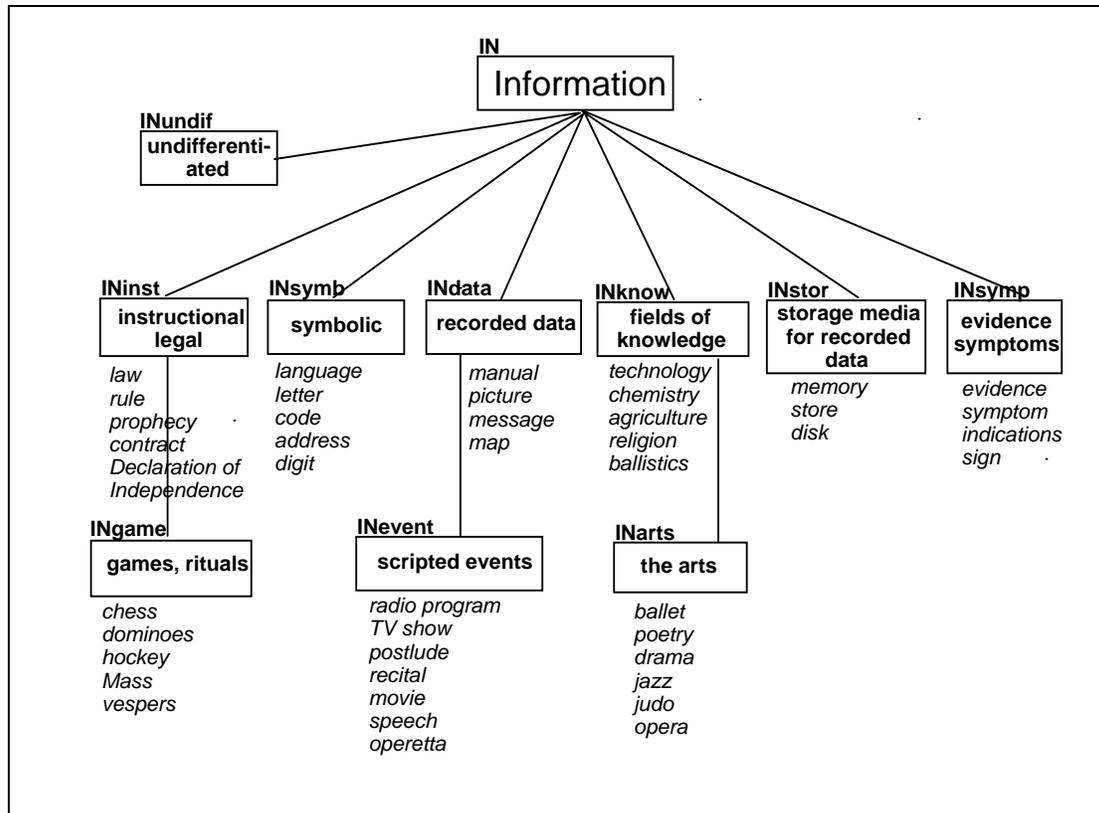


Fig. 7 - SAL Noun Superset: Information. The ‘information’ noun superset comprises a broad class of nouns that denote information, knowledge, symbols, data, rules, games, rituals and other such concepts where communication is a primary implication of the noun’s meaning (excluding deverbal nouns). This noun superset also includes the storage medium or format in which informative expression is recorded, represented, or communicated. Certain of these ‘information’ Type nouns (e.g., INdata) followed by the preposition *on* generally cause this preposition to have the meaning of ‘concerning,’ provided the noun following the preposition is not a ‘support surface’ Type noun (e.g., *a book on the presidency* versus *a book on the shelf*).

- ❑ SAL was developed as a numeric language and remains so internally, although mnemonics are available to the developer and are used in this paper.
- ❑ SAL is not a true metalanguage or interlingua but does approximate one in certain of its word classes. SAL noun elements, for example, are common to all languages. This is only slightly less so the case with pronouns, determiners, prepositions, adverbs and conjunctions. On the other hand, SAL verbs and adjectives are interlingual only at the superset level.

Thus far, SAL has been fully developed only for English and German. The complete SAL taxonomy for English adjectives is given in Appendix A.

4.1.2 Effectiveness of SAL for Deterministic Parsing

Because of its levels of abstraction, SAL allows developers to deal with meaning in a relatively manageable way. In the following examples, we illustrate how SAL is used to resolve (with varying degrees of success) certain classic parsing problems where English is the source language. Raw, unedited output is provided to show SAL’s effectiveness in resolving structural

ambiguities. (You are reminded that the source analysis being conducted here is common to both the French and German target translations shown here.)

Case One

SAL codes are used in analyzing syntactic patterns like ‘N *and/or* N N’, as in (1a) and (1b), below. Effects of analysis are illustrated in the unedited French and German output which follows:

- (1a) *I like the ham and cheese sandwiches.*
- (1a’) *J’aime les sandwichs de jambon et de fromage.*
- (1a’’) *Mir gefallen Schinken- und Käsestullen.*
- (1b) *I never go to that bank or TV store.*
- (1b’) *Je ne vais jamais à cette banque ou à ce magasin télévision.*
- (1b’’) *Ich gehe nie zu jener Bank oder Fernseh Lager.*

In (1a), unlike (1b), *ham* and *cheese* are both members of the same ‘edible’ noun set (**MAedib**) under the ‘mass’ noun superset (**MA**). The semantic homogeneity between **N1** and **N2** allows a generalized **Parse1** rule, comprising the SAL pattern ‘**N(X;SG) CJ(CRD;u) N(X;SG) * N(u;u)**’, to see both *ham* and *cheese* as attributive to **N3** (*sandwiches*) and thus to concatenate the string as **NP**. (The **X** signifies any SAL Type shared between **N1** and **N2**. The **u** in **N3** signifies universal Form in the case of **N3**, or non-relevance in **CJ**. **SG** signifies singular morphology. The ***** in the SAL pattern is a constrained Kleene star.) In writing rules like that applying to (1a), the developer specifies whether the semantic commonality constraint associated with **X** must be at the SAL superset, set, or subset level. Because SAL’s semantic granularity is fairly coarse, its effectiveness in these situations is limited; more consistently correct parsing results await a much finer-grained taxonomy. (A new semantic taxonomy is currently under consideration for this purpose, and if installed will be supplemental to the present semantico-syntactic SAL taxonomy (See Section 8).)

Case Two

An important noun property captured via SAL classification is *agentiveness*. The following will illustrate its use in typical **Parse1** parsing (shown with unedited French and German output, any defects of which are unrelated to source analysis):

- (2a) *corn eating insects*
- (2b) *insects eating corn*
- (2a’) *Les insectes qui mangent le maïs*
- (2b’) *Les insectes qui mangent du maïs*
- (2a’’) *Maisessen-Insekten*
- (2b’’) *Insekten, die Mais fraßen*

In (2a), the **ING** form is bounded on the left by a non-agentive (*corn* = **MAedib**, ‘edible’ Type ‘mass’ noun), on the right by an agentive (*insects* = **ANbugs**, ‘bugs’ Type ‘animate’ noun). Constraints for the rule that handles the pattern in (2a) require that **N1** have a non-agentive SAL classification and that **N2** have an agentive SAL classification. All constraints being satisfied, the rule will parse the SAL pattern as **NP**. The rule cannot apply to (2b), causing the **ING N** string to be seen by other rules as an elided relative clause. (Note, in the unedited French translations (2a’) and (2b’), how the analysis has allowed the target to see (2a) and (2b) as semantic near equivalents.)

Case Three

In the following, we illustrate typical use of SAL classifications for resolving ambiguities involving verbal elements. The ambiguity here concerns the attachment of the participial phrases *effected/affected by digitalis*. (The unedited French translations shown are raw output from an earlier release of the Logos System. (For uninvestigated reasons, the **Parse3** rule effecting this analysis has been removed in the current release.) Again, any flaws in translation are unrelated to analysis.

- (3a) *changes in tissue effected by digitalis.*
- (3b) *changes in tissue affected by digitalis.*
- (3a') *changements de tissu effectués par la digitaline.*
- (3b') *changements de tissu affecté par la digitaline.*

In (3a) the verb *effect* belongs to a SAL Type set that *must* have as object a noun deverbal (process noun). The noun *changes* is a process noun in SAL and alone satisfies this condition in the given pattern, thus causing *effected by digitalis* to be attached to it. In (3b) the SAL Type set for *affect* offers no reason to attach *affected by digitalis* to anything but the left-adjacent noun, *tissue*, the default action.

Case Four

Many verbs have multiple meanings and attendant argument structures, only one of which can serve as the basis for a verb's SAL code. The SAL code is usually chosen to reflect a verb's most complex complementation pattern, on the grounds that this will afford maximum parsing benefit. It is the function of the **Semantic Table** to compensate for this single-code limitation. Rules in the **Semantic Table** will analyze a verb's actual sentential context and revise the SAL code and transfer accordingly.

We illustrate this using the multifaceted verb *keep*. (The verb *keep* has 39 rules in the **Semantic Table**, which though seemingly many actually represent still rather incomplete coverage). In (4a), below, we see operative the sense of *keep* for which it was SAL-coded. In (4b)-(4d), we begin to see variations in argument structure and attendant shifts in verb meaning, made evident here in the unedited translations. In (4c), the classic Chomskian **PP** attachment ambiguity, *John kept the car in the garage*, is resolved by a **Semantic Table** rule designed to capture the sense of *keep* when complemented by a locative **PP**, thereby also causing the **PP** to be parsed as converbial to *kept* rather than connominal to *car*. (See 4.2.4 for more on the **Semantic Table**.)

- (4a) *John kept driving the old car.*
- (4a') *John fuhr das alte Auto weiter.*
- (4b) *John kept the old car.*
- (4b') *John behielt das alte Auto.*
- (4c) *John kept the new car in the garage.*
- (4c') *John bewahrte das neue Auto in der Garage auf.*
- (4d) *He did not try to keep his children from driving the old car, but he told them to keep the old car away from the new car in the garage.*
- (4d') *Er versuchte nicht, zu verhindern, dass seine Kinder das alte Auto fuhren, aber er wies sie an, das alte Auto vom neuen Auto in der Garage fernzuhalten.*
- (4d'') *Il n'a pas tenté d'empêcher ses enfants de conduire la vieille voiture, mais il leur a dit de garder la vieille voiture hors de portée de la nouvelle voiture dans le garage.*

Case Five

Noun phrases analyzed as SAL patterns may benefit target. In the *experimental* work below involving the N N noun phrase pattern, we show (i) three literal strings, (ii) the SAL patterns to which they are converted, and (iii) the analyses that source rules that are specified for these SAL combinations are able to pass on to the target module. The final, generated target results are illustrated for French. (The default rendering in French for the N N pattern, of course, is **N2 de N1**.)

- *gold watch*
 - **N1(MAmetal) + N2(COmeter)**
 - **N2 made of N1 (implicit analysis)**
 - *montre en or*

- *computer tape*
 - N1(COmach) + N2(MAfunc)
 - N2 used by N1 (implicit analysis)
 - *bande pour ordinateur*
- *wine glass*
 - N1(MAliqu) + N2 (COrecp)
 - N2 for containing N1 (implicit analysis)
 - *verre à vin*

In the SAL patterns above, **MA** and **CO** stand for the ‘mass’ and ‘concrete’ noun supersets, respectively. The string in lower case represents either the set or subset within the superset, as the case may be. Note that although the Logos Model is multi-target and thus one source analysis supports multiple targets, rules having applicability only to French, for example, can be ignored by other targets. The above work is purely experimental: exceptions to such rules are not uncommon and while some of these exceptions can be lexicalized, the feasibility of handling N N constructions in general at this level of semantic abstraction has not yet been established. If feasible, exceptions that are not lexicalizable would then be handled via the **Semantic Table**.

4.2 HOW DO YOU STORE LINGUISTIC KNOWLEDGE?

Generally, linguistic knowledge is stored in two principal places: lexicon and rule base. Conventionally, the lexicon is the principal repository of linguistic detail and it is here that whatever semantic information a system will have tends to reside. The lexicon will commonly contain codes or instructions to assist in syntactic and sometimes also semantic disambiguation. In “radical lexicalism” (Karttunen, 1987), such as in so-called ‘Shake and Bake’ systems (Beaven, 1992), virtually all linguistic information is stored lexically. Lexicons thus tend to be information-rich, and because they are by nature indexable, can grow to very large size with negligible performance impact. But because lexical entries are word-specific, the lack of generality in lexically driven operations can be considered a drawback. Another drawback concerns the cost and difficulty users may experience in building and maintaining information-rich lexicons. Historically, some otherwise effective systems have been known to founder largely on these grounds (e.g, TAUM Aviation (Juola, 1989)).

Rule bases, by contrast, are largely confined to syntactic information whereby generality is realizable. Because of their inherent generality, syntactic rules can also be relatively few in number, considered a sought-after virtue with respect to system performance. There are drawbacks as well, chief among which is the parse-forest approach to parsing that “syntax alone” entails, requiring subsequent pruning. This separation of syntax and semantics (and the functions they support) advances an approach to language quite distinct from the assumptions about human sentence processing we have been describing in this paper.

4.2.1 Linguistic Storage in the Logos Model.

Knowledge store in the Logos Model differs rather considerably from the conventional distribution mentioned in the preceding section. The Logos lexicon, for example, has exceptionally lean information content, considering that the lexicon must support semantic processing and the distant, ultimate goal of FAHQT. Lexical information for a given entry is confined to (i) SAL classification for the part of speech (POS code plus three SAL codes for superset/set/subset); (ii) codes denoting morphological class and properties, to support analysis and/or generation; (iii) optional domain codes and user ID’s used to guide lexical selection at run time; (iv) pointers to target transfers (any number of target languages). This informational leanness was motivated by the perceived need to keep lexical work as simple as possible for the commercial user.

The following properties of the lexicon will be of interest:

- Lexical entries are stored in a relational data base (Oracle, since 1995).

- ❑ There is no distinction between source and target entries in the lexicon: the same entry in a given language will serve both source and target purposes. This is also true of all morphological tables.
- ❑ Morphological data required for a new lexical entry (including stem generation) is derived automatically by language-specific logic (and exception tables) in the **TermBuilder** software, freeing the user of this concern.
- ❑ Syntactic homographs also need not concern the user. When making a new entry, users do not need to consider whether or not the entry entails more than one part of speech. Assuming the lexicon already possesses these other parts of speech, **TermBuilder** automatically links associated homographs to the new entry.
- ❑ SAL codes are *automatically* assigned (since 1997) via an expert knowledge base wherein all possible meanings of all nouns are defined (i.e. SAL-encoded). However, accuracy here is far from optimal and users are urged to review SAL assignments and correct them as necessary. (So-called “prompts” are provided for this purpose, making it unnecessary for users to become familiar with the SAL coding scheme.) The automatic SAL-coding provision was designed to accommodate users’ rush jobs and can be considered an adequate accuracy/expediency tradeoff in such circumstances.
- ❑ Entries are also optionally encodable for subject matter domain and user ID. Users can create their own domain hierarchy or employ one supplied by the system. Lexical matching logic gives priority to domains and ID’s specified by the user at run time.
- ❑ Provision has been made in the Oracle implementation of the lexicon to include additional semantic codes, SAL or otherwise, for the full range of meaning that a word has. At present, however, the meaning assigned an entry is limited to (i) a meaning associated with a subject matter or company code; or (ii) a default meaning. It is the function of the **Semantic Table** to detect other meanings, fairly easily accomplished in the case of verbs and prepositions, but far more problematic in the case of adjectives and nouns (See Section 8 and Fig. 18).

The relatively high degree of automation available in lexical work allows users to dump pre-existing user glossaries into the lexicon quickly and easily, assuming they are rationally organized. Users are advised however that, with regard to SAL code assignments, *interactive* processes will yield significantly better results. It may be difficult to persuade the user of this since the impact of less than optimum lexical work is not always evident in the short term though, doubtless, it would become evident over time. Considerable internal debate has taken place regarding the relative *business* advantages of offering full automation with its appeal of speed and effort-reduction versus the imposition of labor-intensive interactive processes for the sake of their positive, long-term effects on translation quality. Given the option, one can foresee a majority of users choosing the route of least effort. Indeed, user psychology here may prove to be the ultimate limiting factor in the developer’s quest for FAHQT.

4.2.2. Logos Model Rule Bases: Source Rules

Although the lexicon is obviously foundational to translation, in the Logos Model the informational richness needed for effective, industrial strength MT lies primarily in the Model’s rule base, not the lexicon. This is accounted for by the fact that source rules are not syntactic but semantico-syntactic. To support deterministic parsing, rules must be prepared to deal with both syntax and semantics as these variously affect decisions regarding both structure and meaning. The advantage in such an arrangement is clear: because these semantico-syntactic rules are

abstract (a consequence of the second-order abstractions of SAL), rule-based semantic processing can avail itself of the advantages of generality as well as specificity.

A second implication for informational richness concerns the size of the rule base, which traditionally must be kept small for performance reasons. In the Model we are describing, the rule base can grow to any size, with such growth having generally sub-linear impact on performance. In the current Logos System for English, there are well over twenty thousand semantico-syntactic source rules distributed across the eight rule modules of the pipeline (Fig. 2). (The overall rule base size for German source is slightly smaller.) In a 25-word English source sentence, on average about 257 rules will contribute in some way to its deterministic analysis: c. 21 for **Res1**, 32 for **Res2**, 43 for **Parse1**, 54 for **Parse2**, 34 for **Parse3**, 62 for **Parse4**, and 11 for the two **Semantic Tables**.

The following features of these rules will be of interest:

- ❑ Source rules are extremely shallow, functionally speaking. Each rule typically deals with a single NL phenomenon, and accomplishes with respect to that phenomenon some small, standardized function or set of functions. The severe restriction on rule scope accounts for their large number. Strict, standardized functionality is imposed on rule writers intentionally: to keep source rules reasonably simple and transparent. Although the set of functions available to the rule writer is fairly extensive, rule writers do not have access to general-purpose programming facilities for specifying rule constraints or actions.
- ❑ Source rules almost always presuppose other rules, and work in free conjunction with rules presumed to fire before or after the current rule. Thus a number of rules, each presupposing the other, are typically needed to accomplish what a single rule in another system might do. This rule conjunction is considered *free* because rule sequence is not legislated either by the rules themselves or by any supervisory logic. (Chiefly, what controls rule sequence is the logic of the input stream itself, although rule writers can influence the sequence in a general way by re-labeling some aspect of the input (e.g., Type field) and then re-examining it, causing a different set of rules to become invoked. Carried to an extreme, this could begin to resemble an algorithm, but that level of control is hardly ever employed.) One advantage of this arrangement (many simple rules *versus* a single complex rule) is that developers, when debugging, can more readily identify the effect of any given rule. The main motivation, however, was the need to address natural language's richness--the endless, often idiomatic details of natural language--in a way that was both effective and efficient. A system with a performance-restricted number of large, powerful rules (only one small portion of which rule might be relevant in a given situation), seemed less likely to support these ends, and more prone to the previously cited problem of logic saturation.
- ❑ Source rules have three parts:
 - Semantico-syntactic pattern expressed as a SAL string. Patterns can comprise up to ten SAL elements and can have up to three Kleene stars. (Kleene stars can be constrained in a variety of ways.) Patterns in most rules tend to be quite short. Each element in the pattern is expressed by the WC(Type;Form) triplet. Patterns obviously can and do include literal words where necessary (expressed by a hash code in the Type field).
 - Constraint satisfaction. Constraints which must be met for a rule to fire may relate to (i) some semantic feature or set of features of an element or elements in the SAL pattern that is not expressible in the element's Type field, e.g. something that could have been learned about the element earlier in the pipeline; (ii) some previous event in the clause or entire sentence, such as the earlier occurrence of a certain SAL element or ad hoc set of elements (e.g., certain verb Types); (iii)

some SAL element or set of such elements which must be found (or not found) at some distance to the right; (iv) some condition of the clausal state which must be true, such as clause or sentence type. In short, constraints can pertain to virtually any kind of top-down or bottom-up information that could be relevant to analysis, whether semantic, syntactic or morphological.

- **Source action:** Actions consist of the string of functions specified by the rule writer, drawn from a library of fixed functions. Actions can be made conditional on various testable linguistic conditions, but are otherwise limited to the library repertoire. Principal action typically is re-write (and concatenation) of the SAL input pattern at a still more abstract level. However, re-writing is not a necessary function of a rule and many rules exist merely to analyze and re-label some part of the SAL signature of an input element, or in some other way to pass on notational information regarding that element for the benefit of some subsequent rule. More rarely, rules may also override decisions of previous rules. Although the ability to do this is rather limited, this provision allows for correction of a false, previously made N/V homograph resolution. Target actions, if any, are accomplished in separate, optional target rules parasitically linked to the source rule so as to make use of source rule analysis. Target rules from any number of target languages may be so linked.
- Source rules accomplish analysis in bottom-up, left-to-right fashion, as the SAL string passes down the six modules of the pipeline. Information about the input string captured by the initial macro parse of the two **Res** modules is made available as top-down data to the later **Tran** modules and accounts for the presence of top-down constraints in these **Tran** modules. (The provision of top-down information during bottom-up analysis will be further accounted for in discussion of the **Res** pipeline modules, Section 4.3.2.)
- Source rule bases in the Logos Model can be viewed as *SAL pattern dictionaries* indexable on their SAL pattern. Such indexing accounts for the sublinear performance impact of rule growth in the Logos Model. In effect, the SAL input string serves as search argument to the rule base *qua* SAL pattern dictionary, analogous to the way NL words are looked up in a NL lexicon. This is how the Logos Model emulates the “content-addressable” memory feature of the mental model. (This feature will be further accounted for in Section 7).
- Rules are self-ordering. Developers do not have to think about where to put a new rule in a given module, or how to insure its being matched. Obviously, developers’ parsing strategy will determine which rule base module in the pipeline is to house a given rule.
- All three parts of a rule are in the form of data, interpreted by the language-neutral software of the corresponding pipeline module.

4.2.3 Logos Model Rule Bases: Target Rules

Target rules in the Logos Model, where such rules are present, are linked to source rules, and are confined to a contrastive linguistic function *vis à vis* the input segment the source rule is dealing with. Target rules therefore only have an action component. The functions available to target linguists nevertheless allow them to implement very effective contrastive linguistic strategies, as the translations in this paper hopefully make evident. Target functions can entail additional analysis of the relevant source pattern, making target actions contingent on source factors left unanalyzed by the parent source rule.

In the previous section on source rules, we stated that simple noun phrases are parsed as **NP** via a sequence of source rules. Target equivalences are effected under this arrangement by a

progressive filling of slots in a target NP template, as elements are handled, one by one, in the source rules. In some cases, new elements not in the source (e.g., a preposition or an article) are added to a template slot, or a source element may be suppressed, etc. Thus, once a target rule (linked to a source rule handling the linguistic element) determines that the target equivalent of, say, an AJ in the SAL string should be post-posed in the target, that target adjective is placed in a post-posed adjective slot within the target NP template. For example, source analysis of the English input string *a tall, white horse*, coupled with linked target rules for French, would yield the source NP and equivalent target NP, below. Note that the French NP rewrite here is top-down, acting on the bottom-up NP of the English analysis.

- **Parse1** source rewrite action: **DET AJ1 PUNC AJ2 N → NP**
 - (*A tall, white horse*)
- **Tran1** target rewrite action: **NP → DET AJ1 N AJ2**
 - (*Un grand cheval blanc*)

In the above example, target slot unloading takes place immediately following creation of the source NP and the associated target rule is called. Target action here unloads the target NP template slots, in effect creating the contrastive sub-tree structure for the target constituent. Target action also includes annotation of this sub-tree node for data (case, gender, number) needed in generation of the literal target noun phrase at the tail end of the translation process. In general, target templates for NP, VP, clause, and sentence are the keys to structure transfer in this Model. These templates are powerful and allow for slot nesting, and for loading of functions to be executed at slot-unload time.

Target rule characteristics:

- Source rules invite target rules to fire (any number of targets) whenever some source analysis is deemed to have potential target implications.
- Target work conducted can be thought of as tree-to-tree transfer. In effect, as each node in the source tree is built, in bottom-up fashion, at successively more abstract levels, a linked target rule acts upon the SAL pattern comprising that node and establishes the target equivalent, in contrastive linguistic fashion.
- Target actions are also limited to a standard repertoire of actions. As in the case for source rules, target rules do not support general-purpose programming logic.
- Target rules that transfer source *verb* constructions (in the **Parse4/Tran4** modules) are designed to be multi-source, i.e., common to any source module. In the case of such multi-source target rules, source parameters (for tense, voice, aspect, etc.) that are needed to drive these verb rules are expressed meta-linguistically. Thus, for example, verb formation rules in **Parse4/Tran4** for French, can be linked to any source (currently German or English). This multi-source feature is not true of target rules in general.
- Because they must often effect radical transformations in narrow confines, target rules in this Model tend to be somewhat more complex than source rules and relatively more difficult to maintain (i.e. less free of complexity effects).

4.2.4 Semantic Tables

The **Semantic Table (Semtab)** is a knowledge resource called upon by rules in the four **Parse/Tran** modules for purposes of finer analysis. (A similar but separate **Semantic Table** is available to the two **Res** modules.) Rules in the **Semantic Table**, called **Semtab** rules, are conceptual, deep structure rules invocable by regular source and/or target rules at any stage of pipeline analysis or

transfer. These deep structure rules can be applied to virtually any relevant surface structure, regardless of word order, passive/active voice construction, etc. **Semtab** properties include the following:

- Resolution of parsing problems. For example, a **Semtab** rule would be invoked in the **Res2** pipeline module to resolve the attachment of the **PP** *to his brother* in the following sentences (shown with raw German output):

- (6a) *John gave the car that he bought to his brother.*
- (6b) *John repaired the car that he gave to his brother.*
- (6a') *John gab seinem Bruder das Auto, das er kaufte.*
- (6b') *John reparierte das Auto, das er seinem Bruder gab.*

In the foregoing, a generic **Semtab** rule applying to any SAL di-transitive verb with governance of the preposition *to*, causes the **PP** *to his brother* to be labeled as *converbal* to the di-transitive verb, in whatever clause it is found. The di-transitive *give* is in the main clause in (6a), and in the relative clause in (6b), an analysis reflected in translations (6a') and (6b'). This labeling has the effect of inhibiting the firing of any subsequent rule that would want to keep the **PP** in the current clause by default, which would work serendipitously for (6b), but not (6a).

- Resolution of semantic ambiguities. **Semtab** rules resolve verb polysemy, usually on the basis of their argument structure. Using the verb *raise*, we illustrate various French transfers effected by **Semtab**, based on argument structure. In these rules, **AN** stands for 'animate' noun superset, **ME** for 'measure' superset, and **MA** for 'mass.' The lower case string stands for set or subset under the respective superset. These rules would typically be invoked in **Parse3/Tran3**.

- (7a) raise a child → V('raise') N(**ANhum**) → *élever . . .*
- (7b) raise the rent → V('raise') N(**ME**) → *augmenter . . .*
- (7c) raise corn → V('raise') N(**MAedib**) → *cultiver . . .*

- Rules are conceptual (deep structure). **Semtab** rules are conceptual in nature and can be invoked to resolve verbal elements without consideration as to actual part of speech. We illustrate this with the **Semtab** rule: V('raise') + N(**ME**) → *augmenter*. This deep structure rule is applied to all surface forms of *raise*, and *augmenter* in turn is automatically given its own appropriate surface form.

- (8a) *he raised the rent* → *il a augmenté le loyer*
- (8b) *the raising of the rent* → *l'augmentation du loyer.*
- (8c) *the rent, raised by. . .* → *le loyer, augmenté de . . .*
- (8d) *a rent raise* → *une augmentation de loyer*

All of this contrastive transfer was effected by the single **Semtab** rule abstracted above, invoked at various stages of the pipeline: (**Parse1/Tran1** for (8d), **Parse2/Tran2** for (8b, 8c), **Parse3/Tran3** for (8a)). (Users of the Logos System can readily create such rules via the **RuleBuilder** tool (previously called *Semantha*)).

- Target transformations. **Semtab** rules analyze the meaning of the source and effect target equivalents, not infrequently involving structural changes. Typical examples follow:

- (9a) *He is afraid of the dark.*
- (9a') *Il a peur de l'obscurité.*
- (9b) *They are competing against local companies.*
- (9b') *Ils font concurrence aux sociétés locales.*
- (9c) *Try to keep him busy.*
- (9c') *Tentez de l'occuper.*
- (9d) *He swam across the river.*
- (9d') *Il a traversé la fleuve à la nage.*

- (9e) *If the light goes on, be sure to turn it off.*
 (9e') *Si la lumière s'allume, soyez sûr de l'éteindre.*
 (9f) *The conflagration may go on for many days before they put it out.*
 (9f') *La conflagration peut continuer pendant beaucoup de jours
 avant qu'ils ne l'aient éteint.*
 (9g) *He lived down the bad reputation that had been following him.*
 (9g') *Il a fait oublier la mauvaise réputation qui l'avait suivi.*
 (9h) *His good name will live on.*
 (9h') *Son bon nom survivra.*
 (9i) *The new product did not live up to expectations.*
 (9i') *Le nouveau produit n'a pas répondu aux espérances.*
 (9j) *The family lived through the storm.*
 (9j') *La famille a survécu à la tempête.*
 (9k) *He lived out the war in a small town far from the conflict.*
 (9k') *Il a passé la guerre dans une petite ville éloignée du conflit.*

- **Semantic Tables** are language-pair specific. There is a separate **Semantic Table** for each source and target language combination. (All other source modules in the Logos Model are target-independent in nature and multi-target in effect.) Plans to separate out purely source-related rules into a separate, target-independent module have never been implemented. Until this happens, current implementations of the Logos Model can only execute one language combination at a time, i.e., cannot execute in actual multi-target mode.
- **Semtab** has over 12,000 rules in the English system. The verb *raise* has 26 rules affecting analysis and transfer, the verb *take* has 130 rules. **Semtab** is expected to grow many times over as the English system continues to mature. (**Semtab** in the present German system is already somewhat larger.)

4.3 HOW DO YOU APPLY THE RULE BASE TO THE INPUT STREAM?

We treat this question in two parts. First we discuss the various ways rules can be applied to the input stream. Then we present a functional overview of input stream/rule base interaction in the Logos Model, as analysis proceeds along the pipeline.

4.3.1 Applying the Knowledge Store.

David Hays, one of the early MT pioneers, claimed that perhaps the most critical and troublesome aspect of machine translation concerned the method by which the knowledge base is to be applied to the input stream², a claim that few developers would care to dispute. In very general terms, the possibilities for this knowledge base/input stream interaction are as follows:

- **One-to-Many Relationship (Traditional Lexicon)**. The one-to-many relationship is typified by lexical lookup where an input term serves as search argument to the stored lexicon. This relationship is made possible because of the *representational monotonicity* shared by input and stored knowledge, viz., literal words. The one-to-many relationship furthermore presupposes an index which efficiently connects search argument to stored knowledge. The finer the index, the smaller the 'many' that must be looked at. As a result of these factors, increases in the size of the lexicon have strictly sublinear impact on system performance. Hence growth in lexicon size need never become an issue.
- **Many-to-One Relationship (Traditional Rule Base)**. The principal difference here is that, in traditional models, rule bases and input stream generally do *not* share strict representational monotonicity, i.e., one is usually attempting to compare apples and oranges. In effect, one starts with the rule base (the many) and attempts to find a match on the input (the one). This raises the question of how to accomplish rule matching

efficiently. Matching strategies generally entail supervisory logic, often in the form of metarules or discrimination networks designed to narrow down the number of rules that need to be applied. Whatever the case, the relationship is many-to-one and for this reason, system performance issues exert great pressure to keep the size of the “many” small.

- Logos Model Rule Base: One-to-Many Relationship. The Logos Model conceives of its rule bases as *indexable pattern dictionaries*, with the effect that the relationship is one-to-many, as in the case of the traditional lexicon. The nature of the index is such that the one-to-many relationship, in certain contexts, begins to approach one-to-one. It is not unusual, for example, that an input segment will find its appropriate rule match by directly consulting a single rule. As in the case of the lexicon, this one-to-many relationship means that growth in rule base size has strictly sublinear impact on system performance. And as in the case of the lexicon, this advantage is made possible because of the representational monotonicity which SAL affords between input stream and knowledge store, thus allowing the SAL input stream to serve as search argument to the series of SAL pattern dictionaries (rule bases) as the SAL input stream passes down the pipeline from module to module.

4.3.2 Pipeline Modules and their Functions (Fig. 8)

Each software module in the pipeline, in succession, take segments of the SAL input stream (working left to right) and, using them as search arguments, seeks the highest scoring match in the module’s rule base. Scores are calculated on the basis of pattern length and semantic specificity (in **Res** some additional factors are used). A successful match occurs when a source rule (i) matches the search argument (SAL input segment), (ii) satisfies all rule constraints, (iii) wins out over competing rules, (iv) and thus wins the right to fire, which means in effect that the action component of the rule is then executed, i.e., the software module interprets and carries out the various functions specified. Depending on where we are in the pipeline, actions may entail any or all of the following: (i) analysis and resolution of syntactic homographs, grammatical relationships, semantic homographs; (ii) concatenation of elements under the head element and creation of a bottom-up parse tree node; (iii) annotating parse nodes by means of a one-hundred-cell array linked to each such node; (iv) recording of both top-down (overview) and bottom-up (local) intelligence for the benefit of subsequent rules; (v) calling linked target rule (for any number of targets) optionally to avail itself of source rule’s analysis.

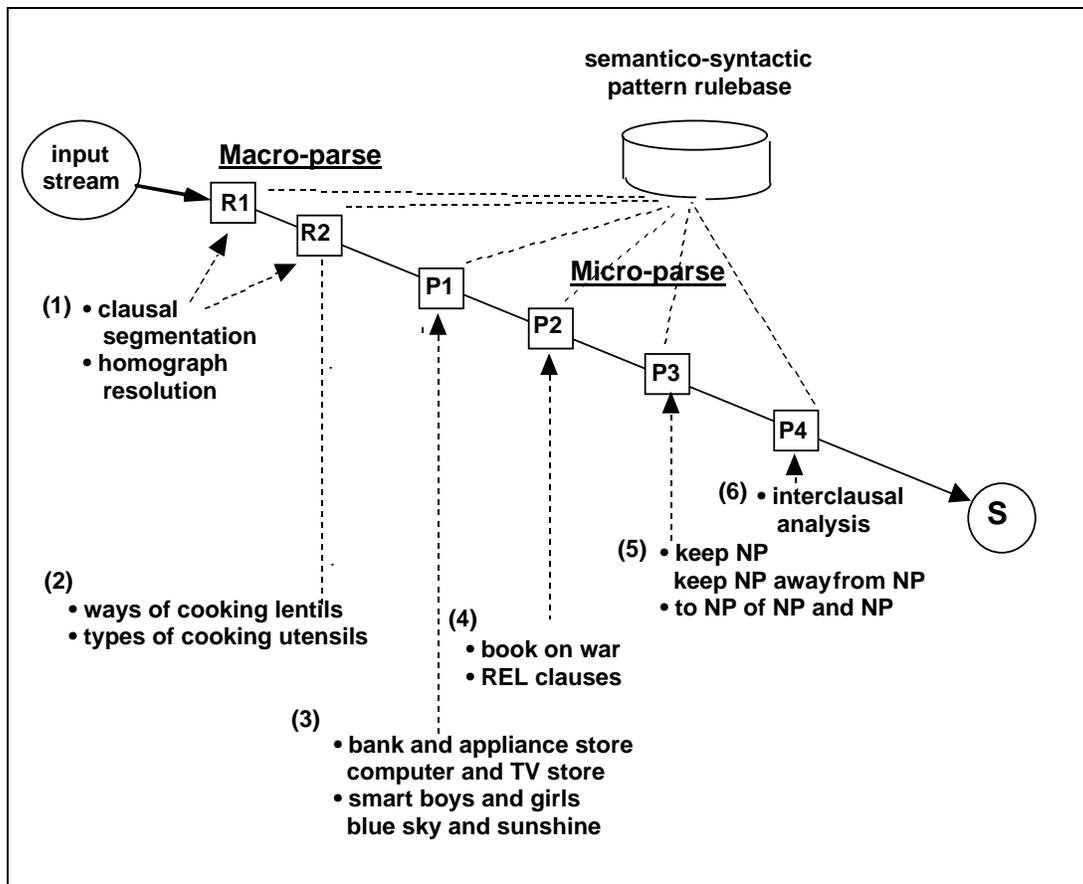


Fig. 8 - Incremental Pipeline Analysis. Res modules accomplish a macro parse (1) of the input sentence, resolving syntactic homographs and clausal transitions. This macro parse affords a top-down view of the sentence that is used to guide the bottom-up micro parse by the four **Parse** modules. Micro parse is accomplished incrementally, each **Parse** module dealing with a specific set of parsing tasks, as illustrated in (3) to (6).

Pipeline modules (Figs. 3 and 8) and their analysis functions are as follows:

- **Format** - This module extracts all mark-up commands from input text (HTML, Interleaf, etc) for subsequent re-application to target output. In some cases, mark-up information is made use of during analysis, e.g., bold facing in a string such as '*Insert key*' helps analysis see *Insert* as a label rather than as an in-line verb. This module also segments text into discrete sentences. (There is as yet no extra-sentential handling, i.e., no sentence carry). Sentence length is arbitrarily set at 70 elements (including punctuation). Sentences exceeding this limit are automatically broken up into smaller sentences at semicolons or other potential breakpoints.
- **Lex** - Words of NL sentence are looked up in this module (on longest match principle) and are immediately converted to their corresponding SAL element (or elements, where entry is syntactically ambiguous). Functional characteristics of lexicon are as follows:
 - no practical limit on word length in a lexical entry.
 - no limit on number of words in a lexical entry. However, an arbitrary ten-word matching limit is currently imposed by the **Lex** software module.
 - no limit on number of parts-of-speech associated with a given

entry. However, a **Res** limitation (see below) requires that only three parts of speech can be extracted for analysis purposes, a limitation meant to be compensated for by various stratagems involving **Semtab**. (The word *building* concerns seven parts of speech.)

- no limit on number of meanings associated with a given part-of-speech of a given entry, insofar as such entries are differentiated by subject matter domain or user ID. However, there is only one default entry that will automatically be selected if no other selectional criterion is found to apply. This can be a serious limitation in the case of polysemous common nouns, one that is only very partially overcome by recourse to **Semtab** (See Section 8 and Fig. 18).
 - domain codes are hierarchical, allowing matching logic to favor the more specific codes in a selectional list and to default to the more generic.
 - part-of-speech disambiguation on morphological grounds immediately eliminates certain word classes from further consideration for a given entry.
 - In German source, compound nouns that have no lexical match are decomposed and separate elements are looked up. Nothing comparable is done for composite nouns in English source.
- **Res1, Res2** - Each **Res** module accomplishes a single pass of the sentence, from left to right, collectively effecting a macro parse. No parse tree is formed (no concatenation) but homographs are resolved and all clausal boundaries and clausal relationships are identified. Information regarding the macro parse is passed on to subsequent **Parse** modules to serve as top-down guide for the progressive, bottom-up micro parse effected by these modules. This top-down picture enables the micro parse, when looking at a noun, for example, to know the structural context of that noun, e.g., what sort of clause the noun is in (e.g., a relative clause inside a dependent clause), whether the noun precedes or follows the verb of the clause, the SAL class of that verb, etc.

For English source analysis, the principal work of the **Res** modules is homograph resolution. The SAL information coming out of **Lex** shows all possible parts-of-speech that the original NL word was found to have and that remained unresolved after morphological analysis. It is the task of **Res1** and **Res2** to resolve such ambiguous forms to a single part-of-speech. The following are typical examples of the work of these **Res** modules. Two sentences of similar construction are shown. Note that both have ‘**ADV ING N**’ strings at the end which have to be resolved quite differently. We show the effect of this resolution on unedited machine output for French, other translation flaws notwithstanding.

(10a) *We spent time doing tasks such as systematically classifying documents.*

(10b) *We spent money eating things such as really satisfying pastry.*

(10a') *Nous avons passé le temps à accomplir des tâches telles que la classification systématique des documents.*

(10b') *Nous avons dépensé de l'argent en mangeant des choses tels que la pâtisserie vraiment satisfaisante.*

In (10a) the **ING** form *classifying* is to be resolved to a verb, in (10b) *satisfying* must be resolved to an adjective. Both **ING**'s came into the **Res** modules as two parts of speech (second infinitive and verbal adjective). In (10a), the key to resolution was provided by *tasks* SAL-coded as a member of a ‘verbal abstract’ noun grouping that anticipates verbal complementation. When a **Res2** rule encounters this Type of noun in the input stream, a top-down verb-expectation bias is established which allows a subsequent rule to interpret the **ING** form as a verb. In (10b), the absence of any such bias (or other determinant) allows the default resolution to obtain, namely to the adjectival form of the **ING**.

Res2 affords limited look-ahead capability for avoiding garden path situations. In the case of the classic garden path construction in (11b), below, a rule attempting to resolve the syntactically ambiguous form, *run*, to the main verb of the sentence must first invoke a look-ahead function to insure that no unambiguous main verb is to be found to the right. Such search-ahead logic is also pattern-rule based, and while generally effective in shorter sentences, may run into difficulties in longer constructions. Notice, in the raw French output below, how *run* has been resolved to an intransitive verb in (11a) and to a transitive verb in (11b).

- (11a) *The horses run by the barn.*
- (11b) *The horses run by the barn are tired.*
- (11a') *Les chevaux courent par la grange.*
- (11b') *Les chevaux dirigés par la grange sont fatigués.*

Res modules share a dedicated **Semantic Table** which supports a variety of resolution strategies. For example, a selectional restriction rule in this table helps resolve the noun/verb homograph *loan* to a noun in the context of words like *office*, *form*, *value*, *officer*, etc, as may be seen in (12a) and (12a'), below.

- (12a) *We sold everything from desks to loan office furniture.*
- (12a') *Wir verkauften alles von Schreibtischen zu Darlehensbüro-Möbeln.*

While the number of rules in this table is in the thousands, relatively few actually deal with selectional restrictions, this stratagem never having been found to be particularly efficient or effective in our experience. Selectional restriction rules generally have weak authority, as may be seen in (12b) and (12b'), below.

- (12b) *We do not allow our employees to loan office furniture.*
- (12b') *Wir ermöglichen unseren Angestellten nicht, Büromöbel zu leihen.*

In (12b), the **Res** rule that registers *allow* as 'pre-verbal' causes a subsequent **Res** rule to outbid the **Semantic Table's** selectional restriction rule otherwise applying, causing *loan* to be resolved to a verb, as the *leihen* in (12b') shows.

In German source analysis, the contribution of the **Res** modules to the parse is more restricted, addressing primarily (i) the ambiguity of noun case markings; (ii) *die*, *der*, *das* part-of-speech resolution (to the extent possible). Given the nature of German morphology, it may take the entire pipeline analysis to resolve certain ambiguities. For example, **Res** can readily resolve the part of speech and case of *der* in *der Mann*, but must defer final resolution of *das* in *das Bier* to a subsequent pipeline module. For example, in *das Mädchen, das Bier bringt*, resolution is not effected until the relative clause handling done in **Parse2**.

A graphic illustration of the **Res** macro parse may be seen in Fig. 11.

- **Parse1** - The four **Parse** modules effect a micro parse of the SAL input stream, building on the output of **Res** and producing a final, single, bottom-up parse tree. Although **Parse** software modules are almost identical programmatically speaking, the compositional approach implied in pipeline architecture presupposes that each **Parse** module will perform a specific range of bottom-up parse functions. The output of each **Parse** module serves as input to the next, affording a progressively more abstract analysis of the sentence.

Typical operations that **Parse1** accomplishes (not always successfully):

- Simple NP formation (excluding noun series, **REL** and **PP** attachments). Parse tree nodes for **NP** are annotated for **NP** properties

(definite, indefinite, SAL Type of adjective modifier if any, etc.).

- Scoping of adjectives. Scoping of **AJ** in **AJ N N**, e.g., is achieved by sending the adjective and each noun in turn to the **Semantic Table**. In the noun phrases below, with unedited translations, note that the adjective in (13a) applies to the *modifier* noun, *boys/garçons*. In (13b) and (13c), the adjective applies to the *head* noun. Default scoping is to head noun. Adjective scoping remains a relatively weak area in **Parse1** analysis.

- (13a) *Smart boys school*
- (13a') *École de garçons intelligents*
- (13b) *Large boys school*
- (13b') *Grande école de garçons*
- (13c) *Smart language students*
- (13c') *Élèves de langue intelligents*

- Auxiliary verb phrase analysis, concatenation and labeling. For example: *ought to have* [+ verb, past t.] → **AUX(MOD;psma)** [Form = past subjunctive modal active].

- (14) *He ought to have gone home.*
- (14') *Er hätte nach Hause gehen sollen.*

- Adverbial phrase recognition and concatenation. For example:

- *in general* → **AV(SENT)** [sentential adverb] (Note that *in general* is not lexicalizable. Cp., *in general terms, in general quarters.*)
- *all morning/day/year/etc. long* → **AV(TIME)** → *toute la journée/etc.*

- Resolution of ING forms. **Res** determines when an **ING** form is to be seen as nominal, but leaves it to **Parse1** to decide whether, in the case of some forms, the **ING** is a concrete noun or gerund. Source analysis and target transfer of **ING** forms are effected by close interaction between **Parse1** and **Semtab**. This remains a still largely unexploited area of **Parse1/Semantic Table** interaction but the examples below, with unedited output, at least show the current possibilities:

- (15a) *He saw a building.*
- (15a') *Il a vu un bâtiment.*
- (15b) *He witnessed the building of the dam.*
- (15b') *Il a vu la construction du barrage.*
- (15c) *The device has a variable speed setting.*
- (15c') *L'appareil a un réglage de vitesse variable.*
- (15d) *A new diamond setting technique has been developed.*
- (15d') *Une nouvelle technique de positionnement de diamant a été développée.*

- Re-labeling of *should, provided*, etc. at beginning of declarative sentences. For example:

- (16) *Should the situation call for such action, we are prepared to act.*
- (16') *Si la situation réclame telle action, nous sommes prêts à agir.*

- Analysis of *as*. The form *as* is in the dictionary only as a **CJ(SUB)**, it being left to source analysis to determine its exact grammatical function, especially when used to introduce a non-lexicalizable phrase. A great many such phrases are handled by **Parse1**, e.g., *as a whole*; *as AV as*; *as AJ as*, etc. **Parse1** analysis concatenates these phrases, labels them as appropriate (**ADV, PP, DET, CJ(SUB)**),

and affords targets opportunity to override the default lexical transfer. For example:

- (17a) *as of April* → *ab April*
- (17b) *as few as five* → *nur fünf*
- (17c) *as many as five* → *bis zu fünf*
- (17d) *as to ...* → *Im Bezug auf ...*
- (17e) *as long as possible* → *so lange wie möglich*
- (17f) *as long as you insist, I will come* → *solange Sie bestehen, komme ich.*
- (17g) *four times as fast* → *viermal so schnell*

- Analysis of any. Translation for *any* commonly depends on analysis of the entire clause, as illustrated in the following:

- (18a) *I do not have any book. Any book will suffice.*
- (18a') *Je n'ai pas de livre. Tout livre suffira.*
- (18a'') *Ich habe kein Buch. Jedes Buch genügt.*

The context available to **Parse1** is limited and **Parse1** translation for *any* can often be wrong, despite the top-down picture afforded by the macro parse. In (18b), we see that the **Parse1** transfer for *any* has necessarily been overridden by subsequent pipeline analysis.

- (18b) *If you do not wish to have any supper tonight, please tell the cook.*
- (18b') *Si vous ne souhaitez pas dîner ce soir, veuillez dire au cuisinier.*

Deficiency in target work can also be a source of error, as the French in (18c'). Note that the German in (18c'') treats *any* more correctly.

- (18c) *Do you want any supper tonight?*
- (18c') **Voulez-vous tout dîner ce soir?*
- (18c'') *Wollten Sie kein Abendessen heute abend?*

- Reversing a parsing decision made by Res. Though infrequently invoked, **Parse1** possesses the means to reverse **Res**, usually regarding N/V homograph resolution. To this limited extent, the pipeline parse is not purely deterministic.

See Fig. 13 for a graphic illustration of **Parse1** parsing.

- **Parse2** - The second **Parse** module builds upon the output of **Parse1**. Among its principal tasks are:
 - Detecting, and effecting connominal PP attachments to NP. Here the **Parse2** rule generally has the form shown below, where **x** is any preposition or class of prepositions.
 - SAL Pattern: NP(GOV_x) PRP(x) NP
 - Constraints: (i) The verb of the clause, if one has occurred, has not made a strong prior governance claim on **PRP**; (ii) the **NP** governs the **PRP**.
 - Actions: (i) Preposition is labeled *connominal*.
(ii) prepositional phrase is attached to **NP** at left.
 - Targets: Targets typically alter lexical transfer for the **PRP** at this point, as in (19b'), below.

In (19a'), below, notice that the gender of the German relative pronoun *die* refers to *revulsion* rather than *money*, reflecting a probabilistic assumption regarding which noun to attach it to.

(19a) *He has a revulsion to money that cannot be overcome.*

(19a') *Er hat eine Revulsion zu Geld, die nicht überwunden werden kann.*

(19b) *He wrote a book about the room.*

(19b') *Er schrieb ein Buch über den Raum.*

(19c) *He placed some flowers about the room.*

(19c') *Er stellte einige Blumen um den Raum.*

In (19c), above, the prepositional phrase *about the room* is *not* seen as connominal (to *flowers*) in **Parse2**. Rather, the converbal attachment of the **PP** and the target transfer *um* are effected in **Parse3**, as seen in (19c'),

- Extraction of relative clauses, embedded absolute constructions, parenthetical material, etc. In all cases, these are removed from the clause in which they appear, leaving behind a trace marker. Extracted materials are processed through **Parse2-Parse4** as separate sentences. (See Fig. 14 for graphic illustration of clause extraction.)
- Discrimination between relative clauses and th-clauses. For example:
 - (20a) *The hope that he had was very strong.*
 - (20a') *Die Hoffnung, die er hatte, war sehr stark.*
 - (20b) *The hope that he would win the race was very strong.*
 - (20b') *Die Hoffnung, dass er das Rennen gewinnen würde, war sehr stark.*

See Fig. 14 for a graphic illustration of **Parse2** parsing.

- **Parse3** - The third **Parse** module builds upon the output of **Parse2**. It will be noted in what follows that much of the work in **Parse3** is accomplished via interaction with **Semtab**. Among phenomena dealt with in **Parse3** are:
 - Verbs and non-contiguous verb particles. The rule abstracted below is typical: a SAL pattern consisting of an undifferentiated verb (V) stretching (★) over any number of non-verbal clause elements to a particle (**PART**) which is following immediately by clausal punctuation (**PUNC(CB)**). Note that up to this point analysis has not yet connected the verb and particle.

SAL Pattern: **V(u;u) ★ PART PUNC(CB)**

Constraints: None.

Action: (i) Send Pattern to **Semantic Table** for match;
 (ii) re-label **PART**, and backspace all the way for next match (re-labeling forestalls rematching).

Semtab Action: (i) If match on **V PART** is found, verb and particle are re-labeled with appropriate new SAL code.

Target Action: Target assigns new transfer to re-labeled verb (this is also done in action portion of **Semtab** rule).

In (21) and (22), below, we see the discriminating power of the **Parse3/Semantic Table** interaction, in particular, and of the pipeline parsing strategy in general. Sentence (21) is artificial but serves to illustrate **Parse3** analysis.

(21) *Please take the cover of the box, put on by my brother, off.*

(21') *Veillez retirer la couverture de la boîte, mise par mon frère.*

(21'') *Nehmen Sie bitte die Haube der Kiste, die von meinem*

Bruder angeschaltet wird, ab.

Below are further examples of **Parse3/Semtab** interaction, all initiated by the single **Parse3** rule abstracted above:

- (22) *Take the unit down, take the unit out, take the unit away, take the unit apart, take the work on, take the items back.*
- (22') *Baissez l'unité, sortez l'unité, retirez l'unité, démontez l'unité, entreprenez le travail, reprenez les articles.*

- Verb argument structure analysis and labeling. The **Parse3** rule effecting the analysis and transformations in the examples below is even simpler:
 - SAL Pattern: **V(u;u) * PUNC(CB)**
 - Constraints: None
 - Action: (i) Send Pattern, including all elements covered by Kleene star, to **Semtab** for match; (ii) inhibit this rule and backspace all the way for next match.
 - Semtab Action: (i) If match on **V** + verb arguments occurs, constituents are re-labeled where appropriate.
 - Target Action: Target may assign new transfer to verb, converbial preposition, object of verb, subject, etc. (also done in action portion of **Semtab** rule).

The following illustrate typical **Parse3/Semtab** interactions invoked by the above **Parse3** rule and the sort of target work that **Semtab** can effect.

- (23a) *Please let me know the result.*
- (23a') *Lassen Sie mich bitte das Ergebnis wissen.*
- (23b) *Please let me have the book.*
- (23b') *Erlauben Sie mir bitte, das Buch zu haben.*

- NP series concatenation. Rule constraints in series rules allow match only where commas and coordinating conjunctions are *non-clausal*, as established by the earlier **Res** macro parse. For example:
 - SAL Pattern: **NP(u;u) PUNC(COM) NP(u;u) CJ(CRD) NP(u;u)**
 - Constraints: **PUNC** and **CJ** (conjunction) are not clause boundaries.
 - Actions: (i) Concatenate as **NP**, using SAL code of **NP1**; (ii) re-label last **NP** with end of series marker; (iii) backspace 1 element.
 - Targets: (i) Chain articles and preposition series (for French); (ii) Make case assignments (for German), etc.

The unedited French translation (24') of sentence (24) well illustrates the effect of this **Parse3** rule, and a shorter **Parse3** rule just like it:

- (24) *My father gave his house, car and boat jointly to his sons and daughters.*
- (24') *Mon père a donné sa maison, sa voiture et son bateau en commun à ses fils et à ses filles.*

- Analysis and concatenation of hitherto unattached prepositional phrases as adverbial **PP**. Adverbial **PP**'s are labeled **PP(SENT)**, **PP(LOC)**, **PP(MANNER)**, etc., according to **PRP** and **NP** Type combination.
- Interclausal chaining. The following depicts a **Parse3** rule designed to handle **ING** series in dependent clause, whether chained by comma or coordinating conjunction. Labeling of the **ING** form in the first dependent clause is

communicated to **ING** forms of the subsequent clauses. Note that the one rule handles **ING** series of any length, as illustrated below.

SAL Pattern: **V(u;ING) * PUNC(COM) /or/ CJ(CRD) V(u;ING)**
Constraints: **PUNC** or **CJ** must be clause boundaries.
Actions: (i) Previous re-labeling of 1st **ING**, denoting presence of a subordinate clause, is communicated to 2nd; (ii) backspace all the way.
Targets: Effect series chaining for **ING** verb forms.
Comments: (i) Rule applies to series chained either by commas or coordinating conjunctions; (ii) re-labeling and backspacing all the way is typical of **Parse3** rules. Backspacing allows for multiple looks at a given pattern, each for a different purpose (as in (26)).

The unedited French (25') illustrates the effect of the above **Parse3** rule:

- (25) *When operating the unit, debugging the unit or performing maintenance tests, be sure to check power levels.*
 (25') *En commandant l'unité, en mettant au point l'unité ou en effectuant les essais d'entretien, soyez sûr de vérifier les niveaux de pouvoir.*

In (26'), below, we see the joint effect of the two very different **Parse3** rules associated with (22) and (25) respectively, executed sequentially over the same pattern. (Note gender error in pronominal reference):

- (26) *When putting the cover on, taking it off, or adjusting it in any way, be sure the power is off.*
 (26') *En mettant la couverture, en le retirant ou en l'ajustant de toute façon, assurez-vous que le pouvoir est coupé.*

See Fig. 16 for a graphic illustration of **Parse3** parsing.

- **Parse4** - The final **Parse** module builds upon the output of **Parse3**. All sentence elements have now been reduced to just five SAL entities: **NP**, **PP**, **AUX**, **V**, **PUNC**, plus place markers for materials extracted earlier in **Parse2**. Earlier modules have already analyzed and labeled constituents for *intraclausal* grammatical function (e.g., re-labeling **NP** for subject, object, indirect object, etc.). Now, a principal work of **Parse4** is verb tense analysis, both intra- and interclausal.

Functions include:

- Intraclausal tense assignments: Punctuation (**PUNC**) and subordinating conjunctions (**CJ(SUB)**) have been concatenated and re-labeled in **Parse3**. These re-labeled elements now trigger tense assignments in **Parse4**.
 (27) *Unless he receives instructions to the contrary, he is going to go home.*
 (27') *À moins qu'il ne reçoive les instructions au contraire, il va aller à la maison.*
- Interclausal verb tense/mood coordination. In the following, the treatment of the complementary infinitive clause is a function of verb SAL Type and verb tense in the principal clause.
 (28a) *They want John to do it*
 (28a') *Ils veulent que John le fasse.*
 (28b) *They do not want him to succeed.*
 (28b') *Sie wollen nicht, dass er Erfolg hat.*
 (28c) *They did not want him to succeed.*

(28c') *Sie wollten nicht, dass er Erfolg hatte.*

- Pronoun resolution. Without extra-sentential processing, pronoun resolution obviously remains a weak area. Handling is somewhat better when the antecedent is intrasentential, as in (29):

(29) *We buy a house that catches our fancy and then resell it the next year.*

(29') *Nous achetons une maison qui attrape notre fantaisie et ensuite la revendons l'année prochaine.*

But, even here, overly complex structures typically cause the pronoun/antecedent relationship to be lost, as in (30):

(30) *We buy a house that catches our fancy and live in it until we are tired of it, and then we sell it again, usually within a few years.*

(30') *Nous achetons une maison qui attrape notre fantaisie et vivons dans elle jusqu'à ce que nous soyons fatigués de lui, et ensuite nous le vendons de nouveau, d'habitude dans quelques années.*

The above functions notwithstanding, the principal work of **Parse4** is directed toward support of targets. It accomplishes this by presenting these abstract source constituents (**NP**, **VP**, **AUX**, **PUNC**), one by one, to linked target rules. Target rules load these constituents in the appropriate slots of a final, high-level target template for each clause and finally for the sentence as a whole. At this point, everything that must be known about the source string is now available to the target, thus allowing targets to place the by now virtually meta-linguistic constituents into the desired target order, effecting such stylistic transformations as is deemed appropriate and possible. At the end of **Parse4** source analysis, target template slots are unloaded and the resultant target parse (bracketed target string) is input to **Tgt Gen**, for generation of literal target output.

See Fig. 17 for a graphic illustration of **Parse4** parsing

5.0 Target Transfer and Generation

In this paper we have focused almost exclusively on source analysis, reflecting a belief that the power to *decode* source is the more fundamental and more difficult aspect of MT. Target work is far from trivial, however, and obviously poses its own set of challenges and difficulties, treatment of which requires appropriate skill. But whenever Logos developers responsible for target work are asked what is most needed to improve translation quality, invariably the answer has to do with improving source analysis.

5.1 TARGET RULES

Sets of target rules make up the **Tran** modules of the pipeline. Target rules presuppose the source rules to which they are linked and therefore do not have a SAL pattern component of their own. In other respects they resemble source rules: i.e., they have their own variety of constraint conditions and, most fundamentally, an action component. A target rule can perform additional source analysis of a nested kind when needed by a particular target language, but the principal actions concern contrastive morphological, syntactic and semantic transfer of the elements comprising the source-rule SAL pattern. All target actions in the **Parse/Trans** pipeline take the form of symbolic target parse-tree notations. These notations subsequently drive actions on literal strings in the final generation phase of machine translation, performed by the **Tgt Gen** module.

5.2 TARGET TRANSFER

As evident in Fig. 2, transfer is effected incrementally, in compositional fashion, at the point where each source constituent is considered to have been analyzed in full (consistent with the system's capabilities). Thus, for example, when a descriptive adjective in **Parse1** has been analyzed in relationship to the noun it modifies, each target will make decisions regarding (a) its transfer (e.g. whether to use the default lexical transfer or to overlay it with a transfer derived from **Semtab**), and (b) its syntactic placement in the target language. Target rules in **Parse4** will decide how and where each constituent of the source clause is to be transferred in the target (e.g., where to position a prepositional phrase analyzed as an adverb of time). Template slots in a clause-level target template are thus gradually filled and ultimately arranged in target sentence order, the target equivalent (*mutatis mutandi*) to constituents directly below **S** in the source parse. At the end of the analysis pipeline, all slots are then unloaded and the output (now essentially an ordered string of pointers to target words in the lexicon, with annotation for morphology) is passed on to the **Tgt Gen** module which, as we have said, then takes this data and synthesizes the literal target sentence.

5.3 STYLISTIC TRANSFORMATIONS

Things expressed one way in the source are often not expressible that way in the target, calling for syntactic transformations of various kinds. To a limited but not insignificant degree, the Logos Model supports the requirements of proper target style, as the examples in (31), (32), and (33) illustrate (showing unedited output). Transformations are the result of rule interaction throughout the pipeline, but chiefly to source/target rule interaction in **Parse4/Tran4**.

- (31) *The situation was alluded to by my friend in his letter.*
- (31') *Mon ami a fait allusion à la situation dans sa lettre.*
- (32) *The situation was alluded to in their letter.*
- (32') *On a fait allusion à la situation dans leur lettre.*

In (33), below, we see French and German output with radically different stylistic treatment, both from each other and from the English original. This shows the limited but real extent to which target linguists working with this Model have been able to overcome the so-called "structure preserving" tendencies said to be inherent in MT (Somers, 1992/3). In particular, note how the English ellipsis in (33) "... and their information input directly..." is handled by each of the targets. Note also the German 'subjectless clause' treatment of the main clause in (33'). All output is unedited.

- (33) *Other forms of storage media, such as magnetic cards and computer tape, can also be accessed through optional devices, and their information input directly to the system.*
- (33') *On peut également accéder à d'autres formes du support d'information, comme les cartes magnétiques et la bande pour ordinateur, par des appareils facultatifs et on peut introduire leur information directement dans le système.*
- (33'') *Auf andere Speichermedienarten, wie magnetische Karten und Magnetband kann auch durch beliebige Geräte zugegriffen werden und ihre Informationen können direkt in das System eingegeben werden.*

6.0 How Do You Deal with Complexity Issues?

We now need to review our original question regarding complexity. Complexity effects in MT concern two issues: system performance and system improvability (a function, we argue, of system maintainability). While performance will always remain an issue, steady increases in raw computer power tend to make system performance a secondary matter. Far more critical is the system improvability and maintainability issue. How strong can an MT system become? Can a system, for example, handle the many thousands of content-sensitive verb transfers to be found in any good, bi-lingual desk-top dictionary? If not, is it because of complexity effects posed when trying to make effective use of such quantities of data? The requisite linguistic knowledge is certainly available: what remains at issue is a computational approach able to deal with it.

To illustrate this complexity problem more concretely, consider what is involved in correctly translating the English word *as*. In (34), below, we see five different senses of this word, each triggered by a variety of contextual clues, and each with its own implication for German. How is such context to be specified, where are these specifications to be stored, and how are they to be applied? Can one burden a lexical entry for *as* with logic of the complexity needed in order to handle examples such as these? Can a single rule be written to cope with such phenomena? At what point in the parsing process would this lengthy rule be applied? How efficient and effective would it be? How maintainable? More than likely, because of the difficulties involved, such phenomena will simply not be dealt with beyond a certain point. This barrier typifies what we mean by complexity and how complexity limits machine translation.

German translations below are unedited output of the current commercial E-G system:

- (34a) *As you can see, he is sick.*
- (34a') *Wie Sie sehen können, ist er krank.*
- (34b) *As he is sick, we cannot ask him to work.*
- (34b') *Weil er krank ist, können wir ihn nicht bitten, zu arbeiten.*
- (34c) *As he was being given his medicine, he began to choke.*
- (34c') *Während ihm seine Medizin gegeben wurde, fing er an, zu ersticken*
- (34d) *As he began to recover his health, he realized that his wife had stood by him through difficult times.*
- (34d') *Als er anfing, seine Gesundheit zurückzubekommen, erkannte er, dass seine Frau ihm durch schwere Zeiten beigestanden hatte.*
- (34e) *As a patient, he was very cooperative.*
- (34e') *Als Patient war er sehr kooperativ.*

There are 80 patterns (rules) indexed on *as* in **Parse1**, 52 in **Parse2**, 5 in **Parse3**, and 11 in **Parse4**. All of the sentences above were dealt with at various times along the pipeline by one or more of these rules (See additional treatment of *as* in discussion of **Parse1** in 4.3.2). The examples above were chosen because they are handled relatively successfully. It is quite easy to find other *as* sentences that translate poorly, and that would require additional rules somewhere in the pipeline.

We have argued in this paper that, for all the obvious importance of linguistics, it is the computational approach that will ultimately determine how good an MT system will be--the computational approach regarding representation, storage, and rule application. We have described an approach which we feel copes optimally with these three fundamentals. We focused on the Model's computational methodology relating specifically to the question of rule application, viz., how an exceedingly rich knowledge store is to be applied, effectively and efficiently, to an unconstrained input stream without giving rise to complexity effects. It is here, perhaps, that the Model we are discussing becomes most novel.

7.0 New Version of the Logos Model (Fig. 18)

The incremental, compositional approach to analysis and transfer that characterizes this Model has meant that decisions must sometimes be made before micro-analysis of the entire sentence is complete. This can occasionally result in faulty translation. For this and other reasons, an upgraded version of the Logos Model has been under development where target transfer will not commence until source analysis is fully completed. In effect, there will be three successive parses of the sentence: (i) the macro-parse accomplished by **Res**; (ii) the micro-parse accomplished by the **Parse** modules, producing a quasi-interlingual parse tree; and finally (iii) a separate parse by the **Tran** modules, operating on this tree for the sake of each target translation.

This new arrangement will allow for improvements to source analysis independent of target considerations, and will allow targets to develop at different paces. It also anticipates introduction of a **Semantic Dictionary** that has been long under development. This new semantic resource has a much finer-grained, purely semantic taxonomy designed to supplement SAL. It will be

accessible from any point in the pipeline and its introduction should substantially improve the Model's semantic power. Finally, provision is also made for the eventual introduction of extra-sentential processing (discourse analysis).

8.0 Drawbacks and Limitations

We have described the Logos Model as a relatively effective method of coping with the complexity of natural language. But one might accuse this Model of a certain complexity in its own right. The Model's implementation, with its many thousands of rules, while not difficult to work with or maintain, seems to defy formal description. There are simply too many rules and their interplay is too multifaceted to keep total track of or to allow for easy characterization, as this paper might be said to demonstrate. One works with the system but one does not exactly master it, not in its entirety. If I may say so, it's somewhat like working with a colleague: you don't understand the colleague completely, but you figure out over time what works and what does not work, resulting in effective teamwork. When it comes to a system, complexity of this sort has sometimes been described as a state or condition residing somewhere between order and chaos. And that may be apt, but what keeps chaos in check in this Model, we may argue, is that there is an underlying, fundamental simplicity at play, the proof of which is that linguistic tasks are accomplished fairly easily and normally with good results. And generally speaking, it pleases linguists to work with such a system. When a linguist sees a problem, he or she can run the offending sentence with diagnostics and quite readily determine where the problem or deficiency lies and also, generally, how to fix it. The linguist then writes or corrects a rule or perhaps a group of rules to address the matter, often doing so more in the space of minutes than hours. And these rules, it may be recalled, find their own appropriate place in the knowledge base. To be sure, the number of new problems that arise when dealing with real world language, and thus the number of new rules needed, is literally endless, but it is rare for a developer to encounter a situation that cannot be handled in this straightforward way. In any case, if the Model exhibits a certain complexity, such complexity rarely translates into *cognitive complexity* for the seasoned developer.

If the Model strikes the reader as somewhat inelegant, in a formalistic sense, one might observe that language itself is not elegant and one might well question whether elegance is the right quality to look for in natural language processing. No one, I think, claims elegance as a characterizing feature of human sentence processing. Indeed, some claim that Chomsky (1990) himself has abandoned his original, more formalistic agenda. On our part, we have not approached natural language as a formal object for good reason; dealing with the complexity and ambiguity of natural language in the real world is like Odysseus making passage between Charybdis and Scylla, an experience one survives only with great cunning and succor from on high.

Regarding the question of how good a system based on this Model can ever get to be, we offer the following. If one in fact has a system that can absorb endless amounts of linguistic data without indigestion and can apply this knowledge effectively, then it seems likely that over time, perhaps as much as several generations, under ideal development conditions, the goal of FAHQT could eventually be reached, at least in a significant portion of the discursive language spectrum. Of course, it is not likely that this will ever happen. The many factors that must be present for it to occur—availability of money, talent, the proper sociological environment and team dedication, wise and patient management, to say nothing of the amount of time it would take, are all too problematic in themselves, quite apart from any consideration of model design and methodology. It is true that the Logos System got as far as it did precisely because, in patches of its 30-year history, many of these factors were more or less present, *mirabile dictu*. But those happy circumstances have ended, still short of the goal, and these conditions are unlikely to be repeated again. In sum, the sought-after FAHQT summit will likely always remain a distant, ultimately unrealizable goal.

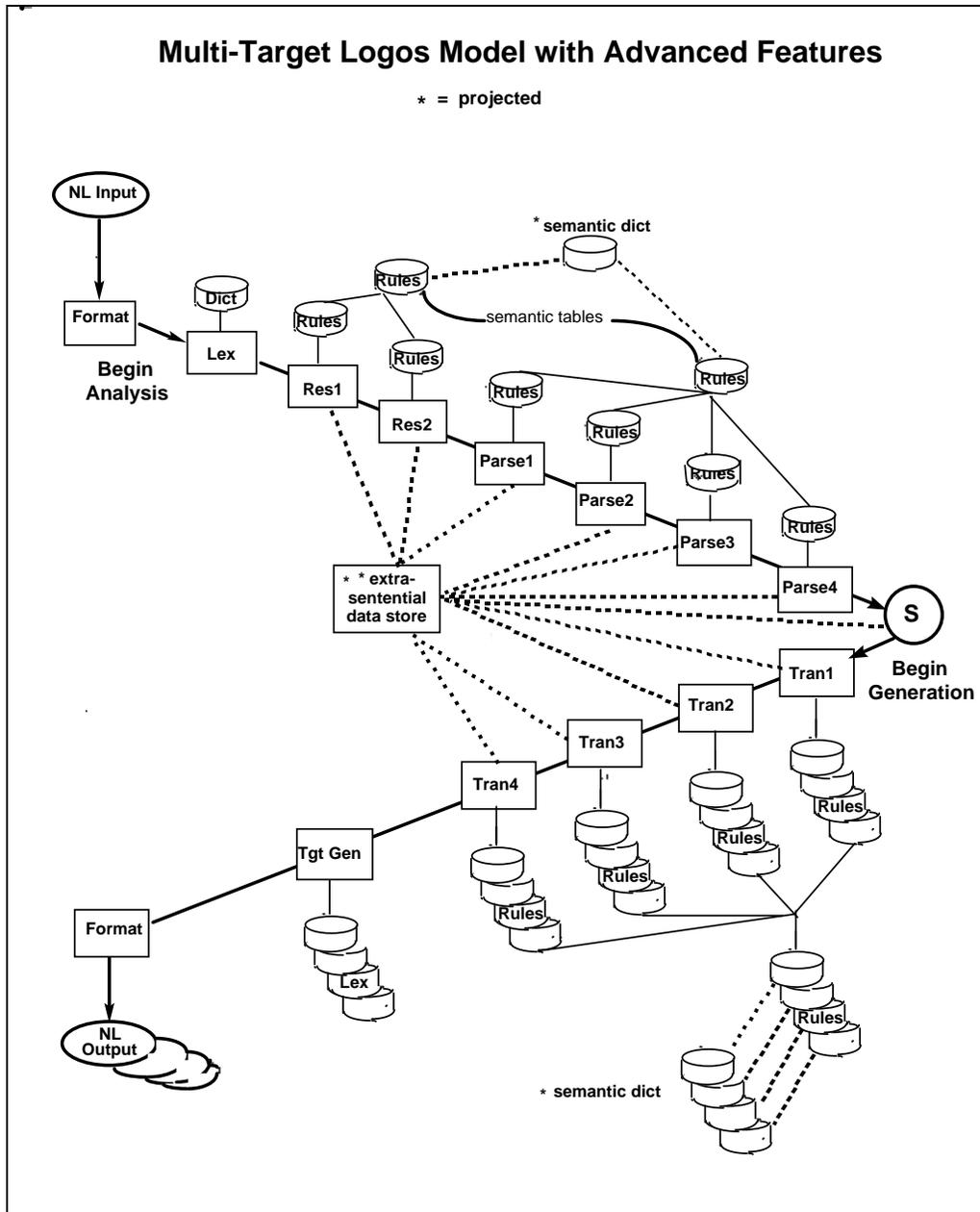


Fig. 18 – New Version of Logos Model under Development. In this new version of the pipeline, target transfer does not begin until source analysis is entirely complete. In effect, target transfer aspect of the pipeline will have as input a completed, quasi-interlingual parse tree of the source. The new version of the Logos Model has already been implemented and tested for English-German. A **Semantic Dictionary** is also planned to be introduced. This dictionary converts natural language words to a finer-grained semantic taxonomy. Designed to supplement SAL, this finer-grained taxonomy is needed for such functions as adjective and common noun disambiguation. This finer-grained taxonomy will also be accessible from any point in the pipeline. A prototype of the dictionary already exists for English and German source (each with c. 80,000 canonical entries) but software changes to pipeline modules to allow for its utilization have yet to be made. The new Model also envisions extra-sentential processing, although at present no work has yet been started in this regard. Discourse analysis will allow for handling of such issues as anaphora, ellipsis, and common noun disambiguation.

Notes

¹ Pyatt, Everett (1973), personal communication referring to Dr. John Foster's classified *1972 Annual Report of the Director, Defense Research and Engineering*. Pyatt was Assistant Secretary of the U.S. Navy.

² David Hays (1964), personal communication.