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

Lexicon-Driven Machine Translation

R. E. Cullingford, Georgia Institute of Technology, Atlanta, GA 30332 (*)
B. A. Onyshkevych, Princeton University, Princeton, NJ 08544

ABSTRACT

Machine Translation (MT) systems have historically relied upon explicit grammars in order to analyze the source text and reproduce it in the target language. In this paper, we argue for a style of MT in which the focus of processing is at the level of the *lexicon,* rather than the grammar. This approach to translation allows an analyzer to map source sentences into an interlingual form, which then can be mapped (perhaps after intermediate inferencing steps) back into target sentence(s) which are paraphrase-equivalent to the original. Advantages of the approach include: 1) the possibility for different paraphrases of the original; 2) the capability for multi-sentence expression of the original when no single word (e.g., a verb) exists in the target language which spans the same meaning complex as a word in the source; 3) a uniform approach to word sense disambiguation and anaphoric reference resolution; and, most importantly, 4) the possibility for robust handling of ungrammatical and ellipsed source text.

1. 0 Introduction: Lexicon-Driven Machine Translation

Systems designed for the Machine Translation (MT) of texts between languages have traditionally relied upon explicit grammars of both the source and target languages, in order to analyze the source text and produce well-formed target-language sentences (cf., e.g., [Tuck84]). Grammar-based systems have been reasonably successful at production-quality MT. The nature of grammatically driven processing leads to certain problems, however.

First of all, an explicit grammar tends to make the system's computation excessively *top-down*. Thus, it is usually not particularly robust under deviant (e.g., ungrammatical, telegraphic or ellipsed) input. Moreover, explicit-grammar approaches tend to be overly concerned with the *form* of the language, rather than its content. Issues of preservation of meaning between source and target texts tend to get downgraded.

In this paper, we shall argue for an alternative style of MT in which the focus of processing for both input and output texts is at the level of the *lexicon,* i.e., the words and phrases of a language, rather than its grammar. No extensive experimentation in MT has been performed within this paradigm (but see, for example, [Wile81, Lyti84]). We shall suggest, however, that the approach very naturally allows for meaning-preserving MT, and provides solutions for difficult problems such as word-sense disambiguation, anaphora resolution, the need for circumlocution when lexical equivalents of source words are not available, etc.

Language analysis, in particular, has a very strong *bottom-up* nature in the approach to be described. Such analyzers [e.g., Ries75, Birn81, Cull84, Dyer84] tend to produce fragmented meaning structures for ungrammatical or ellipsed inputs. Thus, there is the possibility for *diagnosis* of the fragments, in order to determine a reasonable reading of the input [e.g., Boot85a, Boot85b].

We illustrate the approach with a toy MT system which translates Ukrainian texts into English. The overall methodology for designing language interfaces in this paradigm is detailed in [Cull85].

---

* The research described here was performed while this author was visiting the EE&CS Department, Princeton University, Princeton NJJ08544.

2.0 Goals and Methodology

We wish to make it clear at the outset that the systems we have in mind are not "true" MT systems, but ones which *retell* or *paraphrase* the source text in the target language. We have not been concerned with modelling the professional translator's expertise in preserving the form, tone and rhetorical flourishes of the speaker, but rather with his basic bilingual, meaning-preserving translation capabilities. We have argued elsewhere [Carb81] that even this capability requires access to numerous, often culture-specific, knowledge structures (KSs) representing what the translator knows about mundane reality, the plans and goals of motivated people, special-purpose rules for specialized domains of knowledge, and the like. As was noted long ago (BarH60], these structures, and their reflections in natural language, will need to be modelled effectively before bilingual translation, and there fore true MT, will be possible.

Our methodology is derived from the so-called *Conceptual Information Processing* approach to natural language processing (NLP), associated with the names of Charniak, Schank, Wilks and Winograd, and their students. This methodology has been described in a number of places [e.g., Wino72, Wilk73, Wilk75, Scha75, Hirs84, Cull85], and we only summarize it here.

We are assuming that an *interlingua* can be designed which models the conceptual level of understanding that people possess independently of any natural language. We assume, also, that this interlingua can be encoded in terms of a relatively small number of *primitive* meaning units, and that the meanings of words in a language can be represented by structures drawn from this interlingua. For effective use by an intelligent system, the primitive units for a knowledge domain must be selected on the basis of *coverage, economy* and *orthogonality of inferencing.* The meaning structures assigned to sentences are to be composed from the meanings of the individual words in such a way as to provide *unique* and *unambiguous* representations of paraphrase-equivalent sentences (in whatever language). The representation scheme is to be *continuous:* small changes in sentence meaning should not cause large changes in the underlying representation. That is, the scheme should support the evolution of a "semantic field" of words having related meanings [cf., e.g., Mill76].

Some simple principles for deriving representations for sentences have been described in [Cull85]. The representation scheme is based on notions of *bottom-up design,* the *maximal inference-free paraphrase,* the *model corpus,* and *continuous deformation* of meaning structures. We will illustrate the first two of these processes for the following simple English sentence:

(1)
Olivia punched Muhammed in the nose.

A meaning representation for a given sentence is normally derived from what is called a *maximal inference-free paraphrase* (MIFP) of the sentence. In an MIFP, one tries to re-express the sentence in the most verbose or circumlocuted form possible, expanding it in terms of clauses (assertions) based on the primitive types of the knowledge domain. The conjunction of the clauses is to be a restatement of the literal, "exact" meaning of the original sentence. (The word "paraphrase" is used here with a stricter meaning than ordinarily.) What this means is that none of the clauses should involve a *substantive inference* from the meaning of the sentence. They should only re-express what seems to be contained in the words of the sentence themselves. This level of literal meaning is called *surface semantics.*

A "substantive inference" is an assertion drawn from the real-world context surrounding the utterance, or an auxiliary concept formed from the hearer's mental model or belief system. "Surface semantic" inferences based on the ordinary meaning of words are, however, legitimate parts of an MIFP. For example, if one hears:

Ronald took an aspirin from the bottle and ate it,

one is entitled to conclude on surface semantic grounds that "it" refers to "aspirin" rather than "bottle". This claim is based on the ordinary meaning of "eat", which demands an ingestible object; and aspirins are ingestible while bottles are not. On the other hand, concluding that

Ronald did this because he had a headache is a substantive inference. Real-world facts need to be known in order to draw such a conclusion.

Having formed the MIFP, one selects the clause that expresses the "main" or most important component of the event being described as the *kernel* of the representation. The other clauses function as *nuances* serving to distinguish this particular event from others of the same type. Obviously, the point of maximizing the number of nuances formed from a given sentence is to maximize the total number of assertions that can be distinguished.

Thus, for Sentence 1, the clauses:

(la) The female person named Olivia propelled a hand into physical contact with a nose.

(1b) This event was forceful.

(1c) The event transpired in the past.

(1d) The hand was in the form of a fist.

(1e) The hand was part of Olivia.

(1f) The nose was part of Muhammed.

(1g) Olivia was facing Muhammed, and was within arm's reach when this event took place.

all seem to be reasonable components of an exact paraphrase of (1). However, assertions such as:

(2a) Olivia was mad at Muhammed.

(2b) Muhammed had done something to make Olivia punch him.

(2c) Both parties were wearing clothes when the event took place.

are clearly *inferences* from the described behavior, that is they are only plausibly true.

Of the parts of the exact paraphrase of Sentence (l), the first, (la), can be taken to be *basic* for most purposes, since it is the one from which the most interesting consequences flow. Looking at (la), one can begin to speculate on the likely reasons for such an episode, how Muhammed might react, how relations between the two may change, etc.

The kernel assertion for Example 1, "Olivia propelled a hand into contact with a nose" can be reasonably represented in terms of the (conceptual dependency) primitive *propel,* with an underlying action in which an intentional or natural/mechanical actor applies a force to an object, with the possibility of a physical state change to it and/or another object. The verb "punch," in this sense of the word, designates a human hand as the object to which the force is applied, and clearly indicates that the hand came into *contact* with another object. (There are other possibilities: the verbs "swing at" and "throw at" are neutral about contact.)

Working on the representation of (1) from the bottom up, we need representations [Note 1] for the entities "Olivia," "Muhammed," "fist" and "nose." In a role-filler formalism [e.g., Char80], these would respectively be:

1 The primitive types used in this paper are drawn from the ERKS (Eclectic Representations for Knowledge Structures) system, an amalgam of Conceptual Dependency, Preference Semantics and Commonsense Algorithms used for illustrative purposes in [Cull85].

HUM0:
(person gender (fem) persname (Olivia))

HUM1:
(person gender (masc) persname (Muhammed))

BP0:
(bpart bptype (grasper))

BP1:
(bpart bptype (proboscis))

based on the primitive types *person* and *b(ody)part.* The symbols HUM0, etc., name the meaning structures, allowing them to be reused. The (not entirely serious) choice of "grasper" and "proboscis" for the *bptype* fields of BPl and BP2 was made to emphasize that the representation should not contain words, only indicators of function or form which are true of many entities simultaneously. Monkeys, elephants, men and robots, for example, all have functional grasping parts.

One can now propose a simple representation for (la) as follows:

EVNT0:
(propel actor HUM0 object BP0 to (physcont part BP1))

In the role-frame of a *propel,* the (actor) path is expected to be animate and the (obj) path is filled with a physical object. The filler of the (to) path in a propel-concept is required to be a primitive type expressing a relationship of *physical configuration* (e.g., location, orientation or contact) between two objects.

Of the nuance assertions, the first two, (1b) and (1c), clearly function to modify or comment on the event expressed in the kernel assertion. Assertion (1b) allows us to distinguish (1) from a sentence like "Olivia tapped Muhammed on the nose." To avoid complexities with representing relative time and physical quantities, one can simply incorporate (1b) and (1c) into the representation as follows:

EVNT0:
(propel actor HUM0 obj BP0 to (physcont part BP1)
time (PAST) quantity (FORCEFUL))

The assertion "hand was in form of a fist," (1d), is an example of an *attributional* concept, one in which an intrinsic state or attribute of an object (such as color, weight, extent, etc.) is described. The attribute *partform* (i.e., "form of a part") is used to express the state of an object that is "malleable" in some sense, that is, it can take on several forms. A version of an attributional concept expressing (1d) is:

STATE0:
(s-attr actor BP0 attr (partform val (fist)))

The primitive type *s-attr* is used in STATE0 to encode a *stative attributional* assertion about the object in the (actor) path, along the attribute dimension in the (attr) path. The (attr val) filler specifies the particular "value" of the attribute for this object. In STATE0, the filler "fist" is *not* an English word, but represents a selection from the contrast-set for the *partform* role, when expressed in conjunction with a particular object which is an intrinsic part of another, in this case a hand. (Other choices might include "flat," "cupped," "pointing," etc.) Note that this nuance allows us to distinguish verbs such as "slap" and "poke" from "punch."

Assertions (1e) and (1f) are typical examples of *physical configurational* concepts. Configurational concepts express a relationship between two or more entities. In this case, the relation is "physically part of:" that is, one object is attached to another in such an integrated way that a severe negative change in the physical state of health (for animate entities) or usability (for artifactual objects) is likely to occur if the two are separated. A representation for (1e) is as follows:

STATE1:
(p-config conl BP0 con2 HUM0 confrel (partof))

Here, the *confrel* slot contains the particular conf(igurational) rel(ationship) encoded by the form: "partof." This nuance allows us to distinguish (1) from such statements as "Olivia hit Muhammed with a rock."

Example (1f) is the same type of configurational as the above. Example (1g) is a composition of physical configurationals.

Once the MIFP has been encoded in this way, one can form "the" representation of sentence (1) in the following ERKS structure:

(3)
Olivia punched Muhammed in the nose
(ms kernel EVNT0 nuance1 STATE0 nuance2 STATEl nuance3 ...)

This is a form based on the special *m(eaning) s(tructure)* primitive, which is used in setting up dictionary entries for words in the analyzer and generator to be described. Note that the organization allows a search process looking, for example, for a word sense to express a concept to make increasingly fine-grained discriminations. First, one would look at the kernel form, then at (nuance1), (nuance2), etc. One can easily order the nuances by counting up the number of word senses that the associated form distinguishes from one another.

To make meaning structures such as (2) easier to understand, we normally use a "collapsed" form based on the kernel structure. The collapsed form for (2) is:

(4)
(propel actor HUM0
obj (bpart bptype (hand) partform (fist) partof HUM0)
to (physcont part (bpart bptype (nose) partof HUM1)))

Here the stative *partof* and *partform* conceptualizations have been "summarized" by making the associated *confrel* and *con2* fillers into a role-filler pair associated with the respective *con1* filler. For example, the filler:

(bpart bptype (nose) partof HUM1)

is a shorthand for "the nose which is part of Muhammed," i.e., (1f). The role *partof* doesn't belong to the role-frame of *bpart.*

3.0 A Lexicon-Directed Analyzer

The basic source of expectations for a surface-semantic analyzer comes from the words themselves. Certain words can predict that other meaning units are likely to occur in the sentence environment because of the semantic requirements of the meaning structure(s) they build. In Sentence (1), for example, there are high-level predictions for an animate puncher and a physical object being punched, both found in characteristic places in the sentence. What

is needed is a way of associating these predictions with the words in the system's dictionary, from which they can be summoned when the word is actually seen in the input stream.

A flexible and attractive means for. accomplishing this is to organize the analyzer as a *production system* [e. g., Newe72, Rych76]. The surface-semantic analyzer under discussion uses a system of productions, based on an extension of the notion of *requests* [Ries75], which represent positional/semantic predictions about concepts in the input stream. The productions, or test-action pairs, are maintained in a *production memory*. The test parts of the productions monitor a *working memory* which holds the current state of the concepts for words analyzed up to that point. The control of the system resides with a very simple *interpreter*. This repeatedly selects the subset of productions which apply to the phrase or clause being currently considered, and whose test conditions match the current state of working memory. Then, by a process of *conflict resolution,* the interpreter selects one production out of the conflict set, and "fires" it. The action portions of the productions generally add new concepts to the working memory, expand the concepts into greater specificity, or merge available concepts to form larger ones expressing the meaning of phrases or clauses. This cycle continues until no productions are applicable to the working memory, at which point the working memory contains the interlingual representation of the input concept.

3.1 Analyzing English

We will outline the production system cycle that analyzers of this type follow for Sentence 1. (See [Cull85] for an extensive discussion of this style of analysis.) Since the morphology of English is relatively simple, we shall defer discussion of morphology until the next section, which gives an example from Ukrainian.

The cycle begins with the dictionary look-up of words in the input stream. The look-up first attempts to find the word in the dictionary as it appears; failing that, a morphological parser attempts to strip the word to its root form, preceded by the morphemes. The dictionary look-up also attempts to match the input stream against phrases, i.e., for idiomatic or "canned" expressions.

The analyzer enters *noun-group* mode at the start of Sentence 1, because the sentence begins with a personal name. The first item in the input stream, "Olivia," is found in its root form in the dictionary. (Actually, this process is managed with a "named-person" macro form.) The production associated with "Olivia" creates a concept which is a representation of the word -- such as HUM0 above. The concept is placed on the *concept list,* or C-LIST, the working memory of available concepts. The analyzer then exits noun-group mode, because of the presence of the past/perfective fragment produced by the root-stripper. If the next word on the input stream had been "Johnson", it would have been adjoined to "Olivia" (by productions) to form a firstname-lastname concept. The end of the noun group would be signalled, in this case, as soon as a non-name was encountered on the input stream. (Arrangements are easily made for titles, appositives, and the like.)

The next item on the input stream is "punch" (we are ignoring morphology for the present). The word's concept (assuming, for the moment, that it isn't ambiguous) is added to the concept list, in the form:

(5)
(propel actor (person)
    obj (bpart bptype (hand))
    to (physcont val (bpart partof (person))))

The word also adds a number of productions to the production memory: a production which looks for a conceptual actor, and a set of productions which look for the object being punched, and the part of the object where contact was made. These productions reside in the production memory until their test parts fire. In the C-LIST, the concept for "Olivia" satisfies the semantic

test “couldbe-person” and the positional/syntactic test for “actor.” Thus, the production associated with the conceptual actor slot in Concept 5, above, is allowed to fire by the control and conflict-resolution mechanism. The action part of the production removes the “Olivia” concept from the concept list, and inserts it in the actor slot in the concept for “punch.” Since the test parts of the other productions do not test true (i.e., they are only predictions at this point), the control passes back to processing the input stream.

The next item on the input stream, “Muhammed”, is analogous to “Olivia”-- (see HUM1 above). A conceptual representation is added to the concept list, and *end-of-noun-group* is signalled by “in” from the input stream. The control selects a production from the production memory which fills in the “person being punched” slot:

(6)
to (physcont val (bpart partof (person persname (Muhammed))))

The remaining production associated with “punch” looks for a prepositional construction denoting proximity, governing a physical object constituent. The productions set up by “in” can create any of a number of structures, including the locational or physical proximity relation required by “punch.” Other constructions which “in” needs to handle include temporal location (“in April”), a “member of” relation (“in the army”), etc. When any of these productions fire, the rest of the production pool for “in” is removed from the production memory.

Having exhausted all productions testing true, control gets the next word: “the.” This word adds a “definite concept” form to the concept-list, and adds a production which awaits a following concept satisfying “couldbe-entity.” The next word is “nose.” It is placed on the C-LIST as a “bodypart” type, then end-of-noun-group is signalled by the end-of-sentence marker, the period. The conflict resolution scheme selects the most-recently-added production which tests true to fire first. In this case, the production of “definite concept” set up by “the” fires, and marks the “nose” concept as “definite reference”.

(bpart ref (def) bptype (nose) partof (nil))

Next, the “in” production for proximity to or location of a physical object tests true, and fires, picking up the “definite nose” concept. This production creates a prepositional constituent. Now the remaining production associated with “punch” tests true, and the action picks up the prepositional phrase, and merges “the nose” with the bpart concept from (6). The preposition is not needed, and is discarded by the production. The end of the input stream has been reached, and all of the productions in the production memory are quiescent (test false); thus, the resulting concept (see (4)) is returned as the conceptual representation of the input.

This extremely flexible scheme allows for the analysis of all of the forms related to Sentence 1:

Muhammed was punched in the nose by Olivia.
Muhammed was punched.
In the nose Olivia punched Muhammed.
Muhammed Olivia punched in. the nose.
Punch!

The basic process needs to be modified only slightly in order to handle imbedded uses of “punch:”'

Olivia wants to punch Muhammed
Olivia likes Ronald’s punching Muhammed
Muhammed’s having been punched was pleasing to Olivia

Finally, the all-or-nothing test-and-firing nature of requests is easily modified to allow for a notion of “best fit,” as in the Preference Semantics [Wilk73, Daws85]. Thus, we get the effect

of grammatical analysis without the need for an explicit grammar. Since we are concerned with the meanings of words anyway, we imbed the grammatical processing (which is often idiosyncratic to the words) in the lexicon itself.

3.2 Analyzing Ukrainian

How does all of this apply to the analysis of other languages? Since all known languages build upon lexical units, it makes sense to consider lexicon-driven analysis of languages other than English. It turns out that the approach works very naturally for languages as different from English as Chinese [Stut76] and, as we shall see, Ukrainian, a Slavic language.

The reason that Ukrainian was selected as the source language is the extensive morphological information conveyed by the heavy inflection of verbs, nouns, adjectives and pronouns. Therefore instead of relying on word order, as in analytic languages like English, synthetic languages like Ukrainian rely on the morphological information for specifying syntactic structure. The morphology of English is very limited, hence can be handled in a simpler fashion than the extensive inflectional system of a synthetic language.

As it turned out, the bulk of the work involved in setting up the existing analysis scheme to handle Ukrainian involved establishing a morphological parsing scheme for the inflected parts of speech. In a language like Ukrainian the morphology can be quite extensive -- verbs, in particular, can have up to five or six morphemes, such as tensing or conjugation information. Nouns, verbs, and pronouns are declined, with seven cases in the singular and plural.

The Ukrainian analysis scheme uses *exactly* the same production-system control mechanism as the English version, only it augments the dictionary look-up procedure with extensive morphological processing. After the morphemes are stripped off an inflected form, they are inserted into the input stream (as in the English case) preceding the root form which replaces the inflected input form. The morphemes have definitions very similar to the definitions of any lexical item. The morphemes modify the conceptual structure built by the root form, filling the appropriate slots in its role-filler case frame. This information is then available for checking agreement, syntactic role, etc.

As the Ukrainian language uses the Cyrillic alphabet, it is necessary to have a systematic way of transliterating from the Ukrainian Cyrillic to a form usable on the machine. Direct use of the transliteration standard, the Library of Congress system, would complicate the morphological system substantially. The reason for this is that there are a number of single-letter phonemes in Ukrainian which need to be represented by diphthongs in the Library of Congress transliteration system. In the experimental system to be described, we decided to set up a single-character transliteration standard.

The importance of the preservation of the unity of certain phonemes in the transliteration method becomes apparent in the root stripper, when checking for or undoing the effects of certain sound shifts. As an example, certain vowel shifts occur in the context of the velars, some of which need to be represented by a diphthong in the Library of Congress system. In the scheme used here, the unity of the phonemes is preserved in the transliteration system.

Additionally, because the *s* and *sh* sounds, as an example, often function similarly in phonetic shifts, and since they "alternate" (i.e. one can become the other and v.v. in certain intervocalic positions), it is convenient to represent them similarly. However, a representation like *sh* would be troublesome because *h* is phonetically irrelevant to alveolopalatal sounds like *sh* and undergoes sound shifts of its own. For a complete listing of the transliteration system we adopted, see Appendix A.

We will now briefly describe the syntax and morphology of Ukrainian, to clarify the basis of the problem example. In Ukrainian, nouns are declined, with seven cases in the singular and plural. The inflections on any one noun may be only one level deep, so a noun may be stripped to its root form. The dictionary entry for the word is based on the root form, i.e. the Nominative singular form. Each noun has a gender entered as part of the definition. Note that

morphological gender is the same as syntactic gender, but is often not the same as semantic gender.

In many of the declensions, there are instances where up to six cases can have the same form. Hence, the root stripper has to return all of the possibilities. This phenomenon forces a certain representational system upon the **case** slot in the representation of a noun. For example, the full representation for a personal name like "UriY" declined as "UriA", e.g. in the genitive and accusative singular would be:

(person persname (UriY)
    gender (masc)
    sex (male)
    case (cases casegen (sing) caseacc (sing)))

Here, the *gender* role gives the semantic gender, and the *sex* role gives the morphological/syntactic gender. A similar representation is used for pronouns, and adjectives as well, with the modification that gender agreement is included with the case.

In addition to just removing the inflectional suffix, the noun root stripper has to undo the effects of any of a number of sound shifts. Such shifts include consonant gemination, palatization, fronting, raising, and so on due to gain or loss of stress, change in juxtaposition, becoming syllabically medial instead of terminal or v.v. etc. The contexts for these changes vary in precision; sometimes relevant letters for a sound shift rule may be two letters removed from the mutator.

Despite all these complications, Ukrainian noun stripping is a relatively simple one-phase process. The noun stripper returns a string of one or more morpheme fragments, followed by the root form of the noun. These morphemes are inserted into the input stream.

The morphological parsing of verbs is a considerably more difficult task than that of nouns. The main reason for this is that a verb may have many levels of prefixes and suffixes, whereas nouns can only have one suffix. This necessitates a significantly different approach for the root stripper.

The verb morphology subfields are: tense, aspect, mood, form, number, gender, person, transitivity. Although many of these subfields are not really related to the *time* of the utterance, it was convenient to store the information in these slots. Accordingly, the root stripper reflects the different nature of verb morphology and calls a series of routines, each of which searches for a particular affix or group of affixes, and which may prepend a morphological fragment to the fragment list if new information is inferred, as well as remove the affix. The basis of this stripping is the removal of the affixes from inflected forms, from either end, from outermost to innermost, followed by an attempt to recreate the root form. The order of the calls to these routines is critical.

The morphemes are inserted into the input stream, and the productions associated with them insert them into the appropriate **time** slots of the root-form case frame. The formulation of these slots is illustrated in an example below.

Initially, we attempted a strict feature matrix approach -- features flagged + or -, resulting in unique identification of the forms. This was adequate for binary features (e.g., + or - singular) but failed for most features, like person, time, etc. The working approach uses a number of slots, each taking zero or one fillers. Two fillers per slot are not allowed. This is not necessarily the case in other inflected languages. In French, for instance, "aime" can be first or third person, but not second person, so the slot **person** would have to be treated like noun casing (below).

The productions associated with verbs in Ukrainian have a number of syntactic conditions which must be met in the test portion of the productions; these tests are generally agreement relations.

In English, subject-verb agreement does not extend beyond “he runs” vs. “they run”. In Ukrainian, there is number and gender agreement, as well as case agreement (most verbs take a nominative subject). So in order for a conceptual **actor** slot to be filled, at least these checks need to be made:

1. case (usually nominative)
2. gender
3. number
4. semantics (whether actor “couldbe” of the semantic type specified)

Every argument or adjunct of a verb has a specific case which it must take: Due to the casing, many prepositional phrases in a language like English become unnecessary, as they can be replaced by a single cased noun: ‘to Mary’ -> dative , ‘of the President’ -> genitive, The test portion of a production which wants to fill any nominal argument slot of a verb checks the **case** slot (as described above) for the potential filler to be the case required by the definition.

The following sentence will serve as an illustration of the process of root-stripping input forms, picking up morphological fragments, and the agreement/case checks. The English gloss is “Ivan hit his horse with a stick”:

(7)
konA vdaryv ivan patykom.

Although “horse” is the first item in the sentence, this is still an active (vs. passive) sentence, because of the case marking.

The first word is read in from the input stream, and sent to the root-stripper, the morphological parser. The string returned is:

sgen$ sacc$ kin

The returned string is pushed back onto the input stream, and each component is marked as having already been stripped. The production associated with the Genitive-singular particle, sgen$, tests for a following nominal concept. The Accusative-singular particle does the same. Neither of the test parts fire until the nominal concept built by the root form is available. The production associated with the noun, as in the English example above, only places the noun concept on the C-LIST. At this point the case fragment productions fire, and the action portion of the productions create a subfield of the case slot for the appropriate case, and fill the slots with the appropriate number -- singular in this situation.

The verb ‘vdaryv’ is the next item on the input stream. The morphological parser returns

perf$ past$ sing$ masc$

(No information about “person” is available in this tense.) The productions for these fragments are similar to the noun fragments -- they await a following verb-concept on the C-list. Once the verb root form, “vdar”, is reached, and the productions associated with it are placed in the active request pools, the (language-independent) conflict-resolution scheme considers one production test at a time. The first to be considered is the production which places the conceptual representation of “hit” on the C-LIST. Now the morphology fragment tests can fire, placing each of the fragments into the appropriate subfield of the **time** slot in the representation of the verb. The perf$ fragment fills the aspect slot, past$ the tense slot, sing$ the number slot, and masc$ the gender slot.

The next productions considered are the ones which attempt to find the conceptual arguments for the case frame built by the verb root. As always, the test portion of the production which fills the conceptual actor slot considers the available concepts on the C-LIST, in order of their nearness to the verb-concept. When it looks at these concepts, it first checks that the nominal concept is in the Nominative case -- that is to say, the “casenom” subfield has either

"sing" or "plur" as a filler. If so, then it takes that number filler, and checks that it agrees with the number of the verb (the filler of the "number" subfield of the **time** slot). Once these tests have been passed, the next condition considered is gender agreement -- the syntactic gender of the noun definition associated with the concept on the C-list must match the filler of the "gender" subfield of the verb **time** slot. The last test is the same semantic test as the English example, viz., is the proposed conceptual actor a "person"?

An important difference between Ukrainian and English is that there need not be any grammatical checks on word order in Ukrainian. For example, since "Olivia hit Muhammed" differs in meaning from "Muhammed hit Olivia," the production which fills the actor slot must find a noun preceding the verb (in the active voice), and the object must follow the verb. In Ukrainian the word order is not as important, since the object could easily precede the verb and actor (as in our example), since it is marked as Accusative, and the actor as Nominative.

At this point in the analysis of Sentence 7, the nominal concept available on the C-LIST is the concept for "horse," which is in the Accusative case. Thus, the actor slot-filling production fails on the first test, the case test. The production associated with the "object being hit" slot only needs to test that the concept on the list is Accusative. Since the "horse" concept is marked Accusative, and the semantics check out properly, the concept is removed from the C-LIST and inserted in the appropriate slot. The remaining productions associated with the verb which are in the pool do not find any concept at this point (i.e., they are *predictions.)*

The next item in the input stream is the actor "ivan", which gets marked Nominative in the manner described above. The concept is put on the C-LIST, where the "actor" production of the verb is able to test it. Since "ivan" is in the Nominative case, and all the other checks described above test positively, the concept is moved to the "actor" slot of the verb.

The "stick" concept is associated with the remaining word on the input stream, and is treated like the "horse" above, except for the case being marked (by the stripped-off fragments) as Instrumental. The resulting concept is returned:

(8)
```
(propel actor
      (person persname (ivan) gender (masc)
            case (cases caseacc (sing) casenom (sing)))
      obj
      (artifact case (cases caseinst (sing)) artifname (stick))
      to
      (physcont val (animal animname (horse) case (cases caseacc (sing))))
      time
      (times time2 (:perf) time6 (:sing) time9 (:masc) time1 (:past))
 )
```

As we saw, the bulk of the modifications required for changing this system from one language to another involved the morphology. Since the analyzer being described is a lexically driven system, there need be no explicit syntactic rules. Ukrainian word definitions alone took care of the syntactic differences. While our consideration of Ukrainian sentence structure is not yet complete, all grammatical sentence constructions considered so far can be captured in the definition of the root verb or auxiliary verb: for example, predicate nominatives, passives, etc. Other analyzer systems are syntax-based, and would need to have the whole syntax module rewritten for each new language. With a lexicon-driven system, the surface morphological routines need to be changed, the definitions accommodated appropriately.

In the Ukrainian analysis system, the definitions for the morphological fragments (or the agreement check) account for constructions such as negative genitives, implicit subjects, possessive constructions, intransitivization, etc.

3.3 Pronominal Reference

In the remainder of this paper, we'll be concerned with the translation of the following short passage:

(9.1) UriY Cumakuvav vozom.
Uri used to bring back salt from some salt flats in a cart,
then sell the salt.
(9.2) vin skotyvsA z mosta.
(One day) the cart rolled off a bridge
(9.3) rika zmyla viz.
The river washed the cart away.

This passage illustrates the analysis of pronouns and ambiguous words, as well as various generation problems to be described later.

The dereferencing of pronouns is a complex problem which puts the premises of our lexical approach to analysis to test. The discussion below uses the Ukrainian sentences of Passage 9 for illustration. However, *exactly* the same process applies to the analysis of English pronouns.

A pronoun may be thought of as an ambiguous concept consisting of all the co-referent concepts previously seen which "match up" with it in semantic terms. As an example, consider Sentence 9.2. The possible referents of "vin" are either "UriY" or "viz" (the cart), since gender agreement is based not on semantic gender, but on grammatical gender -- "viz" is masculine, and takes the pronoun "vin" (he). In this case, "vin" must resolve to "viz" because of the semantic requirements of "skotytysA" (rolled off). However, if Sentence 9.2 had the sense "He retired after 30 years," the pronoun "vin" would have "UriY" as the antecedent. That is, no purely syntactic process can guarantee that the correct anaphoric reference will be located.

The production associated with the definition of "vin" calls a function which finds the possible referents for the pronoun by applying a certain predicate function. In this case the predicate selects the concepts which satisfy the requirement "masculine grammatical gender". The reference function applies the predicate to concepts on a context list, called the NLP-context. In the simplest case, this list holds the concept names for all the "substantive" concepts that the analyzer has formed. The "substantive" concepts include the noun-group constituents, as well as clause and sentence-level concepts. How far back to look in the NLP-context during referent search is, of course, problematical.

When the list of possible referents is returned to the production for "vin," the pronoun's concept is replaced with the concept of the antecedent if there is only one possible referent, or with a *vel* (Latin for "non-exclusive or") of the possible referents if there are more than one. For the example of (9.2), the resulting concept would be:

```
(vel v1 (person persname (UriY) gender (masc) sex (male))
    v2 (veh vtype (cart)))
```

The casing information of the antecedents is irrelevant to the pronoun, so that is dropped. The pronoun has case information of its own, in this case Nominative-singular, which is used to mark all of the disjuncts of the *vel.* When the pronoun is disambiguated by the verb, the *vel* is replaced with the concept of the one referent selected by the verb productions from the *vel.*

After the morphology fragments are picked up by the verb, it is in the position to pick up the actor concept. Both of the disjuncts of the *vel* satisfy agreement and case requirements -- the disambiguation of the *vel* is up to the semantic restriction. The verb "skotytysA" (roll off) requires a wheeled vehicle or cylindrical object as actor in the reflexive. So the production picks up the *vel,* compresses it into the "veh" concept only, and fills the actor slot of the verb with the result.

If the size of the disjunction is greater than one after the agreement checks and the verb selectional restrictions, the whole disjunction is used to fill the slot.

3.4 Prepositional Constructions

This section will present an approach to the meaning of prepositions, and how other words can take advantage of these meanings. As an example, we shall continue with the analysis of (9.2).

The prepositional phrase "iz mosta" has the meaning "from (off) the bridge." However, the preposition "iz" can take any of a number of meanings: "from" and "(together) with," among others. These two uses, however, take different case for the following noun phrase. Thus, the productions can select between these two based on the case information associated with the noun phrase concept.

However, case alone is not able to distinguish among other meanings of the preposition. For instance, both the "(together) with" and "at the time of" usages of the preposition "iz" require arguments in the instrumental case. The semantic restrictions for the temporal reading require a time phrase such as "New Year's" or "Tuesday." So in order to assert the "at the time of" meaning of the preposition, the preposition's productions must pick up a "time" concept as the argument.

In English, the first means of selecting the appropriate reading of the preposition does not apply. So the selection is achieved by the semantics alone. As an example, consider the readings of the preposition "in" in English:

(10) Olivia was in the house
(11) Olivia was in the army
(12) Olivia graduated in 1984

It seems clear that prepositional phrases such as "in the house," "in the army" and "in 1984" have different readings. For example, "in the house" contributes a meaning fragment that says something like: "if someone (an event, for example) is looking for a particular kind of locational relationship, this phrase can build one."

The word "mosta" (bridge) is marked as genitive case, so the meaning of the preposition is chosen to be "from." The conceptual definition of the verb "roll off" seeks a "from topof" argument; the prepositional construction indicates that it is a "locational relation", so the prepositional phrase's argument is the "from" location. The argument of the prepositional phrase is inserted into the "from topof" slot in the final concept:

```
(13)
(ptrans obj
      (veh case (cases casenom (:snom)) vtype (cart))
      from (topof part
                  (struc case (cases casegen (:sgen) caseacc (:sacc))
                        structype (bridge)))
      time
      (times time2 (:perf) time6 (:sing) time9 (:masc)
            time7 (:intrans) time1 (:past))
      )
```

The primitive element *ptrans* is the conceptual dependency actional rendering events in which an animate actor (here unmentioned) causes a physical transfer of the location of a movable object.

3.5 Word Meaning Disambiguation

The selection of the intended meanings of words in context is a key problem for any language analyzer. The best-known case of the meaning selection problem is *wordsense disambiguation,* the process of choosing the correct underlying representation for a word having several senses. A wordsense disambiguation scheme, therefore, will require a model of context consisting of both the meanings of surrounding words and higher level expectations.

In order for this selection process to proceed, an analyzer needs a means of making the alternative meaning structures of a word explicitly accessible. This is the motivation for the *vel* construction introduced in Section 3.3. Nominal words normally rely on requests of other words to compress the ambiguous structure down to a single meaning; the discussion of the pronouns illustrated the mechanism. Ambiguous nouns, such as "ball" in English, are disambiguated in a similar manner.

Verbs, on the other hand, establish a set of productions to disambiguate themselves. Sentence 9.3, for example, has the verb "zmyty." The reading of the verb in this sentence is "to wash away." Other readings include the sense "to wash off (something):"

(14a) rika zmyla berih. ("The river washed away its shore")
(14b) UriY zmyv ruky ("Uriy washed off his hands")
(14c) rika zmyla Yomu ruky ("The river rinsed Uriy's hands")

When the verb concept is initially placed on the C-list, it sets up a *vel,* or disjunction of the different readings. The productions attempt to find surrounding constituents which allow a decision to be made. The first production looks for an animate entity in the actor spot. If it finds it, it *asserts* that the (14b) sense of "zmyty" may be the preferred reading. We say "may be" because the consideration process for the productions gives *all* the requests a chance to perform a disambiguation. When all of the productions have had a chance to fire, the system packages up the result: a single concept if the word has been completely disambiguated, or another *vel,* if only a partial disambiguation is possible.

Note that what this requires is a means for saving and restoring the state of the analysis process just before a production is considered and just after it fires. Since the state of the working memory is completely described by the C-list, it suffices to remember the state of the C-list as a production pool starts, then remember the revisions to the C-list (i.e., the compressed *vel* subconcepts) caused by a production's firing. Before each production is considered, the analyzer restores the C-list to the saved state. The simplicity of the production system mode! makes the management of processes such as staged disambiguation very easy.

Returning to the "zmyty" example, one can see that the presence of a "natural force" (the river) available on the C-list to fill the actor slot will allow either the (14a) or the (14c) readings. However, the (14c) reading requires an indirect object (The gloss should be "The river washed the hands for Uriy"). The (14a) production finds the natural force subject, and the "physical object" direct object it needs to be asserted. Thus, this reading of the verb is *asserted,* the *vel* is compressed to the meaning selected, and the analysis is able to be completed, resulting in the disambiguated concept:

```
(15)
(ptrans
  actor (movingwater case (cases casenom (:snom)) mwtype (river))
  obj (veh case (cases caseacc (:sacc) casenom (:snom)) vtype (cart))
  time (times time6 (:sing) time9 (:fem) time1 (:past))
)
```

4.0 Surface Semantic Machine Translation

In the rest of this paper, we are going to discuss a simplistic model of MT in which the source analysis and target generation are done on the basis of surface semantics, i.e., literal word meanings, alone. Of course, we know perfectly well that access to detailed world knowledge is necessary in order to perform much of the inferencing that is needed to produce full understanding, and therefore high-quality MT. There is an important class of simpler inferences, however, which can be supported directly at the surface semantic level. These include word meaning selection, many kinds of anaphoric reference resolution, and a process which we will call "distributed target realization."

A block diagram of a surface semantic MT system is shown in Figure 1. As can be seen, a conceptual analyzer of the kind discussed in the last section creates a meaning structure for each of the input sentences, and passes it on to a module called a *surface-semantic annotator.* This slightly modifies the concept in order to allow for certain differences in the modes of expression available in the source and target languages, then hands it to a conceptual generator for expression in the target language.

4.1 Surface Semantic Generation

Conceptual generation is the process which performs the inverse mapping from a meaning structure into a NL string. This process has several distinctive features. First of all, the system begins with a concept to be expressed, and possibly an indication of a sub-concept to be "said" first. The system is not told anything about the words or syntactic constructions to be used. This is in contrast with other models of generation [e.g., Simm72, Swar79, Wien80] in which the program's input is a syntactic phrase structure of some sort, including some or all of the words to be used.

Secondly, the generation process need not in any way be the processing inverse of the analysis process, as. for example, some purely grammatical approaches would claim. The system to be described starts (as people usually seem to) with a complete, well-formed "thought" to be "said."' Thus, the generation process is *top-down,* in a way that analysis never can be. As we have seen, analysis has a very strong bottom-up flavor of recognition, as the listener attempts to match the fragments of meaning from the words that are being heard against his conceptual expectations. A corollary of these two ideas is that literally *everything* (words, syntax, focus, connectives, etc.) that a generator of this kind chooses, in order to express the concept, will be motivated by *conceptual features* of the given concept, its conversational context, or the goal-following activities of the overall system. In many cases, therefore, the generator algorithm to be described will not be able to "say" the most fluent-sounding thing, because a conceptual reason for choosing the fluent construction is not apparent. This is the price one pays for a radically conceptual-level approach.

The generator to be described has data and control structures which are reminiscent of the analysis module discussed earlier. Just as in the analyzer, there is no explicit grammar; "syntax" is stored with the individual words of the lexicon. The generator's primary data structure is a short-term memory, called the C-LIST, consisting of concepts intermingled with words and morphological fragments. The basic control structure of the generator accesses the C-LIST in an iterative process of *looking up* word(s) to express the meaning of a concept that is currently the focus of attention (at the "front" or "top" of the C-LIST); and second, of *inserting* leftover subconcepts in appropriate places around the chosen word(s) on the C-LIST [Note 2]. The subconcepts may be accompanied by "function" words, such as prepositions or conjunctions, which serve to mark the conceptual case in the parent concept from which the subconcept came. From time to time during the basic iteration, "demon" subprocesses may intervene to prescribe a more economical means of expressing a concept than a dictionary entry may allow.

2 The elegant lookup-then-insert iteration to be described was originally developed by Mallory Selfridge, working with the author in the Intelligent Systems Design group at The University of Connecticut

Initially, the C-LIST contains a single conceptual form to be expressed. The overall generation cycle can be described by the following rules:

1) If the front of the C-LIST is empty, then there is nothing to generate; return.

2) If there is a word or fragment on the front of the C-LIST, then after some preprocessing "say" the word by saving it on a special list to be returned as the generator's result.

3) If there is a concept at the front of the C-LIST, check if any of the "demon" processes want to do anything to it. The demons, called *sketchifiers,* are described in [Cull85]. If a demon fires, go to 1) and start over.

4) When none of the demons fires, remove the concept from the front of the C-LIST, and try to find a word in the dictionary to express the concept. The dictionary entries are based on *wordsenses,* associations between words and conceptual forms. The conceptual form of an entry which matches a C-LIST item is a *template* for the item: a pattern containing roles and fillers which must be present in the item if the dictionary entry is to be used.

5) If the current concept is completely "spanned" by an entry, i.e., the template is "equal" to the entry, then replace it on the C-LIST by the word(s) of the entry [Note 3j. Otherwise, insert the fillers not matched into the C-LIST using the positional constraints stored with the wordsense found.

It is worth noting that Rule 3, above, usually embodies a decision *not* to say something that the dictionary would normally want to say. Thus, the model of generation we're presenting can be thought of as an "exhaustive" algorithm (Rules 1, 2, 4 and 5) being *restrained* by rules of type 3.

4.2 Dictionary Entries for English

To outline how a lexicon-driven generator for English works, consider the dictionary entries needed to generate the passive form of example sentence (1) "Olivia punched Muhammed in the nose," from a concept, e.g. c55, produced by the analyzer [Note 4] . In ERKS format, this would be:

```
c55:
(propel actor c27
        obj (bpart bptype ( hand) partof c27)
        to (physcont val ( bpart bytype (nose)
                                partof (person persname (Muhammed) gender (masc))))
        time (times time1 (:past))
        mode (modes mode1 (:t)))

c27:
(person gender (fem) persname (Olivia))
```

3 The pattern-matching operation implied here is implemented by a general-purpose knowledge base manager described in [Cull83]

4 We are currently investigating the extension of the generator algorithm to heavily inflected languages such as Ukrainian

focus path:
(to val partof)

The first thing to note is that the generator, if it is working in the same language as the analyzer, can use many of the same wordsenses as the analyzer did in order to arrive at c55 in the first place. Thus, for example, both the analyzer and generator definitions for "punch" can be based on wordsense wsPUNCH1 as shown in Figure 2. The wordsense entry provides a *structure-frame* (associated with "ws-structure") from which to obtain instances of the concept associated with the sense such as: a set of *constraints* on fillers proposed for the slots of the concept; a specification (not needed here) of any irregular forms of the root word; and a *focus* field to give the generator a sentential focus, a subconcept to "say first" if the concept supplied doesn't contain one. (The sentential focus in our example concept is on "Muhammed.") The generator matches C-LIST items against the "ws-structure" form, and if the entry is selected, uses the "surface-form".

The function *gdictdef* in Figure 2 adds the additional information necessary to make the wordsense wsPUNCH1 available to the generator. This function supplies the specification of syntax for sequencing words. These specifications are sensitive to the "focus" property provided with the input concept, and all use the positioning predicates *pr* (precedes) and *fo* (follows). The dictionary definition for main verbs, such as "punch", contains pairs consisting of a path to a focussed-on subconcept, and a set of positional specifications for leftover fillers. At [l] in Figure 2, for example, are the specifications to be used when the conceptual (*actor)* path is to be the sentential focus. The specification is an association list (alist) consisting of a path into the C-LIST item, and a set of predicates for placing the filler found at the end of the path on the C-LIST. At [2], for instance, is the alist for the conceptual *(actor)* filler:

((actor)
(pr)(pr (to val partof)) (pr (to val)))

What this says to do is: if the *(actor)* path in the C-LIST item matching the template is nonempty, then position it on the C-LIST *preceding* the word ("punch") spanning the item, *preceding* the filler of the *(to val partof)* path, and *preceding* the filler of the *(to val path).* Similarly, the filler of *(to val partof)* is to *follow* the word "punch," *follow* the filler of *(actor),* and *precede* filler of *(to val).* The specification for the *(to val)* filler is:

((to val)
(fo)(fo (actor))(fo (to val partof))(fo in))

This indicates that the *(to val)* filler is to follow the word, the *(actor)* filler, and the *(to val partof)* filler. It is also to follow the *function word* "in," which is simply inserted as a lexical entry on the C-LIST. Thus, the entry specifies the standard ordering of constituents for the active voice of the verb "punch."

The second association of focus and specification in a dictionary entry is assumed by the generator to correspond to the passive voice. At 3] in Figure 2, we see that the passive voice goes with the sentential focus on the *(to val partof)* filler. If c55 were expressed using the passive, the dictionary would specify an ordering of constituents on the C-LIST as follows:

(to val partof) "punch" "in" (to val) "by" (actor)

To handle "Olivia" and "'Muhammed," one has the generator's analog of the analyzer's named-person macro viz.. wsNAMED-PERSON1, as shown in Figure 3. The motivation is exactly the same: to he able to generate all the thousands of names that there are with a single, concise definition.

Figure 3 contains several new things. First of all, the word in the wordsense is "nil," the "empty" lexeme. It will have no direct realization in the sentence; it merely serves as a pivot to

position the naming information. In the call to *gdictdef,* the empty path ( ) indicates that there is no focus, as is typical of nominal concepts. The syntactic predicates position the *persname* filler preceding the empty lexeme, with the *surname* filler following it.

We also have shown some "semantic predicates" (sempreds), arbitrary Lisp code (however, without side-effects!) making special checks on the given item which are hard to encode with simply the structural information in the template. (Any matching process, for NLP or anything else, needs a structured way to "escape to Lisp" to look for things that are difficult to represent.) The predicates here use *filledp* to demand that at least one of the name slots in the input be filled. (Entries without such a sempred would allow the realization of unnamed persons, such as "a man," "he," etc.)

The dictionary entry for "nose" is contained in the following function call:

```
(gdictdef wsNOSE1)
```

Since there are no imbedded concepts to be expressed (at least in the simple cases), one just needs to declare the wordsense defined earlier.

The generator's dictionary lookup routines are responsible for selecting the word or words that span as much of the current concept as possible. Sometimes information in the concept does not map into a complete word, but is expressed by morphological changes in a root form, or by the addition of auxiliary items. Examples in English are the "s" fragment indicating possession, and the "to" that signals the infinitive form in the phrase "to graduate is my heart's desire."

When the concept sent to the dictionary contains temporal or modal information, a surface verb kernel must be built to express the *time* and *mode* slots of the concept, and the verb form must be made to agree in person and number with the focus of the sentence. Temporal and modal information is like sentential focus information in that it is not an integral part of the meaning of the concept but expresses auxiliary information. The *time* information expresses the temporal relationship of the action or state to the time of the speech act ("now"), and possibly to the time of some other event. (Our scheme for representing time is based loosely on the theory discussed in [Bruc72].) Modal information expresses the ability, intent, obligation, etc. of the speaker and/or hearer to participate in the expressed action or state. This processing is handled in our generator by the *verb kernel routines (see* [Cull85]).

A generator of English also needs to be able to create "advanced" syntactic constructions such as infinitives, gerunds, coordinated forms, etc. Most often, the availability of these realizations is signalled by characteristic *redundancies* in the concept to be "said" or in its surrounding context. Creating these forms is the responsibility of demon processes called sketchifiers. Discussion of these complex processes is beyond the scope of this paper (but see [Cull85]).

4.3 Annotating Surface Semantic Forms for Output

The conceptual representation of the source input meaning still has some traces of the source language -- in Ukrainian, for example, the tensing/casing information. These fields must be adjusted for the generator as the intermediate step between analysis and generation in the translation process.

The casing information is never used directly by the target language, since the case of each nominal position is specified by the generator word definitions, if needed. The seven cases in Ukrainian, for instance, are discarded by the intermediate process, the Annotator, since the generator knows what cases are needed in what positions in English (for pronoun casing). For the reverse translation process, i.e. from English to Ukrainian, the definitions of the verbs would have the case for each argument explicitly made available to the generator.

Other intermediate annotations need to be performed on the verb tensing information. Many of the slots of the Ukrainian tensing information are superfluous or unnecessary for

English generation. The person and number fields can be extracted from the "subject" position information. The gender agreement slot is also not necessary for English generation; in a language like French, the gender for agreement would be extracted from the subject. When there is no explicit subject in the source language, as is often the case with first and second person conjugation, and the target language needed gender for agreement, the annotator would have to mark the "implicit subject" with the gender of the agreement slot from the source language, if it were available.

In our Ukrainian-to-English example the annotator discards most of the tensing information, leaving only time and aspect. Since the tenses do not match exactly, the annotator adjusts the Ukrainian tenses to the most nearly equivalent English ones.

One of the strengths of an intermediate-language approach to MT is that the analyser is not target-language specific, and that the generator does not know about the source language. Thus if we were to expand our system to handle Ukrainian to French translation, and the French generator were available, the analyzer would not be changed at all, only the minimal intermediate annotator would need to be added.

4.4 Distributed Target Realization

The lexically-driven nature of the analyzer and generator suggest interesting advantages over syntactic (e.g., transfer) schemes. Among these is the possibility for handling of a distributed target instance -- a one-word concept in the source language can be translated into a multi-word or multi-clause realization just as easily and naturally as into a single-word realization. The fact that there is a meta-representation, which gets created by the analyzer, allows a number of more complex constructions to be analyzed.

An example of such a situation is demonstrated by Sentence 9.3, from our example passage:

(9.3) UriY Cumakuvav vozom.

The English translation of this sentence would be: "UriY (repeatedly) brought salt back from the salt flats in his cart, and sold it." A simple version of the conceptual form corresponding to the verb "Cumak" (which does not contain the content implied by "repeatedly") is:

```
(sequel
  con1 ( ptrans actor (hianimate)
              obj (hianimate)
              from (locrel)
              to (inside part (geofeat geoname (nil)
                                   geotype (saltflats)))
              inst ($drive actor (hianimate) veh (veh))
              time (times) mode (modes))
  con2 (atrans actor (hianimate)
              obj (ingobj phase (granular) ingtype (salt))
              to ( poss part (hianimate)))
  con3 (ptrans actor (hianimate)
              obj (ingobj ingtype (salt))
              inst ($drive actor ( hianimate) veh (veh))
              to (locrel)
              from (inside part (geofeat geoname (nil)
                                   geotype (saltflats))))
  con4 (dual con1 (atrans actor (hianimate)
                    obj (ingobj ingtype (salt))
                    from (poss val (hianimate))
```

```
                )
        con2 (atrans actor (hianimate)
                obj (money)
                to (poss val (hianimate)))
        )
)
```

This conceptual representation of the verb captures much of the nature of the verb; e.g. the "$drive" fillers of the "inst" slots represent a *script* for the activity of "driving", that is a whole sequence of events which is captured by the one verb. Many of the slots of the representation refer to the same concept. For example, the actor of the "driving down to the salt flats" (con1) is the same as the actor of the "driving back from the salt flats" (con3). These equivalences can be specified in the definition explicitly, so that instead of having a copy of the same concept in the multiple slots, the same actual concept appears in as many places in the concept as necessary.

Sentence 9.3 is analyzed into the following concept:

```
(16)
(sequel con1
    (ptrans actor (person persname (UriY) gender (masc)
                    case (cases caseacc (:sacc) casenom (:snom)))
        obj c54
        from (locrel)
        to (inside part (geofeat geoname (nil) geotype (saltflats)))
        inst ($drive actor c54
            veh (veh case (cases caseinst (:sinst)) vtype (cart)))
        time
        (times time2 (:imprf) time1 (:past) time6 (:sing) time9 (:masc))
        mode (modes model (:t)))
    con2                                    *
    (atrans. actor c54
        obj (ingobj phase (granular) ingtype (salt))
        to ( poss part c54))
    con3
    (ptrans actor c54 obj c54
        inst ($drive actor c54 veh c120)
        to c69 from c70)
    con4
    (dual con1 (atrans actor c54 obj (ingobj ingtype (salt))
                from (poss val c54))
        con2 (atrans actor c54 obj (money) to (poss val c54)))
    )
```

Here any concepts specified merely as 'c##' are subsequent references to concepts already expanded in full. For instance, the actors of all the sub-concepts are the same concept -- c54.

As we see the power of a lexically-driven analysis is such that there is minimal language-specific information encoded, yet the range of analyzable sentences includes many examples of very troublesome syntactic constructions and lexical meaning.

5.0 An Example

In this section, we give an annotated run of our toy MT system working on the short passage discussed in the last section. We threw the system together out of existing tools, so it is miles away from any sort of "production" capability. Indeed, as will be seen, there are some problems with the translation produced. The system does serve to illustrate the major points we have been making, however.

The MT system is a collection of Franz Lisp programs running under BSD Unix 4.2 on a Pyramid 90x minicomputer [Note 5]. APE is the analysis module; GEN is the generator. The system picks source sentences out of a file one at a time, analyzes them, annotates the resulting concept, then generates an English realization of the annotated concept. This is an actual transcript of the running program, heavily edited for readability. Lines beginning with ";" are comments.

```
TOOL-> (perklasty souce1)
Tue Jun 25 13:42:53 1985
Translating file: source1
Sentence: (UriY Cumakuvav vozom)

; the analyzer begins on the "noun group" implied by "UriY:" the root word
; preceded by its affixes
Entering ng mode

APE: new word is sacc$
APE: considering topreq pool: ap2 (ar3) sacc
Executed ar3
Available: nil

APE: new word is snom$
APE: considering toprq pool: ap$ (ar5) snom$
Executed ar5
Available: nil

APE: new word is UriY
APE: considering toprq pool: ap6 (ar&) UriY
Executed ar7
Acailable: (c54)
C54: (person persname (UriY) gender (masc))

; one of the "Cumak" afixes forces the system out of noun-group mode
Leaving ng mode

APE: new word is imperf$

APE: considering gapreq pool: ap5 (ar6) snom
Executed ar6
Available: (c54)

APE: considering gapreq pool: ap3 (ar4) sacc$
Executed ar4
Available: (c54)
; after the affixes modify the base concept, we have:
C54: (person persname (UriY) gender (masc)
          Case (cases caseacc (:sacc) casenom (:snom)))

; this adds the completed concept to the NLP context
```

5. These programs are collectively called the NLP Toolkit. They are available for teaching purposes in concert with [Cull85].

```
APE: considering shipreq pool: ap1 (ar1 ar2)  NGP
APE: shipping noun group result: c54
Executed ar2
Available: (c54)

; now the system activates the verb-affix lexemes
; c61 is the imperfective fragment
APE: considering peekreq pool: ap8 (ar11)   imperf$
Executed ar11
Available: (c61 c54)

APE: new word is past$
APE: considering topreq pool: ap12 (ar16)   past$
Executed ar16
Available: (c61 c54)

APE: new word is sing$
APE: considering topreq pool: ap14 (ar18)   sing$
Executed ar18
Available: (c61 c54)

APE: new word is masc$
APE: considering topreq pool: ap16 (ar20)   masc$
Executed ar20
Available: (c61 c54)

APE: new word is Cumak
APE: considering topreq pool: ap18 (ar22)   Cumak
Executed ar22>
Available: (c65 c61 c54)
; c65 has the structure shown in the last section...

; now the fragments mark "Cumak"'s sequel-concept
APE: considering gapreq pool: ap17 (ar21)   masc$
Executed ar21
Available: (c65 c61 c54)

APE: considering gapreq pool: ap15 (ar19)   sing$
Executed ar19
Available: (c65 c61 c54)

APE: considering gapreq pool: ap13 (ar17)   past$
Executed ar17
Available: (c65 c61 c54)

APE: considering gapreq pool: ap9 (ar13)   imperf$
Executed ar13
Available: (c65 c54)

APE: considering gapreq pool: ap20 (ar24)   Cumak

; one of "Cumak"'s requests picks up a conceptual actor
APE: considering gapreq pool: ap19 (ar23)   Cumak
Executed ar23
```

```
Available: (c65)

; now the "noun group" for "a cart"
Entering ng mode

APE: new word is sinst$

APE: new word is viz
APE: considering topreq pool: ap24 (ar29)   viz
Executed ar29
Available: (c120 c65)

Leaving ng mode

; "pr" is "period." the sentence terminating punctuation lexeme
APE: new word is pr

APE: considering gapreq pool: ap23 (ar28)   sinst$
Executed ar28
Available: (c120 c65)

: "a cart" goes into NLP context
APE: considering shipreq pool: ap21 (ar2o ar26)   NGP
APE: shipping noun group result: c120
Executed ar26
Available: (c120 c65)

: then Cumak gets its instrumental vehicle
APE: considering gapreq pool: ap20 (ar24)   Cumak
Executed ar24
Available: ( c65)

: after a bit more, the result
APE: sentence:
(UriY Cumakuvav vozom pr)
result:(c65)
c65:
(sequel con1
        (ptrans actor (person persname (UriY) gender ( masc)
                                case (cases caseacc (:sacc) casenom (:snom)))
                obj c54 from (locrel)
                to (inside part (group typmem (geofeat geotype (saltflat))))
                inst ($drive actor c54
                        veh (veh case (cases caseinst (:sinst)) vtype (cart)))
                time
                (times time2 (:imperf) time1 (:past) time6 (:sing) time9 (:masc))
                mode (modes model (:t)))
        con2
        (atrans actor c54 obj (ingobj phase (granular) ingtype (salt))
                to ( poss part c54))
        con 3
        (ptrans actor c54 obj c54 inst ($drive actor c54 veh c120)
                to c69 from c70)
        con 4
```

```
(dual con1
      (atrans actor c54 obj (ingobj phase (granular) ingtype (salt))
            from (poss val c54))
      con2
      (atrans actor (nil) obj (money) to (poss val c54)))
compnum
(4))
```

Now the annotator gets in to make the concept ready for realization in English. First, it maps the tensing information and clears the Ukrainian surface cases. Then, as a translator would have to, it sees that there is no lexical equivalent in English to the complex "Cumakuvaty" concept. It looks for an economical realization using two or more verbs. The result of this will be a concept expressing "bring back then sell."

Mapping tensing info in: c65

Clearing surface cases in: c65

Seeking realization of conrel: c65

```
; the resultant concept
c102: (sequel conl c103 con2 c134 compnum (2))

; this is "bring back"
cl03:
(sequel conl
      (ptrans actor (person persname (UriY) gender (masc)) obj c54 from (locrel)
            to (inside part (group typmem (geofeat geotype (saltflat))))
             inst ($drive actor c54 veh (veh vtype (cart)))
             time (times time2 (:prog) timel (:past) time6 (:sing) time9 (:masc))
             mode (modes model (:t)))
      con2
      (atrans actor c54 obj (ingobj phase (granular) ingtype (salt))
            to (poss part c54))
      con3
      (ptrans actor c54 obj c54 inst ($drive actor c54 veh c120)
            to c69 from c70)
     compnum (3))

; this is "sell"
c134:
(dual conl (atrans actor c54 obj (ingobj phase (granular) ingtype (salt))
                from (poss val c54))
     con2 (atrans actor (nil) obj (money) to (poss val c54)))
     compnum (2))

; now the generator gets in with the above concept
GEN: top of c-list
(sequel ..... )
```

```
 ; no sentential focus has been specified for the concept, so the generator
 ; picks a default
 get-D focus: assuming default focus conl for c141

 ; the dictionary is probed
DICT to match: (sequel....)
 ; and returns the entry corresponding to "then"
DICT result: Dwl91   (nil)

: it inserts the "bring back" concept preceding "then" on the C-list
inserting ( nil c142 (c141 con2)) at:
((fo c141) (fo (funcword then)) (fo (c141 conl)) (pr (c141 con3))
   (pr(c141 con4)) (pr(c141 con5)) (pr(c141 con6)))

; and "sell" following
inserting (nil c155 (c141 conl)) at:
((fo c141) (pr(c141 con2)) (pr(c141 con3)) (pr(c141 con4))
   (pr(c141 con5)) ( pr (c141 con6)))

; "bring back" reaches the front
GEN: top of c-list
(sequel con3 (ptrans actor (person persname (UriY) gender (masc)) obj c144....))

: a sketchifier notices that a vehicular-instrument phrase is available...
VEHINST: pushing vehicle c158

; the result of the dictionary look up...
; the Ukrainian imperfective has been mapped into the English progressive, the
; nearest available realization...
DICT result: Dw190   (was bringing back)

; the leftover subconcepts are inserted...
inserting (nil c162 (c155 conl to part)) at:
((fo c155) (fo (c155 conl actor)) (fo (cl55 con2 obj))
(fo (cl55 conl from)) (fo (funcword from)))

inserting (nil c160 (c155 conl from)) at:
((fo c155) (fo (c155 conl actor)) (fo (c155 con2 obj))
(fo (funcword to)) (pr (c155 conl to part)))

inserting (nil c166 (c155 con2 obj)) at:
((fo c155) (fo (c155 conl actor)) (pr (c155 conl to part))
(pr (c155 conl from)))

inserting (nil c144 (c155 conl actor)) at:
! ( pr c155) ( pr ( c155 conl from)) (pr(c155 conl to part))
( pr ( c155 con2 obj)))

: as a result, Uri gets to the front...
GEN: top of c-list
(person persnarne (UriY) gender ( masc))

: the entity-reference demon notices this named-definite concept, and adds it
   to the NLP-context
```

entref: def-izing unique entity c187
entref updating nlp context with c187

: the Name sketchifier extracts the entity's name...
Name Maker: examining (person ref c190 persname c188 gender c189)

; and the first words of the sentence "appear"
utterance is (UriY)
utterance is (UriY was bringing back)

; now the salt gets to the front...
GEN: top of c-list
(ingobj phase (granular) ingtype (salt))

; the entity-reference demon notes the phase, and arranges a mass-noun determiner
entref: indef-izing mass noun c191
entref updating nlp context with c191

DICT to match: (ingobj ref (!indef) phase (granular) ingtype (salt))
DICT result: Dwl89 (salt)

GEN: using
(salt)

inserting (nil c194 (c191 ref)) at:
((pr c191))

; the mass-noun determiner is realized
GEN: top of c-list
(!indef)

DICT to match: (!indef)
DICT result: Dw148 (some)

GEN: using
(some)

: more words are "said"
utterance is (UriY was bringing back some)
utterance is (UriY was bringing back some salt)
utterance is (UriY was bringing back some salt to)

; the origin/destination of the "bring back" (which cannot be expressed in Ukrainian)
; reaches the front
GEN: top of c-list
(locrel)

DICT to match: (locrel)
DICT result: Dw206 (somewhere)

GEN: using
(somewhere)

utterance is (UriY was bringing back some salt to somewhere)

utterance is (UriY was bringing back some salt to somewhere from)

: "salt flats" is conceptually a physical group whose "typical member" is
: an individual flat
GEN**:** top of c-list
(group typmem (geofeat geotype (saltflat)))

: the group gets the default determiner: plural indefinite
entref: indefizing group: c195
entref updating nlp context with c195

: and the generic "group" entry (which pluralizes the typmem subconcept)
: is looked up
DICT to match: (group ref (!indef) typmem (geofeat geotype (saltflat)))
DICT result: Dwl40 (nil)
({(nil c196 (c195 typmem)) (pr c195) (fo (c195 grpnum))))

GEN: using
(nil)

: the pluralized typmem gets inserted...
inserting (nil c196 (c195 typmem)) at:
;(pr c195) (fo (c195 grpnum)))

GEN: top of c-list

: geofeat ref (!indef) compnum (plural) geotype (saltflat))

entref updating nlp context with c196

: and looked up...
DICT to match: (geofeat ref (!indef) compnum (plural) geotype (saltflat))
DICT result: Dwl88 (salt flats)
(((nil c198 (c196 ref)) (pr c196) (pr (c196 geoname))))

GEN: using
(salt flats)

: "some" gets produced again, this time as the standard English plural-indefinite
: determiner
GEN: top of c-list
(!indef)
DICT to match: (!indef)
DICT result: Dwl48 (some)

GEN: using
(some)

: more words
utterance is (UriY was bringing back some salt to somewhere from some)
utterance is (UriY was bringing back some salt to somewhere from some salt flats)
UUP ranee is (UriY was bringing back some salt to somewhere from some salt flats)
utterance is (UriY was bringing back some salt to somewhere from some salt flats in)

: now the cart-instrument gets to the front from where vehinst pushed it, a while

```
; back
GEN: top of c-list
(veh vtype (cart))

; vanilla determiner
entref indef-izing entity as default c200
entref updating nlp context with c200
DICT to match: (veh ref (indef) vtype (cart))
DICT result: Dwl87  (cart)
(((nil c202 (c200 ref)) (pr c200)))

GEN: using
(cart)

inserting (nil c202 (c200 ref)) at:
((prc200))

GEN: top of c-list
(indef)

DICT to match: (indef)
DICT result: Dw150  (a)

GEN: using
(a)

; "then" is the realization of the original sequel
utterance is (UriY was bringing back some salt to somewhere from some salt flats in a
                    cart then)

: here is "sell"
; note the absence of tame/mode information; this should have been mapped from
; the "bring back" concept, we don't understand how to do this exactly.
GEN: top of c-list
(dual conl
  (atrans actor (person persname (UriY) gender (masc))
          obj (ingobj phase (granular) ingtype (salt)) from (poss val c144))
  con2 (atrans actor (nil) obj (money) to (poss val c144)))

; as a result, we get back the root form, instead of the past-progressive
DICT to match: (dual...)
DICT result: Dw315  (sell)

GEN: using
(sell)

; the actor and object of "sell" get positioned
inserting (nil c147 (c142 con1 obj)) at:
((fo c142) (pr(c142 con2 actor)))

inserting (nil c144 (c142 conl actor)) at:
((pr c142) (pr (c142 con2obj)))

GEN: top of c-list
```

(person persname (UriY) gender (masc))

; the recently mentioned animate actor is found in the NLP-context
; the demon sees that the pronoun can be used without confusion
entref: pron-izing recently seen c203
entref updating nlp context with c206
DICT to match: (person gender (masc) case (subj) ref (pron))
DICT result: Dw231 (he)

GEN: using
(he)

utterance is (UriY was bringing back some salt to somewhere from some salt flats in
a cart then he sell)

GEN: top of c-list
(ingobj phase (granular) ingtype (salt))

; the demon selects a definite determiner for "salt" here, rather than "it,"
; because of a possible confusion with the prior "cart"
entref: def-izing old entity c209
entref updating nlp context with c209

DICT to match: (ingobj ref (def) phase (granular) ingtype (salt))
DICT result: Dw189 (salt)

GEN: using
(salt)

GEN: top of c-list
(def)

DICT to match: (def)
DICT result: Dw149 (the)

GEN: using
(the)

; the output for the first Ukrainian sentence:
UriY was bringing back some salt to somewhere from some salt flats in a cart
then he sell the salt.

Sentence 9.2 contains a. pronoun with two possible referents as far as surface agreement is concerned: "UriY" and "cart." An ambiguous concept is placed in the C-list, and "rolled off" selects the appropriate surface-semantic referent.

: the second sentence
Sentence: (vin skotyvsA z mosta pr)

: analysis proceeds as above: when the root form of the pronoun "he"

; is processed, the system creates a vel of the possible referents.
; the pronoun lexeme sets the surface case of each subcon to singular
; nominative, its form for this sentence...
APE: new word is _vin
APE: considering topreq pool: ap31 (ar39) _vin

APE: referents for surface-masculine pronoun: (c120 c54)
Executed ar39
Available: (c254)
c254:
(vel v1 (veh case (cases casenom (:snom)) vtype (cart))
      v2 (person case (cases casenom (:snom)) persname (Uriy) gender (masc)))

; the NLP-context update fails because an unambiguous referent for "he"
; isn't yet available
APE: considering shipreq pool: ap33 (ar41)   NGP

; after the affixes of "skot" have been read, the root word gets in...
APE: new word is skot
APE: considering topreq pool: ap46 (ar54)   skot
Executed ar54
Available: (c281 c275 c254)
c2Sl:
(ptrans obj (veh) from (topof) time (times timel (:pres)) mode (modes model (:t)))

APE: considering gapreq pool: ap48 (ar56) skot

; after the skot-affixes have fired, marking the concept appropriately,
; one of "skot"'s productions selects the vehicle sense of "he"
; to fill the conceptual object with the appropriate surface-semantic filler
APE: considering gapreq pool: ap47 (ar55)   skot

VEL: disambiguated c254 as c262
(veh case (cases casenom (:snom)) vtype (cart);)

Executed ar55
Available: (c281)

; the reflexive form of "it" gets in...
APE: new word is iz
APE: considering topreq pool: ap49 (ar57)   iz
Executed ar57

Available: (c294 c281)

APE: considering gapreq pool: ap48 (ar56)   skot

APE: considering gapreq pool: ap45 (ar53)   self$

APE: considering gapreq pool: ap50 (ar58)   iz

Entering ng mode

; The argument of the preposition takes genitive case.

; The fragment "fo" is inserted into the input stream whenever
; a genitive fragment is stripped off: this fragment
: attempts to create a "possessive" construction
; if the genitive noun is not picked up by a preposition or the
; verb, and there is a following available nominal.
APE: new word is fo
APE: considering topreq pool: ap52 (ar61) fo

Executed ar61

Available: (c299 c294 c281)

APE: considering peekreq pool: ap53 (ar62) fo

APE: new word is sgen$

APE: considering peekreq pool: ap53 (ar62) fo

APE: considering topreq pool: ap54 (ar63) sgen$

Executed ar63

Available: (c299 c294 c281)

APE: considering peekreq pool: ap53 (ar62) fo

APE: new word is sacc$

APE: considering peekreq pool: ap53 (ar62) fo

APE: considering topreq pool: ap56 (ar65) sacc$

Executed ar65

Available: (c299 c294 c281)

APE: considering peekreq pool: ap53 (ar62) fo

APE; new word is mist

APE: considering peekreq pool: ap53 (ar62) fo

APE: considering topreq pool: ap58 (ar67) mist

Executed ar67

Available: (c302 c299 c294 c281)

APE: considering peekreq pool: ap53 (ar62) fo

Executed ar62

Available: (c302 c294 c281)

Leaving ng mode

APE: new word is pr
APE: considering gapreq pool: ap57 (ar66) sacc$

Executed ar66
Available: (c302 c294 c281)

```
APE: considering gapreq pool: ap55 (ar64)  sgen$

Executed ar64
Available: (c302 c294 c281)

; as there is no following nominal, and the genitive-case noun "mosta"
; is picked up by the preposition, "fo"'s production never fires.
APE: considering gapreq pool: ap59 (ar68)  fo

APE: considering shipreq pool: ap51 (ar59 ar60)   NGP

APE: shipping noun group result: c302

; the result for the second sentence
APE: sentence:
(vin skotyvsA z mosta pr)
result:(c281)
c281:
(ptrans obj (veh case (cases casenom (:snom)) vtype (cart))
        from (topof part (struc case (cases casegen (:sgen) caseacc (:sacc))
                                structype (bridge)))
        time (times time2 (:perf) time6 (:sing) time9 (:masc) time7 (:intrans)
                timel (:past))
        mode (modes model (:t)))


; the annotator does its work
Mapping tensing info in: c281

Clearing surface cases in: c281

; the result:
(ptrans obj (veh vtype (cart))
        from (topof part (struc structype (bridge)))
        time (times time6 (:sing) time9 (:masc) time7 (:intrans) timel (:past))
        mode (modes mode1 (:t)))

; the generator is called to express this concept
GEN: top of c-list
(ptrans...)

; because no sentential focus is set, the generator selects the passive realization
; of "roll off." Note that the conceptual actor is empty. We know that the likely
; actor is "gravity," but we can't conclude this on surface-semantic grounds
; alone! There really should be an annotation rule for sentential focus; indeed.
; the analyzer should record the constituent that was "said" first.
GEN: using
(was rolled off)

; the NLP-context mechanism is not managed correctly here! Indeed, it was not
: really designed to support cross-lingual referent search. As a result, the
; entity reference demon selects "it" to realize "cart," even though there
: is a prior host of it-concepts (salt, saltflats, somewhere) that could be
; confused with it...
```

GEN: top of c-list
(veh vtype (cart))

entref: it-izing recently seen c325
entref updating nlp context with c328
DICT to match: (entity ref (pron))
DICT result: Dw204 (it)

GEN: using
(it)

utterance is (it was rolled off)

GEN: top of c-list
(struc structype (bridge))

entref indef-izing entity as default c330
entref updating nlp context with c330
DICT to match: (struc ref (indef) structype (bridge)) -
DICT result: Dw359 (bridge)

GEN: using
(bridge)

; the result is not fluent, but reasonably understandable...
It was rolled off a bridge.

The third sentence contains an example of an ambiguous verb-concept. The productions of the verb will compress the vel to the appropriate sub-concept when they see "the cart." Interestingly, when the pronoun "he" is used here, we have a surface-semantically ambiguous concept: "the river washed the cart away" vs. "the river washed Uriy off." The analyzer will produce a vel of these concepts as its result, relying on diagnostic processes from general world knowledge and current context (viz., bridges often cross rivers, things falling off bridges land on/in the thing crossed, etc.) to select the likeliest meaning. Lots of deep problems here!

Sentence: (rika zmyla viz pr)

; processing proceeds normally until the root-form of "zmyla" gets in...
APE: considering topreq pool: ap72 (ar84) zmy
Executed ar84
Available: (c357 c349)
;"a river"
c349:
(movingwater case (cases casenom (:snom)) mwtype (river))

; the vel of "wash away" and "wash off"
c357:
(vel vl (ptrans actor (animate) obj (pobj) time (times timel (:pres))
mode (modes model (:t)))
v2 (propel obj (ingobj-f ingtype (water)) to (physcont-f)

```
                time (times time1 (:pres)) mode (modes model (:t))))

 ; eventually the root word "cart" gets in...
 APE: new word is viz
 APE: considering topreq pool: ap79 (ar94)   viz
 Executed ar94
 Available: (c383 c357 c349)
 c383:
 (veh vtype (cart))

 ; and a production of "zmy" make the appropriate selection: "wash away"
 APE: considering velreq pool: ap73 (ar85 ar86 ar87)  zmy

 VEL: possible disambiguation: c357 as c373

 Executed ar87
 Available: (c357 c349)

 VEL: disambiguated c357 as c373
 (ptrans obj (veh case (cases caseacc (:sacc) casenom (:snom)) vtype (cart))
        time (times time1 (:pres)) mode (modes model (:t)))

 ; the result:
 APE: sentence:
 (rika zmyla viz pr)
 result:(c373)
 c373:
 (ptrans actor (movingwater case (cases casenom (:snom)) mwtype (river))
         obj (veh case (cases caseacc (:sacc) casenom (:snom)) vtype (cart))
         time (times time6 (:sing) time9 (:fem) time1 (:past))
         mode (modes model (:t)))

 ;after annotation, the generator is called
 GEN: top of c-list
 (ptrans...)

 GEN: using
 (washed away)

 GEN: top of c-list
 (movingwater mwtype (river))
 GEN: using
 (river)

 GEN: top of c-list
 (indef)
 GEN: using
 (a)

 ; surface semantics is insufficient to infer "the river" which must be
 ; under "a bridge"!!!
 utterance is (a river)
 utterance is (a river washed away)
```

```
GEN: top of c-list
(veh vtype (cart))

; can't use "it" because of "river"
entref: def-izing old entity c407
entref updating nlp context with c407

GEN: using
(cart)

GEN: top of c-list
(def)
GEN: using
(the)

; and the result
A river washed away the cart.
t
TOOL->
```

6.0 Conclusions

We have argued that lexicon-directed MT is a viable alternative to standard explicit-grammar approaches. As our simplistic experimental system suggests, many difficult problems in meaning-preserving translation are very naturally approached in this style of analysis/generation. Since the interface does create/map out of interlingua] forms, it can easily be adapted to work with an expert reasoning/database system containing models of world knowledge and MT expertise, which will be needed for the fully automatic high-quality MT sytems that we really would like to build.

REFERENCES

[BarH60]
Bar-Hillel, Y., "The Present Status of Automatic Translation of Languages." In F. L. Alt (ed.), *Advances in Computers (Vol. I).* Academic Press, New York, 1960.

[Boot85a]
Booth S.L., Cullingford, R.E., & White, N.H., "DESI, A Robust Natural Language Interface to a Decision Support System," *Proc. IEEE/ACM Conf. on Software Tools,* Sheraton Center Hotel, NYC, NY. April 1985.

[Boot85b]
Booth, S.L., & Cullingford, R.E., "How to Make a Natural Language Interface Robust," *Proc.1985 IEEE Int'l Conference on Cybernetics & Society,* Tucson, Arizona, November 1985 (in press).

[Bruc72]
Bruce, B.. "A Model for Temporal References and Its Application in a Question Answering Program." *Artificial Intelligence,* Vol. 3, 1072, 1-25.

[Carb81]
Carbonell, J.G., Cullingford, R.E. and Gershman, A.V., "Steps Toward Knowledge-Based Machine Translation," *IEEE Trans. on Systems, Man & Cybernetics,* Vol. PAMI-3, No. 4, pp. 376-392, July 1981.

[Char72]
Charniak, E., "Towards a Model of Children's Story Comprehension," (Ph.D. Diss.) AI Laboratory TR-266, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1972.

[Cull79]
Cullingford, R.E., "Pattern Matching and Inference in Story Understanding," *Discourse Processes,* Vol. 2, No.4, pp. 319-334, November 1979.

[Cull81]
Cullingford, R. E. et. al., "Towards Automating Explanations," *Proc. 1981 Int. Joint Conf. on Artificial Intelligence,* pp. 432-438, Vancouver, B.C., August 1981.

[Cull83]
Cullingford, R. E. and Joseph, L. J., "A Heuristically 'Optimal' Knowledge Base Organization Technique," *IFAC Automatica,* Vol. 19, No. 6, Nov-Dec, 1983.

[Cull84]
Cullingford, R. E. and Pazzani, M. J., "Word Meaning Selection in Multimodule Language Processing Systems," *IEEE Trans. PA&MI,* Vol. PAMI-6, No. 4, pp. 493-509, (July).

|Cull85]
Cullingford, R.E., *Natural Language Processing: A Knowledge Engineering Approach,* Rowman & Allanheld, Totowa, NJ, 1985 (in press).

[Daws85]
Dawson, B., "A Preference-Based Conceptual Analyzer," Dept. of EE&CS Research Report (M.S. thesis). University of Connecticut, Storrs, CT 1985.

[Gers79]
Gershman, A.V., "Knowledge-Based Parsing," (Ph.D. diss.) Research Report No. 156, Department of Computer Science, Yale University, New Haven, CT, 1979.

[Lyti84]
Lytinen, S.L., "The Organization of Knowledge in a Multi-Lingual, Integrated Parser," (Ph.D. diss.) Research Report No. 340, Department of Computer Science, Yale University, New Haven, CT, 1979.

[Mill76]
Miller, G.A., & Johnson-Laird. P.N., *Language and Perception,* Belknap/Harvard Press, Cambridge, MA, 1976.

[Scha73|
Schank, R.C., "Identification of Conceptualizations Underlying Natural Language." In R. C. Schank and K. M. Colby (eds.), *Computer Models of Thought and Language,* Freeman, San Francisco, 1973.

[Scha75]
Schank, R. C. (ed.), *Conceptual Information Processing*, North Holland, New York, 1975.

[Simm72]
Simmons, R., and Slocum, J., "Generating English Discourse from Semantic Networks," *Comm. ACM,* Vol. 15, No. 10, 1972.

[Swar77]
Swartout, W., "A Digitalis Therapy Advisor with Explanations," *Proc. 5th International Joint Conf. on AI,* Cambridge, Massachusetts, 1977.

[Tuck84]
Tucker, A. and Nirenburg, S., "Machine Translation: A Contemporary View," *Annual Review of Information Science and Technology,* Vol. 19, Chapter 5, American Society for Information Science.

[Wein80]
Weiner, J. L., "BLAH, A System which Explains its Reasoning," *Artificial Intelligence,* Vol. 15, Nos. 1, 2, 1980.

[Wile81]
Wilensky, R., and Morgan, M., "One Analyzer for Three Languages," UCB/ERL TR-M81-87, University of California, Berkeley, 1981.

[Wilk73]
Wilks, Y., "An Artificial Intelligence Approach to Machine Translation.""In R. C. Schank and K. M. Colby (eds.), *Computer Models of Thought and Language,* Freeman, San Francisco, 1973.

[Wilk75]
Wilks, Y., "A Preferential, Pattern-Seeking Semantics for Natural Language Understanding," *Artificial Intelligence,* Vol. 6, pp. 53-74, 1975.

[Wino72]
Winograd, T., *Understanding Natural Language,* Academic Press, New York, 1972.

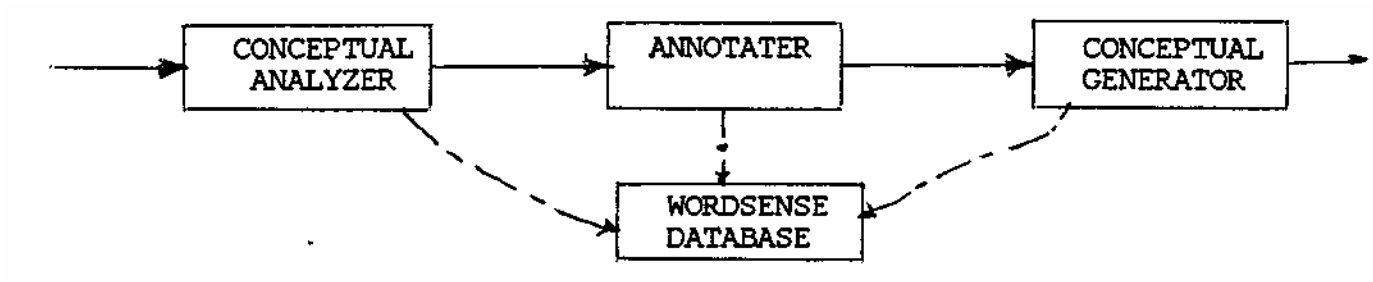

Figure 1: Surface-Semantic Machine Translation System

```
;         Figure 2: Generator Definition for "punch"
; This is EXACTLY the same wordsense as the analyzer uses
(def-wordsense wsPUNCH1
  surface-form (punch)
  ws-structure
  (propel-f actor (nil)
            obj (bpart-f bptype (hand) partof (nil))
            to (physcont val (bpart-f bptype (nil) partof (nil))))
  equivs
  (((actor) (obj partof)))
  ;default focus for the generator
  focus
  (actor)
  )

(gdictdef wsPUNCH1
  ;[1]
  ;syntax for the active voice
  (actor)
   ;[2]
   ;(actor) placement for (actor) focus (active voice)
  (((actor)
    ;the realization of the (actor) is
    ; to precede "punch" on the C-LIST
    (pr)
    ; to precede the realization of the (to val partof) filler
    (pr (to val partof))
    ; and to precede the filler of (to val)
    (pr (to val)))
   ; (to val partof) placement in active voice
   ((to val partof)
    ; following "punch"
    (fo)
    ; following the (actor)
    (fo (actor))
    ; and preceding the (to val) filler
    (pr (to val)))
   ; (to val) placement
   ((to val)
    ;following "punch"
    (fo)
    ; following (actor)
    (fo (actor))
    ; following (to val partof)
    (fo (to val partof))
    ; and following the function word "in"
    (fo in)))

  ;[3]
  ;syntax for the passive voice
  (to val partof)
  (((to val partof)
    (pr) (pr (actor)) (pr (to val partof)))
   ((to val)
    (fo) (fo (to val partof)) (pr (actor)) (fo in))
   ((actor)
```

```
(fo)(fo (to val partof))(fo (to val))(fo by)))
)
```

```
(def-wordsense wsNAMED-PERSON1
  ;"word" is the empty lexeme
  surface-form (nil)
  ws-structure
  (person-f persname (nil) surname (nil))
  )

(gdictdef wsNAMED-PERSON1
  ()
  (((persname)
    (pr)(pr (surname)))
   ((surname)
    (fo)(fo (persname))))

  sempreds
  (or (filledp '(persname))(filledp '(surname)))

)
```

: Figure 3: Generator Definition for "named-person"