ON KNOWLEDGE-BASED MACHINE TRANSLATION

Sergei Nirenburg, Victor Raskin** and Allen Tucker**

ABSTRACT

This paper describes the design of the knowledge representation medium used for representing concepts and assertions, respectively, in a subworld chosen for a knowledge-based machine translation system. This design is used in the TRANSLATOR machine translation project. The knowledge representation language, or interlingua, has two components, DIL and TIL. DIL stands for 'dictionary of interlingua' and descibes the semantics of a subworld. TIL stands for 'text of interlingua' and is responsible for producing an interlingua text, which represents the meaning of an input text in the terms of the interlingua. We maintain that involved analysis of various types of linguistic and encyclopaedic meaning is necessary for the task of automatic translation. The mechanisms for extracting and manipulating and reproducing the meaning of texts will be reported in detail elsewhere. The linguistic (including the syntactic) knowledge about source and target languages is used by the mechanisms that translate texts into and from the interlingua. Since interlingua is an artificial language, we can (and do, through TIL) control the syntax and semantics of the allowed interlingua elements. The interlingua suggested for TRANSLATOR has a broader coverage than other knowledge representation schemata for natural language. It involves the knowledge about discourse, speech acts, focus, time, space and other facets of the overall meaning of texts.

1. Delimiting the Problem.

TRANSLATOR explores the knowledge-based approach to machine translation. The basic translation strategy is to extract meaning from the input text in *source language*, SL, represent this meaning in a language-independent semantic representation and then render this meaning in a *target language*, TL. The knowledge representation language used in such a set-up is called, for historical reasons, interlingua (henceforth, IL).

TRANSLATOR's ultimate aim is achieving good quality automatic translation in a non-trivial subworld and its corresponding sublanguage. The philosophy of TRANSLATOR aims at the independence of the process of translation from human intervention in the form of the traditional pre- and/or post-editing. Interaction *during* the process of translation can be accommodated by this philosophy, but only as a temporary measure. Interactive modules will be plugged into the system pending the development of automatic modules for performing the various tasks as well as more powerful inference engines and representation schemata. This is a device that facilitates early testing of a system even before all the modules are actually built. Another advantage of this strategy is that the system becomes 'dynamic', in the sense that its knowledge is growing with use.

This strategy is an extension of one of the approaches discussed, for example, in Carbonell and Tomita (1985) since it implies knowledge acquisition during the exploitation stage and also involves a broader class of texts as its input. Johnson and Whitelock (1985) are also proponents of the interactive approach, but their motivation is different, in that they perceive the human to be an integral part of their system even in its final incarnation. In any case, interactivity is not the central design feature of TRANSLATOR.

Before proceeding to describe the knowledge clusters in TRANSLATOR we would like to comment very briefly on a number of methodological points concerning MT research. It seems that some of the opinions more or less commonly held by some members of the MT community may need rethinking. In what follows we list some of these opinions, together with our comments. A more detailed treatment of these topics will be given elsewhere.

† This paper is based upon work supported by the National Science Foundation under Grant DCR-8407114.

* Colgate University

** Purdue University

Opinion. It is unnecessary to extract the full meaning from the SL text in order to achieve adequate MT.

Comment. An MT system can do well without (involved) semantics in many cases, but has to use meaning in the rest (or rely on human intervention). Machines, unlike humans, cannot on demand produce interpretations of text at an arbitrary depth sufficient for understanding. Therefore, if one aims at fully automatic, one has to prepare the system for the treatment of even very semantically involved text. One can, of course, think of designing a system that can decide how deeply each sentence can be analyzed semantically in an attempt to minimize semantic analysis. We maintain that the decision making involved is as complex as the initial problem of deep semantic analysis.

Opinion. It is not necessary to finish processing the input sentence before starting the translation. Indeed, people very often do this (consider interpreters) with very good results.

Comment. This Opinion is based on introspection. The real thought processes that go on in the translators' or the interpreters' heads are not known. The (quite considerable) knowledge that the translators have about the subject of the text (speech) and about the speech situation itself prompts them to preempt the text by following their expectations concerning the most probable set of meanings for the text and deciding before the final corroboration arrives. Even if in a majority of cases this strategy works (as it is supposed to, because otherwise humans, being intelligent creatures as they are, would not have had the above expectations in the first place!), there is nothing unusual in making an error of judgement. Those of us who worked as translators surely remember multiple instances of this kind. Of course, this discussion is relative to the quality of product desired in the translation.

Opinion. Approaches to MT based on AI do not pay sufficient attention to the syntactic analysis of SL, while syntactic information is important for MT.

Comment. Syntactic structure of input conveys meaning; this meaning is extracted by the semantic analyzer with the help of syntactic knowledge. All clues are indeed used. No results of syntactic analysis are stored because they are not needed. Any approach that attempts to relate directly various syntactic structure trees between SL and TL strikes us as quite unpromising. It is only some early AI-oriented MT systems that were vulnerable to this criticism.

Opinion. IL-based approaches lead to an overkill because no peculiarities of SL (and of the relationship between, or contrastive knowledge of, SL and TL) can be used in translation. Some languages have quite a lot in common in their syntax and meaning distribution. It is wasteful not to use this additional information in translation.

Comment. While such insights can sometimes be detected and used, most of them comes from human intuition, and cannot be taken advantage of in an MT system, which can hardly be considered a model of human performance. It is also totally wrong to imply, in our opinion, that discovery and implementation of those pieces of contrastive knowledge can be simpler or, in fact, distinct from involved semantic analysis.

Opinion. With IL, the process of translation becomes one of interpretation, The structure of the SL text, when used in addition to IL in MT, governs the choice of one of the paraphrases. Moreover, again, IL is an overkill, because the paraphrases are not needed and add an element of ambiguity.

Comment. Human translators always have a few practically equally acceptable paraphrases for virtually every SL sentence. The degree of meaning similarity among the acceptable paraphrases is determined by external parameters. The translation is executed according to the human translator's intuitive understanding of these parameters. Only in IL approaches can one control the required degree of similarity among the acceptable paraphrases as translations of an SL sentence.

Opinion. Generation of TL is a relatively simple problem for which very little or no knowledge other than lexical or syntactic is needed.

Comment. Generation requires non-trivial decision making, for instance, in the light of the discussion in the previous paragraph, or, for that matter, as regards the computational stylistics, which will have to be a part of the choice-making mechanisms in building TL texts.

2. Configuration of TRANSLATOR

The background of the TRANSLATOR MT project at Colgate is presented in Tucker and Nirenburg (1984). This paper focuses on the static knowledge clusters of TRANSLATOR. The latter are identified as follows:

- IL dictionary
- SL - IL dictionary
- IL - TL dictionary
- SL grammar and syntactic dictionary
- SL - IL translator
- IL grammar
- TL grammar and syntactic dictionary
- IL - TL translator

There are also dynamic knowledge clusters in TRANSLATOR: the parser and the generator modules as well as the inferencing mechanism (known as the Inspector) used to derive additional knowledge from IL representations when troubleshooting becomes necessary.

In this paper we will describe the structure of the IL dictionary and the IL grammar, the central components of the system. These two structures are actually knowledge representation languages. IL dictionary is written in a language for describing the types of concepts that can appear in the subworld of translation. IL grammar is written in a language for representing the assertions about tokens of those types that actually appear in texts. We will call these languages DIL (for Dictionary Interlingua) and TIL (Text Interlingua), respectively. The distinction between DIL and TIL is similar, for instance, to that between the description and the assertion languages in KL-ONE (cf., e.g., Brachman and Schmoltze, 1985).

After discussing these languages we will briefly discuss the structure of knowledge about SL (the SL grammar and the SL - IL dictionary), enough only to help us through an illustration of how the IL dictionary and grammar are used.

3. The IL Dictionary.

The IL dictionary serves as the database where TRANSLATOR stores its knowledge about the subworld of translation. It is purely semantic, conceptual. The IL dictionary is a source of information for representing the meanings of SL texts. In it one does not find any information pertaining to any particular SL or TL. Thus, it is pure coincidence that most of the entry heads in this dictionary, as well as most of the members of the property sets (cf. below) look like English words. This choice was made with the dictionary writers in mind. The other possibility would have been to assign non-suggestive identifiers to entries and values in the IL dictionaries. This would have slowed down the process of dictionary compilation. The dictionary writers must do their best not to mix the semantics of an IL dictionary entry with that of an English word whose graphical form coincides with that of the IL dictionary entry head.

There are two kinds of entities in DIL: concepts and properties. Concepts are IL 'nouns' (*objects*) and IL 'verbs' (*events*). IL 'adjectives', 'adverbs' and 'numerals' are represented by properties. These are organized as sets of property values indexed both by the name of the property set (e.g., 'color', 'time' or 'attitude') and by the individual values, to facilitate retrieval. Property values are applicable to specific concept types. Their tokens do not appear on their own in IL texts, but only as fillers of slots in the frames for concept tokens. Thus, for example, 'red' will be a potential filler for the 'color' property of a token of every physical object. An explanation of the relationship between IL word types and tokens follows.

The IL dictionary is organized as a set of entries (concept nodes) interconnected through a number of link types (properties). However, the structural backbone of the dictionary is the familiar *isa* hierarchy with property inheritance. Note that most of the time the translation system will be working with terminal nodes in this hierarchy. But the nonterminal nodes play a special role in it. By representing sets of entries, thereby providing a link among a number of (related) concepts, they serve as the basis for a variety of inference-making procedures. Even more importantly, these 'nonterminal entries' constitute, together with the sets of various property values, the *schema* of the dictionary, the set of terms that are used to describe the semantics of the rest of the dictionary entries.

Just as all other nodes in the hierarchy, nonterminal nodes represent dictionary entries, which means that they can also have tokens. This device comes handy when, on analyzing a segment of input, we conclude that a certain slot filler is unavailable in the text. At the same time, if we know the identities of other slot fillers in the frame, we can come to certain conclusions about the nature of an absentee. For instance, if the Agent slot of a certain mental process is not filled, we, by consulting the 'agent-of' slot of the nonterminal node 'mental-process', can infer (or, rather, abduce) that, whatever it is, it must be a 'creature'. This knowledge helps in finding referents for anaphoric phenomena.

The dictionary entries represent IL concept and property *types*; IL texts consist of IL concept *tokens* (as well as IL clause and sentence tokens). Every token of an IL concept stands in the *is-token-of* relationship to its corresponding type. Structurally both IL concept types and IL concept tokens are represented as frames. The frame for a type and the frame for a corresponding token are not identical in structure, though the intersection of their slot names is obviously non-zero. One must note, however, that even in this case the semantics of the slots in the dictionary frames is different from that of the corresponding slots in the text frame.

Some of the slot names in the type frames refer to the paradigmatic relationships of this concept type with other concept types. These are the *type parameters* of an IL dictionary entry. The rest of the information in an entry describes syntagmatic relationships that tokens of this particular type have with tokens of other types on an IL text. These are called *token parameters*. Among the type parameters one finds the pointers in the *isa* hierarchy, relationships like *part-of, belongs-to*, etc.

The token-parameter slots in the dictionary entries contain either *default* values for the properties (the 'no-value' value is among the possible default choices) or acceptable *ranges* of values, for the purpose of validity testing. IL concept tokens, which are components of IL *text*, not its dictionary, have their slots occupied by actual values of properties; if information about a property is not forthcoming, then the default value (if any) is inherited from the corresponding type representations.

In what follows we will describe DIL, the IL dictionary language. We will do this by presenting the top levels of the *isa* hierarchy of concepts in our world and listing the frames for high-level nodes. Next, we'll present examples of IL dictionary frames, including one complete path in the *isa* hierarchy, from the root to a terminal node.

The actual contents of the tree are, as we already said, idiosyncratic: it may be overdeveloped in some of its branches and underdeveloped in many others. This state of affairs corresponds to the strategy of working within a subworld.

3.1. Frames.

```
all ::= ('all'
        ('id' string)
        ('properties' properties)
        ('subworld' subworld*))
```

This is the root of the *isa* hierarchy. The three slots present here mean that every node in the tree has an *id*; every node features some *properties* (which exactly, will be shown in lower-level nodes); and every node represents a concept that belongs to one or more *subworlds*.

```
event ::= ('event'
          ('isa' all)
          ('patient' object))
```

At this level we meet the 'isa' slot for the first time. This is the pointer to a node's parent in the hierarchy. Events divide into processes and states. The only overtly mentioned property common to all events is the conceptual case of 'patient' (this reflects our opinion that in the sentence (1) *John* is not an agent, but rather a patient). Note that 'patient' in DIL subsumes the semantics of 'beneficiary'.

(1) *John is asleep.*

process ::= ('process'
('isa' event)
('is' process-sequence)
('part-of' process*)
('agent' creature)
('object' object)
('instrument' object)
('source' object)
('destination' object)
('preconditions' state*)
('effects' state*))

In addition to the conceptual case slots, the process frame contains information about preconditions and effects. These are states that must typically hold before and after the process takes place, respectively. A process can also be a part of other processes. Thus, for instance, *move* is a part of *travel* and, at the same time, of *fetch* or *insert*. The 'is' slot of a process frame contains either the constant *primitive*, if the process is not further analyzable in DIL, or the description of the sequence of processes which comprise the given process. The *process-sequence* is a list of process names connected by the operators *sequential*, *choice* and *shuffle*. In other words, a process may be a sequence of subprocesses (*sequential*), a choice among several subprocesses (*choice*), a temporally unordered sequence of subprocesses (*shuffle*) or any recursive combination of the above. This treatment of processes is inspired by Nirenburg et al., 1985. For the purposes of machine translation it seems unnecessary to introduce a more involved temporal logic into consideration for the 'is' slot.

physical-process ::= ('physical-process'
('isa' process))

mental-process ::= ('mental-process'
('isa' 'process')
('is' primitive)
('agent' creature)
('object' object | event))

Only creatures can be fillers for the 'agent' slot. Mental objects classify into reaction processes (cf. the English 'please' or 'like'), cognition processes ('deduce') and perception processes ('see'). Objects of mental processes can be either objects, as in (2) or events, as in (3).

(2) I know John
(3) I know that John has traveled to Tibet.

speech-process ::= ('speech-process'
('isa' process)
('is' primitive)
('agent' person)
('patient' person* | organization*)
('object' event)
('source' 'agent')
('destination' 'patient'))

Speech processes are *primitives*. The speech processes recognized by DIL include assertions (that further subdivide into definitions, opinions, facts, promises, etc.) and requests (questions or commands). The 'agent' slot filler has the semantics of the speaker. The 'patient' is the hearer. Note that there is a possibility for the hearer to be a group or an organization, as in (4).

(4) I promised the band to let them have a ten-minute break every hour.

The 'agent' is the 'source' and the 'patient' is the 'destination' of a speech process.

state ::= ('state'
('isa' event)
('part-of' state*))

The actant in states, which is the patient rather than the actor, is inherited from the event frame.

object ::= ('object
('isa' all)
('part-of' object*)
('consists-of' object*)
('belongs-to' creature | organization)
('object-of' (Mental-Process Speech-Process))
('patient-of' event)
('instrument-of' event)
('source-of' event)
('destination-of' event)
('source-of' event))

The '...-of' slots are used for consistency checks.

3.2. Properties.

Property values are primitive concepts of IL used as values for slots in concept frames. We give here just an illustration of these. Many more exist and will be used in the implementation.

size-set ::= nil | infinitesimal | ... | huge

color-set ::= nil | black | ... | white

shape-set ::= nil | flat | square | spherical ...

material-set ::= nil | (gold (specific-gravity 81) (unit-value 228))| ...

subworld-set ::= nil | computer-world | business-world | everyday-world ...

boolean-set ::= nil | yes | no

texture-set ::= nil | smooth | ... | rough

properties ::= ('properties'
'none'
('size' size-set)
('color' color-set)
('shape' shape-set)
('texture' texture-set)
('belongs-to' creature | organization)
('part-of' object | event)
('consists-of' object | event)
('power' real)
('speed' real)
('mass' real)
('edibility' boolean-set)
('made-of' material-set)
...)

3.3. From the Root to a Leaf.

A path of concept representations from the root to a leaf node is presented below.

all -> object -> pobject -> +alive -> creature -> person -> computer-user

Frames for 'all' and 'object' see above.

pobject ::= ('pobject'
('isa' object)
('object-of' (+ (Take Put))
('size' size-set)
('shape' shape-set)
('color' color-set)
('mass' integer))

The '+' sign in slots means all inherited information plus the contents of the current slot.

```
+alive ::= ('+alive'
           ('isa' pobject)
           ('edibility' boolean-set))

creature ::= ('creature'
             ('isa' +alive)
             ('agent-of' (Eat Ingest Drink Move Attack))
             ('consists-of' (Head Body))
             ('object-of' (+ (Attack))
             ('power' real)
             ('speed' real))

person ::= ('person'
           ('isa' creature)
           ('agent-of' (+ (Take Put Find Speech-process Mental-Process)))
           ('source-of' Speech-process)
           ('destination-of' Speech-process)
           ('consists-of' (+ (Hand Foot ...)))
           ('power' 50)
           ('speed' 50)
           ('mass' 55))

computer-user ::= ('computer-user'
                  ('isa 'person)
                  ('agent-of' (+ (Operate)))
                  ('subworld' computer-world))
```

The complete frame of the leaf of this path, 'computer-user', including all inherited slots and default values is listed below. In reality frames like this do not exist, because the tokens of this type do not contain all the possible slot fillers.

```
(computer-user
   ('isa' person)
   ('agent-of' (Operate Take Put Find Speech-process Mental-Process
            Eat Ingest Drink Move Attack))
   ('object-of' (Find Mental-process Speech-process Attack Take Put))
   ('destination-of' Speech-process)
   ('source-of' Speech-process)
   ('consists-of' (Hand Leg Head Body))
   ('power' 50)
   ('speed' 50)
   ('mass' 55)
   ('subworld' computer-world))
```

4. The Interlingua Grammar.

In the previous section we dealt mostly with IL lexicon. This section is devoted to the syntax of IL text. Unlike a natural language text, an IL text is not linear. It is a (potentially) complex network of IL sentences, interconnected by *IL discourse markers*. An IL sentence consists of a main clause and a number of subordinate clauses, possibly interconnected through discourse markers, with the speech act and focus information added. IL clauses are the place where the tokens of events are put into the modal and spatio-temporal context. IL events are processes and states. It is in representations of the latter that tokens of IL 'verbs' and 'nouns' (retrieved from the dictionary and augmented by various property values identified during SL text analysis) meet for the first time.

The above consideration led us to declare the language of the grammar a separate representation language, TIL. There are important differences between TIL and DIL. At the same time there are regular correspondences. The values of the properties in entity tokens typically correspond to the data types listed as fillers for the corresponding slots in the IL dictionary. Thus, for instance, the *color* property slot in the IL dictionary frame for 'flower' can be occupied by a list (white yellow blue red purple pink ...), the one for 'snow' will presumably contain only (white). At the same time, 'rose11' will have the value 'red' as the contents of its 'color' slot. This underscores the difference in the semantics of similarly named slots in DIL and TIL.

4.1. Text.

```
text ::= nil |
         sentence |
         (discourse-structure-type text text+)
```

The above means that an IL text is either an empty string, a single sentence, or a number of sentences interconnected through discourse structure markers.

4.2. Sentence.

```
sentence ::= ('sentence-token'
             ('main-clause' clause)
             ('clauses' clause*)
             ('id' string)
             ('subworld' subworld)
             ('modality' modality)
             ('focus' focus)
             ('speech-act' speech-act))
```

Every sentence is declared to contain a speech act. Thus, we propose to represent (5) as (6), provided we can infer the identities of the speaker and the hearer, as well as the identity of the process:

(5) I'd rather not do it.

(6) Boss ordered Employee X not to agree to the terms of Sales Offer Y.

Both direct and indirect speech acts are represented with the help of speech process tokens. With direct speech acts, the information to be put into the sentence frame is present in the text, while with indirect speech acts it has to be inferred.

Thematic information about the sentence is restricted to the values of the focus slot in the sentence frame. This slot contains pointers to the entities that constitute the 'given' and the 'new' in this particular sentence. This entity can be a concept, a property of a concept, or an entire clause (cf. 4.11) The value of the modality slot for the IL sentence is chosen from the set of modalities (cf. 4.10). The subworld slot is a marker that shows that the sentence belongs to a 'semantic field' related to computers. In TRANSLATOR this is the designated topic for translation. In broader environments the subworld information will be helpful to prune unneeded inference paths.

The fact that we allow only one clause to occupy the 'main-clause' slot of a sentence means that IL sentences cannot be compound (i.e., consist of a number of sentences connected through commas and coordinate conjunctions like the English 'and', 'or' or 'but'. The fact that other clauses can be present means that it can be complex. Sentences that are compound in SL are translated as texts in IL, the representations of the immediate constituents of the compound SL sentences being IL sentences. Appropriate discourse structure markers are used to represent the meaning carried by the conjunction.

4.3. Clause.

```
('clause-token'
   ('id' string)
   ('discourse-structure' discourse-structure)
   ('focus' focus)
   ('modality' modality)
   ('time' time)
   ('space' space)
   ('event' event)
   ('quantifier' quantifier2)
   ('subworld' subworld)
```

The major difference between the interlingua clauses and events is that clauses contain information that actually appears in the input text (augmented by anaphora resolution), while events can be either contained in the input or *inferred* from it.

A clause may be connected discourse-wise not only with another clause but also with an object or an event, as well as with a sentence, a paragraph or even a whole text; also note that discourse structure assigns the given clause as one of the two arguments in the discourse structure; one clause can be an argument in more than one discourse-structure expression.

4.4. Process.

```
('physical-process-token'
    ('id' string)
    ('is-token-of' string)
    ('agent' object-token)
    ('object' object-token)
    ('patient' object-token)
    ('instrument' object-token)
    ('source' object-token)
    ('destination' object-token)
    ('negation' negation)
    ('quantifier' quantifier2)
    ('phase' phase-set)
    ('manner' manner-set)
    ('space' space)
    ('time' time)
    ('subworld' subworld))
```

An actual process token is represented as follows:

```
(move-token
    ('id' move21)
    ('is-token-of' move)
    ('is' primitive)
    ('agent' person12)
    ('object' person12)
    ('source' (in house2))
    ('destination' (in house3))
    ('negation' nil)
    ('quantifier' nil)
    ('phase' static)
    ('manner' easily)
    ('part-of' travel5)
    ('time' (before 1700))
    ('subworld' everyday-world))
```

4.5. State.

```
('state-token'
    ('id' string)
    ('is-token-of' string)
    ('negation' negation)
    ('quantifier' quantifier2)
    ('patient' object-token)
    ('phase' phase-set)
    ('part-of' state-token*)
    ('space' space)
    ('time' time)
    ('subworld' subworld))
```

Events in IL have a property of 'phase': they are either 'static', 'beginning' or 'end'. This device is needed to represent changes of state. Changes of state are sometimes represented as a separate class of processes. The solution in IL may be more economical.

4.6. Object.

A typical frame for an object token in TIL is as follows. The 'string' in the 'is-token-of' slot stands for the name of the corresponding object type.

```
('object-token'
    ('id' string)
    ('is-token-of' string)
    ('subworld' subworld)
    ('negation' negation)
    ('quantifier' quantifier1))
```

An example object token follows:

```
('person-token'
    ('id' person23)
    ('is-token-of' person)
    ('subworld' everyday-world)
    ('negation' no)
    ('quantifier' any)
    ('power' 50)
    ('speed' 50)
    ('mass' 55))
```

Note the difference from DIL object frames. No '...-of' slots here. More emphasis on syntagmatic relationships and default overriding.

4.7. Time.

```
time ::= nil | absolute-time | relative-time

absolute-time ::= ('time'
                   ('quantifier' quantifier2)
                   ('point' integer) |
                   ('interval-begin' integer)
                   ('interval-end' integer))

relative-time ::= ('time'
                  (temporal-operator event)
                  ('quantifier' quantifier2))

temporal-operator ::= simultaneous | before | during | around | always | none
```

Relative time markers will predominantly appear in texts.

4.8. Space.

```
space ::= nil | absolute-space | relative-space

absolute-space ::= ('space
                    ('quantifier' quantifier2)
                    ( 'coordinate1' real)
                    ( 'coordinate2' real)
                    ( 'coordinate3' real))

relative-space ::= ('space'
                    (spatial-operator object)
                    ('quantifier' quantifier2))

spatial-operator ::= left-of | equal | between | in | above |
                     near | none
```

As in the case of time, relative (topological) space specifications will predominate in texts.

4.9. Slot Operators.

```
quantifier1 ::= nil | all | any | most | many | some | few | 1 | 2 | ...

quantifier2 ::= nil | hardly | half | almost | completely
```

4.10. Modality.

```
modality ::= ('modality' modality-set)

modality-set :: = real | desirable | undesirable | conditional |
                  possible | impossible | necessary | nil
```

4.11. Thematic Information

```
focus ::= ('given'
          ('object' obj) |
          ('event' event) |
          ('clause' clause) |
          ('quantifier' event-quantifier | quantifier))
```

('new'
('object' obj) |
('event' event) |
('clause' clause) |
('quantifier' event-quantifier | quantifier))

The thematic information, together with the discourse structure and speech act information, explicitly represents the rhetorical force of SL texts. The lack of this type of knowledge led many MT researchers to declare that SL traces are necessary in the internal representation. The above information may prove sufficient for abandoning that requirement.

4.12. Discourse Structure.

discourse-structure ::= (discourse-structure-type
(clause1 clause-n | sentence | text) |
(clause-n | sentence | text clause1)*)

discourse-structure-type ::= none | temp | equiv | +expan | -expan |
condi | +simil | -simil | choice

For a more detailed description of the discourse cohesion markers in TRANSLATOR see Tucker et al., 1986.

A clause may be connected discourse-wise not only with another clause but also with a sentence, a paragraph or even a whole text; also note that discourse structure assigns the given clause as one of the two arguments in the discourse structure; one clause can be an argument in more than one discourse-structure expression.

4.13. Speech Act.

speech-act ::= ('speech-act'
('type' speech-process)
('direct?' yes | no)
('speaker' object)
('hearer' object+)
('time' time)
('space' space))

Every IL sentence features a speech act, irrespective of whether it was overtly mentioned in the SL text. If it was, it is represented through a token of a speech process. Otherwise, it is inferred. The time and space of the speech act can be quite different from that of the proposition which is the information transferred through this speech act.

4.14. Other Slots and Slot Fillers.

negation ::= boolean-set

referent-set ::= nil | above | below | object-token

manner-set ::= nil | difficulty | attitude

difficulty ::= nil | easily | ... | difficultly

attitude ::= nil | caring | ... | nonchalantly

phase-set ::= nil | static | beginning | end

5. Conclusion

This paper suggested an approach to conceptual representation of a text in a natural language for the purposes of translation. An important distinction has been maintained between the representation of descriptions and assertions. We even suggested two different representation languages, DIL and TIL for the two tasks.

The next task in the project is to actually implement the procedures for analysis, inference making and synthesis. One crucial prerequisite for that is the compilation of a substantial knowledge base (IL dictionary) for the subworld of computers. Now that the structure of IL has been specified, we can actually do it. Strategies and aids for uniform and computer-aided knowledge acquisition are being developed.

Acknowledgement. The authors wish to thank Irene Nirenburg for reading, discussing and criticizing the numerous successive versions of the manuscript. Needless to say, it's we who are to blame for the remaining errors.

References.

Brachman, R. and J. Schmoltze 1985. An overview on the KL-ONE knowledge representation system. *Cognitive Science*, vol. 9, issue 2.

Carbonell, J. and M. Tomita 1985. New approaches to machine translation. In: S. Nirenburg (ed.).

Johnson, R. and P. Whitelock 1985. Machine translation as an expert task. In: S. Nirenburg (ed.).

Nirenburg, S. (ed.), Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation. Hamilton, NY, August 1985.

Nirenburg, S. and J. Brolio (in preparation). A parsing strategy for knowledge-based machine translation.

Nirenburg, S., V. Raskin and A. Tucker, Interlingua design in TRANSLATOR. In: S. Nirenburg (ed.)

Tucker, A., S. Nirenburg and V. Raskin, Discourse, cohesion and semantics of expository text (this volume).