TR-I-0035

Representation and computation of units of translation for Machine Interpretation of spoken texts

会話テキストの機械通訳のための翻訳単位の表現と計算

Christian Boitet
クリスチャン・ブワテ

August, 1988

Abstract

This paper deals with an overall organization of the Machine Interpretation system, in which heterogeneous elements such as speech recognition, machine translation and speech synthesis should be put together in a unified framework (functional, computational and linguistic organization). Specifically a layered weighted lattice as the coarsest representation of a unit of translation is proposed in the spirit of the blackboard technique.

This work was done when the author was a visiting research scientist of ATR Interpreting Telephony Research Laboratories.

ATR Interpreting Telephony Research Laboratories
ATR 自動翻訳電話研究所

This is an excerpt from the report as a visiting research scientist

Period: from April 6, 1988 to August 5, 1988

Company: ATR Interpreting Telephony Research Laboratories
Osaka, Japan

Topic: Considerations on Technical Problems in Machine Translation of Spoken Dialogues

Name: Christian Boitet, Professor,
GETA (Groupe d'Etudes pour la Traduction Automatique),
Université Joseph Fourier & CNRS,
Grenoble, France

Representation and computation of units of translation for Machine Interpretation of spoken texts

Christian Boitet*
ATR Interpreting Telephony Research Laboratories, Osaka

and

GETA (Study Group for Machine Translation)
Joseph Fourier University & CNRS, Grenoble

ABSTRACT

The first goal of Machine Interpretation is to translate spoken dialogues. Methods from Speech Processing, Machine Translation and Discourse Understanding have to be used in an integrated way. In the spirit of the *blackboard technique,* the paper proposes to use a *layered weighted lattice* as the coarsest representation of a unit of translation. Weights on nodes or arcs may be thought of as scores or activation/inhibition factors. Each node contains an object the exact structure of which depends on the layer of the node. It always contains a *class* from the repertory of its layer, and optionnally a *representation structure,* an ordered tree where in turn each node contains 1) a *label,* 2) a *decoration* (hierarchical and bounded attribute structure), and 3) an optional *interpretation structure* (complex typed f-structure with possible reentrancy). At each layer, nodes corresponding to the same input fragment are clustered, and the ones sharing the same class are packed, thus allowing factorized treatment of ambiguities. Weighted connections are maintained between nodes corresponding to successive steps in a given analysis of some fragment, thus making the *derivation structure(s)* explicit (but hidden from other layers). It is hoped that this arrangement can provide a basis for using statistical techniques at higher levels of linguistic abstraction than what is currently done, and for paving the way for future use of the promising neural network techniques, without loosing the benefit of well proven structural and symbolic techniques, which can not be replaced by statistical methods when it comes to explicit discourse understanding and planning.

**This study was done while the author was staying with ATR as visiting researcher.*

INTRODUCTION

1. The task and its constraints

Machine Interpretation, a new field pioneered by ATR, aims at translating speech. The first application envisaged is to produce a system for interpreting Japanese-English dialogues in the context of the organization of an international conference. Here, the system must obviously be speaker-independent, and operate in real time. Its basic vocabulary has been *a priori* limited to 2/3000 lemmas (usual dictionary entries, not wordforms). However, the system must be prepared to handle unknown words, such as names of places or persons, because it is impossible to limit them to those contained in a dictionary (of any size). For example, a typical conversation can begin as follows :

(E1) This is Miss Stachowski of Speechtech limited. I'm coming with Doctor Morny and Miss Lee...

"Stachowski", "Speechtech", "Morny" and "Lee" could hardly be all in the dictionary. They have to be dynamically recognized as proper names, and possibly stored for further reference.

Because the situation is interactive, it might be easier to constrain the syntax by asking the participants to rephrase, or to choose betwen alternate formulations proposed by the system. Concerning the semantics and pragmatics of the system, a core model describing international conferences and the various types of discourse seems to be necessary in order to solve the best part of the numerous anaphoras and ellipses observed in such dialogues. However, much as for the lexicon, the system should be *open*, in order not to fail utterly when some participant begins to talk about unrelated matters such as current exhibitions and cultural events, so that the conference organizer answers something like :

(E2) All right, I'll check a summer festival schedule... ah, in Tôkyô, there is the "Edo-Shumi-Noryo-Taikai", and then we have the very unusual [ma, ah] matsuri called "My Town Festival" in Yoyogi Park, and then there is the Plum festival in Fuchu.

(...)

Ah, if you are interested in the Yokohama and Kamakura area, there is the Kawase-matsuri festival in Chichibu-city in Saitama prefecture, and there are two [hana, ah] hanabi taikai's or fireworks, one in Yokohama and the other in Kawasaki.

This example has been taken from (Iida&al 1987), with minor changes.

Another potential type of application would be to interpret during a business teleconference. Here, some short tuning to each speaker's voice is envisageable, although speaker-independency would be preferred. It is possible to use a comparable knowledge about the discourse structure, but the domain-specific knowledge can only be minimal, and mainly of a lexical nature. Typically, when a human interpreter prepares for a meeting on biotechnologies, s/he does not learn biology, chemistry and medicine, but the specific technical terms, their rough semantic connotations (a method, a substance, a device, etc.), and their equivalents in the various languages at hand.

A third application would not involve dialogue at all, but monologue. We think of situations where the end user would dictate a letter, a report, a comment on some ongoing events, etc. Expert domain knowledge would not generally be included in the system. Rather, two main parameters of linguistic nature, the typology (letter, report...) and the domain (car rental, computer services...), would be used to select the adequate versions of the grammars and dictionaries (or to construct them from available parts). Such a system might be speaker-dependent, and quite a bigger dose of interaction might be considered : the task is more translation than interpretation, and there is no need to perform in real time (think of a businessman dictating in a plane and feeding the tape later to his computer). Also, the production of a written form of the input and output texts, certainly useful in the first two applications, is mandatory here.

2. Possible architectures

2.1 Pure sequential composition of existing techniques

To construct a Machine Interpretation system, it is obviously necessary to use methods of Speech Processing, Machine Translation, and Discourse Understanding – the third, at least for the interpretation of purpose-specific dialogues. The first idea that comes to the mind is to simply connect the three techniques through interface structures. To give a possible schema, one would :

1) use speech recognition techniques to produce a *written text,* possibly containing special punctuation marks standing for pauses, stress and melody ;
2) use a standard MT analyzer to transform the text into a suitable *representation structure* (e.g., decorated tree, Q-graph, syntaxo-semantic net) ;
3) use domain- and discourse-related knowledge to put the text in some *interpreted form* (e.g., logical formula, frame structure, semantic net, f-structure) ;
4) produce a corresponding *representation structure* in the target language ;
5) synthesize a corresponding *surface structure,* enriched with prosodic marks ;
6) synthesize a corresponding *coded phonetic string* (a string of characters) ;
7) pass it to a voice synthesizer.

Of course, the speech recognition part would make use of some kind of linguistic knowledge, such as the frequencies of sequences of 2 or 3 morphosyntactic classes (common noun, proper noun, verb, adjective, particle, conjonction...), the frequency of each word in its class(es), and possibly of a simplified regular or context-free grammar. But there would be no cooperation between the running processes of speech recognition and linguistic analysis.

Considering the cost of developing such systems, and the advances in each of them, this may be a solution. However, it seems somewhat counterproductive. For example, if the speech recognition part has determined the morphosyntactic classes of the words, or even a rough syntactic tree, why duplicate the work during the following analysis ? Also, why produce only one solution (path) and

hide alternate possibilities (cites/sights/sites, this man/these men...) when there is no real best candidate ? The analyzer would have to resynthesize phonetically similar forms if it fails to produce a good enough analysis with the delivered text. In the same vein, why try to translate at the "understanding" level if the utterance, or part of it, is not related to the task domain ?

2.2 Total integration of each major step

Another possibility is to eliminate the need for interfacing different processes, within each major step (1-2-3, 4-5-6), or at least for analysis (1-2-3), by using just one process to take care of the whole task. Synthesis is much easier than analysis, because the process starts with no noise and no ambiguities, so that a sequential approach is suited. On the contrary, previous experiments in speech recognition and understanding (Lea 1980, Lee 1988) have demonstrated that the various "knowledge sources" must interplay during processing, to limit the combinatorial explosion without drastically reducing the quality of the output. It has also been demonstrated that it is extremely difficult to let different processes associated to these various "knowledge sources" interact tightly during processing, as in the Hearsay-II system (Reddy 1980, Erman&Lesser 1980). The resulting systems are also slower.

Encouraged by successes in speech-to-text transformation (according to (Lee 1988), Sphinx is speaker-independent and handles continuous input with a 1K vocabulary – of wordforms, not lemmas – and a perplexity of about 30), some researchers in speech recognition seem to think that a unique (possibly enormous) Hidden Markov Model (HMM) could be the solution. In the same vein, some specialists of computational linguistics would like to cover all the ground with a unique grammar formalism coupled with an efficient LR method (Saito&Tomita 1988). Still others, more interested by the AI aspect of the task, have considered using symbolic, unification-based inferential methods to do everything (Kogure&al, 1988).

The above proposals amount in fact to compile the different knowledge sources which have to bear on the problem in a unique form (a network, an extended context-free grammar, a production or a dedution system). While this may be a goal for a final product, one should not forget that :

- such a compilation is usually extremely computer-intensive, and difficult to perform incrementally, because it is essentially a *global* optimization technique – see the experience of the Harpy system (Lowerre&Reddy 1980) – 13 hours on a .4 Mips machine ;

- in Hearsay-II, Harpy and Sphinx, the result of speech recognition is a string of words, which is passed to the semantic routine (the simplicity of the grammars makes parsing cheap) : in the most successful speech understanding systems, there is no complete integration. On the other hand, the very promising and well designed HWIM system of BBN (Wolf&Woods 1980) integrated really everything, but, according to (Lea 1980a:385), its computation time was far

superior to that of Harpy (500/28 Mipss, or million instructions per second of speech) and its quality notably inferior (44/95%) ;

- although good results have been obtained in speech understanding with statistical (extremely "data-driven") methods of the HMM type, more extendable, distributed methods have fared almost equally well (e.g., Hearsay-II, 85Mipss, 91%) ;

- linguistic parsers of acceptable speed used or usable in MT systems (which are usually less constrained than speech understanding systems by two or three orders of magnitude) have been successfully constructed with various methods, combinatorial or heuristic : context-free base with attached transformations and weights (data-driven with variable instantiation, but no pattern-matching), as in METAL (Slocum 1984), tree transformational systems with heuristic control and preferences (pattern-matching), as in ARIANE (Boitet 1986a, 1987a), MU (Nakamura&al 1986), AS-Transac (Toshiba) or HICAT (Hitachi), and more recently unification and deduction, as in LMT (McCord 1985) ;

- no other methods than symbolic and heuristic, often based on unification (e.g., logical programming), have led to successful results in explicit understanding and inferencing.

2.3 Enriched interface structures embedded in a blackboard

Total integration and pure sequential composition offer complementary advantages and inconveniencies. Why not construct a MI system by using the best of both ? To be more specific, we could propose to :

- use a unique data structure, called *blackboard* in reminiscence of Hearsay-II, organized in *layers* corresponding to different levels of linguistic description ;
- specialize some of these layers as *enriched interface structures ;*
- construct *integrated processes* for each main phase (1...7), by compiling into them parts of various knowledge sources (phonological, lexical, grammatical, task-specific...) ;
- run them under a *central control program,* taking care of any interprocess communication or feed-back, but essentially leaving them free to use their own strategy.

For example, the linguistic parser should have access to a richer structure than a sequence of wordforms, namely a word lattice (or grid), where each node would contain not only the recognized word, but also the category used by the recognizer, if any (e.g., light(V), light(N)...), its score, its endpoints (with error margins), its phonetic transcription, and some prosodic parameters, such as pitch or energy, usable for syntactic analysis (Lea 1980b, Waibel 1986). This is of course reminiscent of the phonetic lattice structure of the HWIM system.

3. Outline

The proposed blackboard architecture uses a *layered weighted lattice* as the coarsest representation of a unit of translation. In the spirit of neural networks, weights on nodes or arcs may be thought of as scores or activation/inhibition factors – each node would admittedly correspond to a very large group of neurons. Each node contains an object the exact structure of which depends on the layer of the node. This object always contains a *class* from the repertory of its layer, and optionnally a *representation structure*, an ordered tree where each node contains 1) a *label*, 2) a *decoration* (hierarchical attribute structure), and 3) an optional *interpretation structure* (complex typed f-structure with possible reentrancy). At each layer, nodes corresponding to the same input fragment are *clustered*, and the ones sharing the same class are *packed*, thus allowing factorized treatment of ambiguities.

The nodes and arcs above are readable by all processes. We propose to call them *white*. Our blackboard will also contain *black* elements, private to the processes. Auxiliary or intermediate information such as expectations (active arcs in a chart parser) or various markings will be created as black nodes or arcs by each process. Weighted connections (special nodes and links) will be maintained between nodes corresponding to successive steps in the treatment of some fragment at a given layer, thus making the *derivation structure(s)* explicit (but hidden from the other layers). As they should be visible by the main controlling process, but invisible by the other processes, we will give them a *grey* color. For instance, a normalization rule like *"ain't –> are not"* would have one input link to the node containing "ain't" and two output nodes to the nodes containing "are" and "not".

Each layer will have 3 dimensions, *time*, *depth* and *class*. The idea is that a node at position (i,j,k) will correspond to the input segment of length j ending at time i and be of class (number) k. All realisations of class k corresponding to this segment will be packed in this node, and all nodes corresponding to approximately equal input segments will automatically be geometrically clustered.

In the first part of this paper, the basic structure envisaged is presented in more detail. The nature and role of derivation, representation and interpretation structures are discussed in the second part. Finally, some methods and tools for computing these structures are presented in the third part.

I. THE BASIC STRUCTURE

I.1 Units and sizes

First, we argue that units of translation should not be sentences, but complete utterances or utterance segments. We then define some phonetic and linguistic units, and conclude by some considerations on the dimension(s) of the blackboard.

1.1 Units of translation

In the considered applications, a speaker may speak a few seconds or several minutes. We call this a *speech period.* A speech period is a sequence of one or more *utterances,* clearly separated by comparatively long silences. In a dialogue, a speech period is usually shorter than 30 words and consists of one utterance. In a meeting, a long speech period could be of the order of a page, or approximately 250 words, last about 1.5 mn (speaking relatively slowly at 2–3 words/second), and consist of several utterances. There is no maximum for monologues. In written texts, paragraphs, not sentences, correspond to utterances. It is relatively easy to determine sentence boundaries in a written text, with a slight amount of preprocessing, but this is much more difficult in a spoken utterance.

We have no idea yet on how to perform simultaneous translation (where translation begins after a slight delay, before the end of the utterance). Hence, an MI system must rely on *consecutive translation of units of translation.* From what has been said above, the smallest unit of translation of an MI system should be the utterance, not the sentence. Another reason to consider units larger than sentences is the abundance of anaphors in dialogues, the majority of which is internal to the utterance. However, all MT systems today translate sentence by sentence, 100 words being a maximum (very rarely attained), with the exception of systems ARIANE – which basic software is not designed to cope with very noisy input, and hence not usable *as is* for MI.

In the case of dialogues, *we will take a unit of translation to be a complete speech period (usually one utterance of less than 100 words) by one of the speakers.* In other cases (teleconference, or monologue), *we will limit it to a sequence of utterances, for a maximum of 200–250 words* (about a page). In practice, the system should warn the speaker to terminate his utterance some 50 words (about 15 s) away from the admitted maximum. That is done in human interpretation and should not be resented by the users.

The proposed maximum is quite large and could be reduced for efficiency reasons (e.g., non linear computation cost), or simply because the translation delay would exceed some psychological limit. However, 120–150 words would be the minimax, because a system which would continuously interrupt the speaker would not be accepted.

1.2 Linguistic and phonetic units

We have already used the linguistic units of *words* and *sentences.* We will use *word* only for *wordform* (spoken or written), and *lemma* for the usual dictionary word (a simple entry, associated with one or several stems and a paradigm). A *term* may be simple (a lemma) or complex (a minimal phrase, such as "registration form" or "yellow fever"). Note that, in many languages, such as Japanese, Chinese or Thaï, the notion of word is not clearly defined, as there are no spaces in the written form and no necessary separation in speech.

We will not be very precise about intermediate phonetic units, because quite a variety has been used in speech recognition system : traditional ones such as *syllables* (Mérialdo 1988), *phonemes* (Quinton 1980, Wolf&Woods 1980), *allophones* (D'Orta&al 1988), and more esoterical ones such as *demi-syllables, phones, diphones, triphones, context-dependent phones,* or *word-dependent phones* (Lee 1988). Some speech recognition systems produce an explicit representation of the utterance in terms of a set of such phonetic units, while others use them only for constructing the recognition device, which produces directly the final result in terms of wordforms.

1.3 Dimension of the blackboard

The most objective (and most implicit) description of the unit of translation is the signal, or rather its representation. In many systems, a sequence of *frames* is the first representation of the waveform. In Sphinx, for example, frames are 20 ms long and overlap by 10 ms. Each frame is smoothed by multiplying it with a "Hamming window". Then, LPC coefficients are computed, and vector quantization finally produces discrete "phonetic codes" (or "phonetic characters"). In Sphinx, such a code is a byte-sized integer interpreted as the index of a "prototype vector", and 1 or 3 codes are associated to each frame (1 for the baseline version, 3 for the most sophisticated one). The difference with written input is considerable : one second of speech corresponds roughly to 15 bytes in a written English text, and to 100 or 300 bytes in the coded spoken Sphinx input, 7 or 20 times more. In the written form, a character can also be coded on more than one byte (JIS code, Xerox-STAR convention,...). Hence, the number of characters of a unit of translation is not a very good measure of its size.

We will take the basic unit of length to be the time between two successive frames (10 ms). A node at position (i,j,k) in the basic structure will cover the intervals [t1,2] ($0 \le t1 \le t2 \le tmax$) of the input such that $t1-e \le i-j+1 \le t1+e$ and $t2-e \le i \le t2+e$, e being some error margin associated to the node, and tmax being about 10000.

I.2 Implicit/explicit arcs : grid or lattice

In a *grid,* there are no explicit arcs. A node N covering [t1,t2] is implicitly connected to another node N' covering [t'1,t'2] iff (if and only if) [t1,t2] is anterior to [t'1,t'2], that is $t1 \le t'1$, $t2 < t'2$, and $t2-e1 \le t'1 \le t2+e2$, e1 and e2 being respectively the *gapping* and *overlapping thresholds* (Quinton 1980). Because of the first condition, there can be no cycles.

In a *lattice,* there are only explicit arcs. Cycles are forbidden, and there must be a unique first node and a unique last node. Figure 1 shows two kinds of lattices used in natural language processing (NLP in the following), a chart and a Q-graph.

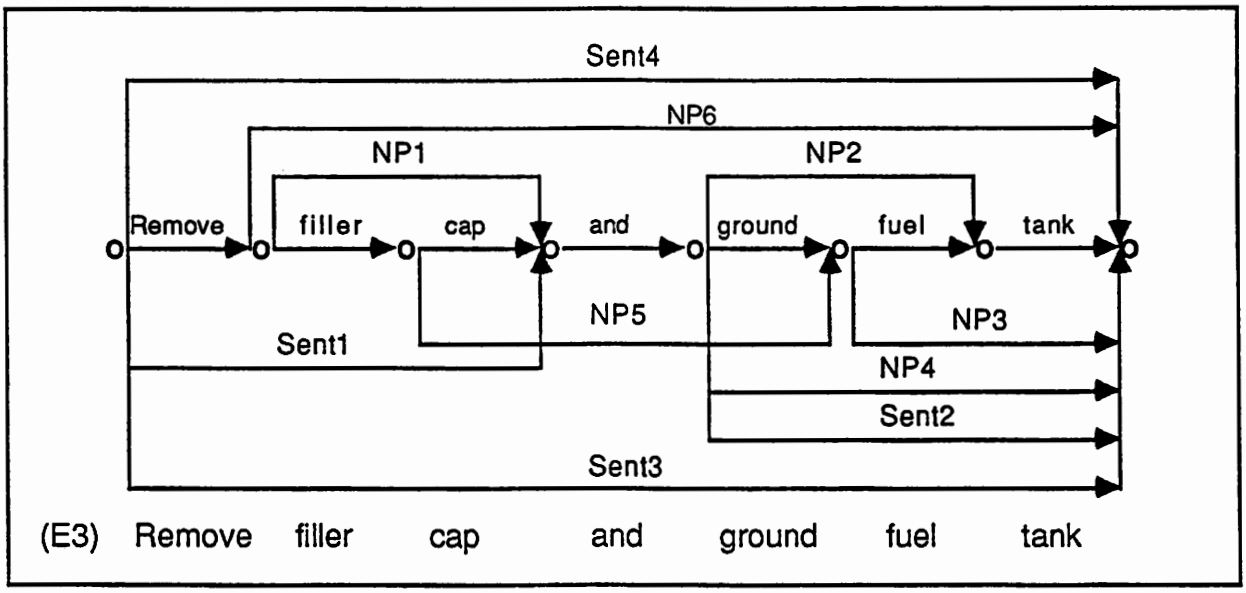

Figure 1 : a chart (built on a syntactically ambiguous sentence)

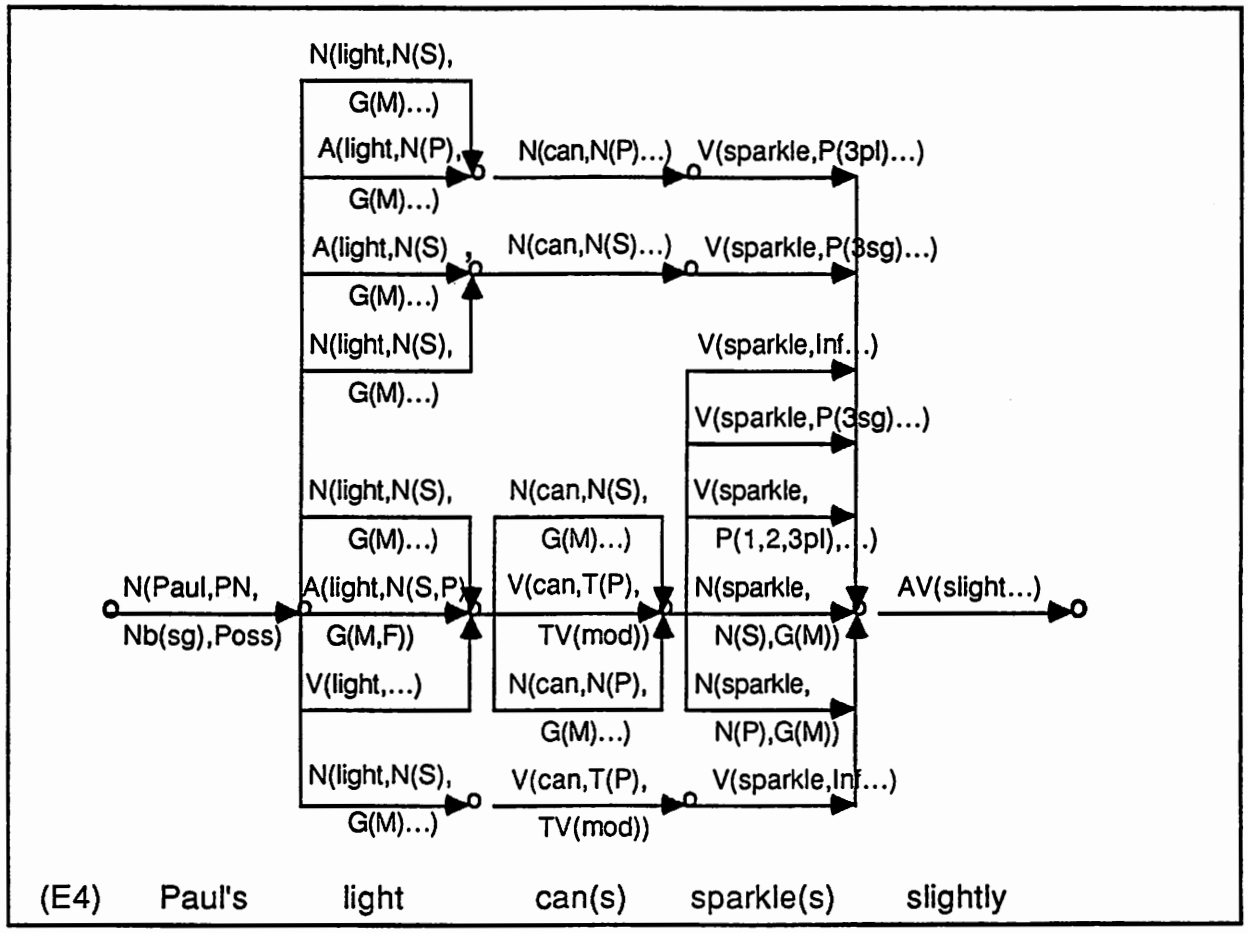

Figure 2 : a Q-graph (built on a phonetically ambiguous sentence)

2.1 Usage in Speech Recognition

Both types of structures have been used in Speech Recognition. Quinton (1980) presents that of the Kéal system, where the output of the phonetic component, the "lexical spectrum", is such a grid. Each node contains a "detection", made of a word and a score. There are two special detections, initial and final, which contain special endmarkers instead of words. ATR's phoneme recognizer also produces a grid. On the other hand, the HWIM system used a "phonetic lattice" on which an extended ATN operated.

2.2 Usage in Machine Translation

Grids have only been used in MT to implement some working structures (like that of the Cocke algorithm). However, we may imagine to use them for representing an input text obtained by scanning a bad original, or a stenotypy tape (Mérialdo 1988).

On the other hand, lattices have been used extensively, in two varieties. First, the *chart structure* has originally been introduced by M. Kay in the MIND sytem (around 1965). In a chart, the nodes are arranged linearly, so that there is always a path between any two given nodes, and the arcs bear the information, not the nodes. This data structure is also used by many unification-based natural language analyzers (Schieber 1985).

The *Q-graphs* of (Colmerauer 1970) and their extension (Stewart 1976) are the basic data structure for text representation in the METEO (Chandioux&Guérard 1981) and TAUM-Aviation (Isabelle&Bourbeau 1984) systems. A Q-graph is a loop-free graph with a unique entry node and a unique exit node. It is possible for two nodes not to be on any common path from the entry to the exit. The information is beared on the arcs, in the form of simply labelled trees, or, in the extension, of trees with labels and binary features.

Actually, none of these structures is strictly a lattice, because two different arcs may link the same pair of nodes ; moreover, a Q-graph may contain two different arcs linking the same two nodes, and bearing the same tree. However, it is always possible to transform a chart or a Q-graph into an equivalent lattice (with the information on the nodes) by replacing arcs with nodes and creating appropriate arcs.

For example, ATEF (a SLLP, or Specialized Language for Linguistic Programming, developed at GETA for writing morphological analyzers) produced initially Q-graphs and decorated trees (Chauché 1975). It was easily adapted (Boitet 1976) to also produce lattices, used as input to "algogrammars" (kinds of bilevel ATNs where one level describes the grammar and the other the heuristic control).

Q-graphs and lattices are also natural structures to represent a text with alternate formulations, like a first draft or a rough translation. For example, the following alternate formulations :

```
(E5)              | it if you came    |
   | would like  | you to   | come | early   | tomorrow
                                   | earlier |
                              | be  early      |
```

could be represented by the lattice of figure 3, or by a Q-graph (with repetition of at least one arc), but not by a grid or a chart.

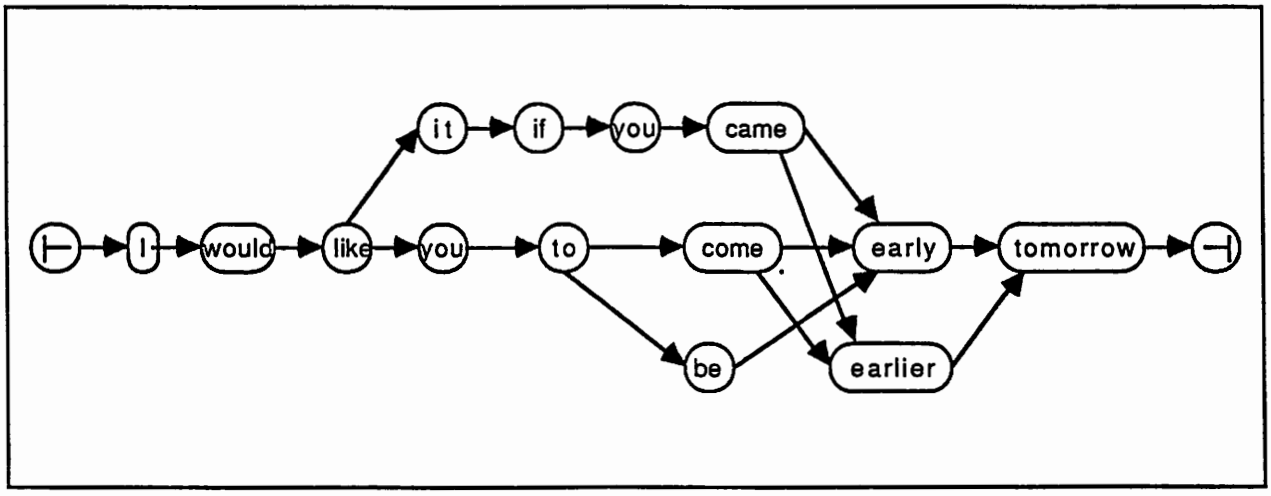

Figure 3 : a word lattice (representing a sentence with alternate formulations)

2.3 Neural Network analogy

Neural networks are the future of speech recognition, and perhaps of language understanding. Their main advantages are *processing speed,* due to heavy parallelism and simplicity of mechanism, *trainability*, and *robustness* in presence of noise, partial destruction, etc. For phoneme recognition, the results are already better than with HHM (Waibel 1988). However, it is not yet proven that they can be scaled up to handle real-life problems, although the very fact that they are founded on biological analogy is encouraging. As far as language processing is concerned, researchers in neural networks are perhaps exaggerating the present practical potential of this technique, but they are certainly opening promising directions (Waltz&Pollack 1985, Touretzky 1987).

We feel that a good design choice would be to use the analogy with neural networks wherever possible, to enable future use of "real" neural networks at more and more levels of NLP. A first consequence of this choice is to *consider lattices, not grids*. No connexions will be implicit. The only information on an arc will be a weight, understood as an excitation or an inhibition potential, and all other information will be contained in the nodes.

To adhere strictly to the analogy, the internal structure of a node should itself be such a lattice (or network), and so recursively until arriving at a stage where, as in real neural networks, the nodes contain no specific information (all execute the same simple operation). However, we will stop at some larger granularity, in order to apply well-known methods involving structured objects.

Of course, a lattice can be trivially constructed from a grid and a transition condition such as that of Kéal (given above). But a lattice is more specific, and the reverse is not true : for example, a grid can not represent the fact that, in a sequence of two phonemes, /o/+/i/, /a/+/i/, /a/+/e/ are possible, but not /o/+/e/. Figure 3 gives another example.

I.3 Layers and classes

3.1 Phonetic layers

The bottommost layer is that of *phonetic codes* (a constant number per frame). Each node corresponds to a given frame, and is connected to its predecessor and successor only. Supposing that the phonetic alphabet contains 256 elements, as in Sphinx, with 3 phonetic codes per frame, there are 256^3 classes in this layer.

If words are not recognized directly, there should be an *intermediate phonetic layer*. It does not seem necessary to plan for more than one. Classes correspond to the units chosen, whatever they may be (see I.1.2) : if we choose the syllable, we will get from 200 (in Japanese) to 20000 (in English – Lee 1988) classes, if we choose the phoneme, from 20 to 50 classes, etc.

3.2 Textual layers

We will consider two textual layers, thought of as representing the text in its written form, with some elaboration. The first one is a *tagged text lattice*. Each node contains a wordform as label (class), and a simple decoration structure. For concreteness, let us sketch a possible declaration of the associated (node) data type, using an extension of Robra's syntax (Boitet&al 1978) :

```
label   classtxt1  =  string     (pot maxwordforms) ;  -- Wordforms will be defined elsewhere.
decor   dectxt1    =  nex all    (                     -- Set type (nex), all attributes obligatory.
        (t1, t2) :    integer [0..tmax],               -- Endpoints. Part of speech scalar (exc):
        cat :         (N, V, A, R, D, S, C),           -- noun, verb, adjunct, representant,
                                                       -- deictor, subordinator, coordinator.
        val :         real [0..1] ) ;                  -- Score (valuation).
```

For example, an occurrence of the verb "can" could be represented as :

'can'[dectxt1 (t1 (195), t2 (205), cat(V), val(0.2))]

Each arc would bear a weight w of type real [minarcw..maxarcw]. In this small example, we have limited ourselves to 7 morphosyntactic classes. In a real setting, we would expand to subcategories (such as various adjective and adverb subclasses of adjunct). "Tagging" with as many as 400 classes has been used successfully in speech recognition.

The second layer, the *lemmatized text lattice*, corresponds to the result of morphological analysis in an MT system. Going from the first to the second layer implies access to dictionaries (to get the valency frames, the semantic features and restrictions, etc.), and use of a morphological grammar (to compute information form roots, prefixes, endings, and various affixes). One node in the first layer can be expanded into a sublattice in the second layer. For example, "can" as a verb could be expanded as two alternate nodes (modal / nonmodal), and "won't" as two successive nodes ("will" + "not").

In this layer, it may be preferred to take as class the morphosyntactic class, and to put the wordform, the lemma, the lexical unit, and the rest of the information in the decoration.

3.3 Linguistic layers

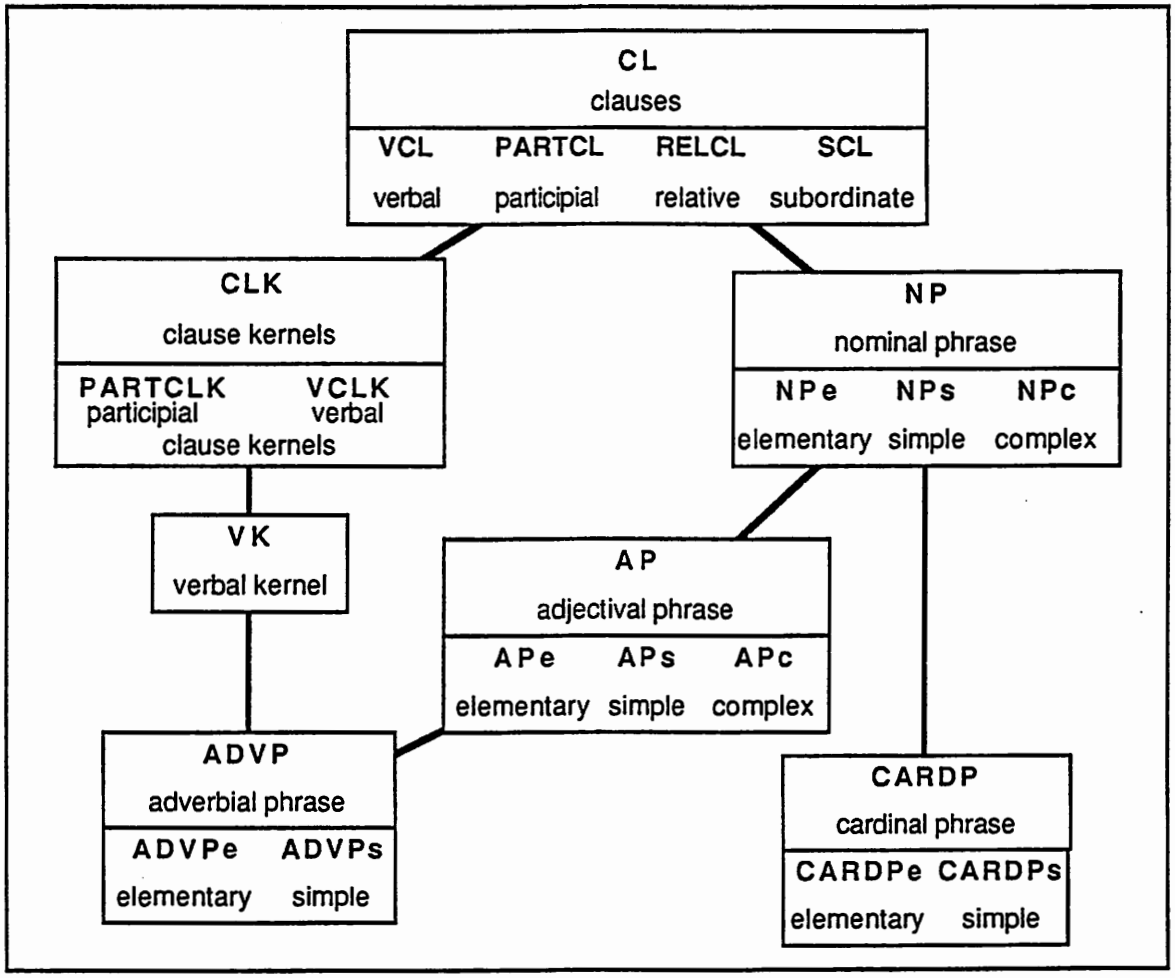

Figure 4 : simplified syntagmatic hierarchy (PP & DP omitted)

Following (Vauquois&Chappuy 1985), we distinguish *three linguistic layers*, corresponding to the *elementary, simple*, and *complex syntagmas*. Elementary syntagmas are usually lexical (N, A–N, N–N, as "Adam Smith"), and simple syntagmas (e.g., Japanese bunsetsus) don't contain embedded syntagmas of higher degree (in a hierarchy where clauses at the top, see figure 4 for an example) : both types may be conveniently described by finite-state grammars or automata.

Finally, we get 6 or 7 layers to represent a unit of translation treated in analysis. We suggest to consider transfer (whether it involves explicit understanding or not) as a transcription phase creating the top layer of a corresponding structure for the target sequence of utterances to be generated.

II. DERIVATION, REPRESENTATION, AND INTERPRETATION STRUCTURES

In each linguistic layer, the nodes will root *derivation trees (d-trees)*. The same node can root several such trees, in case of ambiguity. Inside each node, there will be one or several *representation trees (r-trees)*, each corresponding to one or several d-trees. Each node of an r-tree will in turn (also optionnally) bear a *representation structure (r-structure)*. This section presents the motivations behind this scheme and illustrates it.

A general idea is that, in analysis, going from one layer to the next and from d-trees to i-structures, one makes more and more explicit what was implicit, thereby going away from the "surface" expression and relating parts of the units to preexisting knowledge.

II.1 Derivation trees

A d-tree is the direct image of the production of a string by a syntagmatic grammar, most often context-free (although context-sensitive grammars in normalized form and local (non-distributive) adjunct grammars also naturally yield trees). By nature, a d-tree is in projective correspondence with the string, a homomorphic image of its frontier.

1.1 Motivations

Of course, there are cases of non-projective correspondences, such as :

(E6) *We have all seen them*

where "we...all" is a "discontinuous constituent". D-trees can not represent this explicitly in their geometry. Also, they can not represent explicitly elision (one should "restore" nodes in the tree). But good algorithms exist to compute all d-trees for a given string, with the axiom or any other nonterminal at the root, and this is important if we want to factorize ambiguities in an efficient way, and handle them explicitly, via preference rules, or implicitly, via scoring and heuristic search.

There are some other arguments in favor of d-trees. They can be used as "shallow structures", acting as a safety net in case of failure to reach a more abstract level. In generation, it is also necessary to produce projective trees. Finally, d-trees have been used successfully in systems of extremely large coverage (Heidorn&al, CRITIQUE system, IBM-JWRC).

1.2 Structure of a node and representation of a d-tree

Each node has a class, such as NP, AP, CARDP, VP... and a set of decorations. Each decoration is associated to one or more *rule instances*. A rule instance is a "grey" node relating such a decoration to a list of other decorations (contained in other nodes of the same layer). Decorations of the left and right hand side of a rule are put in correspondence by an equation associated to the rule, in the usual manner (an example is given in II.3.3 below).

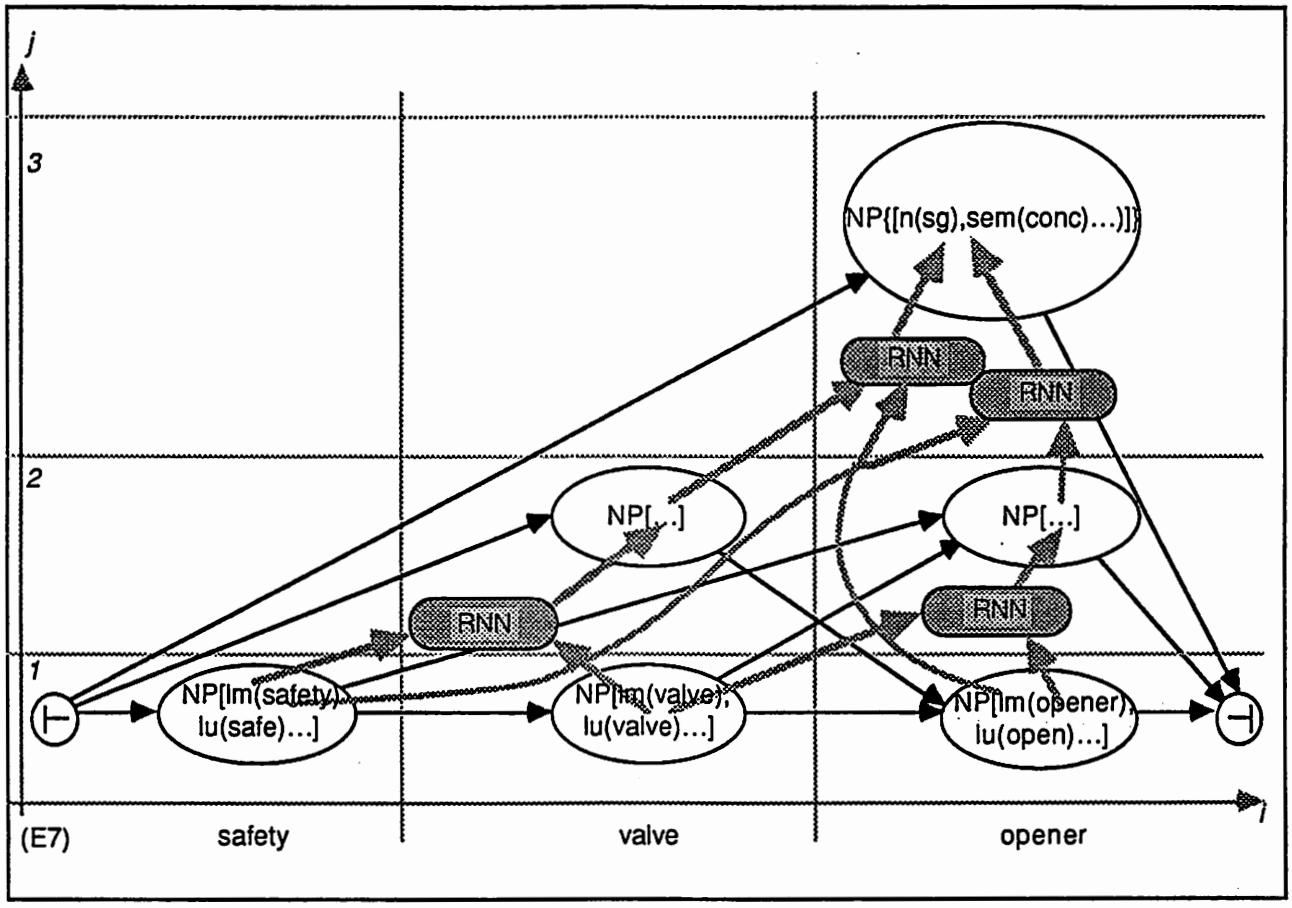

Figure 5 : example of nodes at the simple syntagmatic layer

Figure 5 illustrates this concept (RNN is meant to be the name of a simple rule with context-free skeleton NP –> NP NP). The node at level 3 in this diagram has one decoration, corresponding to two different structures of the simple nominal phrase "safety valve opener". Replacing "safety" by "test", we would get a second node at level 3, with label VCL.

1.3 Decorations

A *decoration* is an unordered finite tree with a symbol on each node, such that no two sister nodes bear the same symbol. A *symbol* is an identifier or the denotation of a *primitive object.* Primitive objects are characters, boolean values **true** and **false**, integers, reals, and vectors of those, of which character strings are a particular case. An internal node must bear an identifier, and any two sister nodes must bear the same type of symbol. Here is an example :

```
dec1 =    deceng  (cat (V (fin, inf)), lm (['give'-'back']),
                        val1 (K(NP), sem (thg)), val2 (K (toNP), sem (persnf)))
```

Symbols on internal nodes are always understood as *attribute names*. The complete name of an attribute is obtained by concatenating the symbols on the path from the root to its node. For instance, deceng.val1.K is the complete name of subattribute K of (deceng.)val1. On a leave, an identifier can be thought of as denoting an elementary value, just as in a Pascal enumerated scalar or set type, or as denoting an attribute with the conventional value **null**. For example, we could replace "V(fin, inf)" by "V" in dec1, getting a decoration dec2 where we would still understand V as a potentially complex attribute.

An attribute has three natural functional interpretations on the domain of decorations : *boolean, immediate* and *complete*. As a boolean function, an attribute is true on a decoration iff its complete name appears as a path in the decoration : truth is equated with presence, falsity with absence. As an immediate function, an attribute is undefined if false, and otherwise equal to the set of labels of its daughters. As a complete function, an attribute is undefined if false, and otherwise equal to the complete set of trees it dominates. For example :

boolf cat (dec1)	= **true**	**immf** cat (dec1)	= V	**compf** cat (dec1)	= V(fin, inf)
boolf drv (dec1)	= **false**	**immf** drv (dec1)	= **undef**	**compf** drv (dec1)	= **undef**
boolf fin (dec1)	= **true**	**immf** fin (dec1)	= **null**	**compf** fin (dec1)	= **null**
boolf val1.K (dec1)	= **true**	**immf** val1.K (dec1)	= NP	**compf** val1.K (dec1)	= NP

Remark that **boolf** K (dec1) cannot be evaluated, because K(dec1) is undefined, as there is a name conflict between val1.K and val2.K. For any function f, it is natural to pose f(**undef**) = **undef**. Note also that **undef** should not be treated as a normal value. Rather, expressions involving this symbol should be first reduced by syntactic (rewriting) rules until it is eliminated and normal evaluation can take place. If convenient, any syntactic sugar can be added, such as "+fin (dec1)" for "fin (dec1) = **true**". For efficiency and engineering reasons, it is useful to *declare decoration types*. We have given a small example above and won't go into more detail here.

II.2 Representation trees

A simple example from computer science can show the difference between a derivation tree and a representation tree. Suppose we want to produce a tree associated to the following expression :

If y = z then x := a else x:= a * (b + c)

The context-free grammar on the left of figure 6 gives rise to derivation tree T1, with 42 nodes, shown on the left of figure 7. The abstract tree T2 for the same string, with only 14 nodes, is shown on the right. T2 is far more simple than T1, and more adequate for various optimization and code generation operations. Notice that its correspondence with the string is neither projective nor total, but that it can be produced by simple transformations attached to the CF-rules (on the right of figure 6).

<cond_expr>	::=	If <cond> then <assig> else <assig>	@0 =	If_then_else (@2,@4,@6)
<cond>	::=	<expr> = <expr>	@0 =	= (@1,@3)
<assig>	::=	Idf := <expr>	@0 =	=: (@3,@1)
<expr>	::=	<expr> + <term>	@0 =	+ (@1,@3)
<expr>	::=	<term>	@0 =	@1
<term>	::=	<expr> * <factor>	@0 =	* (@1,@3)
<term>	::=	<factor>	@0 =	@1
<factor>	::=	(<expr>)	@0 =	@2
<factor>	::=	Idf	@0 =	@1

Figure 6 : a simple syntagmatic grammar and associated transformations

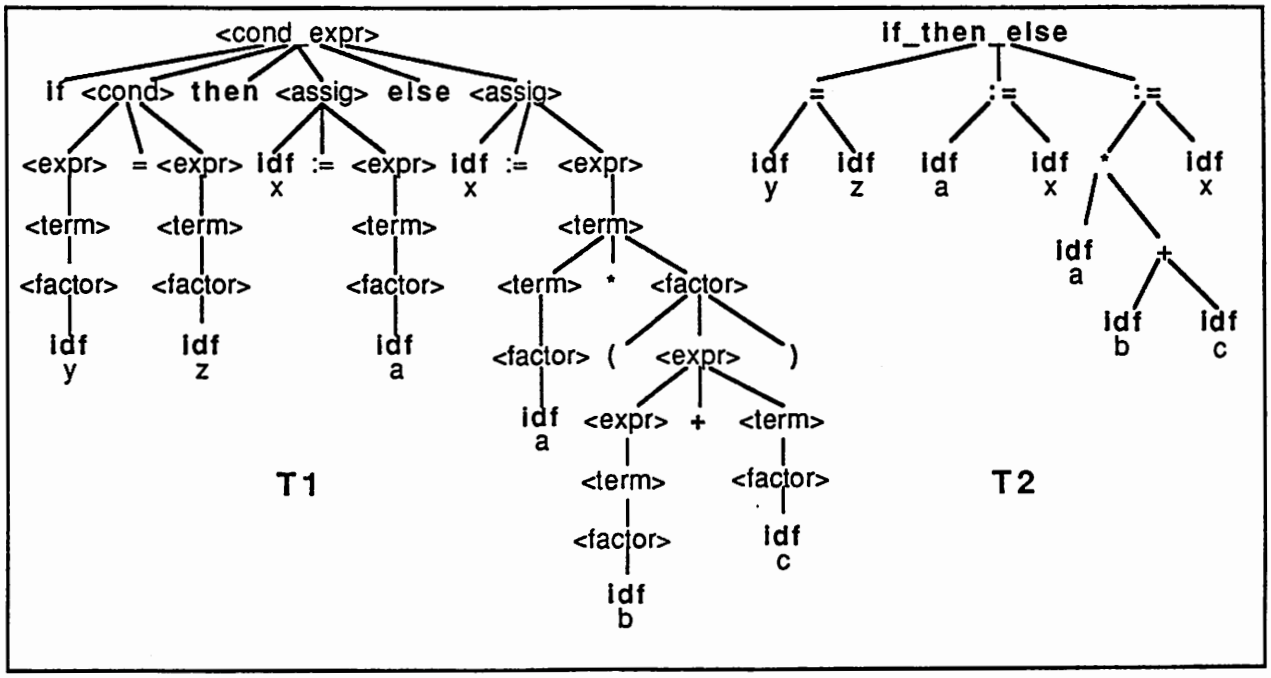

*Figure 7 : a derivation tree and a representation tree for "If y = z then x := a else x:= a * (b + c)"*

2.1 Representation trees in Machine Translation

In MT, an r-structure for a unit of translation must at the same time represent the meaning of the unit and its form, because translation must preserve the meaning and also, if possible, keep some parallelism in the form. Not all paraphrases are translations (see example E9 below). In most translational situations, the domain of discourse is not closed – METEO (Chandioux&Guérard 1981) is a notable exception –, even if it revolves around a formalizable domain, as in the case of conference registrations or business meeting arrangements.

Hence, it makes sense to found translation on an abstract linguistic representation of the unit of translation. To choose between competing r-structures, it may certainly be useful to use an interpretation in a formalized (sub)domain ("explicit" or "expert" understanding), and even to use a model of the ongoing communication and situation ("pragmatic" understanding). But *r-structures should correspond only to linguistically meaningful distinctions*. Starosta (1988) gives a number of powerful arguments to this end.

For example, many semantic features should be used in r-structures, because the corresponding distinctions are *observable* in many languages : human, animate, animal, plant, concrete, abstract, container, time, space, closed, open... But we should stop short of trying to describe a domain of discourse by such means, or else the resulting systems will be completely specific (as METEO). Of course, the line may sometimes be difficult to draw, because language reflects our apprehension of the world, and evolves with it : new words, and even new syntactic constructions, appear constantly (think of *retroviruses,* or of genes *coding for* proteins). For example, a semantic feature of "chemical element" may be justified, but probably not that of "metalloid" .

An r-structure contains lexical, grammatical and relational information. The *lexical elements* should not be wordforms, but lemmas, or even better lexical units (derivational families), chosen to get identical or very similar r-structures for translational paraphrases.
For example :

(E8) *Check that the lift works correctly / Check the correct working of the lift*
are acceptable paraphrases in translation, but not :

(E9) *The repairman has fixed the lift / The man who (does) repairs has fixed the lift*

However, such an "unacceptable" paraphrase may be necessary, if the corresponding word does not exist in the target language. A good place to put information necessary to produce more general paraphrases is a dictionary *à la Mel'čuk,* containing *lexical functions* ("oven" is *Means_of* "bake", "far" is *Antonym_of* "near"...), which are more general than the derivational relations.

The *grammatical information* is centered on the words and the phrases to which they pertain – be they connex or not. Morphosyntactic and syntagmatic categories, number, person, time, aspect, mode, semantic features, are examples of that type of information. The *relational information* shows the role of the elements in a group. It typically includes the syntactic functions (subj, obj1, obj2, atrsuj, sujf, atrobj, compl, epit, determ, reg...), the logical relations (arg0, arg1, arg2, arg01, arg02, trl10, trl20...), representing the predicate-argument structure, and the semantic relations (agt, pat, bnf, qual, means, meas, qtf, qtfier, manner, local, accomp, cause, conseq, goal, orig, dest...). Coreference is another example.

Traditionnally, r-structures are *ordered trees*, and there are good reasons for that. First, *trees are understandable,* because a subtree is essentially independent of the rest of the tree. Second, and much for the same reason, *trees can be processed efficiently*. Third, their inherent (left-right) ordering can be used to *implicitly encode or remember* some types of linguistic information for which there is no good formal description. For example, the theme/rheme/pheme statutory distinction, or the degree of emphasis, are difficult and sometimes impossible to compute in non-configurational languages, but they can still be translated by preserving the order or modifying it in a systematic way. Discourse coherence and good scoping of anaphoric elements also depend on a good treatment of order. Finally, it is possible to look at ordered trees *as if* they would be unordered (e.g., "unordered nodes" in ROBRA tree patterns), whereas the converse is clearly impossible or at best extremely awkward (positions must be remembered as annotations to the nodes).

There is also a series of good arguments in favor of using *decorated* trees. First, this obviously reduces the number of nodes. Second, the geometrical view of the structure is separated from the algebraic view of the attached information. Trees are based on sequences and decorations on sets. Third, decorations are bounded in size and usually declared, which allows for compilation and efficient processing. This is to be contrasted with feature structures (see below). To use only "flat" decorations, analogous to property lists, is of course possible, but this does not reflect the successive refinements in description (e.g., cat(N) with subcategories of noun SubN(CN, PN)), and it is awkward to ensure that, for example, there is no SubN in the absence of N. A hierarchical organization like that illustrated above is almost as easy to implement and far more informative.

There is a lot of talk about multistratal versus monostratal linguistic theories and representations. Following linguists of Western and Central Europe, who have extensively studied natural languages, and not only formalisms, thereby writing real grammars and dictionaries in the context of NLP, we adhere to a *multistratal analysis* of language (Mel'čuk 1981). However, for the purpose of MT, it is most convenient to use a *monostratal representation.* This technique has been originally introduced by B. Vauquois in 1974, and used successfully ever since in several MT systems. More recently, some modern linguistic theories such as Lexicase (Starosta 1988) or GPSG (Gazdar&al 1985) have also advocated a unique representational level.

B. Vauquois' idea is to use the same graph (a tree), on which several levels of interpretation are encoded. A satisfactory compromise is to use a "flat tree" (Starosta 1988 also advocates the use of flat trees for d-trees), where each subtree represents a syntagm and has a particular daughter, the "governor". The governor is the main element, which would be put at the top in a pure dependency representation. Most often, it is a leave, but it can sometimes root a subtree (e.g. in the case of a compound noun, or of a complex verbal kernel). An example is given in figure 8 below.

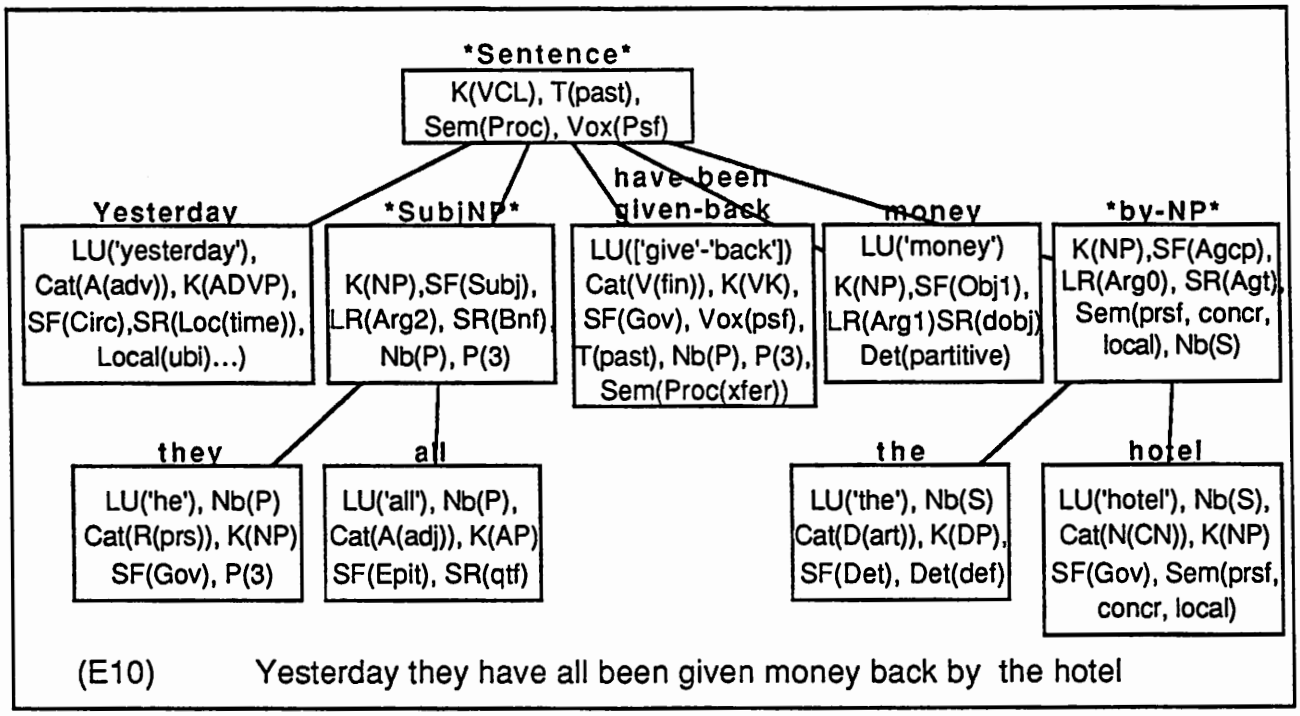

Figure 8 : an r-tree

Decorations have been (very partially) shown on the diagram, within boxes, and labels appear on top of them. Labels are usually used only for giving a condensed information about the decorations on graphic tree displays, which allows to show the detailed decorations separately, or, under a convenient tree editor, on pop-up windows.

Note the absence of a node for "by" : in the case of an argument, the preposition is almost always redundant with the realized valency frame (not shown here). In the case of a circumstancial ("outer case"), the preposition should best be put on the top node of the group (as in classical dependency representations), where it can combine with the SR (semantic relation) attribute to show the exact nuance of the relation. Alternatively, and almost necessarily in case of coordinated prepositions ("I'll stay *before and after* the conference"), they can be left in place, in a subtree with the syntactic function of *"regent" ("régisseur")* at the same level as the governor. Incidentally, there is no more reason to consider that the governor of a prepositional nominal phrase is the preposition rather than the main noun than there is to consider that, in a relative clause, the relative pronoun is the governor rather than the verb.

2.2 Structure of a node and representation of an r-tree

On top of a label and a decoration, each node of an r-tree should also contain a *weight* (score), and, optionnally, an *interpretation structure (i-structure).* In an actual implementation, sub-r-trees will often be r-trees associated to sub-trees of the d-tree rooted at the considered node of the layered lattice, so that structure sharing should easily be implemented (perhaps again by means of "grey" nodes). A typical case is shown on figure 7 above.

Another use of r-trees is to *represent certain types of ambiguities, polysemies or doubts.* In a case like :

(E11) Which hotel runs this office?

it is best to produce only one r-tree, indicating on the top node the ambiguity type (Subj-Obj1 vs Obj1-Subj) by means of attributes (e.g., "Doubt (SF), Ambtype (SubjObj1)"), and assigning the most probable analysis to the daughter nodes (here, Subj-Obj1). The same goes for structural ambiguities of the "safety valve opener" type : choose one standard attachment, and again indicate the other possible choices in the decoration.

This implies that one r-tree may correspond to several d-trees (associated to the same syntagm). The converse should be kept as rare as possible, but may happen.
For instance :

(E12) Mr Adams and Mrs Smith will talk on Tuesday and Friday

has only one d-tree, which shows two coordinated NPs, but the two r-trees corresponding to the distributive and respective interpretations cannot be represented by only one "ambiguous" r-tree in a natural way. The correspondences between r-trees and d-trees could also be represented by "grey" nodes (see figure 5).

2.3 Maintaining text-tree correspondences

The correspondence between an r-tree and a string (a path in the tagged text lattice) is not as easy to describe as that between a d-tree and a string. However, it may be useful to keep it, in particular if some clarification dialogue is planned : if something is not clear to the system, the corresponding part(s) of the unit of translation should be recoverable *verbatim.*

(Boitet&Zaharin 1988) have proposed to attach to each node of an r-tree two (non necessarily connex) substrings of the string "covered" by the entire tree, called SNODE and STREE, which are the substrings corresponding to the node as individual node and as root of its subtree, repectively. With the data structures described here, however, there is a better possibility : SNODE and STREE could refer to lists of d-trees, because a d-tree corresponds naturally to a partial path in each of the two text lattices.

II.3 Interpretation structures

3.1 Understanding as producing i-structures

By definition, understanding is a mapping from a language into a domain, both being formal or formalized. We see interpretation structures are the means to describe such a mapping from the unit of translation to the representation of the domain, the situation, and the ongoing communication. It is desirable to choose i-structures in such a way that the domain, the situation and the communication can themselves be modelled by means of i-strucures. However, for reasons of modularity and adaptibility, i-structures can remain clearly separated from the linguistic and phonetic descriptions.

Following universal practice dating back to Montague, i-structures should depend on the r-structures in a compositional way. Hence, each node of an r-tree should bear an i-structure, computed from the i-structures of its daughters and from any other information, available either on the node (such as label, decoration, weight...) or in the representation of the discourse (identity of speakers, possible referents, etc.).

3.2 Possible kinds of i-structures

There are several possible types of i-structures. The first candidates are logical formulas *à la Montague* (and *à la GPSG*). Their main drawback is that they are rather ill-suited to represent a knowledge base. Remaining in the framework of logic, the next possibility is to use Prolog directly, an i-structure being simply a set of Horn clauses. This technique is well understood and has been used successfully in many applications, most notably for building natural language interfaces with data bases or expert systems, themselves also written in Prolog, although this is by no means mandatory. However, the fact that Prolog does not allow to name subterms of terms forces to encode all information in a positional fashion (see Aït-Kaci 1986 for interesting comments).

Departing from logic representation, frames or semantic networks are worth considering, as they are used extensively in AI for knowledge representation. But these structures offer no clearly defined operation(s), and no standard processing mechanism(s), such as Prolog's unification and resolution. Aït-Kaci speaks of a "glaring lack of formal semantics".

ATR's researchers have chosen to implement i-structures as *feature structures (f-structures),* in the spirit of current "unification-based" grammar formalisms. F-structures have been introduced in (Kay 1981) for representing functional grammars (Dik 1978) and associated linguistic structures. *Usual f-structures are dags* (directed acyclic graphs with a unique root), where labels on the arcs are *feature names* and where leaves (and leaves only) can bear atomic values. *Extended f-structures* are allowed to contain not only reentrant arcs, but also cycles, and will be called *dfgs* (directed feature graphs). Unification applies to dfgs as well as to dags.

It is quite straightforward to represent logical terms or situation-theoretic types by means of f-structures. K. Kogure has kindly suggested the following examples.

1) logical formula :

a) love (John, Mary) -> [[predicate love] [arg0 John][arg1 Mary]]

b) (∀ X)[man(X) ⇒ love (X, Mary)] -> [[quantifier ∀]
[scope [[logical-connective ⇒]
[antecedent [predicate man] [arg0 X]]]
[precedent [predicate love] [arg0 X][arg1 Mary]]]]]

2) situation-theoretic type :

[s | s |= <<run, agent : John ; 1>>] -> [[relation holding-in]
[situation s]
[state_of_affairs [[relation run]
[assignment [[agent John]]]
[polarity 1]]]
[polarity 1]]

In the currently available implementations (Schieber 1985, Tomita&al 1988)), f-structure equations are associated to the rules of a context-free grammar, so that f-structures are associated directly to the nodes of a d-tree, and that there is no r-tree. It is always possible to represent an r-tree in an f-structure, but not in a natural way, because dags are unordered : the geometry of a tree must be encoded by means of special features (usually called first and rest). For the purposes of MT, we are simply proposing to "insert" r-trees between d-trees and i-structures, keeping d-trees for the reasons exposed above.

3.3 Typed f-structures as i-dfgs

The beauty of working with f-structures is that, much like in Prolog, unification is *the* unique operation. However, in practice, there is a need for other operations. To give an example, the non-availability of addition makes it very clumsy to propagate a feature indicating the total length of the associated string. To see this, take the simple language {a+b+}. If we are allowed to use integer features, the following augmented grammar will compute the correct feature (n) on each node of the d-tree :

S->AB / n(0)=n(1)+n(2) ; A->aA / n(0)=n(2)+1 ; A->a / n(0)=1 ;
B->bB / n(0)=n(2)+1 ; B->b / n(0)=1 ;

In D-PATR (Schieber 198), on the other hand, we must encode the length in unary. If we choose the "successor" representation (<n s s 0> = 2), two auxiliary features must be used to encode the beginning and end of a "vertical" list representing the length (without a final 0) :

S–>AB / (0 in) = (1 in), (2 out)=0, (0 out) = (2 out), (1 out) = (2 in) ; A–>aA / (0 n s) = (2 n) ; A–>a / (0 n) = (s 0), (0 in s) = (0 out) ; B–>bB / (0 n s) = (2 n) ; B–>b / (0 n) = (s 0), (0 in s) = (0 out) ;

This is reminiscent of what happened at the beginning of Lisp and Prolog. Very soon, predefined operators, functions and predicates had to be defined to make it possible to program real applications, thereby sacrificing simplicity for usability. It is very useful, and more uniform, to type not only elementary features (those having atomic values), but all features. Good formal semantics for typed f-structures have been developed (Aït-Kaci 1986), together with a Prolog-like calculus. Several universal operations can be defined, such as unification, meet, and difference, on f-structures containing disjunctions. Type specific operations are also naturally available. Natural graphic representations are dags where arcs bear feature names, as before, and nodes bear type names.

The syntax introduced by Aït-Kaci immediately shows that *decorations are in effect simplified typed f-structures,* where dags reduces to trees. The pairs *label⇒type* can be thought of as being on the nodes, with nothing on the arcs. Hence, *the computation of typed f-structures from f-structures and decorations can be done uniformly,* simply by treating decorations as f-structures.

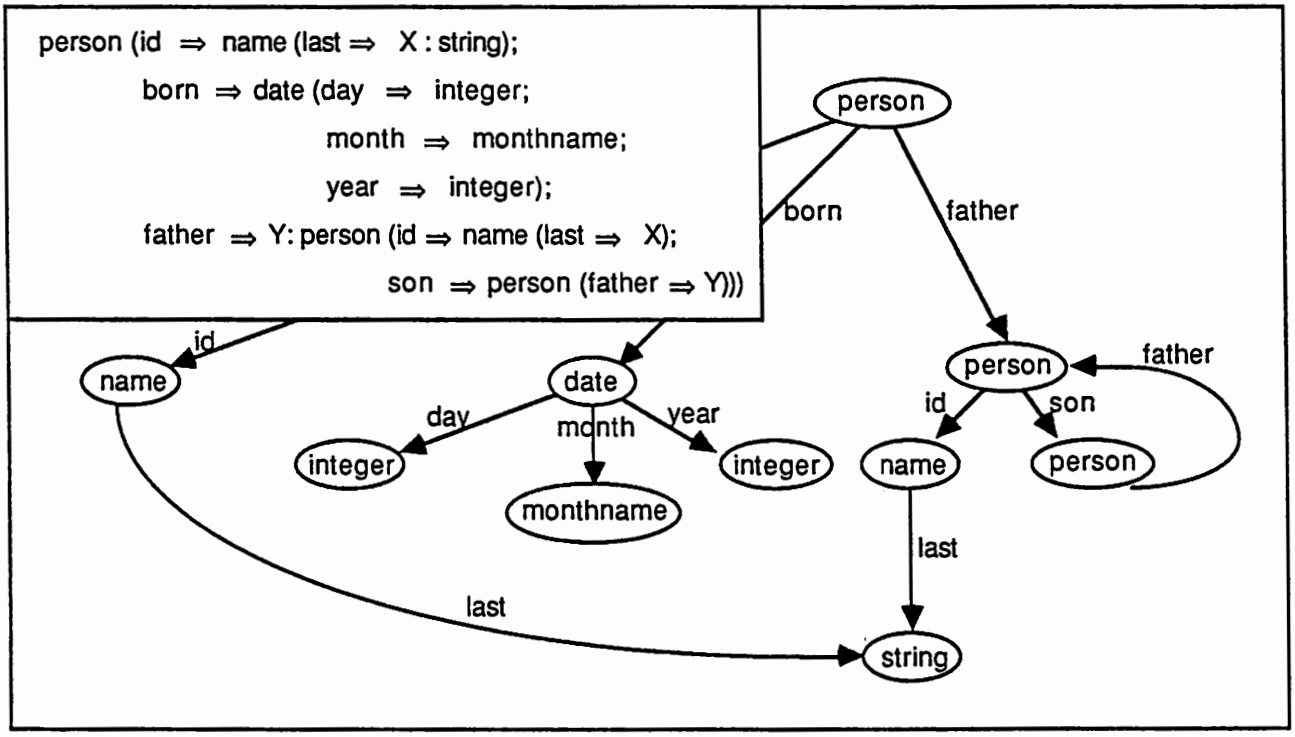

Figure 9 : linear and graphical representation of a typed f-structure (example from Aït-Kaci)

In short, typed f-structures, with their clear formal semantics, open character, and graphic expressive power, seem to be an excellent type of data structure for the interpretation level. They can be used to represent knowledge extracted from the unit of translation (the i-structure) as well as preexisting or accumulated knowledge about the domain of reference, the discourse structure, and the situation. In the following, *we will take i-structures to be typed f-structures, and call them i-dfgs.*

III. COMPUTATIONAL METHODS AND TOOLS

III.1 General strategy

If we want to adhere to the idea of neural networks, the various layers should be computed left to right and bottom up, in a parallel fashion, each process starting (from the left) as soon as the layer immediately below begins to be filled with results. The adequate granularity should be determined by the connections. For example, if there are at most 250 words for 10000 frames, word limits could be rounded up using 4 or 5 frame units. Moreover, the same idea should apply for the computations "in depth", that is going from d-trees to r-trees to i-dfgs.

However, a considerable amount of work remains to be done before a complete neural network implementation of the phonetic level can be used in practice, and even more for other levels. Hence, control can not be ignored, or "left to the cells", and more classical methods must be used. Completely distributed control as in Hearsay seems to be quite difficult to debug, and very costly. *Hence, it can be suggested to use a central scheduler to control all layer-specific processes, but to leave each process free to apply its own strategy.* For instance, purely left-right sequential decoding could be used at the phonetic level, while the computation of d-trees could be some-path left-corner.

An essential remark is that (in analysis) techniques vary from data-driven to pattern-matching to unification-based when going up in the layers and deep in the nodes, and that, at the same time, the elementary operations become increasingly complex. To make the whole process workable (recall the goal of real time processing), *costly computations should be delayed until they become necessary.*

1.1 Interplay of various knowledge sources and overall control

For reasons of modularity, the various knowledge sources should be developed separately, and in a "static" way, that is without reference to a particular way of use. This is true for the phonetic, lexical, grammatical, and domain-specific data bases. Then, as has been advocated by the CMU group (Tomita&Carbonell 1986), knowledge necessary for each process should be extracted and compiled into an integrated representation suitable for the process. This ensures that different computational strategies can be experimented with without having to rewrite the basic knowledge.

This kind of *integration* of various knowledge sources is essentially static. By *interplay,* we mean the dynamic interactions between the running processes attached to each layer. As said before, no process should directly call another, communication between two processes occurring only through the data passed from one to the other (i.e., the lattice between them). However, there are cases where some real interplay seems necessary.

Suppose for example that the linguistic parser, using the syntaxo-semantic valency frame of a verb, is able to create the expectation of a nominal phrase with certain semantic properties (substance, means of transportation, location, container, document...), at some distance from the verb (and possibly *before* it, in languages like Japanese). This might help the speech recognition part. But, even if the expected phrase occurs after the verb, it is hard to see how the phonetic process, probably based on a Markovian model using digrams or trigrams, could incorporate it, because the distance to the verb may be quite longer than 2 or 3. Another situation, often experimented by humans, is that a part of the message "does not make sense". In such a case, it seems necessary to recall the phonetic form of the message and to reanalyze it in a more detailed way, maybe with some set of expectations.
Take for example :

(E13) I'll be flying on the 40th with Mr Johnson and running by JAL from L.A. to Narita

Could it have been "14th" and not "40th", could "running" be not a verb, but a proper noun ("Mr Running") ?

In our view, such communication should be handled by the overall control program, which, receiving a message from a process, and having access to part of its working structure (the white and grey parts), would send an appropriate message to another process, acting as an *intelligent scheduler*. Note that this permits to distribute processing on physically different machines, which would be impossible with a more interleaved control. As it is, some specialized hardware may be necessary to achieve real time (such as the Beam hardware used by Sphinx), while not being suited to run all processes.

1.2 Bilevel translation ?

It has been advocated by (Tsujii 1987) to translate at two levels of quality. "Important" parts should be translated fluently, in a "content-bound" way, while "not important" parts could be translated not so well, in a "structure-bound" way. Tsujii gives the following example :

(E14) Tomodachi to disuko ni ikitai no de, Roppongi no chisaku no hoteru ga ii no desu ga.

(1) I prefer to stay at a hotel near Roppongi, because I would like to go to discos with friends.

(2) Because I want to go to a disco with friends, a hotel near Roppongi is good.

While (1) is "content-bound", (2) is "structure-bound". Actually, a purely linguistic translation, which can not use the information that the sentence is a wish (this comes from the dialog structure and the description of the current situation), can be quite better than (2) :

(3) A hotel near Roppongi would be good, because I would like to go to discos with friends.

Remark also that it is not clear whether one or several hotels, discos and friends are considered (because number is usually not marked in Japanese), and whether the speaker speaks for himself or for a group ("we" should then replace "I").

This idea can be made somewhat more precise. First, the clarification dialogue, if any, could dispense with the "unimportant" things. In the example, the number of discos and friends is not important, but the number of hotels certainly is. How can the system know what is important or not ?

A very simple criterion could be the following : *by definition, important things are denoted by words which have an interpretation in the description of the domain.* Obviously, the important parts will be better translated than the unimportant ones, if they are better analyzed, due to interactive disambiguation.

A second aspect of the idea is to translate the "meaning" of the important parts, and only the "expression" of the rest. How to do this in practice seems difficult, and no solution has yet been proposed. In the framework presented here, however, we can propose a method :

- *decompose the result of analysis in smaller i-dfg(s) and r-tree(s) ;*
- *perform content-bound translation on i-dfgs ;*
- *perform structure-bound translation on r-trees ;*
- *recompose the results to get a complete target r-tree, passed to generation.*

Having obtained a complete r-tree for the unit of translation, one would compute the i-dfg of each node, and its weight, in such a way that the weight reflects the "importance" of the corresponding subtree (e.g., the proportion of nodes which lexical units have been unified with domain-specific concepts of the knowledge base). The important and not important parts would then be separated, translated at the appropriate level, and recombined to get a target r-tree, to be passed to the generator (see III.3 for more details).

1.3 Computational tools as adaptors and constructors

The computational tools are envisaged as *specialized languages for linguistic programming (SLLPs)* adapted to different kinds of computations on linguistically motivated data structures. GRADE, ROBRA, the Q-systems, D-PATR... are well known examples. Each static knowledge base and each dynamic process should be written in an appropriate SLLP.

In MT systems, it is usual to call processes *phases*. Typically, lexical phases (morphological analysis, lexical transfer, morphological generation) incur for less than 10% of the overall cost, while structural phases are heavy (for example, 50% for structural analysis, 10% for structural transfer – if adequate r-trees are used –, and 30% for syntactic generation). In Ariane-85 (Boitet&al 1985), to provide added modularity, we have also introduced intermediate phases used to map whole structures from one decoration type to another, or to perform "lexical expansion" by simple dictionary look-up (lexical transfer is a special kind of lexical expansion). Their cost is negligible.

We propose to call the processes which don't involve extensive search *adaptors*, and the others *constructors*. Running adaptors involves only simple computations and (possibly) access to dictionaries, so that it is relatively straightforward to optimize them, by choosing good representations (hash-tables, B-trees...). Another difference is that, because of the risk of combinatorial explosion, constructors are often equipped with a search strategy designed to avoid exhaustive search, thereby producing only one or a few of the possible solutions.

As a general rule, *any two independent constructors should be separated by at least one adaptor,* so that a modification in one does not necessitate a modification in the other (certain, like those of the syntagmatic layers, are naturally interdependent).

Let us now study the computational methods and tools usable for the two main parts of the interpretation process, namely the *extraction part*, analysis, made of recognition, structuration and understanding, and a the *production part,* made of transfer, generation, and synthesis (of speech and text).

III.2 Analysis : recognition, structuration and understanding

Recognition is the passage from the signal or the coded phonetic string to the tagged text lattice or to the lemmatized text lattice, depending on whether morphological analysis is complex, or simple enough to be integrated in the phonetic recognition. *Structuration* is the computation of the complex syntagmatic lattice, with a final adaptor extracting the "best sublattice" (ideally, one arc with one r-tree). *Understanding* is the computation of i-dfgs attached to the r-tree nodes.

2.1 Recognition

It is now possible to obtain impressive results in speech recognition with very large dictionaries (Mérialdo 1988) of the order of 45000 lemmas (200000 words). The results are better than with dictionaries of 2000 lemmas (10000 words), already quite large by speech recognition standards, because the problem of unknown words disappears. In an application such as conference registration, using a very large dictionary would permit to include a lot of proper names, and to be almost sure that any unknown word is a proper name. It must be said that these results have been obtained on French, the speakers being asked to pause slightly between syllables, which is acceptable in this language. It is however claimed that this limitation will soon be removed, by suppressing the silence part contained in the Markov machine representing a syllable, in order to process continuous speech.

The *language model* is based on trigrams of morphosyntactic classes, because, with such a large dictionary, it is impossible to train directly on words. The probability that word W_i of class C_i follows $W_{i-2}\ W_{i-1}$ of classes C_{i-2} and C_{i-1} is taken to be $p(W_i \mid W_{i-2}\ W_{i-1}) = p(W_i \mid C_i) * p(C_i \mid C_{i-2}\ C_{i-1})$.

The use of order 2 (trigrams) rather than higher orders reflects the concern with space (with 200 classes, there are 8M trigrams and 1.6G quadrigrams) and with trainability (as it is, corpuses of dozens of millions of running words are required to get meaningful estimates of trigram probabilities).

For the purpose of further structure and content analysis, the recognition system should deliver a tagged or lemmatized word lattice rather than a string of wordforms. Homophone paths *(this man / these men)* can be disambiguated later. If the written text produced by the recognizer is to be corrected interactively before translation, it makes sense to propose some choices to the user (e.g., by presenting the text as in figure 3), so that s/he has only to tick the correct one, and not to retype.

We take *morphological analysis* to be a part of recognition. For highly inflected languages, this phase may involve search, but its computational complexity is low. Convenient SLLPs for writing morphological analyzers may be based on finite-state non-deterministic transducers, as ATEF (Chauché 1975), or on simple context-free analyzers as in SALAT (Hauenschild&al 1979).

2.2 Structuration

Construction of d-trees

The next two layers to be computed are those of elementary and simple syntagms. Several groups working on texts have already developed very efficient methods based on statistics to compute them (Débili 1982, Garside&al 1987). They go further than morphosyntactic tagging. For example, Débili, Fluhr and their colleagues have developed SPIRIT, a complete textual data base system (accessible through the Minitel network), which retrieves documents on the basis of elementary syntagmas represented as dependency structures extracted from the questions and matched against the similarly preprocessed documents.

To construct these two layers, there is a choice between :

- *adapting statistical methods from speech recognition,* by developing a simple SLLP with rules of the form "<sequence of nodes> –> <small tree>" ;

- *using the same formalism and algorithmic method as for the complex syntagmatic layer.*

For the complex syntagmatic layer, we may turn again to some augmented context-free formalism, and use Quinton's algorithm, specifically designed to handle word lattices in the context of the Kéal speech recognition system (Quinton 1980).

Rather than to implement a unique algorithm, Quinton defines a *class of algorithms* obtained by specifying separately

- *the general framework :* a bottom-up adaptation of Earley's technique, involving the use of *dotted rules ("configurations"),* a *set of "situations",* and *three basic operations (read, predict, concatenate) ;*

- *the strategy :* using 3 standard lists of situations and simple operations on them, Quinton has effectively implemented and tested deterministic and nondeterministic variants of various strategies (best-first, backtrack search, beam search, beam search with backtrack, sequential stack decoding).

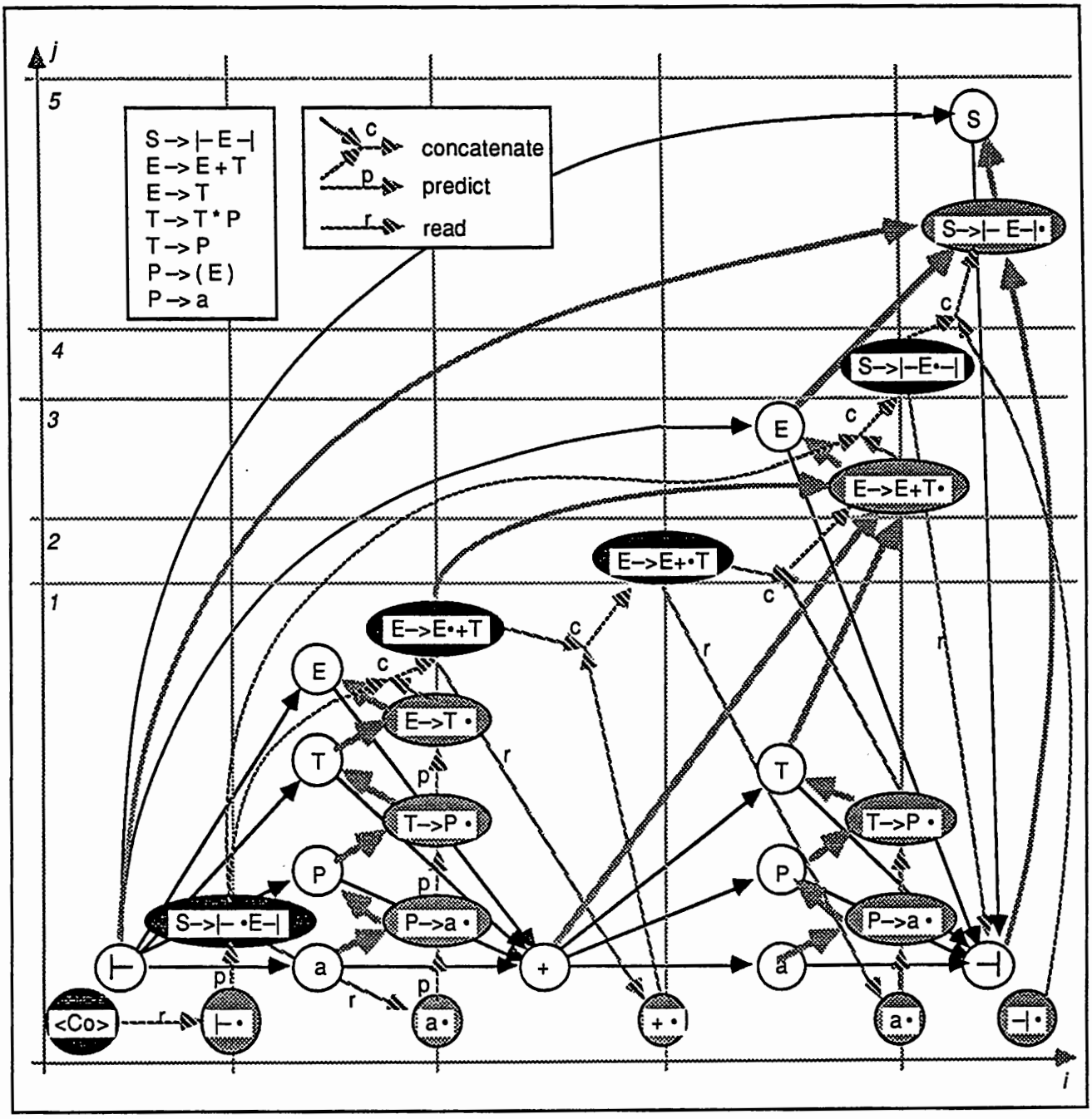

Figure 10 : the syntagmatic layer with a representation of structures used by the Quinton algoritm

A situation is made of a path in the lattice (called "sequence of detections") and of a stack of configurations. A situation can be simply coded as S = (n, c, p, v), where n is the last node of the path, c the topmost configuration of the stack, p a pointer to the situation containing the preceding configuration, and v the valuation of S. This amounts to encode the set of situations in a factorized way. The algorithms are all bottom-up, and behave like shift-reduce parsers (of which the LR parsers are a well-known subclass), augmented with a *prediction* operation.

Quinton's method has several interesting properties :

- *no heavy compilation is needed, as* the relations between grammar symbol and rules used by the elementary operations can be computed incrementally ;

- *elementary operations involve at most two stack symbols* (in effect, a transformation to Chomsky's binary normal form is performed dynamically and transparently) ;

- *the prediction operation,* which replaces the topmost configuration of a stack, is inspired by that of Earley, but is much more efficient, prediction of useless configurations being avoided by examining the next possible nodes in the lattice and the preceding configuration to ensure that reduction will indeed be possible.

Figure 10 above illustrates how the working structure of this algorithm can be represented in one of the layers. The white nodes constitute the lattice, constructed here from a very simple initial linear path representing "a + a", the black nodes encode Quinton's "situations", and the grey nodes and arcs, showing the derivational history, implement the d-trees.

Construction of r-trees

The construction r-trees from d-trees is not always as simple as what has been shown in II.2 above. Sometimes, it necessitates the power of a tree-transformational mechanism like that of ROBRA (Boitet&al 1978) or GRADE (Nakamura&al 1984). Suppose for example that we want to model distributive (and nonprojective) natural language constructs, such as :

Linguists,	*lexicographers*	*and engineers*
describe,	*index*	*and implement*
grammars,	*dictionaries*	*and NLP-systems.*

We can abstract them by considering the strictly context-sensitive formal language $L = \{ a^n b^n c^n \mid n \geq 1 \}$. At least two structural descriptors seem "natural" for a string $w_n = a^n b^n c^n$, namely :

(T1) $S(A(a_1,A(a_2,...A(a_n)...)), B(b_1,B(b_2,...B(b_n)...)), C(c_1,C(c_2,...C(c_n)...)))$

(T2) $S(a_1,b_1,c_1, S(a_2,b_2,c_2,... S(a_n,b_n,c_n)...))$

Tree T1 gives rise to a projective correspondence, but does not imply any particular relation between a_i, b_i, and c_i . If we want to consider the triples a_i, b_i, c_i directly as discontinuous constituents, T2 is better.

T1 can obviously be produced by attaching adequate translations to the rules of the grammar

S –> ABC A –> aA | a B –> bB | b C –> cC | c

in the following manner :

S –> A B C / n(0) = n(1) = n(2) = n(3) / @0 = S (@1, @2, @3)

-- @0, @1, @2, @3 denote the r-trees associated to S, A, B, C, respectively.

-- In the following rules, @1 denotes the r-tree associated to a terminal element (a, b, or c), that

-- is, by convention, the r-tree made of one node containing the element as label, and any

-- other information coming from a dictionary in the decorations.

A –> a / n(0) = 1 / @0 = A (@1) A –> a A / n(0) = n(2)+1 / @0 = A (@1, @2)

B –> b / n(0) = 1 / @0 = B (@1) B –> b B / n(0) = n(2)+1 / @0 = B (@1, @2)

C –> c / n(0) = 1 / @0 = C (@1) C –> c C / n(0) = n(2)+1 / @0 = C (@1, @2)

To generate T2 is a problem, because it is necessary to wait until the three groups (A, B, C) have been constructed to begin the construction of T2. This requires the ability to "decompose" trees. Here is an example, using a tree-transformational grammar with just two rules.

S –> A B C / n(0) = n(1) = n(2) = n(3) / @0 = *G* (S (@1, @2, @3))

A –> aA | a B –> bB | b C –> cC | c -- Same equations as above.

-- *G* is a tree-transformational grammar, where R2

G = {R1, R2 (*G*, S.2)} -- recursively calls *G* itself on the subtree of its

R1 : S (A (a), B(b), C(c)) –> S (a, b, c) -- image rooted at S.2.

R2 : S .1(A (a, @1), B(b, @2), C(c, @3)) –> S.1 (a, b, c, S.2 (@1, @2, @3))

2.3 Understanding

In II.3.1, understanding has been equated with producing the i-dfgs attached to the nodes of an r-tree. Their computation has been described. A remark is in order at this point : the lowest cost to-date (Kasper 1987) of unification of two f-structures f and g containing n symbols and d disjuncts is in $O(2^d\ n \log_2 n)$. On top of this, the size of i-dfgs grows from the leaves to the roots of the supporting d-trees or r-trees (unlike that of decorations), because unification never removes and usually adds information. Hence, a combinatorial use of unification is to be avoided at all costs. However, once one or a few d-trees have been produced, there will be an upper limit on the cost of constructing the i-dfgs, and linear speed-ups are always possible.

One should not forget that the final lattice extracted from the last syntagmatic layer will contain exactly one node and one r-tree only in exceptional cases. Often, there will be several complete analyses, represented in one node (covering the entire unit of translation) by competing r-trees, or, even worse, there will be no complete analysis, so that a lattice of several nodes, corresponding to partial analyses, will be produced. In this case, a simple adaptor should transform this structure into a unique r-tree (where the "top" mirrors the geometry of the lattice) – or the discourse-level process would be designed to accept a lattice of i-dfgs.

III.3 Transfer, generation and synthesis

The complete architecture envisaged is shown on figure 11. In the text below, we have often simplified by speaking of trees instead of lattices of trees. In the diagram, we have tried to make up for that by picturing lattices wherever they could appear.

3.1 Bilevel transfer

Having computed the i-dfgs attached to the nodes of the resulting r-tree(s), there seem to be essentially *two main methods of separating the "important" and "unimportant" parts.* The first method gives priority to the r-tree : "prune" the r-tree by erasing every forest dominated by a node having an i-dfg weight exceeding a certain threshold, getting a *top r-tree,* or *t-r-tree.* The pruning operation should store in each leave the identifier of its i-dfg, for later recombination.

As said before, *the t-r-tree itself (without i-dfgs) would then be passed to a classical transfer component, while the i-dfgs of the leaves of the t-r-tree would be translated by a "deep transfer", at the "explicit understanding" level.*

The "deep" transfer will involve two steps. *First,* athough i-dfgs should be language independent, it is unavoidable that they reflect some characteristics of the expression from which they are computed, so that a restructuring step, or *conceptual transfer,* will be needed to take care of differences between languages. For example, proper names and appropriate honorifics have to be introduced when translating from English to Japanese, and the same goes for determinacy (a hotel, the hotel, hotels, the hotels) in the reverse direction. *Second,* a *conceptual generator* of the target language will transform the i-dfgs into as many *translated "bottom" r-trees,* or *tb-r-trees.*

The last transfer operation consists in *recombining* the translated t-r-tree (tt-r-tree) and the tb-r-trees to produce a complete r-tree representing the unit of translation in the target language. A simple way is to submit the forest made of the tt-r-tree and the tb-r-trees to a tree-transformational system written in the same formalism as all other tree transformations performed elsewhere in the system (production of r-trees in analysis, structural transfer, structural generation). This eliminates the need of any special conventions.

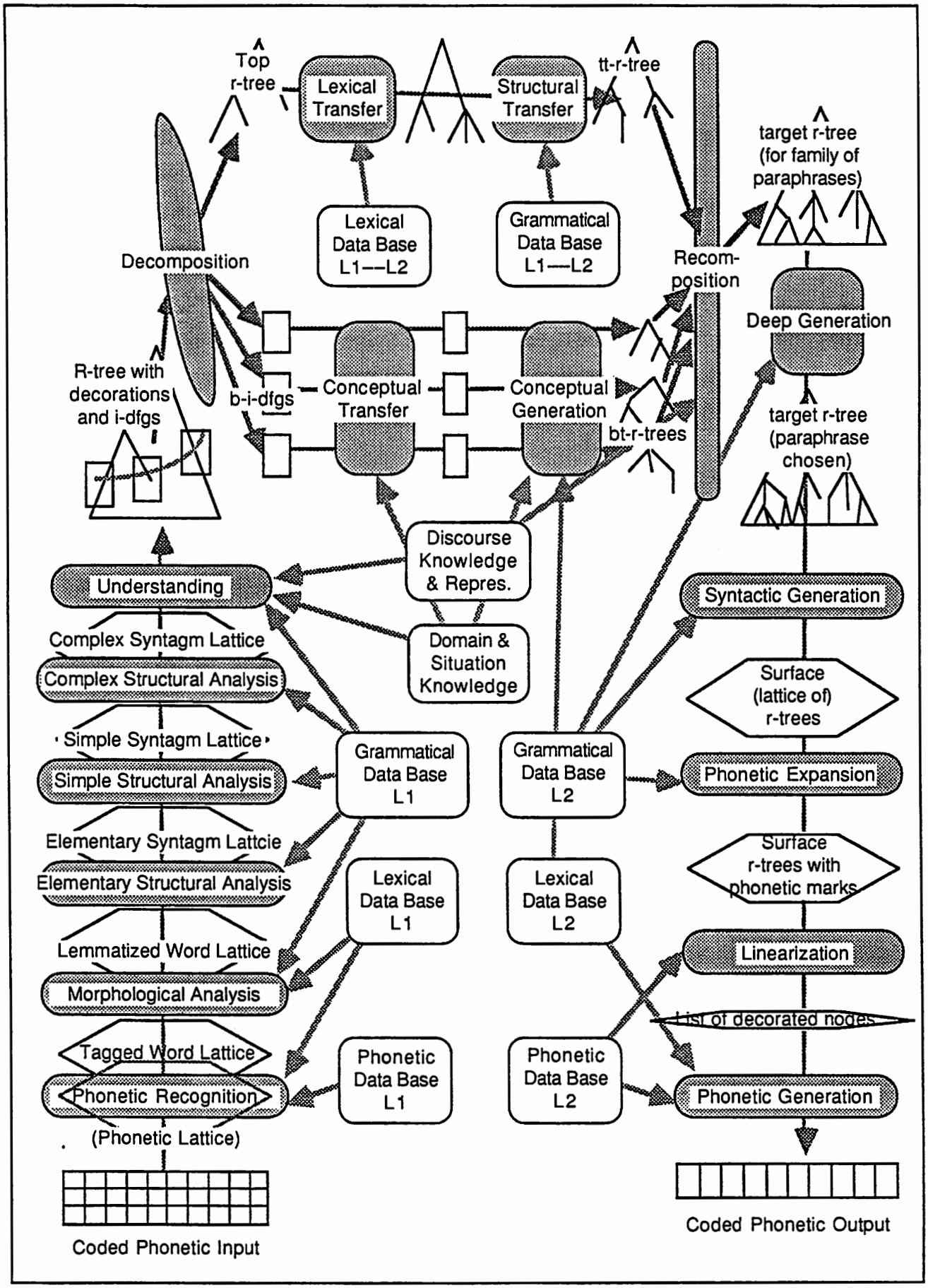

Figure 11 : general diagram

The second method for separating content-bound and structure-bound parts would give priority to the i-dfg. Remember that the unit of translation is represented by the i-dfg associated to the root of an r-tree. Because the computation is largely compositional, this i-structure will contain substructures identical to some i-structures associated to lower nodes of the r-tree. If, during the construction of each i-structure, a special feature is attached to its root, with the identifier of the node as value, it will be possible to traverse the i-structure (from its root) and to prune it of "unimportant" parts marked as associated to sub-r-trees, replacing them with the identifiers of these trees, again called b-r-trees. Then, everything would happen as before, with the only difference that the decomposition would be in one i-dfg and several r-trees instead of one r-tree and several i-dfgs.

Which method to choose should be decided on experimentally. The second method seems intuitively more appealing, because the topmost level is used first. Decomposition may be more difficult to perform, because i-dfgs are not trees, but it may be argued that deep generation, required in both methods, involves a similar decomposition operation anyway.

3.2 Generation

Generation transforms the (lattices of) r-trees delivered by transfer into (lattices of) d-trees representing the target text in a projective and complete manner. The natural computational tools for this are *tree-transformational systems, possibly augmented* to handle lattices – as REZO (Stewart 1976) or the Q-systems (Colmerauer 1970).

One should actually distinguish two steps in generation. First, *deep generation* consists in *choosing a paraphrase* from the set of possible expressions associated to the target r-tree (see II.2.1, examples E8 and E9). This usually involves the recomputation of the syntactic functions, syntagmatic classes and morphosyntactic classes from the highest levels of interpretation, namely logical and semantic relations, as well as the determination of the surface ordering. At this stage, the use of lexical units rather than lemmas avoids a potentially costly consultation of dictionaries from within rules. In practice, one writes a transformational system which starts from the root and performs these operations in a recursive top-down fashion, the initial goal being to generate the syntagmatic class predicted on the root.

The second step consists in *generating a surface tree* (d-tree), by producing all missing nodes (articles, valency-bound prepositions, auxiliaries...), suppressing parts to be elided, and computing all surface attributes, thereby ensuring correct agreement where needed. In a SLLP like ROBRA, this can be done by a relatively simple tree-transformational system, working in totally parallel mode.

3.3 Text and speech synthesis

The *synthesis of a written text* from a d-tree is very simple : the frontier (left-right sequence of leaves) is passed to a morphological generator. Using the information contained in the decorations (lexical unit, morphosyntactic class, gender, person, number, tense, mode...) and a bounded context, a simple grammar controls the assembly of morphs (roots, affixes, punctuations...) found in dictionaries and takes care of all particular cases (e.g., elision in French).

To give an idea, GETA's morphological generation grammar of French is exhaustive, and only 2 pages long (in SYGMOR), while the morphological analysis grammar of Russian, also exhaustive, is about 40 pages long (in ATEF).

The synthesis of a spoken text is somewhat more complex. Its goal is to produce a *coded phonetic target text,* to be passed to a physical (programmed) device producing speech. The phonetic coding for synthesis is obviously different from the phonetic coding from which recognition starts. We suppose that, as in some existing text-to-speech systems, it consists in a string of symbols representing :

- *the phonetic units* to be generated ;
- *the speech parameters* (stress, melody, intensity, voice type...).

Starting from the same d-tree as text synthesis, speech synthesis would then begin by a process of *phonetic expansion* in which the d-tree would be transformed, essentially by the addition of new leaves containing the desired speech parameters in their decorations.

The designers of text-to-speech systems have all encountered the necessity of performing a syntactic analysis to produce a natural prosody. However, resource limitations have forced them to rely only on very shallow analyses. In the case of MI, not only a complete syntactic and semantic analysis will be available, but it will even be possible to transmit from the situation and discourse understanding level some information useful for speech synthesis, like theme, emphasis, or speech act type.

The resulting d-tree would then be passed to *phonetic generation,* a straightforward adaptation of morphological generation. The last process, of course, is *speech generation* from the resulting phonetic target text.

CONCLUSION AND PERSPECTIVES

As Machine Translation, Machine Interpretation is an engineering entreprise. Involving speech recognition and synthesis, it is even more complex. The architecture presented above may also seem complex. In a sense, this is unavoidable. However, we have tried to keep it as simple as possible, by proposing a unique type of data structure, compatible with all algorithmic methods which have been successfully applied in Speech Recognition, Machine Translation and Language Understanding, and also compatible with future developments in Neural Networks.

We hope that work in this direction will provide a sound basis for developing operational MI systems and for improving the robustness of MT systems. From a more speculative point of view, this type of architecture may also be seen as a framework for using statistical techniques at higher levels of linguistic abstraction than what is currently done, and for paving the way for future use of the promising neural network techniques, without loosing the benefit of well proven symbolic techniques, which in our opinion can not be replaced by statistical methods when it comes to explicit discourse understanding and planning.

ACKNOWLEDGEMENTS

Several parts of this paper have benefitted from numerous discussions with ATR's researchers, most notably A. Kurematsu, K. Kogure and T. Aizawa for questions concerning the goals and the technical organization of an MI system, K. Yoshimoto, H. Ueda, M. Kume and H. Iida for linguistic aspects, K. Shikano, T. Morimoto and K. Kita for questions of speech recognition, R. Zajac, Y. Nicolas and K. Kogure for the definition and implementation of unification of typed structures, and, last but not least, A. Waibel and P. Haffner for their enthusiastic presentation of neural networks.

For everything concerning translation proper, I am most indebted to late Pr Vauquois, and to all colleagues of GETA. However, the views contained in this paper should don't necessarily represent those of all or any of the preceding persons, and all inaccuracies, misrepresentations or outright errors are of course mine.

REFERENCES

Hassan Aït-Kaci (1986) *An algebraic semantics approach to the effective resolution of type equations.* Theoretical Computer Science 45 (1986), 293–351.

T.S. Anantharaman, R. Bisiani (1987) *Computer Architectures for Speech Recognition.* in "The Characteristics of Parallel Algorithms", L.H. Jamieson, D. Gannon, R.J. Douglass, eds, MIT Press, 1987, 169–190.

Hidezaku Arita, Kiyoshi Kogure, Izuru Nogaito, Hiroyuki Maeda, Hitoshi Iida (1987) *Comparison of telephone and keyboard conversation.* ATR report TR-I-0016, December 1987, 21 p.

ATR NLDI workshop (1987) *Summary of workshop on Natural Language Dialogue Interpretation.* ATR report TR-I-0017, November 1987, 53 p.

Jeffrey Barnett, Morton I. Bernstein, Richard Gillman, Iris Kameny (1980) *The SDC Speech Understanding System.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 272–293.

G. Edward Barton Jr., Robert C. Berwick, Eric Sven Ristad (1987) *Computational Complexity and Natural Language*. MIT Press, 335 p.

Robert C. Berwick, Amy S. Weinberg (1984) *The grammatical basis of linguistic performance*. MIT Press, 1984, 325 p.

Christian Boitet (1976) *Problèmes actuels en traduction automatique. Un essai de réponse*. Proc. COLING-76, Ottawa, July 1976.

Christian Boitet (1984) *Research and development on MT and related techniques at Grenoble University (GETA)*. in "Machine Translation today: the state of the art", Proc. third Lugano Tutorial, 2–7 April 1984, M. King, ed., Edinburgh University Press, 1987, 133–153.

Christian Boitet (1985) *Traduction (assistée) par Ordinateur: ingéniérie logicielle et linguicielle*. Colloque RF&IA, AFCET, Grenoble.

Christian Boitet (1986a) *The French National MT-Project: technical organization and translation results of CALLIOPE-AERO*. IBM Conf. on Translation Mechanization, Copenhagen.

Christian Boitet (1986b) *Current Machine Translation systems developed with GETA's methodology and software tools*. ASLIB Conf. London.

Christian Boitet (1987a) *Current projects at GETA on or about Machine Translation*. Proc. II World Basque Congress, San Sebastian, 7–11 Sept. 1987, 28p.

Christian Boitet (1987b) *Current state and future outlook of the research at GETA*. Proc. of the Machine Translation Summit, Hakone, 17–19 Sept. 1987, 26–37.

Christian Boitet (1988a) *Software and lingware engineering in modern M(A)T systems*. To appear in "Handbook for Machine Translation", Bátori, ed., Niemeyer, 1988.

Christian Boitet (1988b) *L'apport de Bernard Vauquois à la Traduction Automatique et au Traitement Automatique des Langues Naturelles*. Actes du colloque sur l'histoire de l'informatique en France, Ph. Châtelin, ed., Grenoble, 4–6 mai 1988, volume 2, 63–82.

Christian Boitet (1988c) *Hybrid pivots using m-structures for multilingual transfer-based MT systems*. Japanese Institute of Electronics, Information and Communication, NLC88–3, 17–22.

Christian Boitet (1988d) *Bernard Vauquois' contribution to the theory and practice of building MT systems : a historical perspective*. Proc. of the Second Int. Conf. on theoretical and methodological issues in the machine translation of natural languages, CMU–CMT, ed., Pittsburgh, June 12–14, 1988, 18 p.

Christian Boitet (1988e) *Pros and cons of the pivot and transfer approaches in multilingual Machine Translation*. Proc. of the int. conf. on "New directions in Machine Translation", BSO, Budapest, 18–19 August, 1988.

Christian Boitet, René Gerber (1984) *Expert systems and other new techniques in MT*. Proc. COLING-84, ACL, 468–471, Stanford.

Christian Boitet, Nikolai Nedobejkine (1981) *Recent developments in Russian-French Machine Translation at Grenoble*. Linguistics 19(3/4), 199–271.

Christian Boitet, Nikolai Nedobejkine (1986) *Toward integrated dictionaries for M(a)T: motivations and linguistic organization*. Proc. COLING-86, IKS, 423–428, Bonn.

Christian Boitet, Pierre Guillaume, Maurice Quézel-Ambrunaz (1978) *Manipulation d'arborescences et parallélisme: le système ROBRA*. Proc. COLING-78, Bergen.

Christian Boitet, Pierre Guillaume, Maurice Quézel-Ambrunaz (1982) *ARIANE-78: an integrated environment for automated translation and human revision*. Proc. COLING-82, North-Holland, Ling. series 47, 19–27, Prague.

Christian Boitet, Pierre Guillaume, Maurice Quézel-Ambrunaz (1985) *A case study in software evolution: from ARIANE-78 to ARIANE-85*. Proc. of the Conf. on theoretical and methodological issues in Machine Translation of natural languages, 27–58, Colgate Univ., Hamilton, N.Y.

Christian Boitet, Yusoff Zaharin (1988) *Representation trees and string-tree correspondences*. Proc. COLING-88, Budapest, 22–27 August, 1988.

Gosse Bouma, Esther König, Hans Uszkoreit (1988) *A flexible graph-unification formalism and its application to natural-language processing*. IBM Journal of Research and Development, 32/2, March 1988, 170–184.

John Chandioux, M.F. Guérard (1981) *METEO: un système à l'épreuve du temps*. Meta 26(1), 17–22.

Sylviane Chappuy (1983) *Formalisation de la description des niveaux d'interprétation des langues naturelles*. Thèse, Grenoble.

Jacques Chauché (1975) *Les langages ATEF et CETA*. AJCL, microfiche 17, 21–39.

Alain Colmerauer (1970) *Les systèmes-Q, un formalisme pour analyser et synthétiser des phrases sur ordinateur*. TAUM, Univ. de Montréal.

Fathi Débili (1982) *Analyse syntaxico-sémantique fondée sur une acquisition automatique de relations lexicales-sémantiques*. Thèse d'Etat, Paris XI, 1982.

Simon Dik (1978) *Functional Grammar*. Publications in Language Science, Foris, Dordrecht, Holland, 1978.

Paolo D'Orta, Marco Ferretti, Alex Martelli, Sergio Melecrinis, Stefano Scarci, Giampiero Volpi (1988) *Large-vocabulary speech-recognition : a system for the Italian language.* IBM Journal of Research and Development, 32/2, March 1988, 217–226.

Jean-M. Ducrot (1982) *TITUS IV.* In: *Information research in Europe.* Taylor, P.J., Cronin, P., eds., Proc. of the EURIM 5 conf., Versailles, ASLIB, London.

Lee D. Erman, Victor R. Lesser (1980) *The Hearsay-II Speech Understanding System : A Tutorial.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 361–381.

Roger Garside, Geoffrey Leech, Geoffrey Sampson (1987), eds. *The Computational Analysis of English. A corpus-based approach.* Longman, 1987, 196 p.

G. Gazdar, E.Klein, G.Pullum, I. Sag (1985) *Generalized Phrase Structure Grammar.* Blackwell, Oxford, 1985, 276 p.

René Gerber, Christian Boitet (1985) *On the design of expert systems grafted on MT systems.* Proc. of the Conf. on theoretical and methodological issues in Machine Translation of natural languages, 116-134, Colgate Univ., Hamilton, N.Y.

Jean-Philippe Guilbaud (1984) *Principles and results of a German-French MT system.* in "Machine Translation today: the state of the art", Proc. third Lugano Tutorial, 2–7 April 1984, M. King, ed., Edinburgh University Press, 1987, 278–318.

Jean-Philippe Guilbaud (1986) *Variables et catégories grammaticales dans un modèle ARIANE.* Proc. COLING-86, IKS, 405–407, Bonn.

Takao Gunji (1987) *Japanese Phrase Structure Grammar.* Reidel, Dordrecht, 1987, 239 p.

E.Hajičova, Zdenek Kirschner (1981) *Automatic translation from English into Czech.* Prague Bulletin of Mathematical Linguistics 35, 5–23.

Christa Hauenschild, Edgar Huckert, R.Maier (1979) *SALAT: machine translation via semantic representation.* In: Semantics from different points of view, Bäuerle & al., eds., Springer, Berlin, 324–352.

Frederick Hayes-Roth (1980) *Syntax, Semantics, and Pragmatics in Speech Understanding Systems.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 206–233.

W.J. Hutchins (1986) *Machine Translation : Past, Present, Future.* Ellis Horwood, John Wiley & sons, Chichester, England, 382 p.

Thomas R. Hofmann, Tarou Kageyama (1986) *10 Voyages in the Realm of Meaning.* Kuroshio series, Hokushin, Toyama, 167 p.

Hitoshi Iida (1987) *Distinctive features of conversations and inter-keyboard interpretation.* in: ATR NLDI workshop (1987), 11–15.

Hitoshi Iida, Masako Kume, Izuru Nogaito, Teruaki Aizawa (1987) *Collection of Interpreted Telephone Conversation Data.* ATR report TR-I-0007, October 1987, 53 p.

Pierre Isabelle (1988) *Reversible Logic Grammars for MT.* Proc. of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Pittsburgh, 1988.

Pierre Isabelle, Laurent Bourbeau (1984) *TAUM-AVIATION: its technical features and some experimental results.* Computational Linguistics, 11:1, 18–27.

Robert T. Kasper (1987) *A Unification Method for Disjunctive Feature Descriptions.* Proceedings of the 25th Annual Meeting of the ACL, 1987, 235–242.

Martin Kay (1981) *Unification Grammars.* Xerox publication, 1981.

Margaret King (1987), ed. *Machine Translation Today.* Proc. of the third Lugano Tutorial, Lugano, 2–7 April 1984, Edinburgh University Press, 1986, 447 p.

Richard Kittredge, A. Polguere, L. Iordanskaya (1988) *Multilingual Generation and Meaning–Text Theory.* Proc. of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Pittsburgh, 1988.

Dennis H. Klatt (1980) *Overview of the ARPA Speech Understanding Project.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 249–271.

Kiyoshi Kogure, Hitoshi Iida, Kei Yoshimoto, Hiroyuki Maeda, Masako Kume, Susumu Kato (1988) *A method of analyzing Japanese Speech Act types.* Proc. of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Pittsburgh, 1988.

Kiyoshi Kogure, Hirosato Nomura (1988) *Computer environment for meaning structure representation and manipulation in Machine Translation system.* in: ATR NLU & KDB report (1988), 121–132.

Ikuo Kudo, Hideya Koshino, Moonkyung Chung, Tsuyoshi Morimoto (1988) *Schema Method: a Framework for Correcting Grammatically Ill-Formed Input.* Proc. COLING-88, Budapest, August 22–27, 1988.

Akira Kurematsu (1987a) *Automatic Telephone Interpretation: a basic study.* ATR report TR-I-0001, May 1987, 22 p.

Akira Kurematsu (1987b) *Problems involved in telephone interpretation.* in: ATR NLDI workshop (1987), 4–10.

Veronica Lawson (1985), ed. *Tools for the trade, Translating and the computer 5.* Proc. ASLIB-81 Conf., London, 10–11 Nov. 1983.

Wayne A. Lea (1980a) *Speech Recognition : Past, Present, and Future.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 39–98.
Wayne A. Lea (1980b) *Prosodic Aids to Speech Recognition.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 166–205.
Wayne A. Lea (1980c), ed. *Trends in Speech Recognition.* Prentice-Hall, 1980, 580 p.
Kai-Fu Lee (1988) *Large-Vocabulary Speaker-Independent Continuous Speech Recognition : the SPHINX System.* Ph. D. Thesis, CMU-CS-88-148, Pittsburgh, 1988, 183 p.
Bruce Lowerre, Raj Reddy (1980) *The Harpy Speech Understanding System.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 340–360.
H.D. Luckhard (1982) *SUSY: capabilities and range of applications.* Multilingua 1(4), 213–219.
Heinz-Dieter Maas (1981) *SUSY I und SUSY II: verschiedene Analysestrategien in der Maschinellen Übersetzung.* Sprache und Datenverarbeitung 5(1/2), 9–15.
Thomas B. Martin, John R. Welch (1980) *Practical Speech Recognizers and Some Performance Effectiveness Parameters.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 24–38.
Hiroyuki Maeda, Susumu Kato, Kiyoshi Kogure, Hitoshi Iida (1988) *Parsing Japanese Honorifics in Unification-Based Grammar.* Proc. COLING-88, Budapest, August 22–27, 1988.
James L. McClelland, David E. Rumelhart (1986) *Parallel Distributed Processing. Explorations in the Microstructures of Cognition. Volume 2: Psychological and Biological Models.* MIT Press, 1986, 611 p.
Michael McCord (1985) *LMT: a Prolog-based Machine Translation system.* Proc. of the Conf. on theoretical and methodological issues in Machine Translation of natural languages, 179–182, Colgate Univ., Hamilton, N.Y.
Eric McDermott, Shigeru Katagiri (1988) *Phoneme Recognition Using Kohonen networks.* Proc. of the ATR Workshop on Neural Networks and Parallel Distributed Processing, Osaka, July 7–8, 1988.
Igor Mel'čuk (1981) *Meaning-Text models: a recent trend in Soviet linguistics.* Annual Review of Anthropology 10, 27–62.
Bernard Mérialdo (1988) *Multilevel decoding for Very-Large-Size-Dictionary speech recognition.* IBM Journal of Research and Development, 32/2, March 1988, 227–237.
Junichi Nakamura, Junichi Tsujii, Makato Nagao (1984) *Grammar writing system (GRADE) of Mu-Machine Translation Project and its characteristics.* Proc. COLING-84, ACL, Stanford.
Sergei Nirenburg, Rita McCardell, Eric Nyberg, Scott Huffman, Edward Kenschaft, Irene Nirenburg (1988) *Lexical Realization in Natural Language Generation.* Proc. of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Pittsburgh, 1988.
Izuru Nogaito, Hitoshi Iida (1988) *Noun Phrase Identification in Dialogue and its Application.* Proc. of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Pittsburgh, 1988.
Hirosato Nomura, Shozo Naito, Yasuhiro Katagiri, Akira Shimazu (1986) *Translation by understanding: a Machine Translation system LUTE.* Proc. COLING-86, IKS, 621–626, Bonn.
Miyo Otani, Nathalie Simonin (1988) *Functional Descriptions as a Formalism for Linguistic Knowledge Representation in a Generation Oriented Approach.* Proc. of the Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages, Pittsburgh, 1988.
Fernando C. N. Pereira (1985) *A Structure-Sharing Representation for Unification-Based Grammar formalisms.* Proceedings of the 23rd Annual Meeting of the ACL, 1985, 137–144.
Raymond C. Perrault (1987) *Speech act theory and extended discourse.* in: ATR NLDI workshop (1987), 26–29 (summarized by H. Maeda).
Patrice Quinton (1980) *Contribution à la reconnaissance de la parole. Utilisation de méthodes heuristiques pour la reconnaissance de phrases.* Thèse d'Etat, Université de Rennes, 1980, 239 p.
L. R. Rabiner, B. H. Juang (1986) *An Introduction to Hidden Markov Models.* IEEE ASSP magazine, January 1986, 4–16.
Andrew Radford (1988) *Transformational Grammar. A First Course.* Cambridge University Press, 1988, 625 p.
Raj Reddy (1980) *Machine Models of Speech Perception.* in "Perception and Production of Fluent Speech", Cole, ed., Erlbaum, N.J., 1980, 215–242.
Hiroaki Saito, Masaru Tomita (1988) *Parsing Noisy Sentences.* Proceedings of COLING-88, Budapest, August 22–27.
Yoshiyuki Sakamoto, Masayuki Satoh, Tetsuya Ishikawa (1984) *Lexicon features for Japanese syntactic analysis in MU-project-JE.* Proc. COLING-84, ACL, Stanford.
Stuart M. Schieber (1986) *An introduction to unification-based approaches to grammar.* CSLI Lecture Notes 4, CSLI, Stanford, 105 p.
Peter Sells (1985) *Lectures on contemporary syntactic theories : an introduction to Government-Binding theory, Generalized Phrase Structure Grammar, and Lexical Functional Grammar.* CSLI Lecture Notes 3, Stanford, 1985, 214 p.

Jonathan Slocum (1984) *METAL: the LRC Machine Translation system.* in "Machine Translation today : the state of the art", Proc. third Lugano Tutorial, 2–7 April 1984, M. King, ed., Edinburgh University Press, 1987, 319–350.

Stanley Starosta (1988) *The Case for Lexicase.* Pinter, London, 1988, 273 p.

Gilles Stewart (1975) *Manuel du langage REZO.* TAUM, Univ. de Montréal.

Masaru Tomita (1986) *Efficient Parsing for Natural Language. A fast algorithm for practical systems.* Kluwer, 201 p.

Masaru Tomita, Jaime G.Carbonell (1986) *Another stride towards knowledge-based Machine Translation.* Proc. COLING-86, IKS, 633-638, Bonn.

Masaru Tomita, Teruko Mitamura, Hiroyuki Musha, Marion Kee (1988) *The Generalized LR Parser/Compiler. Version 8.1: User's Guide.* CMT, CMU, memo, January 1988.

Loong Cheong Tong (1986) *English-Malay translation system: a laboratory prototype.* Proc. COLING-86, IKS, 639-642, Bonn.

David S. Touretzky (1986) *Representing and Transforming Recursive Objects in a Neural Network, or "Trees Do Grow on Boltzmann Machines.* Proc. IEEE Int. Conference on Systems, Man, and Cybernetics, Atlanta, 1986, 5 p.

David S. Touretzky (1987) *Representing Conceptual structures in a Neural Network.* Proc. IEEE First Annual Int. Conference on Neural Networks, San Diego, 1987, 8 p.

Jun-Ichi Tsujii (1987) *Dialogue Translation vs Text Translation: interpretation-based approach.* in: ATR NLDI workshop (1987), 16–21 (summarized by M. Kume).

Jun-Ichi Tsujii, Yukiyoshi Muto, Yuuji Ikeda, Makoto Nagao (1988) *How to Get Preferred Readings in Natural Language Analysis.* Proc. COLING-88, Budapest, August 22–27, 1988.

Jun-Ichi Tsujii, Makoto Nagao (1988) *Dialogue Translation vs Text Translation: Interpretation Based Approach.* Proc. COLING-88, Budapest, August 22–27, 1988.

Bernard Vauquois (1975) *La traduction automatique à Grenoble.* Document de linguistique quantitative n° **29**, Dunod, Paris.

Bernard Vauquois (1979) *Aspects of automatic translation in 1979.* IBM-Japan, scientific program.

Bernard Vauquois (1983) *Automatic translation.* Proc. of the summer school "The computer and the arabic language", ch. 9, Rabat.

Bernard Vauquois, Sylviane Chappuy (1985) *Static grammars: a formalism for the description of linguistic models.* Proc. of the Conf. on theoretical and methodological issues in Machine Translation of natural languages 298–322, Colgate Univ., Hamilton, N.Y.

Bernard Vauquois, Christian Boitet (1985) *Automated translation at GETA (Grenoble University).* Computational Linguistics, **11**:1, 28–36.

Gérard Veillon (1970) *Modèles et algorithmes pour la traduction automatique.* Thèse d'Etat, Grenoble.

Nelson Verastegui (1982) *Utilisation du parallélisme en traduction automatisée par ordinateur.* Proc. COLING-82, North-Holland, Ling. series **47**, 397–405, Prague.

Alex Waibel (1988) *Modular Construction of Phonetic Neural Networks.* Proc. of the ATR Workshop on Neural Networks and Parallel Distributed Processing, Osaka, July 7–8, 1988.

Alex Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang (1987) *Phoneme recognition using time-delay neural networks.* ATR report TR-I-0006, October 1987, 37 p.

Donald E. Walker (1980) *SRI Research on Speech Understanding.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 294–315.

David L. Waltz, Jordan B. Pollack (1985) *Massively Parallel Parsing: a Strongly Interactive Model of Natural Language Interpretation.* Cognitive Science **9**, 51–74 (1985).

Yorick Wilks (1975) *Preference Semantics.* Formal semantics of Natural Language, Keenan, ed., Cambridge, 329–348.

Jared J. Wolf, Williams A. Woods (1980) *The HWIM Speech Understanding System.* in "Trends in Speech Recognition", W.A. Lea, ed., Prentice-Hall, 1980, 316–339.

William A. Woods (1970) *Transition network grammars for natural language analysis.* CACM **13**/10, 591–606.

David A. Wroblewski (1987) *Nondestructive Graph Unification.* Proceedings of the Sixth National Conference on AI, 582–587.

Yusoff Zaharin (1986) *Strategies and heuristics in the analysis of a natural language in Machine Translation.* Proc. COLING-86, IKS, 136–139, Bonn.

Rémi Zajac (1986) *SCSL: a linguistic specification language for MT.* Proc. COLING-86, IKS, 393–398, Bonn.

Rémi Zajac (1988) *Interactive Translation : a new approach.* Proc. COLING-88, Budapest, 22–27 August, 1988.

-o-o-o-o-o-o-o-o-o-