Post-COLING-ACL '98 Workshop on Content Visualization and Intermedia Representations. Montreal, Quebéc.

Visualization of Protocols of the Parsing and Semantic Interpretation Steps in a Machine Translation System

Ulrich Germann
USC Information Sciences Institute
Marina del Rey, CA 90292
email: germann@isi.edu

Abstract

In this paper, we describe a tool for the visualization of process protocols produced by the parsing and semantic interpretation modules in a complex machine translation system. These protocols tend to reach considerable sizes, and error tracking in them is tedious and time-consuming. We show how the data in the protocols can be made more easily accessible by extracting a procedural trace, by splitting the protocols into a collection of cross-linked hypertext files, by indexing the files, and by using simple text formatting and sorting of structural elements.

1 Introduction

The tool described in this paper was developed in connection with the *Gazelle* Machine Translation System (Knight et al., 1995), which is currently under development at the USC Information Sciences Institute. At the moment, *Gazelle* covers machine translation from Japanese and Arabic to English.

Figure 1 sketches the flow of processing. The input text is first segmented and tagged with morphological information. It is then parsed and interpreted semantically. The result of semantic interpretation is finally fed into the text generation module.

Almost all modules relevant to the discussion here employ bottom up chart parsing mechanisms. For any given input, they may return more than one interpretation, as the sample parse sequence for the string *saw the ape with his binoculars* in Figure 2 illustrates.

The processes of parsing and semantic interpretation are recorded step by step in process protocols. A parse step in our system is equivalent to the creation of a new parse node. Each node receives a category label which determines its behavior in the remainder of the parsing process. A new parse node can be created in order to

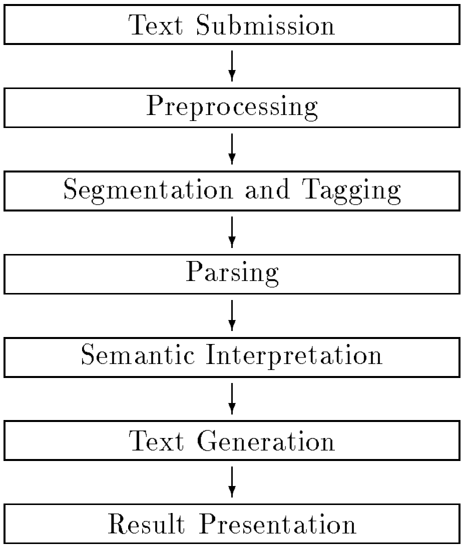
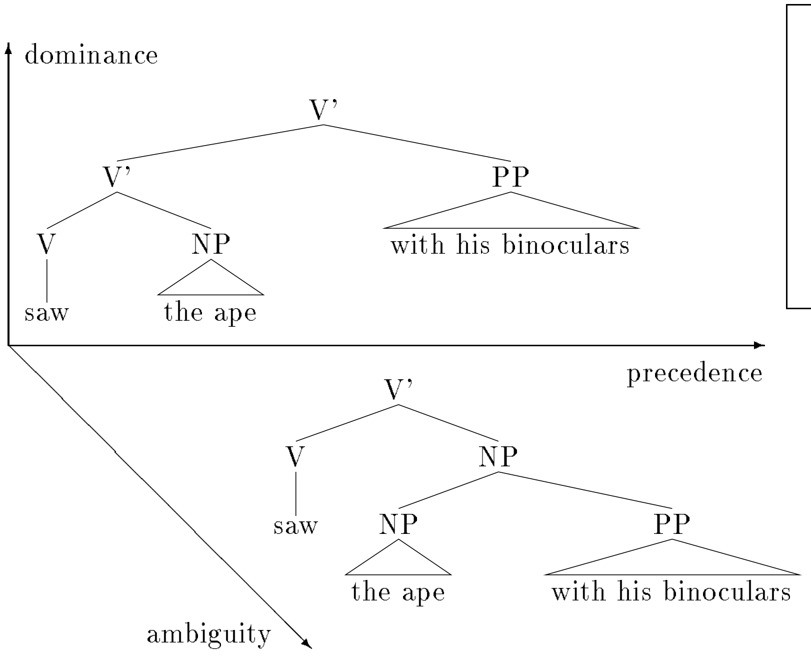


Figure 1: The machine translation process

- integrate a new text element into the parse space,
- combine two or more exisisting nodes into one node, or
- change the category of an existing node by making it the daughter of the new node, and assigning a different category label to the new node.

If, for any given range of input text, there are two interpretations that result in the same category label, they are combined into one node. For example, parse step No. 6 in Figure 2 creates a node that comprises two interpretations.



Parse Sequence (simplified)

Step 1:	V → *saw*
Step 2:	NP → *the ape*
Step 3:	V' → V NP
Step 4:	PP → *with his binoculars*
Step 5:	NP → NP PP
Step 6:	V' → V NP
	OR:
	V' → V' PP

Figure 2: The parse space for the string *saw the ape with his binoculars.* (simplified)

Each node in the parse tree also has a feature structure associated with it. This feature structure specifies various properties of the constituent represented by the respective node. A feature structure is a set of attribute-value pairs, where a value can either be atomic (something that cannot be described in terms of having particular features or properties), or an embedded feature structure. Typical forms of representation are attribute-value matrices (AVMs), and graphs with labeled arcs and nodes. Figure 3 shows a partial description of the word *sees* (3rd person singular form of the verb *see* with the meaning |*see*|) in both formats.

During semantic interpretation, the feature structures introduced by the parser are augmented by semantic features assigned to the nodes by semantic interpretation rules (Knight and Hatzivassiloglou, 1995).

In the protocols, each node receives an individual entry which keeps track of the following information:

- the node number, which uniquely identifies the node / parse step;
- the category label assigned to the node, e.g. *NP* for 'noun phrase';
- the nodes dominated by the node, or, if the node is a terminal node, the corresponding lexical symbol (a word or morpheme) from the input text. Daughter nodes of nonterminal nodes are referenced by their node numbers. Since ambiguities are packed into one node (instead of spelling them out by creating multiple nodes), a nonterminal node may have several sets of daughters (cf. node No. 22 in Figure 4).

$$\begin{bmatrix} \text{SURFACE} & \textit{sees} \\ \text{SYN} & \begin{bmatrix} \text{POS} & \textit{verb} \\ \text{PERSON} & \textit{3rd} \\ \text{NUMBER} & \textit{sg} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INST} & |\textit{see}| \end{bmatrix} \end{bmatrix}$$

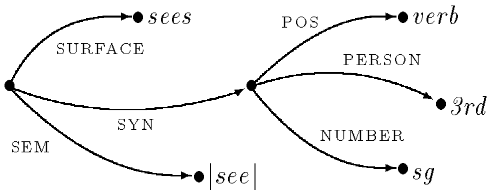
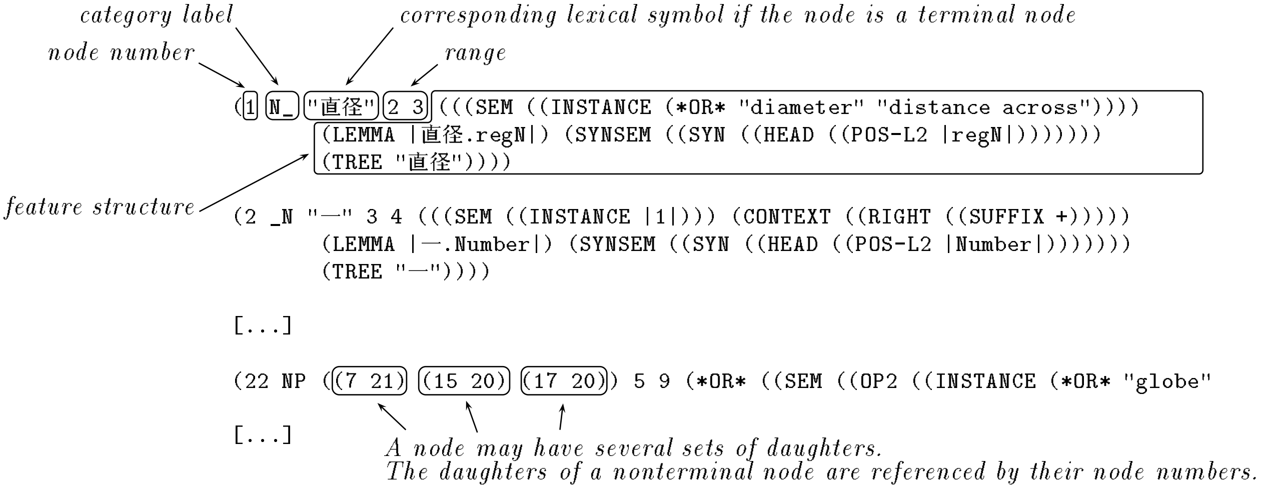


Figure 3: Partial description of the word *sees* as an attribute-value matrix (top) and as a graph with labeled arcs and nodes (bottom)



```
(1 N_ "直径" 2 3 (((SEM ((INSTANCE (*OR* "diameter" "distance across"))))
     (LEMMA |直径.regN|) (SYNSEM ((SYN ((HEAD ((POS-L2 |regN|)))))))
     (TREE "直径"))))

(2 _N "一" 3 4 (((SEM ((INSTANCE |1|))) (CONTEXT ((RIGHT ((SUFFIX +)))))
     (LEMMA |一.Number|) (SYNSEM ((SYN ((HEAD ((POS-L2 |Number|)))))))
     (TREE "一"))))

[...]

(22 NP ((7 21) (15 20) (17 20)) 5 9 (*OR* ((SEM ((OP2 ((INSTANCE (*OR* "globe"

[...]
```

category label
corresponding lexical symbol if the node is a terminal node
node number
range
feature structure
A node may have several sets of daughters.
The daughters of a nonterminal node are referenced by their node numbers.

Figure 4: Sample sections from a process protocol. The samples were slighty modified and reformatted in order to achieve a more compact and readable display.

- the range of input text covered by the node;
- the feature structure(s) associated with the node.

Figures 4 and 5 show sample sections from a process protocol in raw format and as displayed by the visualization tool.

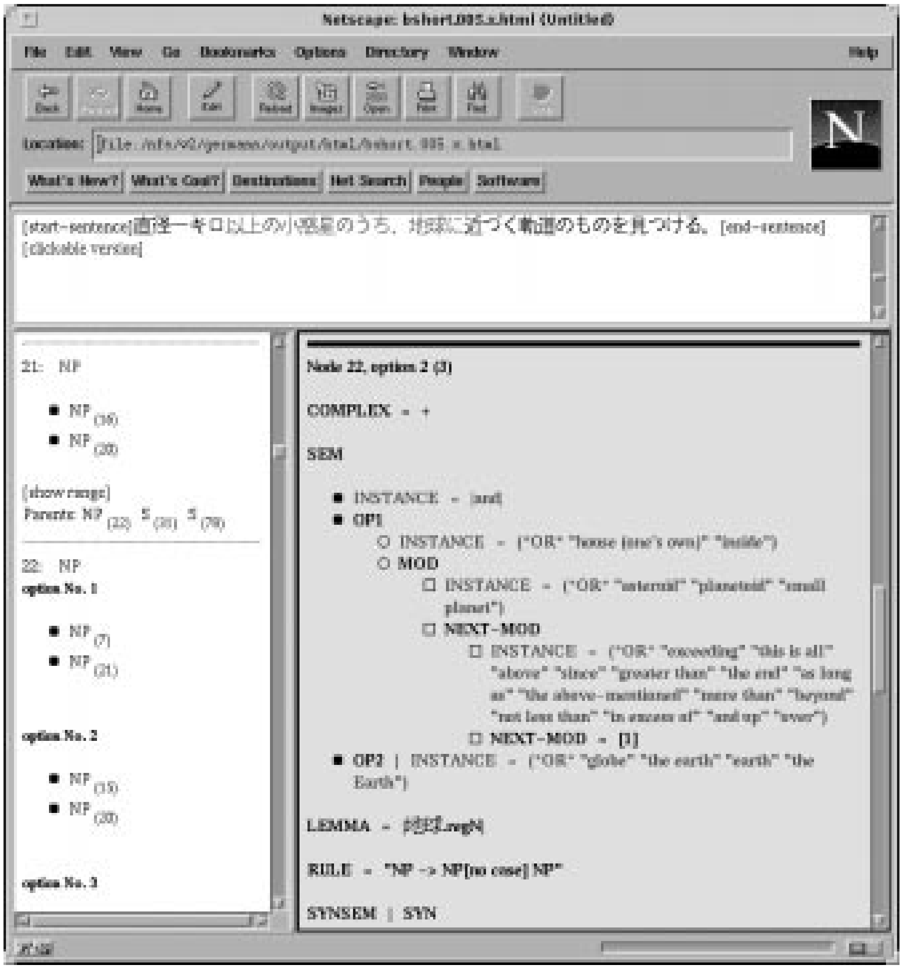
2 Analysis of the Data

Apart from the obvious, explicit information expressed in the protocol entry of each node — the node number, the immediate daughters of the node, and the features associated with it — the process protocol also contains some implicit information which can be retrieved by going through the protocol as a whole. For example, it is possible to check whether a node is used later on in the parsing process and what its mother node(s) is. Going down the tree, it can be determined which part of the input sentence is covered by a particular node. The 'parse forest' (collection of parse trees) can be reconstructed by linking the individual branchings (minimal trees consisting of a mother node and its daughter node(s)) by virtue of the node numbers. Last but not least, the original input sentence can be recovered by concatenating the leaf nodes of the tree.

Individual parse trees are two-dimensional, the two dimensions being *precedence* (the linear order in which the elements occur in the sentence) and *dominance* (the relation between a mother node and its daughter node(s)). Even though both relations are preserved in the process protocols, none of them is immediately obvious from the protocols: Nodes that are neighbors in terms of linear order, or nodes that are connected in terms of dominance do not necessarily appear next to each other in the parse sequence. Consider, for example, the parse sequence in Figure 2. The node created by step 1 is used again in the steps 3 and 6. Whereas for step 3 the sister nodes 1 and 2 happen to be adjacent, this is not true for step 6. Here, the two nodes that make up the first interpretation are separated by the nodes 2, 3 and 4. Equally, the dominance relation between parse nodes is usually obscured by intermediate parse steps. If we also take into account the issue of ambiguity, we have to deal with it as a third dimension, as illustrated in Figure 2.

Keeping in mind the purpose of examining the data, which is first and above all error tracing and quality control, we can thus distinguish four aspects of the information contained in the process protocols. These aspects are:

- the overall structure, i.e., information about the constituent structure (parse forest) of the sentence, and about the way it is built up from lexical items;
- the details, i.e., information about the


Netscape: bshort.005.x.html (Untitled)
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: file:/nfs/u2/germann/output/html/bshort.005.x.html
What's New? What's Cool? Destinations Net Search People Software

[start-sentence]直径一キロ以上の小惑星のうち、地球に近づく軌道のものを見つける。[end-sentence]
[clickable version]

21: NP

- NP (38)
- NP (20)

[show range]
Parents: NP (22) S (31) S (78)

22: NP
option No. 1

- NP (7)
- NP (21)

option No. 2

- NP (15)
- NP (20)

option No. 3

Node 22, option 2 (3)

COMPLEX = +

SEM

```
■ INSTANCE = |and|
■ OP1
    ○ INSTANCE = ("OR" "house (one's own)" "inside")
    ○ MOD
        □ INSTANCE = ("OR" "asteroid" "planetoid" "small
          planet")
        □ NEXT-MOD
            □ INSTANCE = ("OR" "exceeding" "this is all"
              "above" "since" "greater than" "the end" "as long
              as" "the above-mentioned" "more than" "beyond"
              "not less than" "in excess of" "and up" "over")
            □ NEXT-MOD = [3]
■ OP2 | INSTANCE = ("OR" "globe" "the earth" "earth" "the
  Earth")
```

LEMMA = 地球.regN

RULE = "NP -> NP[no case] NP"

SYNSEM | SYN

Figure 5: Sample screen. The top frame displays the original input text, which serves as an index to the parse sequence in the lower left frame. The category labels and node numbers in the lower left frame are hyperlinks. A click on a category label will cause the browser to display the feature structure(s) associated with that node in the frame on the right. A click on a node number will lead to the respective node within the parse sequence.

properties of the constitutents as preserved in the feature structures associated with each node;

- ambiguity, i.e., the number of interpretations associated with each node;
- the relation between the parse sequence and the input, i.e., information about the connection of each node in the parse sequence to the part of the sentence that is covered by this node, and, in reverse, the connection of each word or morpheme of the input to the node that integrates this word or morpheme into the parse sequence.

Whereas the connection of each node to the input can be considered a property of the respective node, the connection of the input to the parse sequence is a matter of indexing that will ease access to 'trouble spots' when testing changes to the syntactic grammar or the semantic interpretation rules.

3 Displaying the Data

3.1 Display of the Constituent Structure

The obvious solution to the issue of visualizing the overall constitutent structure of a sentence, namely displaying it as a conventional parse tree representation, turns out to be not very practical at closer examination. Since some parsers employed in our system may return more than just one structure for any given input, and since partial parses are very relevant for debugging, we are, in fact, not dealing with a single parse tree but rather with a 'parse forest' which may contain dozens or even hundreds of complete and partial trees. A development tool that confronts the developer with a large number of spelled-out ambiguities is not satisfactory, because it does not allow efficient access to relevant areas in the data space. Moreover, even in cases where there is only one single parse tree, the tree may not fit on the screen, and even if it does, there will be little room left to display the feature structures associated with each node.

Therefore, instead of using conventional tree representations, the parse sequence is maintained as such and displayed as a cross-linked sequence of minimal trees, essentially in the same way it is recorded in the process protocol. However, it is formatted in a manner that supports the correct interpretation of the symbols as mother and daughter(s) (cf. the lower left frame in Figure 5). In addition, the explicit information contained in the individual entries of the process protocols is augmented by information that can be inferred from the process protocol as a whole and will not be visible when just looking at the protocol. Note, for example, that category labels are given not only for the mother node but also for the daughter nodes, and that the mother-daughter relation contained in the individual protocol entry is back-referenced to the daughter nodes in the HTML representation, so that the subsequent use of the node can be traced easily. In contrast, detailed information about the properties of the individual constituents, which tends to obstruct the view of the overall structure, is removed from this trace of display and stored in a separate file. Nevertheless it is readily at hand by means of hyperlinks, as will be explained immediately below.

What is not visible in Figure 5 is that all category labels, and the node numbers in subscript after the category labels of daughter and parent nodes, are hyperlinks. A mouse click on a category label causes the browser to display the feature structure associated with that node in the frame on the right. A click on the node number leads to the respective node within the same (left) frame. This setup provides the user with the means to easily navigate the parse tree or parse forest, and, at the same time, allows him or her to inspect the features associated with each node in detail. In both frames, ambiguities (multiple structures associated with one node) are preserved by marking the alternatives with clear headings. Since parse sequence and feature structures are separated and displayed in different, adjoining frames in the browser, a very compact yet informative form of data visualization is achieved.

3.2 Visualization of the Relation between Constitutent Structure and Input Text

Before we turn to the representation of feature structures, let us briefly comment on the top frame in Figure 5. This frame displays the text of the original input sentence and serves two purposes. First, a click on the caption [**show range**] in either one of the lower frames[1] causes the browser to display the range covered by that node in red (as opposed to black for the surrounding context), and, secondly, in the 'clickable' mode of the top frame the input sentence serves as an index: a click on a word leads to the respective node in the parse sequence in the lower left frame, so that particular points of interest can be accessed easily during the grammar development process.

3.3 Representation of Feature Structures

Our approach to the display of feature structures also diverges from representation formats that are well established in the literature. Commonly used representations — representations as graphs with labeled arcs and as attribute-value matrices (cf. Figure 3) — tend to become too large once the feature structures reach a certain complexity. Moreover, both of them require extensive calculations and graphical processing, which have to be paid for in terms of processing time. Instead, our tool represents feature structures as unordered lists in HTML. Each list element represents an attribute-value pair. Atomic values are separated from their attributes by an equal sign (=); complex values

[1] The caption is not visible in the lower right frame in Figure 5. See Figure 9 for the location of the caption in the lower right frame.

$$\begin{bmatrix} \text{SEM} & \begin{bmatrix} \text{INST} & |prize\langle gift| \vee |booty| \\ & \vee\ |trophy, prize| \\ \text{MOD} & \begin{bmatrix}\text{INST} & \text{“best dresser”}\end{bmatrix} \\ \text{T.-LOC.} & \begin{bmatrix} \text{INST} & |time\ unit| \\ \text{YEAR} & [\text{INST } |1992|] \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 6: AVM representation of the semantic content of the phrase *the "Best Dresser" Award 1992.*

(embedded feature structures) are represented as embedded lists.

- SEM
  - INST = $(*\text{or}*\ |prize\langle gift|\ |booty|\ |trophy, prize|)$
  - MOD
    - INST = "best dresser"
  - T.-LOC.
    - INST = $|time\ unit|$
    - YEAR
      - INST = |1992|

Figure 7: The semantic content of the phrase *the "Best Dresser" Award 1992*, represented as an unordered list.

- SEM
  - INST = $(*\text{or}*\ |prize\langle gift|\ |booty|\ |trophy, prize|)$
  - MOD | INST = "best dresser"
  - T.-LOC.
    - INST = $|time\ unit|$
    - YEAR | INST = |1992|

Figure 8: The semantic content of the phrase *the "Best Dresser" Award 1992*, represented as an unordered list with path abbreviations.

In this representation format, the feature structure represented by the AVM in Figure 6 would be rendered as shown in Figure 7.

In order to achieve a more compact representation, we use a convention that is commonplace in the HPSG literature (Pollard and Sag, 1994, and others), namely, we abbreviate single paths by the use of a vertical bar (|). Thus the representation in Figure 8 is equivalent to the one shown in Figure 7.

3.4 Sorting and Use of Colors

In the process protocols, features usually appear in random order. In order to ease the locating of features and the comparison of feature structures, features are generally listed in alphabetical order in the display. However, since not all information contained in the feature structures is equally important, some features receive special treatment.

The main purpose of the parsing and semantic interpretation steps within the translation process is to build up a more language-independent representation of the semantic content of the input text. This information is stored in the SEM feature of the feature structure associated with each node. Within these substructures, the INSTANCE feature plays a central role: it specifies the concept from the *Sensus* ontology[2] (Knight and Luk, 1994; Hovy and Knight, 1993) that the object referred to by the expression in the source language is an instance of. For example the Japanese expression

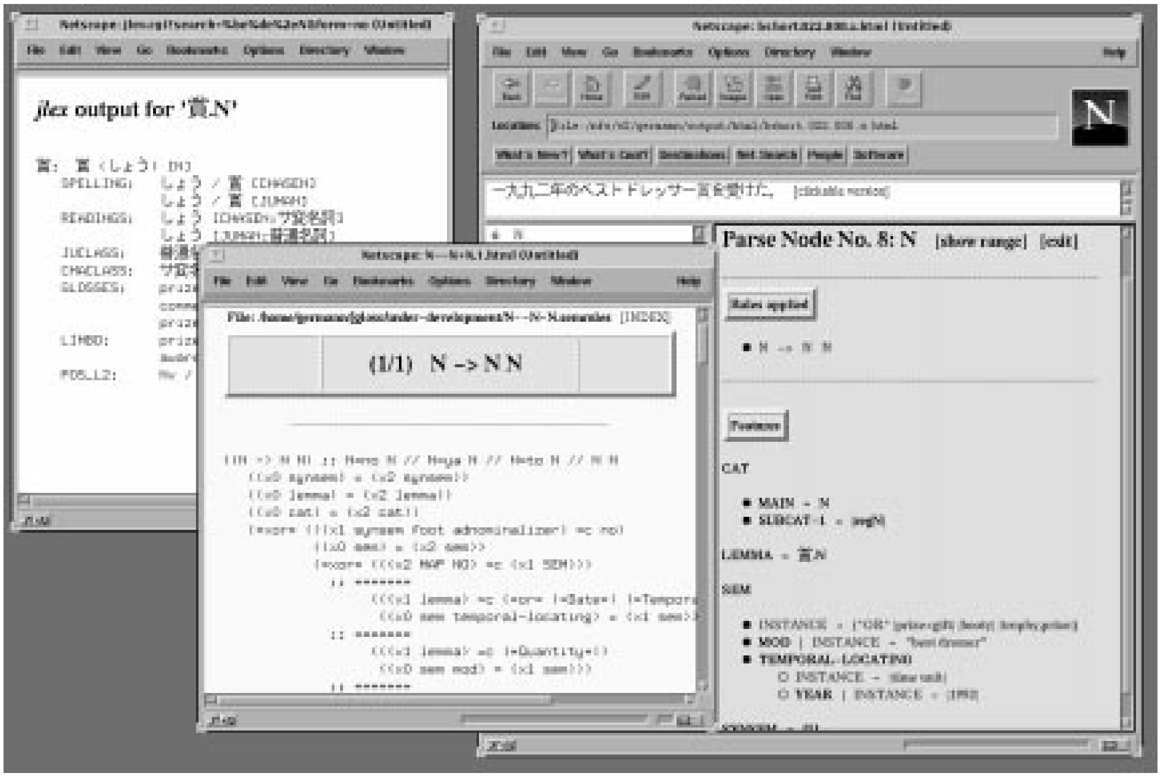
一九九二	年の	ベストドレッサー	賞
1992	year	"best dresser"	award

refers to an instance of a prize or award (賞) which is modified[3] by the string ベストドレッサー ("best dresser") and temporally located in the year 1992: the *"Best Dresser" Award 1992.*

Other features on the same 'level' as the INSTANCE feature specify arguments (in the case of predicates and relations) and further modifications of the object in question. In order to accomodate the central role of the INSTANCE feature, it is always pushed to the top of the list of features under its governing attribute in the feature structure, and displayed, together with its value, in red instead of black. This increases its visibility in the display of the feature structures, so that it can be spotted easily. The lower right frame of the main window in

[2] If an appropriate concept cannot be found in the ontology, an English gloss is used instead.

[3] This representation does not specify the nature of the modification. We currently do not have the means to automatically determine the specific character of the modification.



jlex output for '賞.N'

```
賞:  賞 (しょう) [N]
   SPELLING:    しょう / 賞 [CHASEN]
                しょう / 賞 [JUMAN]
   READINGS:    しょう [CHASEN:サ変名詞]
                しょう [JUMAN:普通名詞]
   JUCLASS:
   CHACLASS:
   GLOSSES:
   LIMBO:
   POS_L2:
```

(1/1) N -> N N

```
((N -> N N) :: N=no N // N=ya N // N=to N // N N
   ((x0 synsem) = (x2 synsem))
   ((x0 lemma) = (x2 lemma))
   ((x0 cat) = (x2 cat))
   (*or* (((x1 synsem foot adnominalizer) =c no)
          ((x0 sem) = (x2 sem)))
         (*or* (((x2 MAP HD) =c (x1 SEM)))
            :: =======
               (((x1 lemma) =c (*or* |*Date*| |*Tempora
                ((x0 sem temporal-locating) = (x1 sem))
            :: =======
               (((x1 lemma) =c |*Quantity*|)
                ((x0 sem mod) = (x1 sem)))
            :: =======
```

一九九二年のベストドレッサー賞を受けた。 [clickable version]

Parse Node No. 8: N [show range] [edit]

Rules applied

- N -> N N

Features

CAT

- MAIN = N
- SUBCAT-1 = [argN]

LEMMA = 賞.N

SEM

- INSTANCE = ("OR" prize-gift booty trophy,prize)
- MOD | INSTANCE = "best dresser"
- TEMPORAL-LOCATING
  - ○ INSTANCE = time unit
  - ○ YEAR | INSTANCE = 1992

Figure 9: Integration of additional information. From left to right: Dictionary output, grammar, and main window.

Figure 9 shows the feature structure discussed here.

Also, in order to help the developer distinguish the different frames and windows (see below) of the visualization tool, we use different background colors for different frames. For example, the background of the lower right frame of the main window is kept in a light blue, whereas other windows have yellow or white backgrounds.

4 Integration of Additional Resources

Whereas the data discussed so far is almost always of interest for the developer, other types of information are requested less frequently. For example, one may want to check a word's entry in the dictionary, or see the grammar rule that was used to create a particular node. Dedicating a frame in the main window to the display of this kind of data did not seem an appropriate solution. Display space is far too precious to be wasted on a frame that is used only occasionally. Instead, this kind of information is provided in separate browser windows. The grammar rules contain unique IDs that are preserved in the feature structures associated with each node. The converter uses these IDs to set links to an HTML version of the grammar. The links are listed under 'Rules applied' in the lower right frame (cf. Figure 9). For terminal nodes, a click on the lexical symbol in either of the lower frames will access the cgi interface of the dictionary and deliver the complete lexical information that is available for the respective entry. Figure 9 shows a sample of the developer's screen with the main and the two subsidiary windows.

5 Evaluation

The prime criterion for the evaluation of a visualization tool like the one described is its effect on the efficiency of working with the data. Since the tool was mainly developed for in-house use by a small team of developers, we did not carry out formal experiments in order to assess the increase in productivity. Based on my personal experience in working with both the simple process protocols and the visualized version as a grammar developer for the Japanese modules, I estimate the speedup at a factor of two to at least ten, depending on the size of the structure to be analyzed: the larger the structure, the greater the benefit of visualization.

Also, the tool has proven to be valuable for demonstration purposes, as the structuring provided by this kind of display helps people who have no previous experience with our system and maybe are even unfamiliar with natural language processing in general understand the process in more detail. In particular, since many people are now familiar with the internet and the interfaces provided by web browsers, our impression is that people who see our system for the first time tend to be able to focus on the *data* rather than the interface more quickly. Again, this claim has not been tested experimentally.

6 Technical Information

The converter was implemented in Perl and currently works in off-line mode only, that is, the HTML files have to be created first and can then be used. Processing time depends very much on the size of the structures as well as other factors beyond the immediate control of the author such as network traffic and overall processing load on the machine. On a Sun Ultra, processing typically takes between a few seconds for short sentences (converting the protocol for the sentence in Figure 9 takes about 3 seconds) and several minutes for very large and ambiguous structures.

7 Conclusion

In this paper, we have presented a tool for the visualization of a complex parse space by separating global information about the overall structure of the parse space from detailed, local information while preserving the relation between both by means of hyperlinks and indexing. Our solution is anything but fancy: it is completely text-based and employs commonplace, off-the-shelf tools for the display of hypertexts. This allows for ease of use and a high degree of portability, because no new software has to be installed at the user-end. Even though some compromises had to be made, for example by deviating from well-established forms of representation, the interface nevertheless provides appropriate functionality for the given purpose. In a word, HTML is used for what it was invented for — the encoding of nonlinear information.

8 Acknowledgements

Gazelle is funded by the US Government under contract MDA904-96-C-1077.

The use of unordered lists in HTML for the representation of feature structures was inspired by a similar approach taken in a web interface for the *Gazelle* system developed by Philipp Köhn.

I am very grateful to Kevin Knight, Ulf Hermjakob and Daniel Marcu for various comments on earlier drafts of this paper.

References

Eduard Hovy and Kevin Knight. 1993. Motivating shared knowledge resources: An example from the Pangloss collaboration. In *Proceedings of the Workshop on Knowledge Sharing and Information Interchange (IJCAI).*

Kevin Knight and Vasileios Hatzivassiloglou. 1995. Unification-based glossing. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence.*

Kevin Knight and Steve K. Luk. 1994. Building a large-scale knowledge base for machine translation. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI).*

Kevin Knight, Ishwar Chander, Matthew Haines, Vasileios Hatzivassiloglou, Eduard Hovy, Masayo Iida, Steve K. Luk, Richard Whitney, and Kenji Yamada. 1995. Filling knowledge gaps in a broad-coverage machine translation system. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence.*

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar.* Studies in Contemporary Linguistics. University of Chicago Press, Chicago.