MASTER IN COMPUTER SCIENCE

Text Archiving Systems and Document Validation Program in Status

Michalski, Jean

Award date:
1991

Awarding institution:
University of Namur

FACULTÉS
UNIVERSITAIRES
NOTRE-DAME DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

TEXT ARCHIVING SYSTEMS AND DOCUMENT VALIDATION PROGRAM IN STATUS

by Jean MICHALSKI

PROMOTEUR :

PROFESSEUR J. FICHEFET

Mémoire présenté en vue
de l'obtention du titre
de Licencié et Maître
en Informatique

Année académique 1990-1991

ABSTRACT

The aim of this essay's
first part is to provide a good overview
of the currently available tools for text archiving.
It discusses both software and hardware issues. The second
part introduces a textbase management system, Status, while
the third part describes and comments on a document
validation program designed and implemented
for the European Economic Communities.

RÉSUMÉ

Ce mémoire est divisé
en trois parties. La première, la plus
générale, introduit le lecteur aux techniques
d'archivage de textes et aux mémoires de masse que
ce genre d'applications nécessite. La deuxième illustre
par un exemple, Status, ce qu'est un système de gestion
de textes, alors que la troisième est entièrement
consacrée à la description du programme
développé lors du stage au Secrétariat
Général du Conseil des
Ministres.

to love

ACKNOWLEDGEMENTS

Thank you to all the people who gave me the best they had : my parents , of course , and also Anne, my sister , Yaya, my grandmother , my uncle Albert and aunt Suzanne and uncle Doug . Outside the family , I would like to give special thanks to Jacques Jordant, Xavier Decornet and Enrica Massucco .

This is the first formal opportunity I have to thank àll the people who , one way or another , helped me to grow up . Some are missing from this list ; I know they will be kind enough to forgive me . So , thank you very much to Renée Puttemans , Olivier Stenier , Jean-Marc Letroye , Jean-Louis Tummers , Anne and Dale Steele , Sunny Park , Pasquale Leone , Marc Levis , Valérie Melebeck , Jean-Pierre Indot , Marc Delhalle, Isabelle et Eric Gillard and Birte Jakobsen .

Thank you to the Professor Jean Fichefet for giving me a good stage at the European Council of Ministers . Thank you to Philippe Vleminckx and Mr Greco who helped to make it profitable , and to Mia , Giovanni , Rainer , John , Rod , Caroline , Isabelle , Nathalie , Manu , Balou , Claudia and all the others who turned it into a good moment of my life . Thank you also to Mrs Gallez , who delicately accepted to type this essay .

And last , but not least , thank you to all the teachers I had : I might forget what they taught me , but I sure won't forget them.

* * * * *

TABLE OF CONTENTS

PART I : ARCHIVING

3. Storage media

3.1. Disk
3.2. CD-ROM
3.3. CD-WORM
3.4. CD-WARM
3.5. Digital paper
3.6. Quarter-inch cartridge
3.7. 4mm DAT Helical Scan
3.8. 8mm Helical Scan
3.9. Holographic data storage

4. Evaluation

5. Conclusion

PART II : STATUS

1. Text retrieval in Status

2. Inside Status

3. Text input in Status

4. Error handling during input : the major problem in Status

PART III : DOCUMENT VALIDATION PROGRAM (DVP)

1. Introduction to the jur textbase

2. Specifications of the DVP

3. Problems encountered during implementation

3.1. Lexical analyser
3.2. Parser
3.3. Error handling
3.3.1. Error detection
3.3.2. Error recovery

4. Conclusion.

PART I : ARCHIVING .

1 . INTRODUCTION .

It is now universally accepted that organizations , in order to pursue their business , need constant access to information . Increasing amounts of money have been spent on projects aiming at a better management and storage of information .

The old products for processing information in general , and text in particular , have failed to acquire large market shares , because of their expensive hardware needs . But new generations of software and hardware are emerging , making it more attractive for businesses to automate their text archiving process .

The first part of this essay tries to classify the technologies currently in use for text archival . We will have a look at the most commonly encountered methods , and at the appropriate storage media to back these up . But the first thing we will do is realize why text archival is such an important issue for organizations .

1.1. Information , documents and formware .

The term " information management and storage " includes all tasks that can be carried out on information : acquisition , organization , retention , distribution , archiving , and eventually purging . Information can appear in many forms : text , graphics , sound , ... and on many media : paper , microfilm , floppy disk , ...

As this essay will continuously speak about documents , it is important to have a firm grip on this notion . The most comprehensive definition I have found was given by Gerald P. Michalski (no familial relationship !) : " A document is a snapshot of some information that can :

- incorporate many complex information types ;

- exist in multiple places across a network ;

- depend on other documents for information ;

- change on the fly (as subordinate documents are updated);

- have an intricate structure , or complex data types such as full motion video and voice annotations ;

- be accessed and modified by many people simultaneously
(if they have permission to do so) " . (MICH91)

In order to manipulate documents electronically , the need for internal representation has arosen . It has led to all the character sets , including the 7-bit ASCII code for text , wich was soon replaced by the 8-bit ASCII (1). Now , we have some interchange standarts for multi-media documents , like ICL's Normalized Documentation Format , or ISO's Office Documentation Architecture (ODA) and Interchange Format .

The particular field of study and computer systems dealing with document structure is usually referred to as " formware " .

(1) : A neww standard is now emerging : the 16-bit UNICODE .

1.2. The cost of information .

In a small text box, Mark Shapinter writes : " (...) US companies spend more than $ 6 billions every year on preprinted forms ; US companies waste $ 2 billions in preprinted forms ; and US businesses spend from $ 94 to $ 120 billions per year to distribute, store and process forms. While most organizations measure the cost of business forms by the cost of buying a paper form, it has been found that for every $ 1 spent to buy a paper form, $ 60 is spent to process it ! " (MICH91) . Tony Hundley writes : " Coopers & Lybrand found that of all the information stored and accessed by companies in the U.S., only 1% of the total is stored in coded form on digital storage devices ; only 4% is held on microfilm for space saving purposes and a staggering 95% is still held in paper form and the total volume of paper stored is still growing at the rate of 25% per year." (HEND90b) . In " Catch the wave of DIP ", David A. Harvey writes : " American business alone produces close to 1 trillion (1000000000000) pages of paper a year. (...) Not only does American business waste paper, but according to various sources, it costs about $ 25 000 to fill a four-drawer file cabinet and $ 2160 to maintain that cabinet for a year . In addition, about 3% of all documents are incorrectly filed or lost, and the average cost to recover a document is around $ 120. Finally, the average executive spends a grand total of about four weeks per year waiting for documents to be located . You know how it goes - you spend 20 minutes pawing through the file cabinet and half an hour looking through the various unsorted piles in the storeroom." (HARV91a) .

How could it be made more obvious that information is a scarse and expensive resource ? I have no big working experience, but what I saw at the European Communities was just beyond understanding. The European Institutions are not short of money ; their budget is large enough. Take for example the translation process, which is partly automated : the document comes in, written in one of the 9 languages of the Communities. One copy is sent to each translation pool (9 pools), who translate it with the help of various databases, and produce a word processor file. This file is printed, reviewed and anotated by another translator, and then the listing (the listing, not the file) is transmitted to the appropriate typing pool, where the secretaries use the same word processor to produce the final copy. Is this typical to public administrations? From what I heard and read, I wouldn't bet.

1.3. Documents archival .

When you look back on the literature, you find plenties of articles describing a brave new world of documents. The paperless office had become a very near future which would bring mountains of advantages to the modern managers.

The advantages are easy to identify : lower space requirements, easier and faster access to documents, improved file integrity, increased security, longer document life and lower costs through productivity gains and space saving.

The first step in text archival was made with the help of microfilm which was introduced in the 1970s. The major problems of microfilm were that the original documents had to be filmed (up to fifteen days of unavailability) and that the retrieval process was still very slow.

For many years, most organizations used the computers to store the references to their documents. This provided them with a good compromise between efficient use of computers and storage problems. As the users became more and more experienced, with these systems, the indexing proved to be a pitfall : no one could agree on a standard way of indexing the documents. Particularly, the level of details for indexing purposes was a big issue which has never been solved .

As computer science started to grow, the computers became always more powerful, and the first full text retrieval systems appeared. Taking files from word processors, these systems absorb the text in their textbase, and prepare it for retrieval. Some use an inverted file of occurences, others work with signature files,... Their common characteristic is that they offer fast and flexible searching facilities. Their major drawback is their unability to handle data types other than text .

Then came new peripherals such as laser printers, paper scanners, and mass storage devices. These new technologies allowed the development of systems later called Document Image Processors (DIP) , which combined the advantages of the ease and speed of high-definition scanners, the fast searches performed by the new generation of DataBase Management Systems (DBMS), the facilities provided by Local Area Networks (LANs) for carrying the documents in the whole organization, and the very good printing quality of the laser printers .

1.4. Wide Area Information Servers (WAISes)

More and more databases, hypertext systems,... are created for business needs. But the companies maintaining these information wells can also make money out of the activity of sharing the information. Thus, more and more remote databases can be accessed by Wide Area Networks.

For the user, consultation can become rapidly very expensive, because it takes time to browse through very large amounts of data. In order to minimise the human interaction with remote information servers, software has been designed whose main features are that they are in charge of the transactions with the servers, that they provide connectivity with any system, and that they are able to retrieve topical information.

As we are moving towards an information society, the need for good connection software will become of utmost importance. But this is not the essay's subject : I will concentrate on how textual information can be archived, discussing both software and hardware issues.

1.5. Data overload .

In spite of everything I have said before, many people reckon not much progress has been made in information management. Instead of organizing the information we need, computers seem to have buried us under a gigantic mass of data.

The availability of remote databases make this problem even more preoccupying. It is a good thing to be able to question as many databases as possible, but the answers have to be of a good quality.

And that's exactly where the difficulty lies. When you are looking for documents dealing with some particular field of interest of yours , the system gives the result of its scan, which is determined not only by the request you gave at retrieval time but also by the additional searching input introduced along with the documents. So, if you don't give the optimal parameters to your search, the system will probably "forget" some documents in the database, and give you some others that you don't need at all.

It is a good thing for organizations to have access to such information, as long as it doesn't alter the value of these informations.
If the company feels it is a good solution, it can ask specialized people to handle their information service, but our problem is still there : how can we have manage effectively our textual information ?

2. EXISTING METHODS .

2.1. Text retrieval .

This method has many variants ; we will discuss the major ones : full text scanning, access by index, and multiattribute retrieval methods. We will then explain a new interesting approach, the N - Grams, and finally give an example of a widespread full text retrieval product, Folio Views.

A common point between all text retrieval systems is that they store, one way or another, the full textual contents of the documents.

Furthermore, updates on filed documents are not often required. Two types of environments have been identified by most authors (BLAI91 , FAL085 ,...); they are the library environment, where documents pile up, and have more or less the same importance, and the office environment, profit oriented, and more dynamic. The archival systems have to take these organizational features into account and provide adapted access methods.

If the document is left as it originally appeared, the only access method possible is full text scanning. On the other hand, we can assign key words to each document, which is then represented and accessed by its list of key words.

Every query language for text falls in one or two categories : either the Boolean algebra approach, in which any document does or does not qualify for a query made of Boolean and textual operators, or the "fuzzy" queries, typical to users who do not have a clear idea of what they are looking for. For example, you can go to the library and ask for all publications related to one or more areas of computer science.

2.1.1. Full text scanning .

In this first case, the only available data is the textbase ; the way to locate the documents satisfying to a particular query is to scan the data, using the most performant algorithm. Many solutions are known; the fastest is that of Bayer and Moore. They chose to compare characters from left to right in the textbase, and from right to left inside \each string ; whenever a difference is observed, the search string may be shifted up to m positions to the right ; where m is the length of the search string. So, the overall complexity of the algorithm is very good : in the worst case, m+n, where n is the lengh of the documents, comparisons are needed.

Nevertheless, the response time is bad, because the textbase can be enormous. The real advantages of full text scanning are that it requires no storage overhead, the effort on updates and insertions is minimal, and exact string matching is possible.

The use of this method is not widespread ; it is sometimes used for the most recent updates of systems primarily using an other method, or for the index methods at a deep level.
Notice that special-purpose hardware exist in order to speed up full text scanning, such as parallel comparators, cellular comparators and finite automata, which allow many strings to be searched for simultaneously.

2.1.2. Index .

A second idea for text retrieval would be to have a dictionary where any word could be easily found, and the corresponding entry in the dictionary would give the places of the textbase where occurences of this word can be found.

The retrieval process is greatly speeded up using such indexes : instead of looking though the whole document, you only have to make a search in the index, which, if it is properly organized, can be very short, and then all occurences are at direct access.

The advantages, besides speed, are the easy implementation, and the easy handling of synonyms. Of course, a big storage overhead is required, to hold the dictionary, and the insertion, deletion and updates are harder to implement , because the index file has to be maintained.

Most words in a text file are unnecessary to be indexed : "a" , "the" , "and" , "but" , ... They are called to stop list, and leaving them aside can dramatically reduce the index file's size. Other refinements include\ better organization for the index file : methods like hashing, B-trees, or multiple-level index files may be suited. Special hardware can also be used to accelerate list mergings. The inverted list of key words is the method most often used in commercial systems.

2.1.3. Multiattribute retrieval methods .

These methods require to define a (short) list of key words for each document. From that list, the document's signature is extracted using methods such as superimposed coding, which is most suitable for text (the key words are treated equally) . Once the signature has been calculated, two approaches are possible : multiattribute hashing and signature file.

2.1.3.1. Multiattribute hashing .

It is in 1971 that Gustafson came up with this method. He uses a hash function whose argument is the list of key words and which returns a list of possible buckets the related document can be found in.
The function deals with combinations, and for more details, you can consult Faloustos very well written article (FAL085).

The disadvantages of Gustafson's process are that the performances are related to the file size, that the maximal number of key words in the original document is fixed (but you can divide the document in smaller parts, and assign fewer key words to each part), that if you try to retrieve documents with a small list of key words, the number of "hits" will eventually be very big, and that only the conjunctive queries are easily handled.

Variations and improvements have been explored, which could eliminate one of the disadvantages, but lost the advantage of easily handling queries involving a few key words.

2.1.3.2. Signature files .

The \ multiattribute hashing seems to have greater theoretical interest than practical, which is very different from the second approach, that of signature files. The only difference here is that the document's signatures are stored in a separate file, which provide easy-to-implement filtering functionalities. Many implementations have been realized, some of them described in (FAL085). The only disadvantage remaining in this method is that the performances decrease as the file becomes larger.

2.1.4. N-Grams (6) .

This is the last method I chose to incorporate in my essay. The clustering method will not be explored in details , because the basic idea will be explained along with N-Grams, and more specific information is fast getting overwhelmingly technical.

N-Grams is a new approach developed by Raymond D'Amore and Clinton Mah. It is a specific method for "fuzzy" queries, like " Give me all the stories about plane crashes " in a large journal articles' textbase. Such requests are very difficult to handle with key words, or with any other method.

The technique uses small parts of words, called N-Grams. An n-Gram is a sequence of n characters occuring consecutively in a word. The 2-Grams of duck are "du", "uc", and "ck". An N-Gram vector is a list of N-Grams found in a document, each associated with the number of its occurences.

To build a text-retrieval system based on N-Grams, you have to make a "good" N-Gram vector for each document in the textbase. Then, for each query, you make its N-Gram vector, and, using a "good" similarity procedure, you compare it to the documents' vectors and retrieve those documents whose similarity coefficient is "high enough". You noticed the quotes around 3 expressions. These show where choices can be made.

(6) : The following text is inspired from the article referenced as (KIMB88)

2.1.4.1. Making good n-gram vectors for documents .

Prior to everything, you make a list of all possible N-Grams, starting with 2-Grams. From those 26 X 26 = 676 2-Grams, some can help to characterize documents, and some are useless. You insert the relevant ones in your list, and expand the others to the right, thus making 3-Grams. Again, from those 3-Grams, you insert the relevant ones in the list and expand \the others into 4-Grams. You stop when you want to. In their testing system, D'Amore and Mah had built a 12000 N-Grams list, using 2-, 3- and 4-Grams.

Each document has to have a different N-Gram vector ; it will be a kind of fingerprint. The N-Grams are taken directly from the text, and not from a summary or key words. Many words in the text can be removed ((KIMB88) provides a list of 256 words wich cannot identify documents and which make up to 55% of the text), and only the stems of words are relevant, so a stemming (conflating) procedure is used.

Out of this compacted text, you start making the vector. First, you count the 4-Grams, then the 3-Grams, and finally the 2-Grams. When the vector is made , you store it for fast access, along with a reference to the corresponding document.

2.1.4.2. The similarity procedure .

Each N-Gram is assigned a weight (the more common, the lighter) ; a good weight formula is $Wi = 1/\sqrt{Pi}$, where i designates an N-Gram, Wi is its weight and Pi its probability of occurence. Pi's are calculated experimentally, using a large representative population of documents. The Wi's are stored in a constants' table.

To calculate the similarity of two vectors V1 = (ng1,ng2,...,ngn) and V2 = (ng*1,ng*2,...,ng*m) is to take the similar N-Grams in both vectors, and, from that subset, to calculate their similarity coefficient, defined as the sum of the products of each N-Grams'frequency by its weight.

The greater the result, the more related the texts. At search time, you give a threshold and all documents whose similarity coefficient is larger will be retrieved.

The procedure can be refined to take into account the fact that a larger document with a larger vector will have greater frequencies for each N-Gram , and will be more likely to be retrtieved. You can then modulate the coefficient with statistical tools. Also, in order to handle the synonyms, a thesaurus can be maintained.

2.1.4.3. Clustering .

Clustering the data on a disk surface is a method by itself for organizing databases in general and textbases in particular. I will illustrate this method very briefly in the N-Gram context, but it can be applied for other purposes.

The assumption is that similar documents have a tendency to be retrieved together ; and documents with related subject matters tend to have similar N-Gram vectors. Thus, you organize the total set of documents in groups, all documents of a given group having similar vectors. You calculate a vector for each group by adding the vectors of group's documents. The group is called a cluster.

Two methods are then possible. Either you use the document's vectors, as before, and check if the similarity is also respected for others documents in the cluster (if not , you might have a false hit) , or you first select a cluster by using the cluster's vector for comparison, and then retrieve documents with their vector inside that particular cluster.

2.1.4.4. N-Grams, what for ?

N-Grams have been experimented on full-text retrieval systems, but might be suited for more structured data . In the textual environment, it works well, and certainly better then keywords ; the N-Gram indexing can be automated, which is delicate with key words.

But the method has its drawbacks too . To have a fine-tuned system is quite complex and long ; the vector creation step consumes lots of CPU-time, and requires many accesses to secondary storage ; finally, and this is the major drawback, the meaning of documents is not present in N-Grams. Therefore, similarity can be faulted, and avoiding this with better software will lead to even more complexity.

2.1.5. Folio Views .

When I first read about this program, I thought it was useless even to mention it in an essay. But the lack of good text retrieval products leads me to consider my first opinion. Folio Views is presented here as a commercial product which can be considered as a good and affordable text retrieval system. It might be a good example of what is to come in the full-text area.

Folio Views is an integrated product : it combines text retrieval, hypertext linking, word processing, directory management, and electronic publishing. It has a new way of indexing the words ; it indexes all words, without a stop list, and compresses the files . It allows fast Boolean searches, creations of selected views of the textbases, links across the textbase, and easy modifications to the textbase (no reindexing!). The product's faults are its slow text scrolling and bad error handling.

2.2. Hypertexts .

2.2.1. Historical background .

Before speaking about the capabilities of hypertexts , we feel it is helpful to provide a general overview of the concept . In 1945 , Vannevar Bush first introduced the hypertext concept as a process close to human thought . He writes : " Human thought \(...) proceeds by associations . We cannot hope to artificially reproduce this mental process , but we certainly can extract knowledge out of it . We cannot come close to the speed and flexibility typical to the mind's associative path , but it should be possible to surpass thought on issues such as durability and accuracy ".(DANI90) .

But is it reasonable , can we imitate with computers the mind's organisation ? No , of course , but isn't that the goal? The computer scientist feels deprived : he has no efficient tool to tackle down the problems of a human being in front of free access to information .

In 1960 , Theodor Nelson originates the word hypertext . On the same year , the first hypertext implementation is realized by D. Engelbart , who creates the oN Line System (NLS) , designed to store memos , research notes and documentation . Today , this system is called Augment and is well fitted to handle highly structured informations . Nelson invented the Xanadu system , designed to interconnect , with the help of links , electronic documents and other kinds of hypermedia data such as graphics , sound , or full-motion video . Xanadu is currently running on Sun stations .

Beyond these implementations devoted to big computers , some hypertext systems can also run on micros . Guide (Brown , 1982) is a document processor using either IBM PCs or Macintosh systems , and hypercard (Macintosh-based) , while it is not a real hypertext system , has strongly contributed to the popularization of hypertexts .

2.2.2. Definition .

Analysis of hypertext literature is fast leading to the conclusion that there is still no standard definition of the term " hypertext " .
When he came up with the word in 1960 , Nelson gave it the following meaning : " association of a natural language with the computer's ability to make interactive links and dynamic displays of non-linear text " .

This definition , while appropriate for early systems which only stored textual information , is now obsolete because of its lack of generality .
The present hypertext systems are mostly handling multi-media documents which can be dynamically modified by the readers . They contain informations of various types , and this variety makes it difficult to define them .

An important characteristic of hypertexts , often put aside in definitions , is that a hypertext's knowledge can be accessed directly with no skill in programming requests (big difference with DBMSes) .

A more recent hypertext definition was given by J. Fiderio : " Hypertext , at its most basic level , is a DBMS that lets you connect screens of information using associative links . At its most soplisticated level , hypertext is a software environment for collaborative work , communication and knowledge acquisition . Hypertext products mimic the brain's ability to store and retrieve information by referential links for quick and intuitive access " (FIDE88).

We will describe later the concepts used in this definition but , for the moment , Nelson's definition is sufficiently clear and comprehensive .

2.2.3. User - friendlyness and hypertexts .

The main characteristics of a hypertext system are:

a " good " man- machine interface , and one or more processors of text , graphics , and others . The basic principles don't change if animation , video pictures , sounds or other media are introduced , but lead to new technical problems (storage methods , ...) . We will rather use the term hypermedia systems when the stress is put on non-textual media .

Conventional documents compel the reader to follow all the time the sequence defined by the document 's organization. In contrast , a hypertext allows its users to browse the document at will . The user becomes active and implied in his learning process , because he finds his own way through the consulted documents.

The spirit in which the man-machine interaction is handled greatly unifies hypertexts . It aims to make access to information more flexible , more efficient , and , above all, more natural .

2.2.4. Utility of hypertexts .

In general , a hypertext system can be considered as a DBMS which allows to browse through screens of information using the predefined links . In a certain manner , it is like a book you could read in many directions and modify at will . Let's have a look at the general properties of a typical hypertext .

It can be used as a writing assistant , then it is a simple extension to structured documents editors. It has to be suited for tools management , information retrieval , and documents' lifetime management of software engineering systems . Moreover , it should provide features for collaboration between authors of a same document . Criteria such as pagination , presentation , and exploration commodities are most relevant .

Typically , you will find four types of hypertext systems: problem solving systems , interactive browsing systems , encyclopedia systems , and multi-functional systems . The available tools can differ , according to the system type .

A hypertext system is useful for the creation and consulting of information networks and of documents (taken in the broadest meaning of the term) . A primordial aspect of hypertexts is the communication with the outside world , which is denoted by a great interactivity with the users .

2.2.5. Basic concepts .

To understand how hypertexts work , we must introduce the concepts of node, link, and network. The following overview will be very general, covering all concepts encountered in all existing systems , thus describing an ideal hypertext . Yet, there is currently no such complete hypertext system .

2.2.5.1. Nodes and links .

2.2.5.1.1. Definition .

Nodes are information holders considered as semantic entities different from each other . Nodes are related to our natural way of dividing information . A link represents a relation between two nodes or between\ information held in nodes . Links are related to our natural way of structuring information . Links usually originate at a single point , called a link reference . Their destination , called a link referent , is usually a node , a chunk or a region of text . The links allow the reader to browse a hypertext non-sequentially . The set of nodes and links is called a network .

2.2.5.1.2. Representation .

Depending on systems , nodes are implemented as screens (Hypercard) or windows (NoteCards) . Windows allow the simultaneous display of several nodes , which can help the user to find his way faster .

A link is represented by its extremities , called anchors. Anchors are materialized by icons or buttons (2) . The buttons, either iconic or textual , are usually representative of the type of node which they point to .

2.2.5.1.3. Information structuring .

The relationships given by the links allow the overall consistency of informations , but they don't necessarily structure them .We can compare links to some sort of cement which , while keeping a wall's bricks together , would impose no shape for the wall .

(2) : The button represents the way used to open an other node , it is not the interactive object defined in man-machine studies .

A document's structure (3) can be reflected by links , which can also help to see the documents at various levels of detail . They provide at least a path through the document which corresponds to the good old sequential way of reading . Most hypertext systems specialize nodes and links in more precise types , adapted to specific modelisations such as a document's structure .

2.2.5.2\ Types of nodes and links .

2.2.5.2.1. Information nodes and composite nodes .

The information nodes (4) are defined as the structures to which informations are associated . These nodes have the following properties , or attributes :

- a presentation (information's nature, size, font, ...)

- an identification (name, creation date, author, ...)

  and

- a situation in terms of other nodes (enumeration of incoming and outgoing links) .

Properties associated with nodes vary from one hypertext system to the other , we have given the most common .

Composite nodes store informations on nodes they group together , and are convenient for representing a document's structure . They are provided by such systems as NoteCards , but their use doesn't seem generalized ; they most usually serve to make tables of contents and to give partial views of the network .

(3) : The term document is used here to designate a set of informations organized in a logical structure .

(4) : In the literature , information nodes are also called frames , chunks or informational objects .

The information (signified) and its representation (significant) are not differentiated by the authors although they are distinct notions . Therefore , it is admitted an information's significant's modification (for example : change the font) can have an influence on its signified . We will insist on this when we will discuss the modifications to a network .

2.2.5.2.2. Reference links and hierarchical links .

Hierarchical links implement the logical structure existing between nodes . They bind composite nodes with the nodes they group . They form a tree of all nodes, thus providing a linear way of reading (5) .

Reference links are the other ones : they link nodes which are semantically close . They provide a non-linear way of reading and the graph is not necessarily a tree .

Unlike for nodes , authors do make a difference between a link's significant and its signified , because , in this case, assimilating a link to its representation would mean that any representation change (for example : a button's shape) would bring into question its semantic meaning and therefore its very existence .

As a summary , we will say the information nodes represent information entities , they allow the readers to have a modular vision of the hypertext's semantic context .
The semantic relations between those nodes are given by the reference links . We will also say the documents' logical organization is materialized by composite nodes which allow a unique reference to a set of information nodes thanks to hierarchical links.

(5) : We mean here the "path" given whenyou sequentially follow all the hierarchical links .

2.2.5.3. Structures and hypertext approaches .

Two kinds of structures exist : the **primary structure** is given by the hierarchical links and represent the overall organization of information , that is , the document's logical structure , and the **secondary structure** concerns reference links . In a classic document , the primary structure is often put forward , whereas in a hypertext , the secondary structure is generally dominant .

Two ways exist to put forth the structures in a hypertext . The network approach is a classic one , it incorporates the rigid document in a hypertext . "Rigid" , because composite nodes and hierarchical links are taken into account ; the primary structure is explicitely represented . Therfore , in order to browse through a document , the reader can use either the primary or the secondary structure . As opposed to the network approach , the fugitive document approach only stores the information nodes and the reference links (the secondary structure) . The document is often formalized with Petri nets ; it comes to existence only at consultation time . It is created as it is consulted : every use of a link corresponding to transition's firing modifies the marking and enables access to new nodes .

2.2.6. Manipulation of a hypertext network .

2.2.6.1. Network update .

2.2.6.1.1. Creation .

When the user has identified what he considers to be a semantic entity , he will be able to introduce it in his hypertext network as a node . When he wants to add a relationship between two nodes , he will create a link .

We have seen we can associate a type to links and nodes ; as M-C Daniel-Vatonne says , "to create a link or a node is in fact to use a predefined type or to create a new link or node and then define its type " (DANI90) .

Thus , at a node's or a link's creation , you will have to give the object its type and the type's predefined attributes their values . Some attributes are mandatory , others are not , some are common to all the objects of a type , such as those which determine the screen display , and others are particular to one object , such as the creation date .

In certain systems , the user can define himself the types of nodes and links and therefore associate characteristics he feels pertinent . " To create a node consistà in asking the creation , eventually from a predefined type , then in precising its attributes and its contents (text, graphics , or other) . To create a link consists in specifying where its extremities (anchors) are , what these will be attached to , and in defining its attributes and the relationship it represents " (DANI90) . Notice that in certain systems , the creation of links is partially automated .

A very big problem of hypertexts is bound to what has just been explained . The identification process of semantic entities is indeed anything but trivial , particularly when the hypertext is built out of a "normal" text . Let's take for example the hypertext version of the Oxford English Dictionary (OED) . The main problem the authors had to solve was to fragment the existing text , and to create links between those chunks : most texts are not naturally fragmented; moreover , a hypertext should keep a good image of the original , and no implicit structure (especially those not detectable when you read the text for the first time) should be lost .

2.2.6.1.2. Destruction .

"To destroy a node is to destroy the incoming and outgoing links which deal with its contents , and then to destroy it . To destroy a link is to erase its representations and then to destroy it " (DANI90) . The authors agree to say the destruction , as well of nodes as of links , doesn't pose any particular problem .

2.2.6.1.3. Modification .

To modify a link is to modify its attributes or its anchors . No problem arises in this kind of operation . To modify a node is to modify its attributes or its contents.

Here , serious problems can occur . Indeed , if the modified contents was the anchor of one (or more) reference links , then inconsistencies can creep into the system .
Example : The context is a hypertext in which are stored great musicians' works and life . Let there be a reference link between a node "works" whose attribute "composer" is "Bach , J.S." , and a node "life" whose attribute "name" is "Bach , J-S." . Notice the hyphen , which did not appear in the first node . If , in the node "works" , you go and modify "Bach , J.S." ,\what happens to the link ? Is it brought into question if you add a hyphen ? And if you replace "Bach , J.S." with "Bach , J.C." ?

2.2.6.2. Exploring a hypertext network .

Two exploring modes can be differenciated : browsing and searching . In the case of the OED , one of the main reasons for the hypertext implementation is the facilities the system provides , as well for browsing as for searching .

2.2.6.2.1. Browsing .

"The principle of browsing is to go through the network using links to reach and consult the nodes" (DANI90) . For the reader , it is impossible to use a link if he doesn't know one of its anchors . In order to display the node the link refers to , you have to click on the mouse , for example .
You must be aware that in big networks , you get easily lost , and help is welcome . Several kinds of help can be given :

Punctual help

The author gives an adviced way to go through the system . If the user doesn't carefully follow that path , the problem is unsolved .

Spacio-temporal help

Spacial help is typically a map in two or eventually three dimensions . Temporal help keeps record of what has already been done by the user , by marking the nodes , or by updating a list of visited nodes .

Structural help

The author defines the good paths to follow for an efficient browsing of the document .

Petri-network based help

This is the case in which freedom has been reduced to its minimum . The whole hypertext document is modeled by a Petri-net , where nodes and links match places and transitions. Every user is assigned an initial marking , which varies , depending on his skills or access rights ; a node's consultation requires that the transition associated to the link used to arrive at the node is enabled . The author has the power to impose the visit of some node before others .

2.2.6.2.2. Searching .

Searches , which is the other method for exploring the network , can be made on :

- the contents of one or more nodes , or on
- the structure of nodes (for example : search all isolated nodes)

It can be an overall search , or specific to nodes that have already been visited , to nodes of a certain type , ...
The search by contents implies index and key words manipulation , with all the related difficulties , especially when the node's contents is not textual .

2.2.7. Hypertexts and DBMSes

Hypertexts and DBMSes share many features : for example , they are systems providing access to information organized in semantic entities , they are systems that can work on objects of various types (texts , graphics , ...) , ...

Nevertheless , some differences can be underlined . First , DBMSes usually give much more sophisticated searching methods ; then , the DBMSes are not neighbourhood-sensitive , and don't offer the necessary tools for exploring such neighbourhoods . And also , one of the properties of hypertexts is their user-friendlyness , which you don't always find in DBMSes . Systems have been developed that combine DBMSes'performances and hypertexts'user interface .

Every hypertext system uses a DBMS to store its data . This DBMS can be as small as a home-made file manager , and as big as a large relational DBMS . There is no standard , and this could , as time goes by , be an obstacle to Nelson's grand vision of integrating all the world's hypertexts .

A step towards a solution has been made by HAM's designers (Hypertext Abstract Machine) (CAMP88) , who advice , inter else , the introduction in the hypertexts' logical architecture of a new layer , which could be positioned between the DB layer and the application layer , and which could standardize the services , the primitives that the underlying layer provides to the layer above it .

2.3. Binary Large OBjects.

Apart from object-oriented databases, which will not be discussed in this essay, because it would be too large to fit in here, conventional database management systems have evolved in such a way that some now provide the user with a multimedia orientation. In particular, some relational databases, like Informix-Online, \have integrated a new data type category : Binary Large OBjects (BLOBs). BLOBs can be used for storing anything from text to graphics or sounds.

But relational DBMSes have their own restrictions : transactions should always garantee atomicity, consistency, isolation and durability (ACID properties), and the introduction of BLOBs should not alter the way other data types are handled. The only need is to extend existing systems so that they can handle very large unstructured objects.

In fact, two basic solutions exist : either the BLOB is known in the database as a pointer to a remote place where it is physically stored, or it is really inserted among the data. In both approaches, all BLOBs are treated as pointers by the DBMS : unless an application explicitely references it, the BLOB's value is never accessed. For the moment, only a few operations can be carried out on BLOBs, but this is changing.

In order to respect the ACID requirements, while not penalizing the other parts of the database,' some processes inside the DBMSes have to be modified. For example, the BLOBs are not inserted in the log file, becausethey are too large (a free-space map is maintained instead) ; backups should be made incremental, because most BLOBs are static ; and shared buffers should not be used by BLOBs, which would soon full them all.

2.4. Image technology.

Full text retrieval systems provided the user with contents searches and analysis. Their major drawback is obviously that they fail to give access to the other data types then text. A totally different approach has been developed by image technology products. Here, any document is treated as an image, a picture.\ In order to retrieve the meaningless data of pictures, information is added, such as index entries, key words which best characterize the document,...

2.4.1. Requirements .

The functional requirements are that of input, which is most often performed with scanners, of image compression and expanding, using CCITT Faxes of group 3 or 4, respectively using a one-dimensional code (compression rate : 5-15 to 1) and a two-dimensional code (compression rate : 15-30 to 1), of printing, mostly taken in charge of by laser printers , of retrieval and of scaling and rotating.

The integration with conventional information systems and networks is an important issue, as is the quality of the graphical interface and the standard for application integration. In the same idea of integration, the system should support as many image formats as possible : scanned, FAX,...

It is of primary importance to create and maintain an appropriate and efficient index. Ideally, fully or partially automated conversion to full text could generate the date necessary to make searches, as would a link or integration with a multimedia DBMS .

The primary data extraction can be performed either by automatic information extraction of initially designed forms, or by tagging the document from the start, using, for example, bar codes.

2.4.2. Advantages .

The first and obvious advantage is that human minds are tuned to deal with images. Then, you can also point out that images are very flexible in terms of input : you can easily handle any format , and respect the original layout as well as important features such as signatures. Some systems allow the graphical amendment of pictures, for example using an electronic stylus. Direct input and output are possible with the fax, and laser printers insure fast and perfect direct output.

2.4.3. Disadvantages .

Two generic disadvantages are to be underlined. The first one is high hardware and software requirements and costs: monitors and laser printers should be of very good quality, high volumes of main memory are required in order to support advanced user interfaces, the caching of images, and the compression and decompression of images. Storage requirements are immense, because, depending on the scanning resolution, an A4 page requires 483 KB (200 dpi) to 1933 KB (400 dots per inch). Compressed, the same page uses 30 KB. So, you rapidely go up to gigabytes. Moreover , as these image based databases are expensive and searched for by users, you might want to incorporate it on a network which is also an expensive investment.

The second disadvantages are bound to the limitations of the image format. Scanning times can be long ; not all systems give insurance that the text is reusable for further processing, nor do they accept direct input from word processors : a raster print image is needed ; finally, the search strategy is crucial, including the building of the index.

3. STORAGE MEDIA .

Only the relevant technologies for text archival purposes are presented. Such media as tapes, semi-conductor memories, flashes, have been deliberately left aside, because of either their limited capacity or their prohibitive access times .

\

3.1. Disk.

Magnetic disks have been around for about 30 years now, and it is noticeable . The technology has matured, it is perfectly incorporated in the data centers of all companies, and, better, it is still on a rising slope. Rigid magnetic disks now operate with their head flying on an air cushion above the surface.

Over the years, the surface of disks has changed, allowing increased densities through higher coercitivity (the magnetic field required to reverse the direction of magnetization of a bit). It is now, thin films made of cobalt-nickel alloys, covered with anti-corrosion carbon coating, which also protect the surface against the head.

Further improvements are possible, particularly in perpendicular recording and in head design. Meanwhile, the strenghs of magnetic disks lie in very fast access speeds, reliability, high capacity, low cost and proofed technology . Their weaknesses are the fixed character of the media, and its costs when very large capacities are needed.

3.2. CD - ROM.

The first kind of optical storage is also the most standardized. CD-Roms are defined by iso 9660, and so it has become widespread and interchangeable. CD-Rom stands for Compact Disk Read Only Memory, because the support is the same as audio CDs, and it is a read-only\ media.

To make a CD-Rom is a very costly process; it is similar to making a record , it has to be recorded, a master disk is first produced, and the other disks are then industrially produced by pressing.

The data is recorded on the surface as holes, which reflect light a different way than the unaffected surface. A laser beam is used to read the data . The main difficulty is to have the beam correctly focused on the surface ; another problem is the error correction : up to 30% of the recorded data is used to detect and correct errors .

Whereas disks are divided into plates, tracks and sectors, CD-Roms are recorded linearly as a spiral going from the inside out. Constant linear velocity is used, which means that each record has the same length on disk, and, as the head is going away from the center, the rotation speed decreases to make sure the same amont of bits always goes under the head . By the way, disks use constant angular velocity : the rotation speed is always the same, and records on the edge take more room than those in the center .

The CD-Rom's performances are : a storage capacity of about 550 MB , an uncorrectable-error rate below 10E-13, an average access time of 1 second and a relatively slow transfer rate . They are more and more used as a publishing media for large amonts of text such as encyclopedia , or extensive documentation.

3.3. CD - Worm .

Write once, read many optical disks currently suffer from the lack of standards. Nevertheless, this technology is a step beyond CD-Rom : you can write your own data to a CD-Worm, although you can't change what you have written.

Data is written using a strong laser beam and read using a weaker one .
The write process differs from system to system : in some, bubbles are made in a thin absorbing layer above the active layer ; in others the laser ablates a pit in the material. Improvements are still looked for , especially to improve density and to reduce the price.

Current formats are the 12 inches disks, for which there is no standard, but performances can go up to 6.5 GB of capacity, a fair 120 ms average access time and a date transfer rate of 8 MB per second. These numbers will improve as the research gives new results . The 5.25 inch format is a bit more settled, and gives a better average access time (48 ms), while cutting down on capacity (1 GB); the data transfer rate is also slower : 5.5 MB per second.

This is a logical competitor of microfiche and microfilm ; it provides the user with faster access, on-line capability, durability and removability.

3.4. CD - Warm .

The Write and Read Many optical technology has been waited for. And, when it came out, it was'nt as simple as CD-ROMs or CD-Worms. In order to write and read many times, no destroying of the recording layer is allowed.

Two main technologies are now widespread : magneto-optical disks and phase-change disks.

Magneto-optical disks combine the laser and the magnet as tools for recording data. The surface of the disk is here one that can be magnetized, but only at high temperatures (past the Curie point). The laser beam is used to heat the surface on a very small spot, and the magnet is used to give the surface its magnetization. The read process is accomplished using a weak laser beam, and detecting the differences in reflectivity : the light's polarization is different if you have a 0 or a 1 . The write process takes two phases : first you erase the surface, setting all the bits to 0 or to 1, and then writing all the bits.

Phase change technology is purely optical ; the surface can switch back and forth between crystalline and amorphous state. You simply have to use different energy levels of laser beams to go from one state to the other and reverse. The differences in the reflected read-beam are even more detectable than in the magneto-optical approach, and moreover, you need no erase process before you can write.

Because of its older development, the magneto-optical approach is more advanced. Because of the optical components, anoptical head is much heavier than a magnetic disc's. This leads in lower speeds for head movements, and thus the access times are not comparable with those of harddisks : the optical drives are 10 times slower. But, despite of that, they are attractive because of their high capacities, especially if you use jukeboxes, which allow you to mix CD-Roms, CD-Worms and CD-Warms.

3.5. Digital paper.

The latest in optical storage is the digital paper. The name is due to the way information is stored (digitally) , and the ability to produce large, flexible sheets of this media, actually made of polymers. It has been developed by the laboratories of Imagedata, whose parent company is I.C.I.

Its principle for recording data is similar to WORMs. Holes are made on the disk's surface using a laser beam . For reading, a lower intensity beam is reflected differently where holes have been made, thus deciphering the encoded bits.

The differences with WORMs is that digital paper is thin and flexible, and that the writing process doesn't make holes in the mirroring surface, only in the active layer above it. The flexibility of disks made with digital paper enables the use of the Bernouilli effect.

The Bernouilli effect is derived from the principle which states " that if the fluid flow is faster on one side of an object than on the other, the object feels a force toward the faster flow. It it the effect that allows an airplane's wings to lift it into the sky (...) " (POUN89).
Here, the surface of the disk, while spinning at 1800 revolutions per minute (30 per second), is lifted toward the head and stays at an approximate distance of 50 microns.

This property is very important, because the problem of maintaining a constant (and small) distance between the head and the disk is avoided, and the head is therefore lightened, allowing faster moves. Another consequence is that the small distance can provide better accuracy when making the pits ,and the density can therefore be increased.

Let's summarize the advantages of digital paper over CD-WORM : higher density, higher capacity, faster access (40 msec), better mechanical properties (much of the head has been removed), better latency, possibility of true double-sided disk drives, and higher data transfer rate (1.5 MB per sec).

The disadvantages include the write-once technology, the slow access to data, where compared to disks, and the immatureness of a new technology.

In the future, digital paper will develop and be used in removable disks and also in smaller memory devices.

3.6. Quarter-inch cartridge .

The quarter-inch cartridge is an tape-based analog storage media. It is still evolving ; new techniques allow better surface usage .

It is a small removable box containing tape ; it has been developed specifically for data-processing applications. Its main advantages are the low cost, the emerging comptability among multiple sources, and its high data transfer rate (currently 600 KB/sec); arguments against it are its relatively short lifetime (a few hundred passes) and low capacity . But the latter is very likely to increase. It is now of 1,35 GigaBytes, and is promising 6 GB for 1993 .

3.7. " 4mm " DAT helical scan .

" 4 mm " is the width of the tape, " DAT " stands for Digital Audio Tape, and " helical scan " refers to the way the data is aligned on the tape. Derived from the audio industry, this storage medium is particularly cheap (+/- $ 1.50 per 100 MB), offers big storage (2 GB) but slow random access time (average of 30 \seconds). The data is stored digitally and has a long lifetime : tapes can be read 1000 times.

DAT systems provide the user with a relatively cheap and reliable storage medium. As it is slow, it can not be used as the working storage , but is well fitted for backup.

Physically , the data is stored on parallel tracks wich have an angle of 6 degrees with the edge of the tape. Two heads are used : one for read operations, and one for writing. Both heads include a servo information detector. A rotating drum rolls the tape forward at a very low speed : less than 1 cm per second. But searches can be performed 200 times faster, because of the technology used, in particular the small wrap angle (90°).

Preventing against errors is insured by forward error correction, a new method used only on DAT devices. It cuts down the error rate to an exceptional factor of 10^{-15}.

3.8. 8 mm Helical scan .

The 8mm helical scan tapes drives are distributed by only one company (Exabyte), but they are worth to mention, because they have astounding archival capacities. Their capacity-to-volume storage ratio is exceptionaly high : 300 MB per cubic inch, thus making them competitive in terms of price per MB.

Technically, they behave a bit like their 4mm cousin, the DAT. It would be out of the subject to explain in details how both work, and the differences between them.

They have the high capacity (up to 5 gigabytes) and low cost of DAT, but not their fast search capability, while keeping their low data transfer rate, and adding the disadvantage of having only one supplier.

3.9. Holographic data storage .

If this product keeps its promises, we could soon assist to a revolution inside computer science. The idea of storing data in crystallite arrays using optical read and write processes has been in the mood for many years. But all experiments failed, due to the lack of experience in the light modulators field. \In the seventies, they were already putting holographic storage devices together, but they focused researches on increasing the capacity and failed to provide a solution. What was going wrong at the time is that the reading process, using a laser beam, influenced and destroyed the stored data, in such a way that after four or five reads, it would become undecipherable.

In 1988, researchers from Stanford University, along with Microelectronics and Computer technology cooperation found a new read process, which could prevent the signal to noise ratio of read data to decrease. The following characteristics are taken from an article published by Jan Parich in Byte (PARI90) .

Characteristics	Prototype targets	Achievable future targets
Storage module size	3 by 3 by 0.5cm to 5 by 5 by 0.5cm	10 by 10 by 0.5cm
capacity	200 MB to 2 GB	Over 100 GB
Page size	64 Kbits	1 Megabit
Average page read time	1 to 10 microseconds	100 microseconds
Average page write time	100 microseconds	10 microseconds
Average sustained transfer rate	100 to 800 MB/sec	Over 1terabyte/se
Costs	Prototype costs to be determined	Less than two times magnetic or optical disk cost per bit in 1895

Amazing, isn't it ?
But how is this achievable ?
In fact, the storage media itself is a small pack of tiny crystallites, made of photorefractive crystal such as strentium barium niobate (with some cerium added) or lithium niobate. The basic idea is to take advantage of some crystals' photorefractiveness : a laser beam can change their optical properties.
For extensive details on how holostores work , please refer to (PARI 90).

No \need to say, if such a storage media is commercialized, it will bring lots of changes to all computer science areas : the gap between CPU time and data-access times will be filled in . But for the moment, wait and see...

4 . EVALUATION .

A small picture is sometimes worth more than a long speech, said Napoleon. So, let's look at Table 1, which is a recapitulation of advantages and disadvantages of all methods discussed.

\

	Full text scanning	Index	Multiat-tribute methods	N-Grams	Hyper-texts	Blobs	Image tech.
1.	X	X	X		X	X	X
2.	(X)	(X)	X	X	(X)	(X)	(X)
3.	X	X		X			
4.	X	(X)					
5.					X	X	X
6.	X			X			
7.	X		X			X	X
8.		X	X	X		X	X
9.	X		(X)	(X)		(X)	
10.		X		X			
11.	X	X	X	X			

Table 1 : Text retrieval methods comparison.

(X) = partially or on some systems

1. Boolean searches
2. Fuzzy queries
3. Synonyms
4. Positional searches
5. Image storage possible
6. Automated indexing
7. High Textbase updates speed
8. High Retrieval speed
9. No storage overhead
10. Easy implementation
11. Low hardware requierements

As you can see, there is no ideal method yet. The possibility of archiving pictures is costly in terms of complexity and of hardware. No method provides good results on both Boolean searches and fuzzy queries. The positional searches ("all documents in which this word follows this word") are possible with full text scanning systems, which are the only ones to be intresequely slow at retrieval time .

As far as the storage media are concerned, the technologies are evolving very fast, so little can be said. Table 2 is a reminder of each technology's strong and weak points.

MEDIA / PERFORMANCES	MAGNETIC DISK	CD-ROM	CD-WORM	CD-WARM	DIGITAL PAPER	QIC	DAT	8 mm HELICAL SCAN	HOLOSTORE
CAPACITY (GB)	7,5	0,55	6,5	1	1	1,35	2	5	2
ACCESS TIME (sec)	0,010	1	1	0,06	0,04	36	20	90	0,00001
TRANSFER RATE (MB/sec)	50	10	8	10,3	1,5	0,6	0,18	0,5	800
REMOVABLE ?	N	Y	Y	Y	Y	Y	Y	Y	N
STANDARDIZED ?	Y	Y	N	N	N	Y	Y	Y	N
EXPECTED IMPROVEMENTS ?	Y	N	Y	Y	Y	Y	Y	(Y)	Y
COMMERCIALIZED ?	Y	Y	Y	Y	N	Y	Y	(Y)	N
DRIVE COST ($)	VARIOUS	1000	3500	5000		2000	1000	3900	
MEDIA COST ($)		300/1000	50	100		35	15	10	

(Y) : ONLY ONE DISTRIBUTOR

TABLE 2 : TECHNOLOGY COMPARISON (7)

(7) : Inside each technology, there are many formats available, with notable differences in performances.

What is sure, is that each technology has its particularities, and depending on the needs and on the budget, the customer will eventually come to a satisfying compromise. When holostores will be available, it will also be possible to see what they really are, and if they keep up with the announcements, they will soon outrank all other competitors, by just blowing up the storage hierarchy.

Some combinations of retrieval methods and storage devices just won't work. Full text scanning, requiring constant access to the textbase, need a fast access time, which can only be provided by the magnetic discs or by CD-Warm. QIC, DAT and 8mm helical scan are not fitted for primary storage of textbases, because of their prohibitive access time. The read - only characteristic of CD-Roms limits their use : they are not suited for dynamic textbases.

5. CONCLUSION .

As a conclusion, I would like to stress again the importance for some organizations to have a good document retrieval method. All the solutions explained in chapters 2 and 3 are not suited for all organizations. The most decisive work will although be done before any system is implemented.

The key to find the right system is to identify the environment and both current and future needs. Moreover, a migration towards an automated archival system might be a good time to review the way information is processed, and to redesign the whole document flow.

Good questions to ask include the following (8) :

- How is the work handled now ?
- What kind of documents are recieved, reviewed, considered and produced and what volume of each ?
- Where are the bottlenecks ?
- What are the costs and benefits of each stage of the information handling process and what other operations might link to the one being considered, how and with what benefit ?
- Can the various storage media and input options be used as appropriate ?
- How much help does the system give to users, supervisors and analysts ?
- Where and why are proprietary formats, hardware and software used ?
- Is there a transition from old to new ?
- Is there a high startup cost and, if so, why ?
- What is the level of technical risk and the speed of obsolescence ?

And, in order to insure the system chosen and installed will be used, you need to take all "political" aspects into account, and manage people involved the right way .

PART II : STATUS .

This second part of the essay will look in details at one particular text archival product used at the "Secrétariat General du Conseil des Ministres". In the first chapter, we will examine Status from the user's point of view. The second chapter will go inside \the data structures used by Status, and the third and fourth chapter will detail the input process and its deficiencies in terms of error handling.

1 . TEXT RETRIEVAL IN STATUS .

The user's point of view is certainly the best to have a good opinion of Status. When you sit in front of your computer and start playing around in a fully designed textbase, in which macros and screens have been written, you are amazed at the speed and variety of possibilities offered by Status.

Local, global, positional, synonyms handling, searches like " give me all grey cars prized between 1000 and 5000 dollars ", everything seems possible. With the advantage of being very fast, even when handling intricate boolean searches. As a textbase can be assigned a user-interface, systems can become very user-friendly.

2. INSIDE STATUS .

Status allows different privileges to a textbase's managers, who have all access rights, than to its users who have limited read and write access. You may not remove a textbase's last manager. When a manager makes a textbase, he has to fully describe it, in terms of space requirements and of structure.

The Status textbases are always divided in chapters. Each chapter contains one or more articles ; each article is made of a title part and a body part, which are in turn made of section. Sections contain one or more paragraphs. Anywhere in the text, a keyed field can be declared .

Chapters have no relation between them, other than belonging to the same textbase. The first chapter of a textbase usually is a description of the textbase, informing the user about the other chapters, the available methods, and any particularity of the textbase. Access rights are defined at chapter level, and searches can be limited to a subset of chapters.

Articles of a same chapter share the same access privilege, and the same overall structure. In one particular article, you can find any series of predifined sections. Two articles of the same textbase do not need to respect the order of occurence of the sections, and a same section may occur more than once in a specific article.

Sections can be either named or default sections. The difference between the two is that default sections do not have their name in the text. The section name can not be accessed by searches. Named sections can have their text split accross an article and their parts are displayed the way they have been entered. Searches can be restricted to particular sections.

You can make a distinction between concorded sections and non-concorded sections. In fact, "concorded" means "searchable".
No concorded named section can be added after the textbase has been created. Hence, some spare section should be defined.

Paragraphs are defined for even more specific displays and for positional searches. They also provide a better text structure.

Keyed fields have to be defined ; they have a value which is associated with the keyed field name in the concordance file ; they can therefore be accessed very quickly, and arithmetic operations can be performed with numeric keyed fields. They can also occur anywhere in the text : you might have three date-oriented sections (order-date, delivery-date and pay-date), and their contents might be defined as keyed fields for access facilities. You will then have 3 same keyed fields in the same article .

Status stores the text in its own compressed format. It builds a concordance file which is in fact a inverted file of key words, with one entry for each word. A common word is one which is concorded and thus impossible to be searched : we called them stop-words earlier; they include "and", "or", etc...

Each word position is expressed in four levels : chapter, article , paragraph and word number. As the concordance file is a static-size file, every dimension has to be specified at create time : maximum number of chapters in the textbase, maximum number of articles per chapter, of paragraphs per article, and of words per paragraph. This can be particularly dangerous for dynamic textbases.

Let's take a closer look at the concordance file, to see how searching is performed. The file is using the digraph method. The two first letters of a word are identified and searched for in a sorted list. Then a second list is used to search for the second digraph, which then gives the reference to a sequential sorted list of all words beginning with the 4 given letters. When the word is identified in that list, you have a list of all references, each corresponding to one occurence of the word in the text. The total number of occurences is given in the first block of the chained-blocks list. You can find an illustration of this in Appendix A , page A1 .

3 . TEXT INPUT IN STATUS.

The document flow is shown in Appendix A2 : the text coming from the secretaries is normal text. In order to be interpreted by Status the right way ; that is, to insert the documents at their right position in the textbase, the input text has to be marked.

This is the list of the markers used and their definition, given by Status's manual (10) :

$$# on a separate line defines the start of a new chapter.

$$T on separate lines delimits the title text.

$$P on a separate line separates each paragraph.

$$N followed by a section name denotes the start of a named section.

$$S denotes the start of the default named section.

$$A on a separate line terminates each article.

$$Z on a separate line signifies the end of text in create and enlarge modes.

$$W bracketing text signifies an external reference to a file outside Status.

$$Knum denotes a protect key of value num.

An example of marked text is found in Appendix B1. Please excuse the bad character prints : the internal format of characters is not standard.

The input file can consist of many articles belonging to a same textbase. They have to be sorted by chapter number.

(10) : "£" characters have been replaced with their continental version, "$".

4. ERROR HANDLING DURING INPUT : THE-MAJOR PROBLEM IN STATUS.

When you enter text in a textbase, everything works well as long as no error is present in the input text. When errors are introduced, either Status does not see them, or it does and gives somewhat inappropriate messages such as

"WARNING-concordance reference limits exceeded in article C : A" ,

not at check time, but at update time !

No syntactic (or semantic, for that matter !) check is performed, or so little. This means that you can insert anything in a textbase.

I even could enter :

```
$$#
coucou
```

in a fully structured and defined textbase ! The only message Status gave me was a warning of unrecognized named section.

Once my dummy input was checked by Status, I had all kinds of trouble removing it from the textbase, where it had been inserted. So you have to be very careful before entering anything in the textbase.

PART III : DOCUMENT VALIDATION PROGRAM (DVP)

Now that an introduction has been made to Status, we can focus on the work I have done during my stage. Mr. Vleminckx, operating in the New Technologies department, was my counsellor and he formulated the problem this way:

"Within the legal department of the General Secretariat of the Council of European Ministers, a document database, JUR, is maintained. This database contains descriptions of legal documents.

Status, a full text retrieval product, is used as the text engine. The product shines in document retrieval speed and query facilities, but lacks support for proper document input and validation.

The objective of a "Document Validation Program" (DVP) is to have a program that checks the syntax and the semantics of the new documents before they are loaded into the JUR textbase.

The following plan must be followed:

0. Introduction to Status and JUR
1. Agree with the user the validation rules
2. Implementation of the program
3. Program testing
4. User and program maintenance documentation

We recommend to investigate into and eventually use progamming productivity tools like LEX (lexical analyzer generator), YACC (parser generator), and AWK (pattern scanning and processing language).

The constraints are that the program must be developed in the UNIX V environment, using the ANSI C language."

1. INTRODUCTION TO THE JUR TEXTBASE

JUR is a Status textbase used by jurists to defend the council during lawsuits, or to give juridical advice. It depends from the juridical documentation service, which transmits to the JUR administrator the documents to be entered.

The project started internally; at first, a PC was used to store the documents, and, two years ago, credits were accorded to support further development. The JUR textbase was born.

In March '90, around 1500 documents were introduced in the textbase. This number is growing slowly, as only newer documents are added. All documents concerning the first, second and fourth chapters are inserted, but only a selection of chapter 3's confidential documents are allowed to be entered in the textbase.

The users of JUR are not numerous: only 3 jurists use the textbase, and only one of them is interested by the fourth chapter. They find the hardware to be of insufficient quality,

but are rather satisfied by the software. In fact, no interface has been defined, and the users have only the raw Status instructions to make their queries. But they don't complain, and find JUR nicer than CELEX, an other juridical database, centralized in the Commission.

JUR is divided in four chapters, each containing a different type of documents. In chapters 1, 2, and 3, you can find the J, K, and N documents, all produced by the Council, and in chapter 4 are stored the anti-dumping legislation of all countries of the world. N documents are a bit special: they are confidential internal notes, that's why not all of them are put in the textbase. In appendix B1, you will find a typical J document, ready to enter the textbase.

2. SPECIFICATIONS OF THE DVP

The input, or document, is basically a series of successive chapters, which are themselves a series of articles. The syntax of an article involves two levels. On a higher level, we can consider the article to validate as a series of sections. For a particular article, all the valid sections must appear once and only once (except for the optional CONCLUSION section), and in the order of appearance described below. On a lower level, there is the syntax of the CONTENTS of each named section.

The Document Validation Program will check if the syntax is respected by a specified document (input). Every time a mismatch is detected, a line will be written in a report file. The error and warning messages are given next to the related situation or constraint.

2.1. Article-level syntax check

If the article is not correctly designed, three error messages can be displayed, depending on the situation :

```
'Erreur'  numdoc  marker  'Marqueur incorrect.'
'Erreur'  numdoc  marker  'Marqueur manquant.'
'Erreur'  numdoc  marker  'Marqueur déjà présent.'
```

2.1.1. J Documents (chapter 1)

```
$$T
$$N NUMDOC
$$N DATE
$$N COTE
$$N CODE
$$T
$$N TITRE
$$N REFERENCES
$$N TEXTE
$$N CONCLUSION (Optional)
$$A
```

2.1.2. K Documents (chapter 2)

```
$$T
$$N NUMDOC
$$N DATE
$$N COTE
$$T
$$N TITRE
$$N REFERENCES
$$N TEXTE
$$N CONCLUSION (Optional)
$$A
```

2.1.3. N Documents (chapter 3)

```
$$T
$$N NUMDOC
$$N DATE
$$N COTE
$$N LANGUE
$$T
$$N TITRE
$$N REFERENCES
$$N TEXTE
$$N CONCLUSION (Optional)
$$A
```

2.1.4. Antidumping documents (chapter 4)

```
$$T
$$N DOCNUM
$$N DAT
$$N PUB_REF
$$N TYPE
$$N PRODUCT
$$N COUNTRY
$$T
$$N KEYWORDS
$$A
```

2.2. Syntactic check of the CONTENTS of the named sections

In the following description, n will stand for any single digit (0..9), l for any letter, and dd/mm/yy for any date coded in 6 digits (2 for the day, 2 for the month, and 2 for the year). DATE.yy is the year part of DATE.

DATE

Format : dd/mm/yy
If unvalid format, error message :
'Erreur' numdoc date 'Format de date incorrect.'
Valid date
Less than today's date
If unvalid date, error message :
'Erreur' numdoc date 'Date invalide.'
If more than 1 year old, warning message :
'Avertissement' numdoc date 'Date antérieure de plus d'un an à la date du jour.'

COTE

Format : J and K documents : nnnnn_yy
(check first 8 char only)
where COTE.yy = DATE.yy
or COTE.yy = DATE.yy - 1
N documents : NOTEINT (or NOTEINTn if several N documents exist for the same date)

If unvalid format, error message :
'Erreur' numdoc cote 'Format de cote incorrect.'
If unvalid yy, warning message :
'Avertissement' numdoc cote date 'Année de la cote antérieure de plus d'un an à l'année du document.'

CODE

Format : nnn
If unvalid format, error message :
'Erreur' numdoc code 'Format de code incorrect.'

LANGUE

Format : l or ll
If unvalid format, error message :
'Erreur' numdoc langue 'Format de langue incorrect.'
Correct code in corresponding LANGUE file
If no correspondance in LANGUE file, error message :
'Erreur' numdoc langue 'Pas de correspondance dans le fichier LANGUE.'

TITRE

Free text (at least one letter)
If empty section, error message :
'Erreur' numdoc 'Section titre vide.'

REFERENCES

Each in a different paragraph ($$P separator)
A reference consists of several descriptors each on a different line and incrementally indented.
If invalid format, error message :
'Erreur' numdoc references 'Format de references incorrect.'
Each reference must exist as a single record in the corresponding REFERENCES file.
If no correspondance is found in REFERENCES file, warning message :
'Avertissement' numdoc references 'Pas de correspondance dans le fichier REFERENCES.'

TEXTE

Free text (at least one letter)
If empty section, error message :
'Erreur' numdoc 'Section texte vide.'

NUMDOC

In input, must be J, K or N.
If unvalid format, error message :
'Erreur' numdoc 'Format de numdoc incorrect.'
This named section in the output must be constructed from the input file, as follows :

J documents :
J_yymmdd_yynnnnn_nnn (date/cote/code)

K documents :
K_yymmdd_yynnnnn (date/cote)

N documents :
N_yymmdd_NOTEINTn (date/cote/optional n)

DOCNUM

Format : 3nnRnnnn or 3nnDnnnn
If unvalid format, error message :
'Erreur' docnum 'Format de docnum incorrect.'

DAT

Format : dd/mm/yy
If unvalid format, error message :
'Erreur' docnum dat 'Format de dat incorrect.'
Valid date
Less than today's date
If unvalid date, error message :
'Erreur' docnum dat 'Date invalide.'
If more than 1 year old, warning message :
'Avertissement' docnum dat 'Date antérieure de plus d'un an à la date du jour.'

PUB_REF

Format : OJ L nnnn, dd/mm/yy, P.nnnn
where dd/mm/yy is a valid date.
If unvalid format, error message :
'Erreur' docnum pub_ref 'Format de pub_ref incorrect.'

TYPE

Each on a new line of the same paragraph
If invalid format, error message :
'Erreur' docnum type 'Format de type incorrect.'
Correct code existing in corresponding TYPE file
If no correspondance is found in TYPE file, warning message :
'Avertissement' numdoc type 'Pas de correspondance dans le fichier TYPE.'

PRODUCT

Free text (at least one letter)
If empty section, error message :
'Erreur' docnum 'Section product vide.'

COUNTRY

On one line, separated from each other by one space
Format : any number of strings of 1 to 10 characters (any but space and carriage return) separated by one space and terminated by a carriage return.
If unvalid format, error message :
'Erreur' docnum country 'Format de country incorrect.'
Correct code existing in corresponding COUNTRY file
If no correspondance in COUNTRY file, error message :
'Erreur' docnum country 'Pas de correspondance dans le fichier COUNTRY.'

KEYWORDS

Each in a different paragraph
Format : Wnnn nn_nn
If unvalid format, error message :
'Erreur' docnum keywords 'Format de keywords incorrect.'
The nn_nn part of each keyword should be found in the corresponding KEYWORDS file.
If the nn_nn code is found in the KEYWORDS file, the corresponding full keyword must be inserted in the input (next to the corresponding code).
If the nn_nn code is not found in the KEYWORDS file, an error message should be printed :
'Erreur' docnum code 'Pas de correspondance dans le fichier KEYWORDS.'
The paragraphs containing keywords should be sorted ascendingly on the first four characters (Wnnn).
If the paragraphs appear in an unsorted order, a warning message should be given :
'Avertissement' docnum keywords 'Les keywords ne sont pas triés par ordre croissant.'

Apart from the validation, the user asked for a summary. At the end of the report file, the number of each kind of markers will be added : one line by marker, with first the name of the marker, and then the number of times it has been detected in the input. $$P and $$# will not be present in this list. Example of a line :

```
$$N COTE        présent     18   fois.
```

3. PROBLEMS ENCOUNTERED DURING IMPLEMENTATION

Besides small but bothering problems like the character-set incompatibility, illustrated in appendix B1, I was faced with problems which were difficult to overcome. Some dealt with the lexical analyzer, LEX, some with the parser, YACC, and a lot with error handling.

3.1. Lexical analyzer

A lexical analyzer is a program which scans a file and returns a different value, depending on the character pattern found. LEX is a tool which helps writing lexical analyzers; the character patterns are called tokens.

The lexical analyzer's specifications must be written in LEX format. Then LEX converts these specifications into a function written in C that has the same effects. The main problem with LEX is that it is impossible to make it work in a procedural way: "if you find this pattern, then do this, else do that". LEX works differently: "this is the next encountered token". And you have to handle all special cases by yourself at a higher level.

Finally, I decided to include in LEX only the date validations, and the marker counts. In appendix C1, you will find the LEX source, and in appendix C2, the associated C function.

There are some bugs in LEX; in particular, it is not always true that the longest token is returned. For example, if you define a word to be any series of alphanumeric characters, and an identifier to be a series of up to eight alphanumeric characters, LEX will sometimes give you the token identifier (value: "suchawor") when encountering "suchaword", and sometimes it will return the word token on "identity", even if identifier had been declared first. Bugs like that are almost impossible to detect in the C program produced by LEX.

3.2. Parser

A parser is a program which scans a file and checks if it respects a predefined structure or syntax. YACC (Yet Another Compilers' Compiler) is a tool which helps writing parsers. The syntax is defined in terms of exact strings or in terms of tokens returned by a lexical analyser. LEX and YACC are two closely related tools for compilers' design.

Two problems arose when writing the YACC program: on one hand, the interface with LEX turned out to be far from perfect, and on the other hand, the debugging was very difficult. The biggest problem of all, error handling, will be discussed in the third chapter.

As I mentioned before, LEX rules are written a bit like inference rules. YACC works completely differently: you specify the expected order of tokens, and, if it is not respected, a syntax error occurs.

Suppose you have the following YACC rules:

```
date :      DATE
            anything
            NEWLINE
            ;
```

```
anything :  /* empty */
            |
            anything
            WORD
            |
            anything
            NEWLINE
            ;
```

where DATE, NEWLINE, and WORD are LEX tokens respectively made of 6 digits, a carriage return, and any number of alphanumeric characters. If the input line is "880808a", LEX will return a single token: WORD, and a syntax error will be detected, where there should be none. The complete YACC rules of the DVP can be found in appendix C3.

The debugging facility was awkward to use: you have to go in the C program generated by YACC (see appendix C4), which is anything but readable, and change it, in order to have very rudimentary debugging. A file which is supposed to help the programmer can be generated; part of it is in appendix C5. I only put a few sheets, because the whole file is more than 50 pages long!

3.3. Error handling

Two difficult problems are to be distinguished: error detection is the ability of a parser to detect an error and its location, and error recovery is the process of finding a good place to start parsing again after an error has been detected. First, we'll speak about error detection.

3.3.1. Error detection

If no error rule has been specified, the only message uttered by a YACC parser when bumping into an error is "syntax error". Moreover, no further parsing is accomplished. It is possible to trap the errors by using the "error" token. When an error is encountered, the parser acts just like if it had received from its lexical analyzer the predefined token "error".

Using this token in rules can help you in tracking the errors, and give them appropriate actions. But it gives you no idea of where the error occured. So you have to add a section number variable (globnumsect), which indicates what is the section number to be parsed, and which will be used in the error handling procedure.

The kind of error which occured is stored in the variable "globnumerr"; it is updated every time an error cannot occur anymore, and it takes the value assigned to the next possible error. This is true only for format errors; all other errors do not give an error for the parser: the fact that a reference is not present in the corresponding file is not an error for YACC, it is a voluntary action taken by the program upon detection of any unvalid reference.

3.3.2. Error recovery

An even more intricate problem is that of knowing where to go in the input when an error has been found. I have no satisfying answer to that question, and I am convinced there is no elegant way to handle this with LEX and YACC, at least in the specific context of text structure checking. Let's suppose you have the following inputs:

```
Input 1

$$#
$$T
$$N NUMDOC
$$N NUMDOC
J_910315_9105263_028
...


Input 2

$$#
$$T
$$N NUMDOC
$$N DATE
15/03/91
...
```

The ideal solution in the first case would be to display a "Marker déjà présent" message, and discard the second "$$N NUMDOC" marker, while in the second case, the appropriate actions include a "Format de numdoc incorrect" error message and the restart of the parsing with "$$N DATE". These two methods work only for their respective input; applying a method to the other input would be a disaster.

4. CONCLUSION

It has been very interesting to experiment with techniques and tools used in the compilers' design context. But LEX and YACC suffer from their lack of documentation and poor error handling. I have realized, during this project, how unknown technologies (new environment, new programming language, etc...) can dramatically increase a project's difficulty.

R E F E R E N C E S

[ALLE89] DENIS ALLEN, Text retrieval with a twist, Byte July 1989

[BLAI91] DAVID C. BLAIR, MICHAEL D. GORDON, The management and control of written information, information and management, vol. 20, n° 4, April 1991

[BURK89] JAMES. J. BURKE, BOB RYAN, Gigabytes on line, Byte, October 1989

[CAMP87] JAN CAMPBELL-GRANT, Introducing ODA, ICL technical journal, November 1987

[CAMP88] B. CAMPBELL, J.M. GOODMAN, HAM : A general purpose hypertext abstract machine, Communications of the ACM, vol. 31, (7), 1988

[DANI90] M-C. DANIEL-VATONNE, Hypertextes : des principes communs et des variations, TSI, numéro spécial : Les hypertextes, vol. 9, (6), 1990

[DIEU91] ALAIN DIEUDONNE, JEAN MICHALSKI, EVA-MARIE ROBERTFROID, Introduction aux hypertextes, Institut d'informatique de Namur, 1991

[DULU86] JEFFREY R. DULUDE, The application interface of optical drives, May 1986

[FALO85] CHRISTOS FALOUSTOS, Access methods for text, Computing surveys, vol. 17, n° 1, March 1985

[FIDE88] J. FIDERIO, A grand vision, Byte, October 1988

[FISH89] MARSHA J. FISHER, Digging out with image technology, Datamation, April 15th, 1989

[FRIS88] MARK FRISSE, From text to hypertext, Byte, October 1988

[FURU90] R. FURUTA, P. SCOTTS, Generalising hypertext : domains of the trellis model, TSI, Numéro spécial : les hypertextes, vol. 9, (6), 1990

[GIGU89] ERIC GIGUERE, Electronic Oxford, Byte, December 1989

[GLAS89] L. BRETT GLASS, Digital video interactive, Byte, May 1989

[HARV90] DAVID A. HARVEY, State of the media, Byte, November 1990

[HARV91a] DAVID A. HARVEY, Catch the wave of DIP, Byte, April 1991

[HARV91b] DAVID A. HARVEY, BOB RYAN, Practically paperless, Byte, April 1991

[HEND90a] TONY HENDLEY, The developping field of information and records management, OIS International, 1990

[HEND90b] TONY HENDLEY, The technical introduction to document image processing, OIS International 1990

[HOUG89] DEAN HOUGH, The paperless office, Byte, July 1989

[KIMB88] ROY E. KIMBRELL, Searching for text ? Send an N-Gram !, Byte, May 1988

[KINN91] PAUL KINNUCEN, Imaging speeds forms processing, Datamation, May 1st, 1991

[LAHT90] WALTER LAHTI, DEAN MCCARRON, Store data in a flash, Byte, November 1990

[LAUB86] LEONARD LAUB, The evolution of mass storage, Byte, May 1986

[LION90] KARINA LION, DAT's a solution, Byte, November 1990

[LIPP90] ROB LIPPINCOTT, Beyond hype, Byte, February 1990

[LOCK91] CHROSTOPHER LOCKE, The dark side of DIP, Byte, April 1991

[MACL91] I.A. MACLEOD, A query language for retrieving information from hierarchic text structures, The computer journal, vol. 34, n° 3, 1991

[MANN91] JANET MANN, Software to manage the paper mountain, Datamation, July 15th, 1991

[MARS91] R. MARSHALL, Manipulating full-texts scientific databases : logic-based semantico-pragmatic approach, The computer journal, vol. 34, n° 3, 1991

[MART91] T. PATRICK MARTIN, JUDY I. RUSSELL, Data cachins strategies for distributed full-text retrieval system, Information systems, vol. 16, n° 1, 1991

[MCCU91] TOM MCCUSKER, Production power !, Datamation, February 15th 1991

[MCMU91] JOHN MCMULLEN, Rewritable still not in overdrive, Datamation, March 1st, 1991

[MICH91] GERALD P. MICHALSKI, The world of documents, Byte, April 1991

[NELS88] THEODOR H. NELSON, Managing immense storage, Byte, January 1988

[NIEL88] J. NIELSEN, The art of navigating through hypertext, Communications of the ACM, vol. 33, (3), 1990

[ODOC91] M.H. O'DOCHERTY, C.N. DASKALAKIS, Multimedia information system - The management and semantic retrieval of all electronic data types, The computer journal, vol. 34, n° 3, 1991

[PARI90] TOM PARISH, Crystal clear storage, Byte, November 1990

[PETE91] R.A. PETERS, Giga-storage, Byte, May 1991

[POUN89] D. POUNTAIN, Digital paper, Byte, February 1989

[RAYM88] D.R. RAYMOND, F.W. TAMPA, Hypertext and the Oxford English Dictionary, Communications of the ACM, vol. 31, (7), 1988

[RICH90] G. RICHARD, A. RIZK, Quelques idées pour une modélisation des systèmes hypertextes, TSI, Numéro spécial : les hypertextes, vol. 9, (6), 1990

[RING87] G. RINGLAND, Standards and office information, ICL Technical journal, November 1987

[ROCH91] J.B. ROCHESTER, D.P. DOUGLASS, The emerging world of multimedia, I/S Analyser, vol. 29, n° 3, March 1991

[ROBI89] P. ROBINSON, Easy reading, Byte, May 1989

[ROBI90] P. ROBINSON, The four multimedia gospels, Byte, February 1990

[RYAN90a] BOB RYAN, Entering a new phase, Byte, November 1990

[RYAN90b] BOB RYAN, The once and future king, Byte, November 1990

[RYAN91] BOB RYAN, The data swamp, Byte, May 1991

[SHER89] I. SHERR, Pepperoni and paperwork, Byte, December 1989

[SHET90] TIM SHETLER, Birth of the BLOB, Byte, February 1990

[SMIT88] J.B. SMITH, S.F. WEISS, Hypertext, Communication of the ACM, vol. 31, (7), 1988

[STAM90] DAVID STAMPS, The challenge of integration, Datamation, July 15th, 1990

[STEI91] R.M. STEIN, Browsing through terabytes, Byte, May 1991

[TAPE91] D. TAPELLINI, How imaging can change your business, Datamation, April 1st, 1991

[TOPE91] TOM TOPERCZER, From pyramids to peers, Byte, May 1991

[WILL89] TOM WILLIAMS, Optical storage inches toward standards, Computer design, October 1st, 1989

[ZOEL86] BILL ZOELLICK, CD-ROM software development, Byte, May 1986

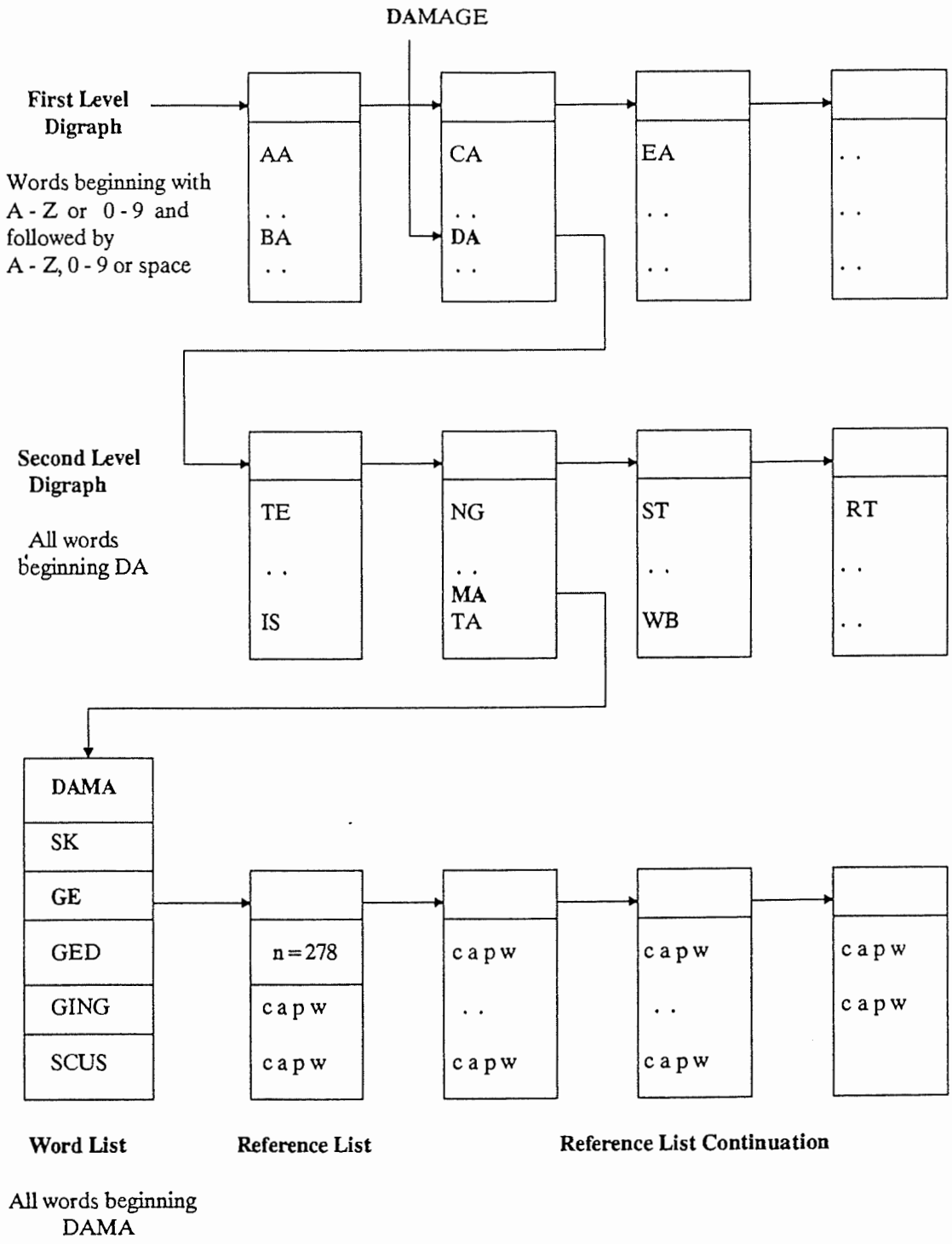
Appendix A

Drawings

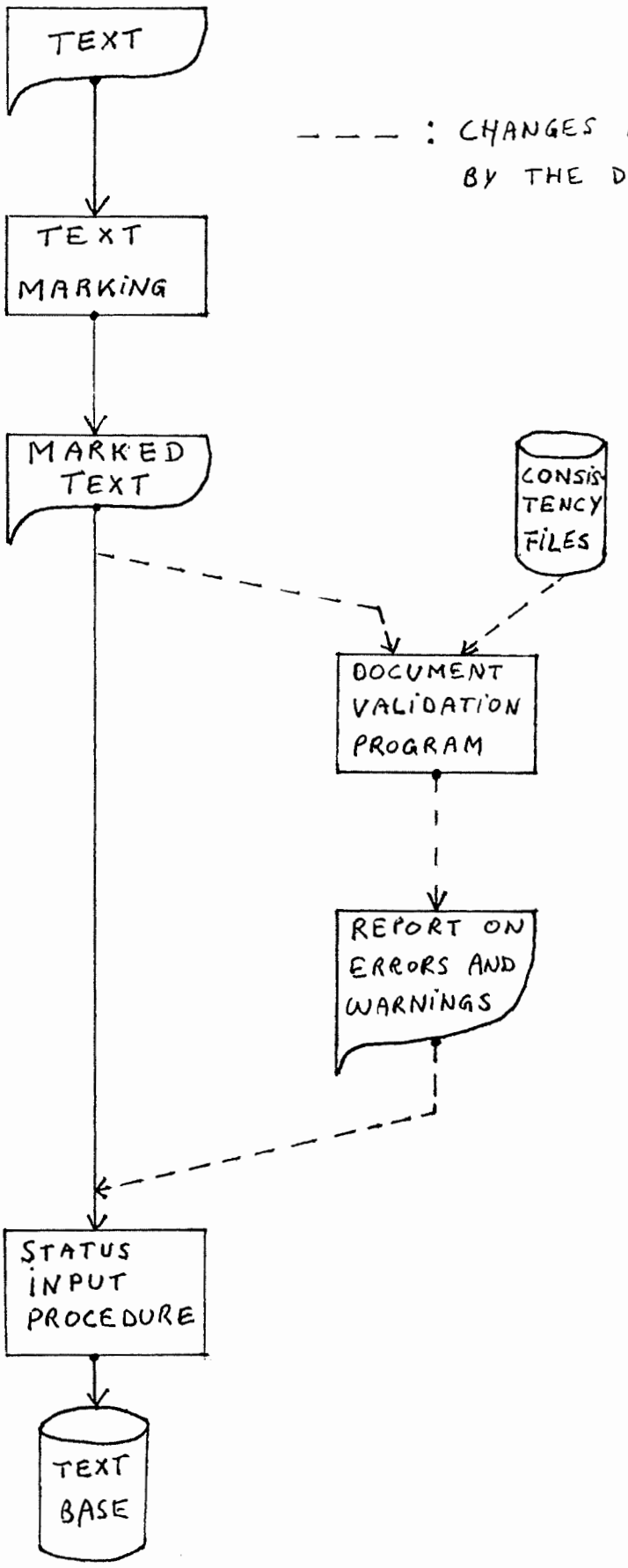
A1 : Accessing the references to DAMAGE

A2 : Document flowchart

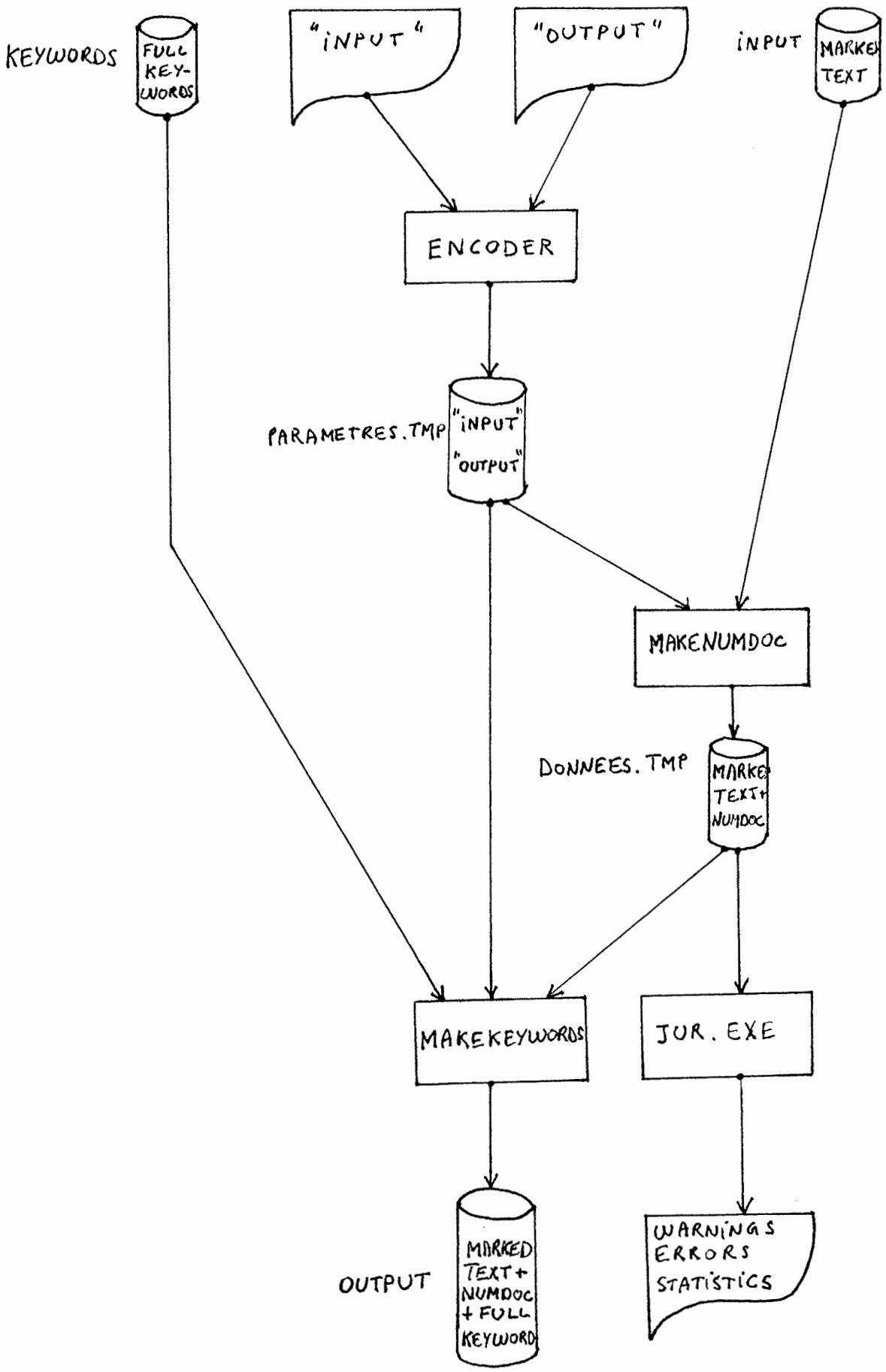
A3 : Program chain of the DVP



Accessing the References to DAMAGE


TEXT
TEXT MARKING
MARKED TEXT
CONSIS TENCY FILES
DOCUMENT VALIDATION PROGRAM
REPORT ON ERRORS AND WARNINGS
STATUS INPUT PROCEDURE
TEXT BASE
- - - : CHANGES INDUCED BY THE DVP.

A2


KEYWORDS
FULL KEY-WORDS
"INPUT"
"OUTPUT"
INPUT
MARKED TEXT
ENCODER
PARAMETRES.TMP
"INPUT"
"OUTPUT"
MAKENUMDOC
DONNEES.TMP
MARKED TEXT+ NUMDOC
MAKEKEYWORDS
JUR.EXE
OUTPUT
MARKED TEXT+ NUMDOC + FULL KEYWORD
WARNINGS
ERRORS
STATISTICS

A3

Appendix B

Texts

B1: Example of a JUR input document

B2: List of the JUR sections

$$T
$$N NUMDOC
J_910315_9105263_028
$$N DATE
15/03/91
$$N COTE
05263_91
$$N CODE
028
$$T
$$N TITRE
Interpr<tation de l'article 116 du trait< CEE]
a)ÛChamp d'application de cette disposition par rapport ` l'articleÛ113 du trait<]
b)ÛContenu de l'action commune y pr<vue.
$$N REFERENCES
CEE_005]
    ETATS MEMBRES]
        OBLIGATIONS]
$$P
CEE_113]
    POLITIQUE COMMERCIALE]
$$P
CEE_114]
$$P
CEE_116]
    ETATS MEMBRES]
        ACTION COMMUNE]
$$P
DROIT_A]
    ACTES]
        SIGNATURE]
$$P
DROIT_A]
    COMPETENCES]
        ETATS MEMBRES]
            COMMUNAUTE]
                REPARTITION
$$N TEXTE
1. Lors de la r<union du 6Ûd<cembreÛ1990, le Pr<sident du groupe "Produits de base" a demand< au Service juridique du Conseil de rendre un avis sur certaines questions concernant l'interpr<tation de l'article 116 du trait< CEE, ` la lumi>re d'une note pr<sent<e par une d<l<gation ` ce sujet. Le repr<sentant du Service juridique a donn< oralement cet avis lors de la r<union du groupe du 7 marsÛ1991. Suite ` son intervention, le Pr<sident du groupe a demand< au Service juridique de mettre son avis par <crit.]
$$P
2. Les questions pos<es peuvent se r<sumer ainsi :]
$$P

A) Quel est le champ d'application de l'article 116 par rapport ` l'articleÛ113Û? Notamment, l'articleÛ116 peut-il s'appliquer aux domaines relevant de la politique commerciale commune (articleÛ113) ou aux domaines relevant de la comp<tence des Etats membresÛ?]
$$P
B) Quel peut §tre le contenu de l'action commune pr<vue par l'article 116Û? Notamment, l'article 116 peut-il §tre utilis< pour obliger les Etats membres ` prendre un engagement internationalÛ? Un corollaire de cette question est celle de savoir si l'on peut obliger les Etats membres, sur la base de cette disposition, ` signer ou ` notifier simultan<ment un accord.]
$$P
A) Champ d'application de l'articleÛ116 du trait< CEE par rapport ` l'articleÛ113]
$$P
3. A titre pr<liminaire, le Service juridique souligne que, l'articleÛ113 <tablissant une comp<tence exclusive de la Communaut<, les cas dans lesquels la Communaut< d<tiendrait une comp<tence virtuelle ne font pas l'objet du pr<sent avis.]
$$P
4. Selon l'article 116, premier alin<a :]
$$P
"Pour toutes les questions qui rev§tent un int<r§t particulier pour le march< commun, les Etats membres ne m>nent plus, ` partir de la fin de la p<riode de transition, qu'une action commune dans le cadre des organisations internationales de caract>re <conomique. A cet effet, la Commission soumet au Conseil, qui statue ` la majorit< qualifi<e, des propositions relatives ` la port<e et ` la mise en oeuvre de cette action commune".]
$$P
Il d<coule de cette disposition que son application est limit<e aux domaines relevant de la comp<tence des Etats membres. C'est <videmment seulement dans les cas o{ ils d<tiennent une comp<tence que les Etats membres peuvent mener une action et donc que l'article 116 peut recevoir application.]
$$P
5. Ainsi, le Service juridique estime que dans les cas o{ il existe une comp<tence exclusive communautaire, c'est ` la Communaut< d'exercer sa comp<tence. Les Etats membres ne disposent plus de comp<tences et ne peuvent donc plus mener une action commune au sens de l'article 116 CEE.]
$$P
Une comp<tence communautaire exclusive dans le domaine des relations ext<rieures existe]
$$P
i)Ûdans les cas express<ment pr<vus par le trait<, ce qui est le cas notamment pour l'article 113 CEE concernant la politique commerciale communeÛ;]
$$P
ii)Ûen application de la jurisprudence de la Cour de Justice dans l'affaire AETR selon laquelle la Communaut< est comp<tente pour conclure tous les accords internationaux qui affecteraient des r>gles internes instaur<es par la Communaut<.]
$$P

6. Dans son avisÛ1/78, la Cour de justice, se r<f<rant ` son avisÛ1/75, a consid<r<, en ce qui concerne la d<limitation du champ d'application respectif des articlesÛ113 et 114, d'une part, et 116, d'autre part, que "ce qui compte, au regard de l'application du trait<, est la question de savoir si une n<gociation entreprise dans le cadre d'une organisation internationale est destin<e ` aboutir ` un "engagement pris par des sujets de droit international et ayant une force obligatoire". Dans un tel cas, ce sont les dispositions du trait< relatives ` la n<gociation et ` la conclusion d'accords, en d'autres termes, les articles 113, 114 et 228, qui sont d'application, et non l'article 116" (attenduÛ51).]
$$P
Il d<coule de ce consid<rant que dans le cas d'une n<gociation pour laquelle une comp<tence communautaire exclusive existe, tel que dans le cas de la politique commerciale commune, c'est la Communaut< qui doit exercer sa comp<tence et elle ne peut pas la faire exercer par les Etats membres, sur la base de l'articleÛ116.]
$$P
7. Au cas o{ la Communaut< ne serait pas membre de l'organisation en cause, les Etats membres devraient agir, au titre non pas de leurs propres comp<tences, mais des comp<tences communautaires. Dans un tel cas, il ne s'agirait pas pour le Conseil de pr<voir une action commune ` mener par les Etats membres qui exerceraient ` cet effet leurs propres comp<tences (ces comp<tences, par d<finition, n'existent plus) mais d'organiser l'exercice des comp<tences communautaires par les Etats membres qui, selon la jurisprudence de la Cour, sont tenus dans un tel cas d'agir dans l'int<r§t et pour le compte de la Communaut<. Cela devrait se faire sur la base des dispositions qui donnent la comp<tence exclusive ` la Communaut<, et non sur la base de l'article 116.]
$$P
En cons<quence, du fait que la politique commerciale commune rel>ve de la comp<tence exclusive de la Communaut<, dans le cas o{ il serait impossible ` cette derni>re d'exercer sa comp<tence dans le cadre d'une organisation internationale ` caract>re <conomique, ce sont les Etats membres qui exerceraient les comp<tences communautaires au nom de la Communaut<. Cet exercice rel>verait des articles 113 et 114 et non de l'article 116.]
$$P
8. Le champ d'application de l'article 116 est donc limit< aux questions qui ne rel>vent pas de la comp<tence de la Communaut<, mais qui rev§tent n<anmoins un int<r§t particulier pour elle. Selon le Service juridique, l'applicabilit< de l'articleÛ116, qui sp<cifie, pour les questions soulev<es dans le cadre d'une organisation internationale de caract>re <conomique, l'obligation g<n<rale <dict<e par l'articleÛ5 du trait<, n'est pas limit<e ` certains domaines du trait< mais peut concerner tous les domaines du trait<, pour autant qu'une comp<tence des Etats membres subsiste. La jurisprudence de la Cour elle-m§me fait r<f<rence ` l'article 116 dans les domaines du transport (affaire AETR att.Û76) et de la p§che (affaire Kramer, att.Û43).]
$$P
B)ÛÛContenu de l'action commune]
$$P

9. Selon la premi>re phrase de l'article 116, premier paragrapheÛ: "Pour toutes les questions qui rev§tent un int<r§t particulier pour le march< commun, les Etats membres ne m>nent plus, ` partir de la fin de la p<riode de transition, qu'une action commune dans le cadre des organisations internationales de caract>re <conomique".]
$$P
Cette phrase constitue ` elle seule une obligation pour les Etats membres, obligation qui pr<cise celle d<coulant des dispositions de l'article 5 du trait< CEE. Cette action commune est obligatoire, m§me si la Commission ne soumet pas, ou que le Conseil n'adopte pas, des propositions relatives ` la port<e et ` la mise en oeuvre de cette action commune.]
$$P
10. En outre, selon la deuxi>me phrase du premier paragraphe de l'article 116, la port<e et la mise en oeuvre de l'action commune peuvent §tre, le cas <ch<ant, pr<cis<es. Le contenu d'une "action commune" ne peut pas §tre d<fini in abstracto. Il doit §tre d<termin< cas par cas en tenant compte notamment du cadre international, de l'objet de la n<gociation et de l'int<r§t en jeu pour la Communaut<. A titre d'exemple, le Service juridique estime que, sur la base de l'article 116, le Conseil peut, sur proposition de la Commission, pr<voir que les Etats membres doivent adopter une attitude commune lors d'une n<gociation internationale, ou ne pas prendre un engagement international ou encore, assortir leur acceptation d'un engagement international de certaines conditions.]
$$P
11. Par contre, le Service Juridique estime que le Conseil ne pourrait pas, sur la base de l'articleÛ116, cr<er une obligation pour les Etats membres de contracter des engagements internationaux. En effet, le Conseil ne saurait obliger les Etats membres ` prendre des engagements internationaux dans des domaines qui rel>vent de leurs comp<tences.]
$$P
12. Toutefois, il convient d'examiner si le m§me raisonnement s'applique en ce qui concerne un accord international "mixte", ` savoir un accord qui r>gle des mati>res relevant en partie de la comp<tence de la Communaut< et en partie de celle des Etats membres, de sorte que ni la Communaut< seule, ni les Etats membres seuls, ne peuvent assumer toutes les obligations stipul<es dans l'accord. La conclusion d'un tel accord par la Communaut< n<cessite donc sa conclusion concomitante par les Etats membres et vice versa.]
$$P
Si, dans un tel cas de comp<tence mixte, le Conseil d<cide, soit d'autoriser la Commission ` ouvrir des n<gociations et de lui donner des directives de n<gociation, soit de signer et, enfin, de conclure un accord, ces d<cisions - m§me si elles sont limit<es ` la partie de l'accord relevant de la comp<tence communautaire - peuvent cr<er implicitement l'obligation pour les Etats membres de n<gocier, signer ou conclure ce m§me accord, lorsqu'il s'av>re que cette action des Etats membres est indispensable ` l'ex<cution de l'accord par la Communaut<. En effet, si, de l'avis du Conseil, il est n<cessaire que la Communaut< conclue l'accord en question, les Etats membres sont tenus (art. 5 CEE) de prendre toutes mesures g<n<rales ou particuli>res propres ` assurer l'ex<cution des obligations r<sultant des actes des institutions de la Communaut<.]
$$P

Ces obligations des Etats membres pourraient, de l'avis du Service juridique, §tre explicit<es par des d<cisions concernant ` la fois la Communaut< et les Etats membres, prises sur une double base juridique visant le ou les articles du trait< qui <tablissent la comp<tence communautaire et l'articleÛ116 en ce qui concerne les parties de l'accord relevant de la comp<tence des Etats membres.]
$$P
13. Finalement, et en ce qui concerne la question pos<e de savoir si l'on peut obliger les Etats membres, sur la base de l'articleÛ116, ` signer ou ` notifier simultan<ment un accord, le Service juridique rappelle que l'exigence de simultan<it< quant ` la signature et ` la notification d'un accord est une exigence pr<vue dans le cadre du documentÛ"PROBAÛ20", arrangement qui a un caract>re politique et qui "laisse de cçt< toute consid<ration de caract>re juridico-institutionnel se r<f<rant aux comp<tences respectives de la Communaut< et des Etats membres" (doc. 5887/81 PROBA 20).]
$$P
Du point de vue juridique, il n'est pas n<cessaire que la signature ou la notification d'un accord intervienne simultan<ment de la part de la Communaut< et des Etats membres. Toutefois, rien n'emp§che le Conseil, sur la base de l'article pertinent du trait< pour ce qui est de la comp<tence communautaire et sur la base de l'articleÛ116 pour ce qui est des comp<tences des Etats membres, de pr<voir que la Communaut< et les Etats membres qui ont d<cid< de signer un accord ou de notifier son acceptation, le fassent simultan<ment. Plus concr>tement, dans les domaines couverts par l'article 116, la Commission et le Conseil pourraient bien estimer l'exigence de la simultan<it< comme une modalit< de mise en oeuvre de l'action commune.]
$$P
Dans ce sens, le Service juridique estime qu'il n'y a pas de contradiction entre la non-existence d'une possibilit< d'obliger juridiquement les Etats membres ` prendre un engagement international (mis ` part le cas des accords mixtes) et l'exigence d'une action simultan<e des Etats membres qui auraient d<cid< de prendre cet engagement.
$$N CONCLUSION
14. Le Service juridique estime que]
$$P
a) La premi>re phrase du premier alin<a de l'article 116 contient, pour toutes les questions qui rev§tent un int<r§t particulier pour le march< commun, une obligation pour les Etats membres de mener une action commune dans le cadre des organisations internationales de caract>re <conomique, obligation qui pr<cise celle d<coulant des dispositions de l'article 5 du trait<. Cette action commune est obligatoire, m§me si la Commission ne soumet pas, ou que le Conseil n'adopte pas, des propositions relatives ` la port<e et ` la mise en oeuvre de cette action commune.]
$$P
b) La deuxi>me phrase du premier alin<a de l'article 116 permet au Conseil, statuant ` la majorit< qualifi<e sur proposition de la Commission, d'adopter des actes relatifs ` la port<e et ` la mise en oeuvre de ladite action commune. Selon le Service juridique, cette disposition ne peut pas §tre utilis<e dans les cas o{ il existe une comp<tence communautaire, ` savoir lorsqu'une question rel>ve du pouvoir de d<cision de la Communaut<.]

$$P
c) Mis ` part le cas des accords mixtes, la deuxi>me phrase du premier alin<a de l'article 116 ne peut pas §tre utilis<e pour obliger les Etats membres ` prendre un engagement international. En revanche, elle peut l'§tre pour pr<voir que les Etats membres adoptent une attitude commune lors d'une n<gociation internationale, que les Etats membres ne prennent pas un engagement international ou qu'ils assortissent leur engagement international de certaines conditions.]
$$P
d) Dans le cas des accords mixtes, si le Conseil d<cide, soit d'autoriser la Commission ` n<gocier, soit de signer ou de conclure un accord, cette d<cision peut cr<er implicitement l'obligation pour les Etats membres, soit de n<gocier, soit de signer ou de conclure ce m§me accord, s'il s'av>re que cette action des Etats membres est indispensable ` l'ex<cution de l'accord par la Communaut<. Ces obligations des Etats membres pourraient §tre explicit<es par des d<cisions prises sur la base de l'article 116 pour ce qui concerne les parties de l'accord relevant de la comp<tence des Etats membres.]
$$P
e) Bien que cela ne soit juridiquement pas n<cessaire, il est possible, sur la base de l'article 116, de pr<voir que la Communaut< et les Etats membres qui auraient d<cid< de prendre un engagement international, signeront cet accord ou d<poseront la notification de cet accord simultan<ment.
$$A

JUR is a heterogeneous textbase. In fact, the fourth chapter, about anti-dumping measures, has nothing in common with the others. They have been put in the same textbase only for performance and maintenance needs.

NUMDOC

Chapter 1	J<date><cote><code>
Chapter 2	K<cote><date>
Chapter 3	N<date>

DATE

Date of the original document.

COTE

Number given by the Coordination Générale for all official council documents.

CODE

Refers to the document's juridical subject (code matière coordination).

LANGUE

Language of the original document. Only for N documents.

TITRE

Describes, in natural language, the subject of the document.

REFERENCES

Describes the document by using selected keywords out of an open list of juridical terms. Very often this named section contains the treaty article.

TEXTE

For J and N documents, the full text is given here, while for the K documents, only part of the text is recorded. Note that this section is optional.

CONCLUSION

When formatting the text so that it fits in the JUR textbase, the analyst may find in the original document a part that can be considered as the conclusion. That particular part of text is then inserted in the CONCLUSION section.

DOCNUM

Reference of the document in the CELEX Database. The leading '3' indicates it is an act of Secondary Community law; 'D' and 'R' stand for Directive and Règlement.

DAT

Date of the original document.

PUB_REF

Gives the reference of the Official Journal where the law has been published. 'OJ' stands for Official Journal, 'L' for Legislation, and 'P' for Page. The first four digits indicate the journal number, the next six the publishing date, and the last four the page number inside the journal.

TYPE

Gives the generic type of anti-dumping measure. This is chosen out of a closed list of a few elements.

PRODUCT

Lists the products concerned by the measure. One product by line.

COUNTRY

Lists the countries concerned by the measure. Each country is referred to by a code.

KEYWORDS

W means whereas, and the four last figures identify a code in a corresponding file. This file contains a closed detailed list of anti-dumping measures.

Appendix C

Programs

C1: Input to LEX

C2: Output of LEX

C3: Input to YACC

C4: Output of YACC

C5: Y.OUTPUT, YACC's debugging feature

```
  /* DEFINITION OF SMALL UNITS : INTEGERS OF 1, 2, 3, 4, AND 5 DIGITS */

%{
#include "y.tab.h"
%}

dd      [0-3][0-9]
mm      [0-1][0-9]
yy      [0-9][0-9]

n       [0-9]
nn      [0-9][0-9]
nnn     [0-9][0-9][0-9]
nnnn    [0-9][0-9][0-9][0-9]
nnnnn   [0-9][0-9][0-9][0-9][0-9]

%e 1000
%p 2000
%n 500
%k 1000
%a 2000
%o 2000

%%      /* RULES SECTION */

^"$$#"  {
        ++nbnewchapcount;
        return (NEWCHAP);
        };
^"$$A"  {
        ++nbnewartcount;
        return (NEWART);
        };
^"$$T"  {
        ++nbtitlecount;
        return (TITLE);
        };
^"$$N NUMDOC"   {
        ++nbnumdoccount;
        return (NEWNUMDOC);
        };
^"$$N DATE"     {
        ++nbdatecount;
        return (NEWDATE);
        };
^"$$N COTE"     {
        ++nbcotecount;
        return (NEWCOTE);
        };
^"$$N CODE"     {
        ++nbcodecount;
        return (NEWCODE);
        };
^"$$N TITRE"    {
        ++nbtitrecount;
        return (NEWTITRE);
        };
^"$$N REFERENCES"       {
        ++nbrefcount;
        return (NEWREF);
        };
^"$$N TEXTE"    {
        ++nbtextecount;
        return (NEWTEXTE);
        };
^"$$N CONCLUSION"       {
```

```
        ++nbconclusioncount;
        return (NEWCONC);
        };
^"$$N LANGUE"   {
        ++nblanguecount;
        return (NEWLANG);
        };
^"$$N DOCNUM"   {
        ++nbdocnumcount;
        return (NEWDOCNUM);
        };
^"$$N DAT"      {
        ++nbdatcount;
        return (NEWDAT);
        };
^"$$N PUB_REF"  {
        ++nbpubrefcount;
        return (NEWPUB_REF);
        };
^"$$N TYPE"     {
        ++nbtypecount;
        return (NEWTYPE);
        };
^"$$N PRODUCT"  {
        ++nbproductcount;
        return (NEWPRODUCT);
        };
^"$$N COUNTRY"  {
        ++nbcountrycount;
        return (NEWCOUNTRY);
        };
^"$$N KEYWORDS" {
        ++nbkeywordscount;
        return (NEWKEYW);
        };

^"$$P"  {
        ++nbnewparcount;
        return (NEWPAR);
        };

"$$"    {
        printf ("\nErreur : $$ est reserve pour les marqueurs STATUS.\n")
        };

[ \n]*\n        {
        linenb++;
        return (NEWLINE);
        };

"    "  {
        return (TAB);
        };

" "     {
        return (SP);
        };

[\]\<]  {
        return (LESS);
        };

\$      {
        return (DOLLAR);
        };
```

```
NOTEINT{n}?     {
        return (COTEn);
        };

J_{yy}{mm}{dd}_{yy}{nnnnn}_{nnn}        {
        return (NUMDOCj);
        };
K_{yy}{mm}{dd}_{yy}{nnnnn}      {
        return (NUMDOCk);
        };
N_{yy}{mm}{dd}_NOTEINT{n}?      {
        return (NUMDOCn);
        };

"OJ L "{nnnn}", "{dd}"/"{mm}"/"{yy}", P."{nnnn} {
        int jour, mois, annee;
        char j[3], m[3], a[3];

        j[0] = yytext[11];
        j[1] = yytext[12];
        j[2] = '\0';
        m[0] = yytext[14];
        m[1] = yytext[15];
        m[2] = '\0';
        a[0] = yytext[17];
        a[1] = yytext[18];
        a[2] = '\0';
        jour = atoi (j);
        mois = atoi (m);
        annee = atoi (a);
        if ((jour < 1) || (jour > 31)) {erreur (12, 2); return (PUBREF);}
        if ((mois < 1) || (mois > 12)) {erreur (12, 2); return (PUBREF);}
        switch (mois) {
                case 2:
                        if (annee%4)
                                if (jour > 29) erreur (12, 2);
                                else;
                        else    if (jour > 28) erreur (12, 2);
                        break;
                case 4:
                case 6:
                case 9:
                case 11:
                        if (jour > 30) erreur (12, 2);
                        break;
        }
        return (PUBREF);
        };

3{nn}[RD]{nnnn} {
        return (DOCNUM);
        };

{nnnnn}_{yy}.*  {
        char ac[3], ad[3];
        int ancote, andate;

        ac[0] = yytext[6];
        ac[1] = yytext[7];
        ac[2] = '\0';
        ad[0] = numdoccour[2];
        ad[1] = numdoccour[3];
        ad[2] = '\0';
        ancote = atoi (ac);
        andate = atoi (ad);
        if (!((ancote == andate) || (ancote == andate - 1))) erreur (3, 1
```

```
        return (COTEjk);
        };

W{nnn}  {
        return (KEYW1);
        };

{nn}_{nn}       {
        return (KEYW2);
        };

{nnn}   {
        return (CODE);
        };

{dd}"/"{mm}"/"{yy}      {
        #include <time.h>

        int jour, mois, annee, dayoftoday, monoftoday, yeaoftoday;
        char j[3], m[3], a[3];
        time_t tnow;
        struct tm *tmnow;

        j[0] = yytext[0];
        j[1] = yytext[1];
        j[2] = '\0';
        m[0] = yytext[3];
        m[1] = yytext[4];
        m[2] = '\0';
        a[0] = yytext[6];
        a[1] = yytext[7];
        a[2] = '\0';
        jour = atoi (j);
        mois = atoi (m);
        annee = atoi (a);
        time (&tnow);
        tmnow = localtime (&tnow);
        dayoftoday = tmnow->tm_mday;
        monoftoday = tmnow->tm_mon + 1;
        yeaoftoday = tmnow->tm_year;
        if ((jour < 1) || (jour > 31)) {erreur (2, 2); return (DATE);}
        if ((mois < 1) || (mois > 12)) {erreur (2, 2); return (DATE);}
        if      (       (annee < yeaoftoday - 1)
                ||      (  (annee == yeaoftoday - 1)
                        && (mois < monoftoday)
                        )
                ||      (  (annee == yeaoftoday - 1)
                        && (mois == monoftoday)
                        && (jour < dayoftoday)
                        )
                )       erreur (2, 11);
        switch (mois) {
                case 2:
                        if ((annee%4) == 0)
                                if (jour > 29)
                                        {
                                        erreur (2, 2);
                                        return (DATE);
                                        }
                                else;
                        else    if (jour > 28)
                                        {
                                        erreur (2, 2);
                                        return (DATE);
                                        }
                                else;
```

```
                        break;
                case 4:
                case 6:
                case 9:
                case 11:
                        if (jour > 30) {erreur (2, 2); return (DATE);}
                        break;
        }
        return (DATE);
        };

[^ \n\t\$\<\]]+ {
        return (WORD);
        };

.       {
        printf ("\nErreur : caractere non admis : ");
        ECHO;
        printf (".\n");
        };
%%
```

```
# include "stdio.h"
# define U(x) x
# define NLSTATE yyprevious=YYNEWLINE
# define BEGIN yybgin = yysvec + 1 +
# define INITIAL 0
# define YYLERR yysvec
# define YYSTATE (yyestate-yysvec-1)
# define YYOPTIM 1
# define YYLMAX 200
# define output(c) putc(c,yyout)
# define input() (((yytchar=yysptr>yysbuf?U(*--yysptr):getc(yyin))==10?(y
# define unput(c) {yytchar= (c);if(yytchar=='\n')yylineno--;*yysptr++=yyt
# define yymore() (yymorfg=1)
# define ECHO fprintf(yyout, "%s",yytext)
# define REJECT { nstr = yyreject(); goto yyfussy;}
int yyleng; extern char yytext[];
int yymorfg;
extern char *yysptr, yysbuf[];
int yytchar;
FILE *yyin = {stdin}, *yyout = {stdout};
extern int yylineno;
struct yysvf {
        struct yywork *yystoff;
        struct yysvf *yyother;
        int *yystops;};
struct yysvf *yyestate;
extern struct yysvf yysvec[], *yybgin;
  /* DEFINITION OF SMALL UNITS : INTEGERS OF 1, 2, 3, 4, AND 5 DIGITS */
#include "y.tab.h"
# define YYNEWLINE 10
yylex(){
int nstr; extern int yyprevious;
while((nstr = yylook()) >= 0)
yyfussy: switch(nstr){
case 0:
if(yywrap()) return(0); break;
case 1:
{
        ++nbnewchapcount;
        return (NEWCHAP);
        }
break;
case 2:
{
        ++nbnewartcount;
        return (NEWART);
        }
break;
case 3:
{
        ++nbtitlecount;
        return (TITLE);
        }
break;
case 4:
{
        ++nbnumdoccount;
        return (NEWNUMDOC);
        }
break;
case 5:
{
        ++nbdatecount;
        return (NEWDATE);
        }
break;
```

```
case 6:
{
        ++nbcotecount;
        return (NEWCOTE);
        }
break;
case 7:
{
        ++nbcodecount;
        return (NEWCODE);
        }
break;
case 8:
{
        ++nbtitrecount;
        return (NEWTITRE);
        }
break;
case 9:
{
        ++nbrefcount;
        return (NEWREF);
        }
break;
case 10:
{
        ++nbtextecount;
        return (NEWTEXTE);
        }
break;
case 11:
{
        ++nbconclusioncount;
        return (NEWCONC);
        }
break;
case 12:
{
        ++nblanguecount;
        return (NEWLANG);
        }
break;
case 13:
{
        ++nbdocnumcount;
        return (NEWDOCNUM);
        }
break;
case 14:
{
        ++nbdatcount;
        return (NEWDAT);
        }
break;
case 15:
{
        ++nbpubrefcount;
        return (NEWPUB_REF);
        }
break;
case 16:
{
        ++nbtypecount;
        return (NEWTYPE);
        }
break;
```

```
case 17:
{
        ++nbproductcount;
        return (NEWPRODUCT);
        }
break;
case 18:
{
        ++nbcountrycount;
        return (NEWCOUNTRY);
        }
break;
case 19:
{
        ++nbkeywordscount;
        return (NEWKEYW);
        }
break;
case 20:
{
        ++nbnewparcount;
        return (NEWPAR);
        }
break;
case 21:
{
        printf ("\nErreur : $$ est reserve pour les marqueurs STATUS.\n")
        }
break;
case 22:
{
        linenb++;
        return (NEWLINE);
        }
break;
case 23:
{
        return (TAB);
        }
break;
case 24:
{
        return (SP);
        }
break;
case 25:
{
        return (LESS);
        }
break;
case 26:
{
        return (DOLLAR);
        }
break;
case 27:
{
        return (COTEn);
        }
break;
case 28:
{
        return (NUMDOCj);
        }
break;
case 29:
```

```
{
        return (NUMDOCk);
        }
break;
case 30:
{
        return (NUMDOCn);
        }
break;
case 31:
{
        int jour, mois, annee;
        char j[3], m[3], a[3];

        j[0] = yytext[11];
        j[1] = yytext[12];
        j[2] = '\0';
        m[0] = yytext[14];
        m[1] = yytext[15];
        m[2] = '\0';
        a[0] = yytext[17];
        a[1] = yytext[18];
        a[2] = '\0';
        jour = atoi (j);
        mois = atoi (m);
        annee = atoi (a);
        if ((jour < 1) || (jour > 31)) {erreur (12, 2); return (PUBREF);}
        if ((mois < 1) || (mois > 12)) {erreur (12, 2); return (PUBREF);}
        switch (mois) {
                case 2:
                        if (annee%4)
                                if (jour > 29) erreur (12, 2);
                                else;
                        else    if (jour > 28) erreur (12, 2);
                        break;
                case 4:
                case 6:
                case 9:
                case 11:
                        if (jour > 30) erreur (12, 2);
                        break;
        }
        return (PUBREF);
        }
break;
case 32:
{
        return (DOCNUM);
        }
break;
case 33:
{
        char ac[3], ad[3];
        int ancote, andate;

        ac[0] = yytext[6];
        ac[1] = yytext[7];
        ac[2] = '\0';
        ad[0] = numdoccour[2];
        ad[1] = numdoccour[3];
        ad[2] = '\0';
        ancote = atoi (ac);
        andate = atoi (ad);
        if (!((ancote == andate) || (ancote == andate - 1))) erreur (3, 1
        return (COTEjk);
        }
```

```
break;
case 34:
{
        return (KEYW1);
        }
break;
case 35:
{
        return (KEYW2);
        }
break;
case 36:
{
        return (CODE);
        }
break;
case 37:
{
        #include <time.h>

        int jour, mois, annee, dayoftoday, monoftoday, yeaoftoday;
        char j[3], m[3], a[3];
        time_t tnow;
        struct tm *tmnow;

        j[0] = yytext[0];
        j[1] = yytext[1];
        j[2] = '\0';
        m[0] = yytext[3];
        m[1] = yytext[4];
        m[2] = '\0';
        a[0] = yytext[6];
        a[1] = yytext[7];
        a[2] = '\0';
        jour = atoi (j);
        mois = atoi (m);
        annee = atoi (a);
        time (&tnow);
        tmnow = localtime (&tnow);
        dayoftoday = tmnow->tm_mday;
        monoftoday = tmnow->tm_mon + 1;
        yeaoftoday = tmnow->tm_year;
        if ((jour < 1) || (jour > 31)) {erreur (2, 2); return (DATE);}
        if ((mois < 1) || (mois > 12)) {erreur (2, 2); return (DATE);}
        if      (       (annee < yeaoftoday - 1)
                ||      (  (annee == yeaoftoday - 1)
                        && (mois < monoftoday)
                        )
                ||      (  (annee == yeaoftoday - 1)
                        && (mois == monoftoday)
                        && (jour < dayoftoday)
                        )
                )       erreur (2, 11);
        switch (mois) {
                case 2:
                        if ((annee%4) == 0)
                                if (jour > 29)
                                        {
                                        erreur (2, 2);
                                        return (DATE);
                                        }
                                else;
                        else    if (jour > 28)
                                        {
                                        erreur (2, 2);
                                        return (DATE);
```

```
                                        }
                                else;
                        break;
                case 4:
                case 6:
                case 9:
                case 11:
                        if (jour > 30) {erreur (2, 2); return (DATE);}
                        break;
        }
        return (DATE);
        }
break;
case 38:
{
        return (WORD);
        }
break;
case 39:
{
        printf ("\nErreur : caractere non admis : ");
        ECHO;
        printf (".\n");
        }
break;
case -1:
break;
default:
fprintf(yyout,"bad switch yylook %d",nstr);
} return(0); }
/* end of yylex */
int yyvstop[] = {
0,

38,
39,
0,

39,
0,

22,
0,

24,
39,
0,

26,
39,
0,

38,
39,
0,

38,
39,
0,

38,
39,
0,

25,
39,
```

```
0,

38,
39,
0,

38,
39,
0,

38,
39,
0,

38,
39,
0,

38,
39,
0,

26,
39,
0,

38,
0,

21,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

21,
0,

38,
0,

36,
```

```
38,
0,

38,
0,

36,
38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

1,
0,

2,
0,

20,
0,

3,
0,

23,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

34,
38,
```

```
0,

38,
0,

38,
0,

35,
38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

27,
38,
0,
```

```
38,
0,

14,
0,

37,
38,
0,

33,
38,
0,

32,
38,
0,

38,
0,

38,
0,

27,
38,
0,

38,
0,

7,
0,

6,
0,

5,
0,

16,
0,

33,
0,

38,
0,

38,
0,

38,
0,

10,
0,

8,
0,

38,
0,

38,
```

```
0,

38,
0,

13,
0,

12,
0,

4,
0,

38,
0,

38,
0,

38,
0,

18,
0,

17,
0,

15,
0,

38,
0,

38,
0,

38,
0,

19,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

38,
0,

11,
0,

9,
```

```
0,

38,
0,

38,
0,

38,
0,

38,
0,

29,
38,
0,

30,
38,
0,

38,
0,

30,
38,
0,

38,
0,

38,
0,

28,
38,
0,

31,
0,
0};
# define YYTYPE int
struct yywork { YYTYPE verify, advance; } yycrank[] = {
0,0,    0,0,    1,3,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    1,4,    1,5,
5,5,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    8,0,
8,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    1,6,    5,19,   6,20,
3,18,   1,7,    7,21,   2,17,
17,31,  20,32,  32,48,  45,59,
3,0,    3,0,    8,0,    57,68,
0,0,    1,8,    8,0,    1,8,
1,9,    1,10,   2,9,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    1,11,   8,22,   0,0,
8,22,   0,0,    8,22,   3,0,
72,89,  1,3,    87,110, 3,0,
71,88,  75,93,  8,0,    1,12,
1,13,   2,12,   2,13,   1,14,
1,15,   2,14,   2,15,   3,18,
41,57,  3,18,   69,85,  3,18,
```

```
1,16,   73,90,  2,16,   9,0,
9,0,    12,0,   12,0,   3,0,
10,0,   10,0,   13,0,   13,0,
86,109, 70,86,  88,111, 3,18,
14,0,   14,0,   74,91,  8,0,
31,43,  74,92,  89,112, 90,113,
91,114, 76,94,  9,0,    70,87,
12,0,   76,95,  9,0,    10,0,
12,0,   13,0,   92,115, 10,0,
93,116, 13,0,   94,117, 14,0,
3,0,    95,118, 9,23,   14,0,
9,23,   76,96,  9,23,   10,24,
96,119, 10,24,  31,44,  10,24,
15,0,   15,0,   9,0,    105,128,
12,0,   18,0,   18,0,   10,0,
106,129,        13,0,   107,130,        31,45,
108,131,        31,46,  85,105, 14,0,
109,132,        31,47,  16,0,   16,0,
110,133,        111,134,        112,135,        15,0,
85,106, 113,136,        114,137,        15,0,
18,0,   115,138,        85,107, 85,108,
18,0,   116,139,        14,27,  9,0,
117,140,        12,0,   118,141,        12,25,
10,0,   16,0,   13,0,   119,142,
13,26,  16,0,   22,18,  129,148,
14,0,   131,149,        14,28,  15,0,
133,150,        134,151,        22,0,   22,0,
18,0,   16,30,  135,152,        16,30,
136,153,        16,30,  137,154,        138,155,
139,156,        15,29,  140,157,        141,158,
147,162,        16,0,   148,163,        23,0,
23,0,   149,164,        150,165,        151,166,
152,167,        22,0,   153,168,        154,169,
155,170,        22,0,   156,171,        162,175,
15,0,   163,176,        164,177,        166,178,
169,179,        18,0,   170,180,        171,181,
22,33,  22,34,  23,0,   22,34,
176,186,        22,34,  23,0,   178,187,
24,0,   24,0,   16,0,   25,0,
25,0,   22,0,   181,188,        186,193,
188,194,        23,33,  23,36,  192,198,
23,36,  22,18,  23,36,  175,185,
175,185,        175,185,        175,185,        193,199,
59,69,  59,70,  23,0,   24,0,
194,200,        208,211,        25,0,   24,0,
59,71,  59,72,  25,0,   59,73,
215,217,        59,74,  217,218,        59,75,
218,219,        59,76,  22,0,   24,34,
22,35,  24,34,  25,37,  24,34,
25,37,  219,220,        25,37,  27,0,
27,0,   26,0,   26,0,   24,0,
28,0,   28,0,   25,0,   23,0,
0,0,    23,35,  29,0,   29,0,
68,84,  68,84,  68,84,  68,84,
68,84,  68,84,  68,84,  68,84,
68,84,  68,84,  27,0,   0,0,
26,0,   0,0,    27,0,   28,0,
26,0,   198,204,        198,204,        28,0,
0,0,    29,41,  0,0,    0,0,
24,0,   29,0,   24,35,  25,0,
26,38,  0,0,    26,38,  28,40,
26,38,  28,40,  0,0,    28,40,
30,0,   30,0,   27,0,   0,0,
26,0,   33,0,   33,0,   28,0,
0,0,    0,0,    0,0,    34,0,
34,0,   29,0,   0,0,    0,0,
```

```
0,0,     0,0,     0,0,     0,0,
0,0,     0,0,     0,0,     30,0,
0,0,     0,0,     27,39,   30,0,
33,0,    0,0,     35,0,    35,0,
33,0,    0,0,     34,0,    27,0,
0,0,     26,0,    34,0,    30,42,
28,0,    30,42,   0,0,     30,42,
33,49,   0,0,     29,0,    36,0,
36,0,    0,0,     34,50,   30,0,
34,50,   35,0,    34,50,   0,0,
33,0,    35,0,    0,0,     0,0,
37,0,    37,0,    34,0,    0,0,
0,0,     0,0,     0,0,     0,0,
0,0,     35,51,   36,0,    35,51,
0,0,     35,51,   36,0,    0,0,
0,0,     38,0,    38,0,    0,0,
0,0,     35,0,    0,0,     37,0,
30,0,    0,0,     36,50,   37,0,
36,50,   33,0,    36,50,   0,0,
39,0,    39,0,    0,0,     34,0,
0,0,     0,0,     36,0,    37,53,
38,0,    37,53,   0,0,     37,53,
38,0,    0,0,     36,52,   40,0,
40,0,    0,0,     0,0,     37,0,
42,0,    42,0,    35,0,    39,0,
38,54,   0,0,     38,54,   39,0,
38,54,   0,0,     55,0,    55,0,
0,0,     0,0,     0,0,     0,0,
38,0,    0,0,     40,0,    36,0,
0,0,     0,0,     40,0,    42,0,
0,0,     0,0,     0,0,     42,0,
0,0,     49,0,    49,0,    39,0,
37,0,    55,0,    40,56,   0,0,
40,56,   55,0,    40,56,   42,58,
39,55,   42,58,   0,0,     42,58,
50,0,    50,0,    40,0,    0,0,
0,0,     38,0,    0,0,     42,0,
49,0,    0,0,     53,0,    53,0,
49,0,    0,0,     0,0,     51,0,
51,0,    55,0,    0,0,     0,0,
39,0,    52,0,    52,0,    50,0,
49,60,   0,0,     49,60,   50,0,
49,60,   0,0,     55,66,   0,0,
0,0,     53,0,    0,0,     40,0,
49,0,    53,0,    51,0,    50,61,
42,0,    50,61,   51,0,    50,61,
52,0,    0,0,     0,0,     0,0,
52,0,    53,64,   55,0,    50,0,
54,0,    54,0,    51,62,   0,0,
51,62,   0,0,     51,62,   0,0,
52,63,   53,0,    52,63,   0,0,
52,63,   0,0,     51,0,    0,0,
0,0,     49,0,    56,0,    56,0,
52,0,    58,0,    58,0,    54,0,
143,143,          0,0,     0,0,     54,0,
0,0,     60,0,    60,0,    0,0,
50,0,    143,0,   0,0,     61,0,
61,0,    0,0,     0,0,     54,65,
0,0,     56,0,    53,0,    0,0,
58,0,    56,0,    0,0,     51,0,
58,0,    0,0,     0,0,     54,0,
60,0,    52,0,    0,0,     0,0,
60,0,    56,67,   61,0,    0,0,
62,0,    62,0,    61,0,    0,0,
0,0,     63,0,    63,0,    60,77,
0,0,     56,0,    0,0,     143,143,
```

```
58,0,    143,143,          0,0,     143,143,
0,0,     0,0,     0,0,     0,0,
60,0,    0,0,     0,0,     62,0,
54,0,    0,0,     61,0,    62,0,
63,0,    0,0,     0,0,     143,143,
63,0,    0,0,     0,0,     64,0,
64,0,    66,0,    66,0,    0,0,
65,0,    65,0,    56,0,    0,0,
63,79,   58,0,    63,79,   0,0,
63,79,   0,0,     0,0,     62,0,
0,0,     60,0,    0,0,     0,0,
63,0,    0,0,     64,0,    61,0,
66,0,    61,78,   64,0,    65,0,
66,0,    0,0,     0,0,     65,0,
0,0,     67,0,    67,0,    0,0,
77,0,    77,0,    64,80,   0,0,
64,80,   0,0,     64,80,   65,81,
0,0,     65,81,   0,0,     65,81,
62,0,    0,0,     64,0,    0,0,
66,0,    63,0,    0,0,     65,0,
67,0,    0,0,     0,0,     77,0,
67,0,    0,0,     0,0,     77,0,
0,0,     0,0,     0,0,     78,0,
78,0,    0,0,     66,82,   0,0,
67,83,   0,0,     67,83,   77,97,
67,83,   77,97,   0,0,     77,97,
79,0,    79,0,    0,0,     64,0,
67,0,    66,0,    0,0,     77,0,
65,0,    0,0,     78,0,    0,0,
0,0,     0,0,     78,0,    0,0,
0,0,     0,0,     0,0,     80,0,
80,0,    82,0,    82,0,    79,0,
0,0,     0,0,     78,98,   79,0,
78,98,   0,0,     78,98,   0,0,
120,0,   120,0,   81,0,    81,0,
0,0,     67,0,    78,0,    79,99,
77,0,    79,99,   80,0,    79,99,
82,0,    0,0,     80,0,    0,0,
82,0,    0,0,     0,0,     79,0,
0,0,     83,0,    83,0,    120,0,
0,0,     81,0,    80,100,  120,0,
80,100,  81,0,    0,0,     0,0,
0,0,     0,0,     122,0,   122,0,
0,0,     0,0,     80,0,    78,0,
82,0,    81,101,  0,0,     81,101,
83,0,    0,0,     0,0,     0,0,
83,0,    0,0,     0,0,     120,0,
79,0,    81,0,    0,0,     0,0,
0,0,     122,0,   0,0,     0,0,
83,103,  122,0,   83,103,  0,0,
82,102,  97,0,    97,0,    0,0,
98,0,    98,0,    0,0,     80,0,
83,0,    82,0,    84,104,  84,104,
84,104,  84,104,  84,104,  84,104,
84,104,  84,104,  84,104,  84,104,
120,0,   122,0,   81,0,    0,0,
97,0,    0,0,     0,0,     98,0,
97,0,    0,0,     0,0,     98,0,
0,0,     99,0,    99,0,    0,0,
100,0,   100,0,   0,0,     0,0,
97,120,  83,0,    97,120,  98,121,
97,120,  98,121,  0,0,     98,121,
0,0,     0,0,     0,0,     0,0,
97,0,    0,0,     122,0,   98,0,
99,0,    0,0,     0,0,     100,0,
99,0,    0,0,     0,0,     100,0,
```

```
0,0,     0,0,     0,0,     101,0,
101,0,   0,0,     102,0,   102,0,
99,122,  0,0,     99,122,  100,123,
99,122,  100,123,          0,0,     100,123,
0,0,     0,0,     0,0,     0,0,
99,0,    97,0,    0,0,     100,0,
98,0,    0,0,     101,0,   0,0,
0,0,     102,0,   101,0,   0,0,
0,0,     102,0,   0,0,     103,0,
103,0,   0,0,     0,0,     0,0,
0,0,     0,0,     101,124,          0,0,
101,124,          102,125,          101,124,          102,125,
0,0,     102,125,          0,0,     121,121,
0,0,     99,0,    101,0,   0,0,
100,0,   102,0,   103,0,   121,143,
121,0,   0,0,     103,0,   104,127,
104,127,          104,127,          104,127,          104,127,
104,127,          104,127,          104,127,          104,127,
104,127,          0,0,     103,126,          0,0,
103,126,          0,0,     103,126,          0,0,
123,0,   123,0,   121,143,          0,0,
0,0,     0,0,     103,0,   101,0,
124,0,   124,0,   102,0,   0,0,
0,0,     125,0,   125,0,   0,0,
0,0,     0,0,     121,121,          0,0,
121,121,          0,0,     121,121,          123,0,
0,0,     126,0,   126,0,   123,0,
0,0,     0,0,     121,143,          124,0,
0,0,     0,0,     0,0,     124,0,
125,0,   0,0,     121,121,          103,0,
125,0,   0,0,     0,0,     0,0,
0,0,     0,0,     0,0,     0,0,
126,0,   0,0,     0,0,     123,0,
126,0,   0,0,     0,0,     0,0,
146,0,   146,0,   0,0,     124,0,
0,0,     0,0,     144,0,   144,0,
125,0,   127,147,          127,147,          127,147,
127,147,          127,147,          127,147,          127,147,
127,147,          127,147,          127,147,          0,0,
126,0,   145,0,   145,0,   146,0,
0,0,     0,0,     0,0,     146,0,
123,0,   144,0,   123,144,          0,0,
0,0,     144,0,   0,0,     0,0,
124,0,   0,0,     124,145,          0,0,
0,0,     125,0,   0,0,     0,0,
145,0,   144,159,          0,0,     144,159,
145,0,   144,159,          0,0,     146,0,
0,0,     126,0,   0,0,     126,146,
0,0,     144,0,   159,0,   159,0,
145,160,          0,0,     145,160,          0,0,
145,160,          160,0,   160,0,   0,0,
0,0,     146,161,          0,0,     0,0,
145,0,   161,0,   161,0,   0,0,
0,0,     0,0,     0,0,     0,0,
0,0,     159,0,   0,0,     0,0,
146,0,   159,0,   0,0,     0,0,
160,0,   0,0,     144,0,   0,0,
160,0,   0,0,     172,0,   172,0,
161,0,   159,172,          0,0,     159,172,
161,0,   159,172,          174,0,   174,0,
160,173,          145,0,   160,173,          0,0,
160,173,          159,0,   0,0,     173,0,
173,0,   0,0,     0,0,     0,0,
160,0,   172,0,   0,0,     0,0,
0,0,     172,0,   0,0,     0,0,
161,0,   174,0,   0,0,     0,0,
```

```
0,0,    174,0,  0,0,    0,0,
0,0,    172,182,        173,0,  172,182,
0,0,    172,182,        173,0,  0,0,
182,0,  182,0,  159,0,  161,174,
0,0,    172,0,  183,0,  183,0,
0,0,    160,0,  173,183,        0,0,
173,183,        174,0,  173,183,        0,0,
0,0,    161,0,  0,0,    0,0,
0,0,    0,0,    173,0,  182,0,
0,0,    184,0,  184,0,  182,0,
0,0,    183,0,  0,0,    0,0,
0,0,    183,0,  0,0,    0,0,
0,0,    174,184,        172,0,  182,189,
0,0,    182,189,        0,0,    182,189,
0,0,    183,190,        174,0,  183,190,
184,0,  183,190,        0,0,    182,0,
184,0,  0,0,    0,0,    173,0,
0,0,    183,0,  185,192,        185,192,
185,192,        185,192,        185,192,        185,192,
185,192,        185,192,        185,192,        185,192,
0,0,    189,0,  189,0,  0,0,
190,0,  190,0,  191,0,  191,0,
184,0,  0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
182,0,  184,191,        0,0,    0,0,
0,0,    0,0,    183,0,  0,0,
189,0,  0,0,    0,0,    190,0,
189,0,  191,0,  0,0,    190,0,
0,0,    191,0,  0,0,    195,0,
195,0,  0,0,    196,0,  196,0,
189,195,        184,0,  189,195,        190,196,
189,195,        190,196,        0,0,    190,196,
197,0,  197,0,  0,0,    0,0,
189,0,  0,0,    0,0,    190,0,
0,0,    191,0,  195,0,  0,0,
0,0,    196,0,  195,0,  0,0,
0,0,    196,0,  0,0,    0,0,
0,0,    0,0,    191,197,        197,0,
0,0,    0,0,    195,201,        197,0,
195,201,        196,202,        195,201,        196,202,
0,0,    196,202,        201,0,  201,0,
0,0,    189,0,  195,0,  0,0,
190,0,  196,0,  191,0,  205,0,
205,0,  0,0,    0,0,    0,0,
0,0,    202,0,  202,0,  197,0,
0,0,    0,0,    0,0,    203,0,
203,0,  201,0,  0,0,    0,0,
0,0,    201,0,  0,0,    0,0,
0,0,    0,0,    205,0,  0,0,
0,0,    197,203,        205,0,  195,0,
202,0,  201,205,        196,0,  201,205,
202,0,  201,205,        203,0,  0,0,
0,0,    0,0,    203,0,  0,0,
197,0,  201,0,  0,0,    0,0,
202,206,        0,0,    202,206,        0,0,
202,206,        0,0,    205,0,  0,0,
206,0,  206,0,  0,0,    0,0,
202,0,  207,0,  207,0,  0,0,
0,0,    0,0,    203,0,  204,208,
204,208,        204,208,        204,208,        204,208,
204,208,        204,208,        204,208,        204,208,
204,208,        0,0,    201,0,  206,0,
0,0,    0,0,    0,0,    206,0,
207,0,  0,0,    0,0,    205,0,
207,0,  205,209,        203,207,        209,0,
209,0,  202,0,  0,0,    0,0,
```

```
210,0,  210,0,  0,0,    203,0,
207,210,        0,0,    207,210,        0,0,
207,210,        0,0,    0,0,    206,0,
0,0,    0,0,    0,0,    0,0,
207,0,  0,0,    209,0,  0,0,
0,0,    0,0,    209,0,  210,0,
0,0,    0,0,    0,0,    210,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    209,212,        0,0,
209,212,        0,0,    209,212,        212,0,
212,0,  0,0,    0,0,    0,0,
206,0,  0,0,    209,0,  0,0,
0,0,    207,0,  0,0,    210,0,
211,213,        211,213,        211,213,        211,213,
211,213,        211,213,        211,213,        211,213,
211,213,        211,213,        212,0,  0,0,
214,0,  214,0,  212,0,  213,215,
213,215,        213,215,        213,215,        213,215,
213,215,        213,215,        213,215,        213,215,
213,215,        0,0,    212,214,        209,0,
212,214,        0,0,    212,214,        0,0,
210,0,  216,0,  216,0,  214,0,
0,0,    0,0,    212,0,  214,0,
220,221,        220,221,        220,221,        220,221,
220,221,        220,221,        220,221,        220,221,
220,221,        220,221,        0,0,    214,216,
0,0,    214,216,        0,0,    214,216,
216,0,  0,0,    0,0,    0,0,
216,0,  0,0,    0,0,    214,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    212,0,
221,222,        221,222,        221,222,        221,222,
221,222,        221,222,        221,222,        221,222,
221,222,        221,222,        0,0,    0,0,
216,0,  222,223,        222,223,        222,223,
222,223,        222,223,        222,223,        222,223,
222,223,        222,223,        222,223,        0,0,
214,0,  223,224,        223,224,        223,224,
223,224,        223,224,        223,224,        223,224,
223,224,        223,224,        223,224,        0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    216,0,  0,0,    0,0,
0,0};
struct yysvf yysvec[] = {
0,      0,      0,
yycrank+-1,     0,              0,
yycrank+-3,     yysvec+1,       0,
yycrank+-35,    0,              yyvstop+1,
yycrank+0,      0,              yyvstop+4,
yycrank+2,      0,              yyvstop+6,
yycrank+3,      yysvec+5,       yyvstop+8,
yycrank+2,      0,              yyvstop+11,
yycrank+-14,    yysvec+3,       yyvstop+14,
yycrank+-82,    yysvec+3,       yyvstop+17,
yycrank+-87,    yysvec+3,       yyvstop+20,
yycrank+0,      0,              yyvstop+23,
yycrank+-84,    yysvec+3,       yyvstop+26,
yycrank+-89,    yysvec+3,       yyvstop+29,
yycrank+-95,    yysvec+3,       yyvstop+32,
yycrank+-131,   yysvec+3,       yyvstop+35,
yycrank+-149,   yysvec+3,       yyvstop+38,
yycrank+4,      0,              yyvstop+41,
yycrank+-136,   yysvec+3,       yyvstop+44,
yycrank+0,      yysvec+5,       0,
yycrank+9,      yysvec+5,       0,
```

```
yycrank+0,      0,              yyvstop+46,
yycrank+-185,   0,              yyvstop+48,
yycrank+-202,   yysvec+22,      yyvstop+50,
yycrank+-231,   yysvec+3,       yyvstop+52,
yycrank+-234,   yysvec+3,       yyvstop+54,
yycrank+-280,   yysvec+3,       yyvstop+56,
yycrank+-278,   yysvec+3,       yyvstop+58,
yycrank+-283,   yysvec+3,       yyvstop+60,
yycrank+-289,   yysvec+3,       yyvstop+62,
yycrank+-327,   yysvec+3,       yyvstop+64,
yycrank+73,     0,              yyvstop+66,
yycrank+10,     yysvec+5,       0,
yycrank+-332,   yysvec+3,       yyvstop+68,
yycrank+-338,   yysvec+3,       yyvstop+70,
yycrank+-357,   yysvec+3,       yyvstop+73,
yycrank+-374,   yysvec+3,       yyvstop+75,
yycrank+-387,   yysvec+3,       yyvstop+78,
yycrank+-404,   yysvec+3,       yyvstop+80,
yycrank+-419,   yysvec+3,       yyvstop+82,
yycrank+-434,   yysvec+3,       yyvstop+84,
yycrank+8,      0,              0,
yycrank+-439,   yysvec+3,       yyvstop+86,
yycrank+0,      0,              yyvstop+88,
yycrank+0,      0,              yyvstop+90,
yycrank+11,     0,              0,
yycrank+0,      0,              yyvstop+92,
yycrank+0,      0,              yyvstop+94,
yycrank+0,      yysvec+5,       yyvstop+96,
yycrank+-468,   yysvec+3,       yyvstop+98,
yycrank+-483,   yysvec+3,       yyvstop+100,
yycrank+-498,   yysvec+3,       yyvstop+102,
yycrank+-504,   yysvec+3,       yyvstop+104,
yycrank+-493,   yysvec+3,       yyvstop+106,
yycrank+-535,   yysvec+3,       yyvstop+108,
yycrank+-449,   yysvec+3,       yyvstop+110,
yycrank+-553,   yysvec+3,       yyvstop+112,
yycrank+15,     0,              0,
yycrank+-556,   yysvec+3,       yyvstop+114,
yycrank+193,    0,              0,
yycrank+-564,   yysvec+3,       yyvstop+117,
yycrank+-570,   yysvec+3,       yyvstop+119,
yycrank+-595,   yysvec+3,       yyvstop+121,
yycrank+-600,   yysvec+3,       yyvstop+124,
yycrank+-630,   yysvec+3,       yyvstop+126,
yycrank+-635,   yysvec+3,       yyvstop+128,
yycrank+-632,   yysvec+3,       yyvstop+130,
yycrank+-664,   yysvec+3,       yyvstop+132,
yycrank+252,    0,              0,
yycrank+7,      0,              0,
yycrank+36,     0,              0,
yycrank+3,      0,              0,
yycrank+3,      0,              0,
yycrank+4,      0,              0,
yycrank+24,     0,              0,
yycrank+4,      0,              0,
yycrank+44,     0,              0,
yycrank+-667,   yysvec+3,       yyvstop+134,
yycrank+-698,   yysvec+3,       yyvstop+136,
yycrank+-711,   yysvec+3,       yyvstop+138,
yycrank+-730,   yysvec+3,       yyvstop+140,
yycrank+-745,   yysvec+3,       yyvstop+142,
yycrank+-732,   yysvec+3,       yyvstop+144,
yycrank+-764,   yysvec+3,       yyvstop+146,
yycrank+778,    0,              0,
yycrank+86,     0,              0,
yycrank+16,     0,              0,
```

```
yycrank+3,      0,              0,
yycrank+13,     0,              0,
yycrank+32,     0,              0,
yycrank+34,     0,              0,
yycrank+33,     0,              0,
yycrank+56,     0,              0,
yycrank+54,     0,              0,
yycrank+38,     0,              0,
yycrank+45,     0,              0,
yycrank+56,     0,              0,
yycrank+-808,   yysvec+3,       yyvstop+148,
yycrank+-811,   yysvec+3,       yyvstop+150,
yycrank+-840,   yysvec+3,       yyvstop+152,
yycrank+-843,   yysvec+3,       yyvstop+154,
yycrank+-874,   yysvec+3,       yyvstop+156,
yycrank+-877,   yysvec+3,       yyvstop+158,
yycrank+-906,   yysvec+3,       yyvstop+161,
yycrank+895,    0,              0,
yycrank+74,     0,              0,
yycrank+81,     0,              0,
yycrank+81,     0,              0,
yycrank+74,     0,              0,
yycrank+87,     0,              yyvstop+163,
yycrank+82,     0,              0,
yycrank+74,     0,              0,
yycrank+91,     0,              0,
yycrank+97,     0,              0,
yycrank+98,     0,              0,
yycrank+74,     0,              0,
yycrank+104,    0,              0,
yycrank+92,     0,              0,
yycrank+96,     0,              0,
yycrank+114,    0,              0,
yycrank+-743,   yysvec+3,       yyvstop+165,
yycrank+-930,   0,              yyvstop+168,
yycrank+-777,   yysvec+3,       yyvstop+171,
yycrank+-951,   yysvec+3,       yyvstop+174,
yycrank+-959,   yysvec+3,       yyvstop+176,
yycrank+-964,   yysvec+3,       yyvstop+178,
yycrank+-976,   yysvec+3,       yyvstop+181,
yycrank+977,    0,              0,
yycrank+0,      0,              yyvstop+183,
yycrank+111,    0,              0,
yycrank+0,      0,              yyvstop+185,
yycrank+105,    0,              0,
yycrank+0,      0,              yyvstop+187,
yycrank+107,    0,              0,
yycrank+114,    0,              0,
yycrank+113,    0,              0,
yycrank+121,    0,              0,
yycrank+117,    0,              0,
yycrank+121,    0,              0,
yycrank+122,    0,              0,
yycrank+137,    0,              0,
yycrank+138,    0,              0,
yycrank+0,      0,              yyvstop+189,
yycrank+-567,   yysvec+121,     yyvstop+191,
yycrank+-1013,  yysvec+3,       yyvstop+193,
yycrank+-1028,  yysvec+3,       yyvstop+195,
yycrank+-1007,  yysvec+3,       yyvstop+197,
yycrank+164,    0,              0,
yycrank+125,    0,              0,
yycrank+131,    0,              0,
yycrank+137,    0,              0,
yycrank+133,    0,              0,
yycrank+147,    0,              0,
```

```
yycrank+151,    0,              0,
yycrank+152,    0,              0,
yycrank+151,    0,              0,
yycrank+153,    0,              0,
yycrank+0,      0,              yyvstop+199,
yycrank+0,      0,              yyvstop+201,
yycrank+-1065,  yysvec+3,       yyvstop+203,
yycrank+-1072,  yysvec+3,       yyvstop+205,
yycrank+-1080,  yysvec+3,       yyvstop+207,
yycrank+191,    0,              0,
yycrank+142,    0,              0,
yycrank+137,    0,              0,
yycrank+0,      0,              yyvstop+209,
yycrank+159,    0,              0,
yycrank+0,      0,              yyvstop+211,
yycrank+0,      0,              yyvstop+213,
yycrank+144,    0,              0,
yycrank+160,    0,              0,
yycrank+153,    0,              0,
yycrank+-1101,  yysvec+3,       yyvstop+215,
yycrank+-1118,  yysvec+3,       yyvstop+217,
yycrank+-1109,  yysvec+3,       yyvstop+219,
yycrank+207,    0,              0,
yycrank+163,    0,              0,
yycrank+0,      0,              yyvstop+221,
yycrank+156,    0,              0,
yycrank+0,      0,              yyvstop+223,
yycrank+0,      0,              yyvstop+225,
yycrank+179,    0,              0,
yycrank+-1147,  yysvec+3,       yyvstop+227,
yycrank+-1153,  yysvec+3,       yyvstop+229,
yycrank+-1172,  yysvec+3,       yyvstop+231,
yycrank+1166,   0,              0,
yycrank+168,    0,              0,
yycrank+0,      0,              yyvstop+233,
yycrank+179,    0,              0,
yycrank+-1216,  yysvec+3,       yyvstop+235,
yycrank+-1219,  yysvec+3,       yyvstop+237,
yycrank+-1221,  yysvec+3,       yyvstop+239,
yycrank+204,    0,              0,
yycrank+181,    0,              0,
yycrank+181,    0,              0,
yycrank+-1250,  yysvec+3,       yyvstop+241,
yycrank+-1253,  yysvec+3,       yyvstop+243,
yycrank+-1263,  yysvec+3,       yyvstop+245,
yycrank+269,    0,              0,
yycrank+0,      0,              yyvstop+247,
yycrank+0,      0,              yyvstop+249,
yycrank+-1297,  yysvec+3,       yyvstop+251,
yycrank+-1312,  yysvec+3,       yyvstop+253,
yycrank+-1318,  yysvec+3,       yyvstop+255,
yycrank+1331,   0,              0,
yycrank+-1306,  yysvec+3,       yyvstop+257,
yycrank+-1359,  yysvec+3,       yyvstop+259,
yycrank+-1364,  yysvec+3,       yyvstop+262,
yycrank+218,    0,              0,
yycrank+-1394,  yysvec+3,       yyvstop+265,
yycrank+-1399,  yysvec+3,       yyvstop+267,
yycrank+1412,   0,              0,
yycrank+-1438,  yysvec+3,       yyvstop+270,
yycrank+1427,   0,              0,
yycrank+-1463,  yysvec+3,       yyvstop+272,
yycrank+228,    0,              0,
yycrank+-1484,  yysvec+3,       yyvstop+274,
yycrank+242,    0,              0,
yycrank+196,    0,              0,
```

```
yycrank+239,    0,              0,
yycrank+1452,   0,              0,
yycrank+1484,   0,              0,
yycrank+1497,   0,              0,
yycrank+1509,   0,              0,
yycrank+0,      0,              yyvstop+277,
0,      0,      0};
struct yywork *yytop = yycrank+1577;
struct yysvf *yybgin = yysvec+1;
char yymatch[] = {
00  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,011 ,012 ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
040 ,01  ,01  ,01  ,011 ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
'0' ,'0' ,'2' ,'2' ,'4' ,'4' ,'4' ,'4' ,
'4' ,'4' ,01  ,01  ,'<' ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,'D' ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,'D' ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,'<' ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
0};
char yyextra[] = {
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0};
int yylineno =1;
# define YYU(x) x
# define NLSTATE yyprevious=YYNEWLINE
char yytext[YYLMAX];
struct yysvf *yylstate [YYLMAX], **yylsp, **yyolsp;
char yysbuf[YYLMAX];
char *yysptr = yysbuf;
int *yyfnd;
extern struct yysvf *yyestate;
int yyprevious = YYNEWLINE;
yylook(){
        register struct yysvf *yystate, **lsp;
        register struct yywork *yyt;
        struct yysvf *yyz;
        int yych, yyfirst;
        struct yywork *yyr;
# ifdef LEXDEBUG
        int debug;
# endif
        char *yylastch;
        /* start off machines */
# ifdef LEXDEBUG
        debug = 0;
# endif
        yyfirst=1;
        if (!yymorfg)
                yylastch = yytext;
        else {
                yymorfg=0;
                yylastch = yytext+yyleng;
                }
        for(;;){
```

```
                lsp = yylstate;
                yyestate = yystate = yybgin;
                if (yyprevious==YYNEWLINE) yystate++;
                for (;;){
# ifdef LEXDEBUG
                        if(debug)fprintf(yyout,"state %d\n",yystate-yysve
# endif
                        yyt = yystate->yystoff;
                        if(yyt == yycrank && !yyfirst){  /* may not be an
                                yyz = yystate->yyother;
                                if(yyz == 0)break;
                                if(yyz->yystoff == yycrank)break;
                                }
                        *yylastch++ = yych = input();
                        yyfirst=0;
                tryagain:
# ifdef LEXDEBUG
                        if(debug){
                                fprintf(yyout,"char ");
                                allprint(yych);
                                putchar('\n');
                                }
# endif
                        yyr = yyt;
                        if ( (int)yyt > (int)yycrank){
                                yyt = yyr + yych;
                                if (yyt <= yytop && yyt->verify+yysvec ==
                                        if(yyt->advance+yysvec == YYLERR)
                                                {unput(*--yylastch);break
                                        *lsp++ = yystate = yyt->advance+y
                                        goto contin;
                                        }
                                }
# ifdef YYOPTIM
                        else if((int)yyt < (int)yycrank) {               /
                                yyt = yyr = yycrank+(yycrank-yyt);
# ifdef LEXDEBUG
                                if(debug)fprintf(yyout,"compressed state\
# endif
                                yyt = yyt + yych;
                                if(yyt <= yytop && yyt->verify+yysvec ==
                                        if(yyt->advance+yysvec == YYLERR)
                                                {unput(*--yylastch);break
                                        *lsp++ = yystate = yyt->advance+y
                                        goto contin;
                                        }
                                yyt = yyr + YYU(yymatch[yych]);
# ifdef LEXDEBUG
                                if(debug){
                                        fprintf(yyout,"try fall back char
                                        allprint(YYU(yymatch[yych]));
                                        putchar('\n');
                                        }
# endif
                                if(yyt <= yytop && yyt->verify+yysvec ==
                                        if(yyt->advance+yysvec == YYLERR)
                                                {unput(*--yylastch);break
                                        *lsp++ = yystate = yyt->advance+y
                                        goto contin;
                                        }
                                }
                        if ((yystate = yystate->yyother) && (yyt= yystate
# ifdef LEXDEBUG
                                if(debug)fprintf(yyout,"fall back to stat
# endif
                                goto tryagain;
```

```
                                        }
# endif
                                else
                                        {unput(*--yylastch);break;}
                        contin:
# ifdef LEXDEBUG
                                if(debug){
                                        fprintf(yyout,"state %d char ",yystate-yy
                                        allprint(yych);
                                        putchar('\n');
                                        }
# endif
                                ;
                                }
# ifdef LEXDEBUG
                        if(debug){
                                fprintf(yyout,"stopped at %d with ",*(lsp-1)-yysv
                                allprint(yych);
                                putchar('\n');
                                }
# endif
                        while (lsp-- > yylstate){
                                *yylastch-- = 0;
                                if (*lsp != 0 && (yyfnd= (*lsp)->yystops) && *yyf
                                        yyolsp = lsp;
                                        if(yyextra[*yyfnd]){            /* must b
                                                while(yyback((*lsp)->yystops,-*yy
                                                        lsp--;
                                                        unput(*yylastch--);
                                                        }
                                                }
                                        yyprevious = YYU(*yylastch);
                                        yylsp = lsp;
                                        yyleng = yylastch-yytext+1;
                                        yytext[yyleng] = 0;
# ifdef LEXDEBUG
                                        if(debug){
                                                fprintf(yyout,"\nmatch ");
                                                sprint(yytext);
                                                fprintf(yyout," action %d\n",*yyf
                                                }
# endif
                                        return(*yyfnd++);
                                        }
                                unput(*yylastch);
                                }
                        if (yytext[0] == 0  /* && feof(yyin) */)
                                {
                                yysptr=yysbuf;
                                return(0);
                                }
                        yyprevious = yytext[0] = input();
                        if (yyprevious>0)
                                output(yyprevious);
                        yylastch=yytext;
# ifdef LEXDEBUG
                        if(debug)putchar('\n');
# endif
                        }
                }
yyback(p, m)
        int *p;
{
if (p==0) return(0);
while (*p)
        {
```

```
        if (*p++ == m)
                return(1);
        }
return(0);
}
        /* the following are only used in the lex library */
yyinput(){
        return(input());
        }
yyoutput(c)
  int c; {
        output(c);
        }
yyunput(c)
   int c; {
        unput(c);
        }
```

```
%{
#include <string.h>
#include <fcntl.h>
#include <stdio.h>
#define grave 1
#define pasgrave 0
#define nbsect 17
#define nberr 14
#define longmess 70
#include "messages"
char checkref[150];
char checktype[150];
char numdoccour[25] = {"00000000000000000000"};
char badtoken[150];
char oldkeyw[4];
int linenb = 1;
int globnumsect    = 0;
int globnumerr     = 0;
int nbnewchapcount = 0;
int nbnewartcount  = 0;
int nbtitlecount   = 0;
int nbnumdoccount  = 0;
int nbdatecount    = 0;
int nbcotecount    = 0;
int nbcodecount    = 0;
int nbtitrecount   = 0;
int nbrefcount     = 0;
int nbtextecount   = 0;
int nbconclusioncount = 0;
int nblanguecount  = 0;
int nbdocnumcount  = 0;
int nbdatcount     = 0;
int nbpubrefcount  = 0;
int nbtypecount    = 0;
int nbproductcount = 0;
int nbcountrycount = 0;
int nbkeywordscount = 0;
int nbnewparcount  = 0;
%}

%token NEWCHAP NEWART TITLE NEWNUMDOC NEWDATE NEWCOTE NEWCODE NEWTITRE NE

%token NEWLINE TAB SP DATE COTEjk COTEn CODE NUMDOCj NUMDOCk NUMDOCn DOCN

%%      /* RULES SECTION */

textin :        chaphead
                chapter1
                chaphead
                chapter2
                chaphead
                chapter3
                chaphead
                chapter4
                ;

chaphead :      NEWCHAP
                NEWLINE
                |
                error
                NEWCHAP
                NEWLINE
                ;

chapter1 :      /* empty or */
                |
```

```
                chapter1
                article1
                artfooter
                ;

chapter2 :      /* empty or */
                |
                chapter2
                article2
                artfooter
                ;

chapter3 :      /* empty or */
                |
                chapter3
                article3
                artfooter
                ;

chapter4 :      /* empty or */
                |
                chapter4
                article4
                artfooter
                ;

artfooter :     NEWART
                NEWLINE
                |
                error
                NEWART
                NEWLINE
                ;

article1 :      title1
                body1
                ;

article2 :      title2
                body2
                ;

article3 :      title3
                body3
                ;

article4 :      title4
                body4
                ;

title1 :        titlehead
                contitle1
                titlehead
                ;

titlehead :     TITLE NEWLINE
                |
                error
                TITLE NEWLINE
                ;

contitle1 :     numdoc1
                date
                cote12
                code
                {
```

```
                globnumsect = 0;
                globnumerr = 0;
                }
                ;

numdoc1 :       NEWNUMDOC NEWLINE
                {
                globnumsect = 1;
                globnumerr = 1;
                }
                NUMDOCj
                {strcpy (numdoccour, yytext);}
                NEWLINE
                |
                error
                ;

date :          NEWDATE NEWLINE
                {
                globnumsect = 2;
                globnumerr = 1;
                }
                DATE NEWLINE
                |
                error
                ;

cote12 :        NEWCOTE NEWLINE
                {
                globnumsect = 3;
                globnumerr = 1;
                }
                COTEjk NEWLINE
                |
                error
                ;

code :          NEWCODE NEWLINE
                {
                globnumsect = 4;
                globnumerr = 1;
                }
                CODE NEWLINE
                |
                error
                ;

body1 :         titre
                references
                texte
                conclusion
                {
                globnumsect = 0;
                globnumerr = 0;
                }
                ;

titre :         NEWTITRE NEWLINE
                {
                globnumsect = 5;
                globnumerr = 3;
                }
                contitre
                |
                error
                ;
```

```
contitre :      optnlspaces
                WORD
                restoftitre
                ;

optnlspaces :   /* empty or */
                |
                optnlspaces
                SP
                |
                optnlspaces
                TAB
                ;

restoftitre :   /* empty or */
                |
                restoftitre
                SP
                |
                restoftitre
                DOLLAR
                |
                restoftitre
                LESS
                |
                restoftitre
                TAB
                |
                restoftitre
                NEWLINE
                |
                restoftitre
                WORD
                |
                restoftitre
                NEWLINE NEWPAR NEWLINE
                |
                restoftitre
                DATE
                |
                restoftitre
                COTEjk
                |
                restoftitre
                COTEn
                |
                restoftitre
                CODE
                |
                restoftitre
                NUMDOCj
                |
                restoftitre
                NUMDOCk
                |
                restoftitre
                NUMDOCn
                |
                restoftitre
                DOCNUM
                |
                restoftitre
                PUBREF
                |
                restoftitre
```

```
                KEYW1
                |
                restoftitre
                KEYW2
                ;

references :    NEWREF NEWLINE
                {
                globnumsect = 6;
                globnumerr = 1;
                checkref[0] = '\0';
                }
                conreferences
                |
                error
                ;

conreferences : reference
                |
                conreferences
                {checkref[0] = '\0';}
                NEWPAR NEWLINE
                reference
                ;

reference :     refdescriptor optionalless NEWLINE
                lineref2
                {
                int errinref;

                if (strlen (checkref) > 0)
                        checkref[strlen (checkref) - 1] = '\0';
                errinref = verifier (checkref, "REFERENCES", pasgrave);
                if (errinref != 0) erreur (6, errinref);
                }
                ;

lineref2 :      /* empty or */
                |
                TAB refdescriptor optionalless NEWLINE
                lineref3
                ;

lineref3 :      /* empty or */
                |
                TAB TAB refdescriptor optionalless NEWLINE
                lineref4
                ;

lineref4 :      /* empty or */
                |
                TAB TAB TAB refdescriptor optionalless NEWLINE
                lineref5
                ;

lineref5 :      /* empty or */
                |
                TAB TAB TAB TAB refdescriptor optionalless NEWLINE
                lineref6
                ;

lineref6 :      /* empty or */
                |
                TAB TAB TAB TAB TAB refdescriptor optionalless NEWLINE
                lineref7
                ;
```

```
lineref7 :      /* empty or */
                |
                TAB TAB TAB TAB TAB TAB refdescriptor optionalless NEWLIN
                lineref8
                ;

lineref8 :      /* empty or */
                |
                TAB TAB TAB TAB TAB TAB TAB refdescriptor optionalless NE
                ;

refdescriptor : WORD
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                {
                strcat (checkref, ":");
                }
                ;

restofrefdes :  /* empty or */
                |
                SP
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                DOLLAR
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                TAB
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                WORD
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                DATE
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                COTEjk
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                COTEn
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
```

```
                CODE
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                NUMDOCj
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                NUMDOCk
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                NUMDOCn
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                DOCNUM
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                PUBREF
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                KEYW1
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                |
                KEYW2
                {
                strcat (checkref, yytext);
                }
                restofrefdes
                ;

optionalless :  /* empty or */
                |
                LESS
                ;

texte :         NEWTEXTE NEWLINE
                {
                globnumsect = 7;
                globnumerr = 3;
                }
                contexte
                |
                error
                ;

contexte :      optnlspaces
                WORD
```

```
                restoftexte
                ;

restoftexte :   /* empty or */
                |
                restoftexte
                SP
                |
                restoftexte
                DOLLAR
                |
                restoftexte
                LESS
                |
                restoftexte
                TAB
                |
                restoftexte
                NEWLINE
                |
                restoftexte
                WORD
                |
                restoftexte
                NEWLINE NEWPAR NEWLINE
                |
                restoftexte
                DATE
                |
                restoftexte
                COTEjk
                |
                restoftexte
                COTEn
                |
                restoftexte
                CODE
                |
                restoftexte
                NUMDOCj
                |
                restoftexte
                NUMDOCk
                |
                restoftexte
                NUMDOCn
                |
                restoftexte
                DOCNUM
                |
                restoftexte
                PUBREF
                |
                restoftexte
                KEYW1
                |
                restoftexte
                KEYW2
                ;

conclusion :    /* empty or */
                |
                NEWCONC NEWLINE
                {
                globnumsect = 8;
                globnumerr = 3;
```

```
                }
                contexte
                |
                NEWCONC
                error
                ;

title2 :        titlehead
                contitle2
                titlehead
                ;

contitle2 :     numdoc2
                date
                cote12
                {
                globnumsect = 0;
                globnumerr = 0;
                }
                ;

numdoc2 :       NEWNUMDOC NEWLINE
                {
                globnumsect = 1;
                globnumerr = 1;
                }
                NUMDOCk
                {strcpy (numdoccour, yytext);}
                NEWLINE
                |
                error
                ;

body2 :         titre
                references
                texte
                conclusion
                {
                globnumsect = 0;
                globnumerr = 0;
                }
                ;

title3 :        titlehead
                contitle3
                titlehead
                ;

contitle3 :     numdoc3
                date
                cote3
                langue
                {
                globnumsect = 0;
                globnumerr = 0;
                }
                ;

numdoc3 :       NEWNUMDOC NEWLINE
                {
                globnumsect = 1;
                globnumerr = 1;
                }
                NUMDOCn
                {strcpy (numdoccour, yytext);}
                NEWLINE
```

```
                |
                error
                ;

cote3 :         NEWCOTE NEWLINE
                {
                globnumsect = 3;
                globnumerr = 1;
                }
                COTEn NEWLINE
                |
                error
                ;

langue :        NEWLANG NEWLINE
                {
                globnumsect = 9;
                globnumerr = 1;
                }
                conlangue NEWLINE
                |
                error
                ;

conlangue :     WORD
                {
                int errinlang;

                switch (strlen (yytext))
                        {
                        case 1 :
                                if ((yytext[0] < 'A') || (yytext[0] > 'Z'
                                        erreur (9, 1);
                                break;
                        case 2 :
                                if ((yytext[0] < 'A') || (yytext[0] > 'Z'
                                        erreur (9, 1);
                                else
                                if ((yytext[1] < 'A') || (yytext[1] > 'Z'
                                        erreur (9, 1);
                                break;
                        default :
                                erreur (9, 1);
                        }
                errinlang = verifier (yytext, "LANGUE", grave);
                if (errinlang != 0) erreur (9, errinlang);
                }
                ;

body3 :         titre
                references
                texte
                conclusion
                {
                globnumsect = 0;
                globnumerr = 0;
                }
                ;

title4 :        titlehead
                contitle4
                titlehead
                ;

contitle4 :     docnum
                dat
```

```
                pub_ref
                {
                globnumsect = 13;
                globnumerr = 1;
                }
                type
                {
                globnumsect = 14;
                globnumerr = 3;
                }
                product
                {
                globnumsect = 15;
                globnumerr = 1;
                }
                country
                {
                globnumsect = 0;
                globnumerr = 0;
                }
                ;

docnum :        NEWDOCNUM NEWLINE
                {
                globnumsect = 10;
                globnumerr = 1;
                }
                DOCNUM
                {strcpy (numdoccour, yytext);}
                NEWLINE
                |
                error
                ;

dat :           NEWDAT NEWLINE
                {
                globnumsect = 11;
                globnumerr = 1;
                }
                DATE NEWLINE
                |
                error
                ;

pub_ref :       NEWPUB_REF NEWLINE
                {
                globnumsect = 12;
                globnumerr = 1;
                }
                PUBREF NEWLINE
                |
                error
                ;

type :          NEWTYPE NEWLINE
                contype NEWLINE
                |
                error
                ;

contype :       contype NEWLINE
                lignetype
                |
                lignetype
                ;
```

```
lignetype :     WORD
                {
                strcpy (checktype, yytext);
                }
                restoflntype
                optionalless
                {
                int errintype;

                errintype = verifier (checktype, "TYPE", pasgrave);
                if (errintype != 0) erreur (13, errintype);
                }
                ;

restoflntype :  /* empty or */
                |
                SP
                {strcat (checktype, yytext);}
                restoflntype
                |
                TAB
                {strcat (checktype, yytext);}
                restoflntype
                |
                WORD
                {strcat (checktype, yytext);}
                restoflntype
                ;

product :       NEWPRODUCT NEWLINE
                conproduct
                |
                error
                ;

conproduct :    optnlspaces
                WORD
                restofproduct
                ;

restofproduct : /* empty or */
                |
                restofproduct
                SP
                |
                restofproduct
                DOLLAR
                |
                restofproduct
                LESS
                |
                restofproduct
                TAB
                |
                restofproduct
                NEWLINE
                |
                restofproduct
                WORD
                |
                restofproduct
                NEWLINE NEWPAR NEWLINE
                |
                restofproduct
                DATE
                |
```

```
                restofproduct
                COTEjk
                |
                restofproduct
                COTEn
                |
                restofproduct
                CODE
                |
                restofproduct
                NUMDOCj
                |
                restofproduct
                NUMDOCk
                |
                restofproduct
                NUMDOCn
                |
                restofproduct
                DOCNUM
                |
                restofproduct
                PUBREF
                |
                restofproduct
                KEYW1
                |
                restofproduct
                KEYW2
                ;

country :       NEWCOUNTRY NEWLINE
                concountry NEWLINE
                |
                error
                ;

concountry :    countrycode
                restofcountry
                optionalless
                ;

countrycode :   WORD
                {
                int errincountry;

                errincountry = verifier (yytext, "COUNTRY", grave);
                if (errincountry != 0) erreur (15, errincountry);
                }
                ;

restofcountry : /* empty or */
                |
                SP
                countrycode
                restofcountry
                ;

body4 :         keywords
                {
                globnumsect = 0;
                globnumerr = 0;
                }
                ;

keywords :      NEWKEYW NEWLINE
```

```
                {
                globnumsect = 16;
                globnumerr = 1;
                strcpy (oldkeyw, '\0');
                }
                conkeywords
                |
                error
                ;

conkeywords :   keywordline
                |
                conkeywords
                NEWPAR NEWLINE
                keywordline
                ;

keywordline :   KEYW1
                {
                /* les keywords doivent etre tries par ordre croissant */
                if (strcmp (oldkeyw, yytext) > 0) erreur (16, 13);
                strcpy (oldkeyw, yytext);
                }
                SP
                KEYW2
                {
                int errinkeyw;

                errinkeyw = verifier (yytext, "KEYWORDS", pasgrave);
                if (errinkeyw != 0) erreur (16, 4);
                }
                optionalless
                NEWLINE
                ;

%%      /* PROGRAMS SECTION */

void erreur (int, int);
int verifier (char *, char *, int);
void statdisplay ();

#include "jur.lex.yy.c"

void erreur (numsection, numerreur)
int numsection, numerreur;
{
printf ("\n");
printf ("%s%d\n", "Ligne incriminee : ", yylineno);
if (numerreur < 10) printf ("%-15s", "Erreur");
else printf ("%-15s", "Avertissement");
switch (numsection)
        {
        case 1 :
                strcpy (badtoken, yytext);
                strcpy (numdoccour, yytext);
                break;
        case 6 :
                strcpy (badtoken, checkref);
                break;
        case 13 :
                strcpy (badtoken, checktype);
                break;
        default :
                strcpy (badtoken, yytext);
        }
printf ("%-20s %s\n", numdoccour, badtoken);
```

```
printf ("          %s\n", message[numsection][numerreur]);
}

int verifier (checkstring, filename, errgrave)
char checkstring[150];
char filename[30];
int errgrave;

{
int verif;
FILE *fileptr;
int pastrouve;
char stringlu[150];
char *result;

/*
printf ("\n%s%s.\n", "String a verifier :", checkstring);
*/
fileptr = fopen (filename, "r");
if (fileptr == NULL) return (5);
else    {
        pastrouve = -1;
        result = fgets (stringlu, 150, fileptr);
        while ((result != NULL) && (pastrouve < 0))
                {
                int lastblank;

/*
                printf ("Stringlu original :%s.\n", stringlu);
*/
                stringlu [strlen (stringlu) - 1] = '\0';
                while (stringlu[0] == ' ') strcpy (stringlu, &stringlu [1
                lastblank = strlen (stringlu) - 1;
                while (stringlu [lastblank] == ' ')
                        {
                        stringlu [lastblank] = '\0';
                        lastblank--;
                        }
/*
                printf ("Stringlu modifie  :%s.\n", stringlu);
*/
                if (strcmp (filename, "KEYWORDS") == 0)
                        {
                        stringlu [2] = ' ';
                        stringlu [5] = '\0';
                        }
                if (strcmp (filename, "COUNTRY") == 0)
                        {
                        char *temp;

                        temp = strchr (stringlu, ' ');
                        *temp = '\0';
                        /* printf ("Country lu et tronque :%s.\n", string
                        }
                pastrouve = strcmp (stringlu, checkstring);
/*
                printf ("pastrouve :%d.\n", pastrouve);
*/
                result = fgets (stringlu, 150, fileptr);
                }
        verif = fclose (fileptr);
        if (verif == -1) return (6);
        if (pastrouve)
                if (errgrave) return (4);
                else return (12);
        else return (0);
```

```
        }
}

void statdisplay ()

{
FILE *chris;
char bidon[100];

printf ("\n\n");
printf ("%s", "Pressez RETURN pour avoir le releve des markers ...");
chris = fopen("/dev/tty", "r");
if (chris == NULL)
        {printf("\nErreur lors de l'ouverture de chris.\n"); exit(1);}
fgets (bidon, 100, chris);
fclose (chris);
printf ("\f");
printf ("Releve des markers\n");
printf ("------------------\n\n");
printf ("%-18s%8s%5d fois.\n", "$$#", "present", nbnewchapcount);
printf ("%-18s%8s%5d fois.\n", "$$A", "present", nbnewartcount);
printf ("%-18s%8s%5d fois.\n", "$$T", "present", nbtitlecount);
printf ("%-18s%8s%5d fois.\n", "$$N NUMDOC", "present", nbnumdoccount);
printf ("%-18s%8s%5d fois.\n", "$$N DATE", "present", nbdatecount);
printf ("%-18s%8s%5d fois.\n", "$$N COTE", "present", nbcotecount);
printf ("%-18s%8s%5d fois.\n", "$$N CODE", "present", nbcodecount);
printf ("%-18s%8s%5d fois.\n", "$$N TITRE", "present", nbtitrecount);
printf ("%-18s%8s%5d fois.\n", "$$N REFERENCES", "present", nbrefcount);
printf ("%-18s%8s%5d fois.\n", "$$N TEXTE", "present", nbtextecount);
printf ("%-18s%8s%5d fois.\n", "$$N CONCLUSION", "present", nbconclusionc
printf ("%-18s%8s%5d fois.\n", "$$N LANGUE", "present", nblanguecount);
printf ("%-18s%8s%5d fois.\n", "$$N DOCNUM", "present", nbdocnumcount);
printf ("%-18s%8s%5d fois.\n", "$$N DAT", "present", nbdatcount);
printf ("%-18s%8s%5d fois.\n", "$$N PUB_REF", "present", nbpubrefcount);
printf ("%-18s%8s%5d fois.\n", "$$N TYPE", "present", nbtypecount);
printf ("%-18s%8s%5d fois.\n", "$$N PRODUCT", "present", nbproductcount);
printf ("%-18s%8s%5d fois.\n", "$$N COUNTRY", "present", nbcountrycount);
printf ("%-18s%8s%5d fois.\n", "$$N KEYWORDS", "present", nbkeywordscount
printf ("%-18s%8s%5d fois.\n", "$$P", "present", nbnewparcount);
printf ("\n%s", "Pressez RETURN pour continuer ...");
chris = fopen("/dev/tty", "r");
if (chris == NULL)
        {printf("\nErreur lors de l'ouverture de chris.\n"); exit(1);}
fgets (bidon, 100, chris);
fclose (chris);
}

yyerror (s)
char *s;
{
erreur (globnumsect, globnumerr);
yyerrok;
}

main ()
{
int returnvalue;

printf ("\n\njur.exe\n");
returnvalue = yyparse ();
statdisplay ();
return (returnvalue);
}
```

```
# line 2 "jur.yacc"
#include <string.h>
#include <fcntl.h>
#include <stdio.h>
#define grave 1
#define pasgrave 0
#define nbsect 17
#define nberr 14
#define longmess 70
#include "messages"
char checkref[150];
char checktype[150];
char numdoccour[25] = {"00000000000000000000"};
char badtoken[150];
char oldkeyw[4];
int linenb = 1;
int globnumsect    = 0;
int globnumerr     = 0;
int nbnewchapcount = 0;
int nbnewartcount  = 0;
int nbtitlecount   = 0;
int nbnumdoccount  = 0;
int nbdatecount    = 0;
int nbcotecount    = 0;
int nbcodecount    = 0;
int nbtitrecount   = 0;
int nbrefcount     = 0;
int nbtextecount   = 0;
int nbconclusioncount = 0;
int nblanguecount  = 0;
int nbdocnumcount  = 0;
int nbdatcount     = 0;
int nbpubrefcount  = 0;
int nbtypecount    = 0;
int nbproductcount = 0;
int nbcountrycount = 0;
int nbkeywordscount = 0;
int nbnewparcount  = 0;
# define NEWCHAP 257
# define NEWART 258
# define TITLE 259
# define NEWNUMDOC 260
# define NEWDATE 261
# define NEWCOTE 262
# define NEWCODE 263
# define NEWTITRE 264
# define NEWREF 265
# define NEWTEXTE 266
# define NEWCONC 267
# define NEWLANG 268
# define NEWDOCNUM 269
# define NEWDAT 270
# define NEWPUB_REF 271
# define NEWTYPE 272
# define NEWPRODUCT 273
# define NEWCOUNTRY 274
# define NEWKEYW 275
# define NEWPAR 276
# define NEWLINE 277
# define TAB 278
# define SP 279
# define DATE 280
# define COTEjk 281
# define COTEn 282
# define CODE 283
```

```
# define NUMDOCj 284
# define NUMDOCk 285
# define NUMDOCn 286
# define DOCNUM 287
# define PUBREF 288
# define KEYW1 289
# define KEYW2 290
# define WORD 291
# define LESS 292
# define DOLLAR 293
#define yyclearin yychar = -1
#define yyerrok yyerrflag = 0
extern int yychar;
extern int yyerrflag;
#ifndef YYMAXDEPTH
#define YYMAXDEPTH 150
#endif
#ifndef YYSTYPE
#define YYSTYPE int
#endif
YYSTYPE yylval, yyval;
typedef int yytabelem;
# define YYERRCODE 256

# line 894 "jur.yacc"
        /* PROGRAMS SECTION */

void erreur (int, int);
int verifier (char *, char *, int);
void statdisplay ();

#include "jur.lex.yy.c"

void erreur (numsection, numerreur)
int numsection, numerreur;
{
printf ("\n");
printf ("%s%d\n", "Ligne incriminee : ", yylineno);
if (numerreur < 10) printf ("%-15s", "Erreur");
else printf ("%-15s", "Avertissement");
switch (numsection)
        {
        case 1 :
                strcpy (badtoken, yytext);
                strcpy (numdoccour, yytext);
                break;
        case 6 :
                strcpy (badtoken, checkref);
                break;
        case 13 :
                strcpy (badtoken, checktype);
                break;
        default :
                strcpy (badtoken, yytext);
        }
printf ("%-20s %s\n", numdoccour, badtoken);
printf ("          %s\n", message[numsection][numerreur]);
}

int verifier (checkstring, filename, errgrave)
char checkstring[150];
char filename[30];
int errgrave;

{
int verif;
```

```
FILE *fileptr;
int pastrouve;
char stringlu[150];
char *result;

/*
printf ("\n%s%s.\n", "String a verifier :", checkstring);
*/
fileptr = fopen (filename, "r");
if (fileptr == NULL) return (5);
else    {
        pastrouve = -1;
        result = fgets (stringlu, 150, fileptr);
        while ((result != NULL) && (pastrouve < 0))
                {
                int lastblank;

/*
                printf ("Stringlu original :%s.\n", stringlu);
*/
                stringlu [strlen (stringlu) - 1] = '\0';
                while (stringlu[0] == ' ') strcpy (stringlu, &stringlu [1
                lastblank = strlen (stringlu) - 1;
                while (stringlu [lastblank] == ' ')
                        {
                        stringlu [lastblank] = '\0';
                        lastblank--;
                        }
/*
                printf ("Stringlu modifie  :%s.\n", stringlu);
*/
                if (strcmp (filename, "KEYWORDS") == 0)
                        {
                        stringlu [2] = ' ';
                        stringlu [5] = '\0';
                        }
                if (strcmp (filename, "COUNTRY") == 0)
                        {
                        char *temp;

                        temp = strchr (stringlu, ' ');
                        *temp = '\0';
                        /* printf ("Country lu et tronque :%s.\n", string
                        }
                pastrouve = strcmp (stringlu, checkstring);
/*
                printf ("pastrouve :%d.\n", pastrouve);
*/
                result = fgets (stringlu, 150, fileptr);
                }
        verif = fclose (fileptr);
        if (verif == -1) return (6);
        if (pastrouve)
                if (errgrave) return (4);
                else return (12);
        else return (0);
        }
}

void statdisplay ()

{
FILE *chris;
char bidon[100];

printf ("\n\n");
```

```
printf ("%s", "Pressez RETURN pour avoir le releve des markers ...");
chris = fopen("/dev/tty", "r");
if (chris == NULL)
        {printf("\nErreur lors de l'ouverture de chris.\n"); exit(1);}
fgets (bidon, 100, chris);
fclose (chris);
printf ("\f");
printf ("Releve des markers\n");
printf ("------------------\n\n");
printf ("%-18s%8s%5d fois.\n", "$$#", "present", nbnewchapcount);
printf ("%-18s%8s%5d fois.\n", "$$A", "present", nbnewartcount);
printf ("%-18s%8s%5d fois.\n", "$$T", "present", nbtitlecount);
printf ("%-18s%8s%5d fois.\n", "$$N NUMDOC", "present", nbnumdoccount);
printf ("%-18s%8s%5d fois.\n", "$$N DATE", "present", nbdatecount);
printf ("%-18s%8s%5d fois.\n", "$$N COTE", "present", nbcotecount);
printf ("%-18s%8s%5d fois.\n", "$$N CODE", "present", nbcodecount);
printf ("%-18s%8s%5d fois.\n", "$$N TITRE", "present", nbtitrecount);
printf ("%-18s%8s%5d fois.\n", "$$N REFERENCES", "present", nbrefcount);
printf ("%-18s%8s%5d fois.\n", "$$N TEXTE", "present", nbtextecount);
printf ("%-18s%8s%5d fois.\n", "$$N CONCLUSION", "present", nbconclusionc
printf ("%-18s%8s%5d fois.\n", "$$N LANGUE", "present", nblanguecount);
printf ("%-18s%8s%5d fois.\n", "$$N DOCNUM", "present", nbdocnumcount);
printf ("%-18s%8s%5d fois.\n", "$$N DAT", "present", nbdatcount);
printf ("%-18s%8s%5d fois.\n", "$$N PUB_REF", "present", nbpubrefcount);
printf ("%-18s%8s%5d fois.\n", "$$N TYPE", "present", nbtypecount);
printf ("%-18s%8s%5d fois.\n", "$$N PRODUCT", "present", nbproductcount);
printf ("%-18s%8s%5d fois.\n", "$$N COUNTRY", "present", nbcountrycount);
printf ("%-18s%8s%5d fois.\n", "$$N KEYWORDS", "present", nbkeywordscount
printf ("%-18s%8s%5d fois.\n", "$$P", "present", nbnewparcount);
printf ("\n%s", "Pressez RETURN pour continuer ...");
chris = fopen("/dev/tty", "r");
if (chris == NULL)
        {printf("\nErreur lors de l'ouverture de chris.\n"); exit(1);}
fgets (bidon, 100, chris);
fclose (chris);
}

yyerror (s)
char *s;
{
erreur (globnumsect, globnumerr);
yyerrok;
}

main ()
{
int returnvalue;

printf ("\n\njur.exe\n");
returnvalue = yyparse ();
statdisplay ();
return (returnvalue);
}
yytabelem yyexca[] ={
-1, 1,
        0, -1,
        -2, 0,
-1, 85,
        0, 1,
        -2, 0,
-1, 99,
        276, 66,
        -2, 63,
        };
# define YYNPROD 231
# define YYLAST 463
```

```
yytabelem yyact[]={

   123,   344,   101,   335,   321,   304,   124,   121,   350,   349,
   346,   352,   353,   354,   355,   356,   357,   358,   359,   360,
   361,   362,   351,   348,   347,   232,   231,   228,   234,   235,
   236,   237,   238,   239,   240,   241,   242,   243,   244,   233,
   230,   229,   170,   169,   166,   172,   173,   174,   175,   176,
   177,   178,   179,   180,   181,   182,   171,   168,   167,   102,
   336,   305,   284,   299,   265,   264,   292,    78,   153,   151,
   155,   156,   157,   158,   159,   160,   161,   162,   163,   164,
   165,   154,   269,   152,   323,   322,   105,   104,   105,   104,
   105,   104,   407,   196,   118,    84,   183,   324,   271,   328,
   128,   147,   281,   103,   108,   345,   289,   406,   405,   404,
   403,   402,   401,   396,   395,   394,   393,   392,   391,   386,
   385,   384,   383,   382,   377,   376,   375,   374,   367,   363,
   342,   307,   297,   201,    97,   410,   399,   389,   380,   372,
   369,   343,   333,   327,   326,   311,   310,   306,   295,   294,
   293,   291,   288,   287,   286,    96,   285,   272,   267,   262,
   225,   223,   218,   199,   197,   195,   190,   185,   184,   149,
   130,   129,   116,   106,    82,    75,    72,    63,    58,    54,
    45,    39,    35,    33,    28,    14,     6,   366,   275,   273,
   217,   148,   135,   302,   317,   189,   194,   279,    74,   222,
    81,    92,   139,   100,   246,    91,    57,    80,   193,   188,
   301,   134,   316,   278,   221,   138,    56,    38,   143,    23,
    44,    18,    62,    17,   142,    43,    37,    22,    61,    10,
     3,    53,    13,    27,    34,    52,    41,    26,     7,    13,
    19,    19,     4,     3,     7,   150,    73,   274,    12,   120,
    55,    42,    16,    60,    36,    21,   308,   276,   263,     2,
   219,   133,   334,   337,    32,     8,   318,   332,   331,   330,
   312,   303,   280,    40,   268,    29,   282,   224,   315,   309,
   300,   290,   277,    47,   266,   220,   187,    49,   137,   136,
   283,   270,   226,   227,   144,    68,   192,   141,    90,    70,
    89,   145,    95,    71,    69,    51,    65,    50,   119,   314,
   313,   198,    98,   216,   325,   215,   214,   320,   319,    86,
    93,   213,   329,    88,   212,    94,   211,   210,   209,   208,
   207,   206,   205,   204,   112,   338,   339,   340,   114,   203,
   117,   202,   115,   113,   125,   364,   400,   390,   381,   365,
   373,   341,   296,   200,   298,   122,    99,    76,   126,    77,
    59,   127,   107,   131,   140,    83,   109,   368,    64,   146,
   370,   371,    79,    25,    24,   132,   111,    87,    67,   379,
   378,    48,    31,    20,    11,   186,   110,   191,   388,   387,
    66,    30,     9,    85,    46,    15,     5,     1,   398,   397,
     0,     0,     0,   245,     0,     0,     0,     0,     0,   409,
   408,     0,     0,     0,     0,     0,     0,     0,     0,     0,
     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
     0,     0,     0,     0,     0,     0,     0,     0,   247,   248,
   249,   250,   251,   252,   253,   254,   255,   256,   257,   258,
   259,   260,   261 };
yytabelem yypact[]={

   -14, -1000, -1000,   -91,   -13,   -27, -1000,   -92, -1000,   -35,
   -19,   -37,   -23,   -93, -1000,   -27, -1000,   -94,   -24,   -95,
 -1000,   -39,   -96, -1000,   -20,   -36,   -97, -1000, -1000, -1000,
   -35,   -37,   -25, -1000,   -98, -1000,   -50,   -99, -1000, -1000,
 -1000,   -18,   -34,  -100, -1000, -1000,   -27, -1000, -1000,   -39,
   -20,   -36,  -101, -1000, -1000,   -69,  -102, -1000, -1000, -1000,
   -56,  -103, -1000, -1000,  -189, -1000,   -35,   -37,   -55,   -50,
 -1000,   -34, -1000, -1000,  -122, -1000,  -232, -1000,  -188, -1000,
  -104, -1000, -1000,  -176, -1000,   -20, -1000, -1000,   -39,   -20,
   -36,  -105, -1000,   -69, -1000,  -191, -1000, -1000, -1000, -1000,
 -1000,  -286, -1000, -1000, -1000, -1000, -1000,  -181,  -106,  -107,
   -35,   -64,   -54,   -50, -1000,   -38, -1000, -1000, -1000, -1000,
 -1000,  -190,   -85,  -108, -1000,  -210,  -235,  -187,  -109, -1000,
 -1000, -1000, -1000, -1000,  -110, -1000,   -20,   -61,  -111, -1000,
   -69,   -60,  -112, -1000,  -193,  -113, -1000, -1000,  -114,  -145,
```

C4/5

```
 -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000,
 -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000,
   -86, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000,
 -1000, -1000, -1000,  -115, -1000, -1000, -1000,   -57,  -116, -1000,
 -1000, -1000, -1000,  -117, -1000, -1000, -1000, -1000,  -252,  -232,
 -1000,  -232,  -210,  -210,  -210,  -210,  -210,  -210,  -210,  -210,
  -210,  -210,  -210,  -210,  -210,  -210,  -210,  -118, -1000,  -225,
 -1000,  -119, -1000, -1000,  -205, -1000,  -184,  -120, -1000, -1000,
 -1000, -1000,   -87, -1000, -1000, -1000, -1000, -1000, -1000, -1000,
 -1000, -1000, -1000, -1000, -1000, -1000,  -286, -1000, -1000, -1000,
 -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000,
 -1000, -1000, -1000,   -88, -1000, -1000,   -59, -1000,  -178, -1000,
  -229,  -121, -1000,  -123,  -124,  -125,  -173, -1000,  -126, -1000,
  -222,  -127,  -128,  -129, -1000, -1000, -1000,  -146,  -225,  -227,
   -63,  -230,  -130, -1000, -1000, -1000, -1000,  -147, -1000, -1000,
 -1000,  -131, -1000,  -132, -1000, -1000, -1000,  -232,  -286,   -62,
 -1000,  -230,  -194,  -286,  -133, -1000,  -134, -1000, -1000,  -192,
 -1000,  -286, -1000, -1000, -1000,  -135, -1000,  -231, -1000, -1000,
  -194,  -194,  -194,  -148,  -136,  -174, -1000,  -269, -1000, -1000,
 -1000, -1000,  -149, -1000,  -286,  -231, -1000, -1000, -1000, -1000,
   -89, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000, -1000,
 -1000, -1000, -1000,  -150, -1000,  -174,  -137,  -232, -1000, -1000,
  -286,  -138,  -151, -1000,  -152,  -153,  -154,  -232,  -286,  -139,
  -155, -1000,  -156,  -157,  -158,  -159,  -232,  -286,  -140,  -160,
 -1000,  -161,  -162,  -163,  -164,  -165,  -232,  -286,  -141,  -166,
 -1000,  -167,  -168,  -169,  -170,  -171,  -186,  -232,  -286,  -142,
 -1000 };
yytabelem yypgo[]={

     0,   397,   259,   396,   395,   394,   393,   392,   252,   391,
   390,   386,   384,   383,   382,   381,   378,   377,   376,   375,
   248,   374,   373,   251,   253,   372,   368,   366,   365,   362,
   361,   255,   254,   250,   246,   360,   359,     7,   358,   357,
   356,   203,   355,     2,     0,   353,   352,   351,   350,   348,
   347,   346,   344,   245,   341,   339,   333,   332,   331,   330,
   329,   328,   327,   326,   324,   321,   316,   315,   313,   312,
   249,   311,   308,   307,   305,   302,   301,   300,   298,   297,
   296,   294,   293,   292,   291,   290,   289,   288,   286,   285,
   284,   282,   281,   280,   279,   278,   277,   276,   274,   272,
   271,     5,   270,     4,   269,   268,   267,   266,   263,   262,
     3,     1,   261,   260,   258,    65,   257,   256 };
yytabelem yyr1[]={

     0,     1,     2,     2,     3,     3,     4,     4,     5,     5,
     6,     6,     8,     8,     7,     9,    10,    11,    12,    20,
    20,    21,    26,    27,    22,    22,    28,    23,    23,    29,
    24,    24,    30,    25,    25,    13,    35,    31,    31,    36,
    37,    37,    37,    38,    38,    38,    38,    38,    38,    38,
    38,    38,    38,    38,    38,    38,    38,    38,    38,    38,
    38,    38,    39,    32,    32,    40,    42,    40,    41,    45,
    45,    46,    46,    47,    47,    48,    48,    49,    49,    50,
    50,    51,    51,    52,    43,    53,    54,    53,    55,    53,
    56,    53,    57,    53,    58,    53,    59,    53,    60,    53,
    61,    53,    62,    53,    63,    53,    64,    53,    65,    53,
    66,    53,    67,    53,    68,    53,    44,    44,    69,    33,
    33,    70,    71,    71,    71,    71,    71,    71,    71,    71,
    71,    71,    71,    71,    71,    71,    71,    71,    71,    71,
    71,    34,    72,    34,    34,    14,    73,    75,    76,    74,
    74,    15,    16,    77,    81,    82,    78,    78,    83,    79,
    79,    84,    80,    80,    85,    17,    18,    90,    92,    94,
    86,    96,    97,    87,    87,    98,    88,    88,    99,    89,
    89,    91,    91,   100,   100,   102,   101,   103,   104,   103,
   105,   103,   106,   103,    93,    93,   107,   108,   108,   108,
   108,   108,   108,   108,   108,   108,   108,   108,   108,   108,
   108,   108,   108,   108,   108,   108,    95,    95,   109,   110,
   111,   111,    19,   113,   112,   112,   114,   114,   116,   117,
```

```
   115 };
yytabelem yyr2[]={

      0,    16,     4,     6,     0,     6,     0,     6,     0,     6,
      0,     6,     4,     6,     4,     4,     4,     4,     6,     4,
      6,     9,     1,     1,    12,     2,     1,    10,     2,     1,
     10,     2,     1,    10,     2,     9,     1,     8,     2,     6,
      0,     4,     4,     0,     4,     4,     4,     4,     4,     4,
      8,     4,     4,     4,     4,     4,     4,     4,     4,     4,
      4,     4,     1,     8,     2,     2,     1,    10,     9,     0,
     10,     0,    12,     0,    14,     0,    16,     0,    18,     0,
     20,     0,    20,     1,     7,     0,     1,     6,     1,     6,
      1,     6,     1,     6,     1,     6,     1,     6,     1,     6,
      1,     6,     1,     6,     1,     6,     1,     6,     1,     6,
      1,     6,     1,     6,     1,     6,     0,     2,     1,     8,
      2,     6,     0,     4,     4,     4,     4,     4,     4,     8,
      4,     4,     4,     4,     4,     4,     4,     4,     4,     4,
      4,     0,     1,     8,     4,     6,     7,     1,     1,    12,
      2,     9,     6,     9,     1,     1,    12,     2,     1,    10,
      2,     1,    10,     2,     3,     9,     6,     1,     1,     1,
     19,     1,     1,    12,     2,     1,    10,     2,     1,    10,
      2,     8,     2,     6,     2,     1,     9,     0,     1,     6,
      1,     6,     1,     6,     6,     2,     6,     0,     4,     4,
      4,     4,     4,     4,     8,     4,     4,     4,     4,     4,
      4,     4,     4,     4,     4,     4,     8,     2,     6,     3,
      0,     6,     3,     1,     8,     2,     2,     8,     1,     1,
     14 };
yytabelem yychk[]={

  -1000,    -1,    -2,   257,   256,    -3,   277,   257,    -2,    -7,
    256,   -12,   -20,   259,   277,    -4,    -8,   258,   256,   259,
    -13,   -31,   264,   256,   -21,   -22,   260,   256,   277,    -2,
     -9,   -14,   -20,   277,   258,   277,   -32,   265,   256,   277,
    -20,   256,   -23,   261,   256,   277,    -5,    -8,   -15,   -31,
    -73,   -74,   260,   256,   277,   -33,   266,   256,   277,   -35,
    -24,   262,   256,   277,   -26,    -2,   -10,   -16,   -20,   -32,
    -20,   -23,   277,   -34,   267,   277,   -39,   -36,   -37,   -25,
    263,   256,   277,   -28,   284,    -6,    -8,   -17,   -31,   -77,
    -78,   260,   256,   -33,   -24,   -75,   277,   256,   -69,   -40,
    -41,   -43,   291,   291,   279,   278,   277,   -29,   280,   -27,
    -11,   -18,   -20,   -32,   -20,   -23,   277,   -34,   285,   -72,
    -70,   -37,   -42,   -44,   292,   -52,   -38,   -30,   281,   277,
    277,    -8,   -19,  -112,   275,   256,   -86,   -87,   269,   256,
    -33,   -79,   262,   256,   -81,   -76,   -70,   291,   276,   277,
    -53,   279,   293,   278,   291,   280,   281,   282,   283,   284,
    285,   286,   287,   288,   289,   290,   279,   293,   292,   278,
    277,   291,   280,   281,   282,   283,   284,   285,   286,   287,
    288,   289,   290,   283,   277,   277,   -20,   -88,   270,   256,
    277,   -34,   -80,   268,   256,   277,   286,   277,   -71,   277,
    -45,   278,   -54,   -55,   -56,   -57,   -58,   -59,   -60,   -61,
    -62,   -63,   -64,   -65,   -66,   -67,   -68,   276,   277,  -113,
    -89,   271,   256,   277,   -96,   277,   -83,   -82,   279,   293,
    292,   278,   277,   291,   280,   281,   282,   283,   284,   285,
    286,   287,   288,   289,   290,   -41,   -43,   -53,   -53,   -53,
    -53,   -53,   -53,   -53,   -53,   -53,   -53,   -53,   -53,   -53,
    -53,   -53,   277,  -114,  -115,   289,   -90,   277,   -98,   287,
    -84,   282,   277,   276,   -44,   276,  -116,   -91,   272,   256,
    -99,   280,   -97,   -85,   291,   277,   277,   277,   277,   279,
    -92,   277,   288,   277,   277,   277,   -46,   278,  -115,   290,
    -93,   273,   256,  -100,  -101,   291,   277,   278,  -117,   -94,
    277,   277,  -102,   -43,   -44,   -95,   274,   256,  -107,   -37,
   -101,  -103,   279,   278,   291,   -44,   277,   277,   291,   -44,
   -104,  -105,  -106,   277,  -109,  -110,   291,  -108,  -103,  -103,
   -103,   -47,   278,   277,  -111,   279,   279,   293,   292,   278,
    277,   291,   280,   281,   282,   283,   284,   285,   286,   287,
    288,   289,   290,   278,   -44,  -110,   276,   278,  -111,   277,
```

```
    -43,    -44,    277,    -48,    278,    278,    278,    278,    -43,    -44,
    277,    -49,    278,    278,    278,    278,    278,    -43,    -44,    277,
    -50,    278,    278,    278,    278,    278,    278,    -43,    -44,    277,
    -51,    278,    278,    278,    278,    278,    278,    278,    -43,    -44,
    277 };
yytabelem yydef[]={

      0,     -2,      4,      0,      0,      0,      2,      0,      6,      0,
      0,      0,      0,      0,      3,      0,      5,      0,      0,      0,
     14,      0,      0,     38,      0,      0,      0,     25,     19,      8,
      0,      0,      0,     12,      0,     20,      0,      0,     64,     36,
     18,      0,      0,      0,     28,     22,      0,      7,     15,      0,
      0,      0,      0,    150,     13,    141,      0,    120,     62,     40,
      0,      0,     31,     26,      0,     10,      0,      0,      0,      0,
    145,      0,    147,     35,      0,    118,      0,     37,      0,     21,
      0,     34,     29,      0,     23,     -2,      9,     16,      0,      0,
      0,      0,    157,    141,    146,      0,    142,    144,     40,     -2,
     65,    116,     83,     43,     41,     42,     32,      0,      0,      0,
      0,      0,      0,      0,    152,      0,    154,    151,    148,     40,
    119,      0,      0,      0,    117,     85,     39,      0,      0,     27,
     24,     11,     17,    222,      0,    225,      0,      0,      0,    174,
    141,      0,      0,    160,      0,      0,    143,    122,      0,     69,
     84,     86,     88,     90,     92,     94,     96,     98,    100,    102,
    104,    106,    108,    110,    112,    114,     44,     45,     46,     47,
     48,     49,     51,     52,     53,     54,     55,     56,     57,     58,
     59,     60,     61,      0,     30,    223,    166,      0,      0,    177,
    171,    165,    153,      0,    163,    158,    155,    149,    121,      0,
     68,      0,     85,     85,     85,     85,     85,     85,     85,     85,
     85,     85,     85,     85,     85,     85,     85,      0,     33,      0,
    167,      0,    180,    175,      0,    161,      0,      0,    123,    124,
    125,    126,    127,    128,    130,    131,    132,    133,    134,    135,
    136,    137,    138,    139,    140,     67,    116,     87,     89,     91,
     93,     95,     97,     99,    101,    103,    105,    107,    109,    111,
    113,    115,     50,    224,    226,    228,      0,    178,      0,    172,
      0,      0,    156,      0,      0,      0,      0,    168,      0,    182,
      0,      0,      0,      0,    164,    159,    129,     71,      0,      0,
      0,      0,      0,    176,    173,    162,     70,      0,    227,    229,
    169,      0,    195,      0,    184,    185,    179,      0,    116,      0,
     40,    181,    187,    116,      0,    170,      0,    217,    194,      0,
    183,    116,    188,    190,    192,      0,    230,      0,    197,    186,
    187,    187,    187,     73,      0,    220,    219,    196,    189,    191,
    193,     72,      0,    216,    116,      0,    198,    199,    200,    201,
    202,    203,    205,    206,    207,    208,    209,    210,    211,    212,
    213,    214,    215,      0,    218,    220,      0,      0,    221,    204,
    116,      0,     75,     74,      0,      0,      0,      0,    116,      0,
     77,     76,      0,      0,      0,      0,      0,    116,      0,     79,
     78,      0,      0,      0,      0,      0,      0,    116,      0,     81,
     80,      0,      0,      0,      0,      0,      0,      0,    116,      0,
     82 };
typedef struct { char *t_name; int t_val; } yytoktype;
#ifndef YYDEBUG
#       define YYDEBUG  1       /* allow debugging */
#endif

#if YYDEBUG

yytoktype yytoks[] =
{
        "NEWCHAP",      257,
        "NEWART",       258,
        "TITLE",        259,
        "NEWNUMDOC",    260,
        "NEWDATE",      261,
        "NEWCOTE",      262,
        "NEWCODE",      263,
        "NEWTITRE",     264,
```

```
        "NEWREF",       265,
        "NEWTEXTE",     266,
        "NEWCONC",      267,
        "NEWLANG",      268,
        "NEWDOCNUM",    269,
        "NEWDAT",       270,
        "NEWPUB_REF",   271,
        "NEWTYPE",      272,
        "NEWPRODUCT",   273,
        "NEWCOUNTRY",   274,
        "NEWKEYW",      275,
        "NEWPAR",       276,
        "NEWLINE",      277,
        "TAB",  278,
        "SP",   279,
        "DATE", 280,
        "COTEjk",       281,
        "COTEn",        282,
        "CODE", 283,
        "NUMDOCj",      284,
        "NUMDOCk",      285,
        "NUMDOCn",      286,
        "DOCNUM",       287,
        "PUBREF",       288,
        "KEYW1",        289,
        "KEYW2",        290,
        "WORD", 291,
        "LESS", 292,
        "DOLLAR",       293,
        "-unknown-",    -1      /* ends search */
};

char * yyreds[] =
{
        "-no such reduction-",
        "textin : chaphead chapter1 chaphead chapter2 chaphead chapter3 c
        "chaphead : NEWCHAP NEWLINE",
        "chaphead : error NEWCHAP NEWLINE",
        "chapter1 : /* empty */",
        "chapter1 : chapter1 article1 artfooter",
        "chapter2 : /* empty */",
        "chapter2 : chapter2 article2 artfooter",
        "chapter3 : /* empty */",
        "chapter3 : chapter3 article3 artfooter",
        "chapter4 : /* empty */",
        "chapter4 : chapter4 article4 artfooter",
        "artfooter : NEWART NEWLINE",
        "artfooter : error NEWART NEWLINE",
        "article1 : title1 body1",
        "article2 : title2 body2",
        "article3 : title3 body3",
        "article4 : title4 body4",
        "title1 : titlehead contitle1 titlehead",
        "titlehead : TITLE NEWLINE",
        "titlehead : error TITLE NEWLINE",
        "contitle1 : numdoc1 date cote12 code",
        "numdoc1 : NEWNUMDOC NEWLINE",
        "numdoc1 : NEWNUMDOC NEWLINE NUMDOCj",
        "numdoc1 : NEWNUMDOC NEWLINE NUMDOCj NEWLINE",
        "numdoc1 : error",
        "date : NEWDATE NEWLINE",
        "date : NEWDATE NEWLINE DATE NEWLINE",
        "date : error",
        "cote12 : NEWCOTE NEWLINE",
        "cote12 : NEWCOTE NEWLINE COTEjk NEWLINE",
        "cote12 : error",
```

```
"code : NEWCODE NEWLINE",
"code : NEWCODE NEWLINE CODE NEWLINE",
"code : error",
"body1 : titre references texte conclusion",
"titre : NEWTITRE NEWLINE",
"titre : NEWTITRE NEWLINE contitre",
"titre : error",
"contitre : optnlspaces WORD restoftitre",
"optnlspaces : /* empty */",
"optnlspaces : optnlspaces SP",
"optnlspaces : optnlspaces TAB",
"restoftitre : /* empty */",
"restoftitre : restoftitre SP",
"restoftitre : restoftitre DOLLAR",
"restoftitre : restoftitre LESS",
"restoftitre : restoftitre TAB",
"restoftitre : restoftitre NEWLINE",
"restoftitre : restoftitre WORD",
"restoftitre : restoftitre NEWLINE NEWPAR NEWLINE",
"restoftitre : restoftitre DATE",
"restoftitre : restoftitre COTEjk",
"restoftitre : restoftitre COTEn",
"restoftitre : restoftitre CODE",
"restoftitre : restoftitre NUMDOCj",
"restoftitre : restoftitre NUMDOCk",
"restoftitre : restoftitre NUMDOCn",
"restoftitre : restoftitre DOCNUM",
"restoftitre : restoftitre PUBREF",
"restoftitre : restoftitre KEYW1",
"restoftitre : restoftitre KEYW2",
"references : NEWREF NEWLINE",
"references : NEWREF NEWLINE conreferences",
"references : error",
"conreferences : reference",
"conreferences : conreferences",
"conreferences : conreferences NEWPAR NEWLINE reference",
"reference : refdescriptor optionalless NEWLINE lineref2",
"lineref2 : /* empty */",
"lineref2 : TAB refdescriptor optionalless NEWLINE lineref3",
"lineref3 : /* empty */",
"lineref3 : TAB TAB refdescriptor optionalless NEWLINE lineref4",
"lineref4 : /* empty */",
"lineref4 : TAB TAB TAB refdescriptor optionalless NEWLINE linere
"lineref5 : /* empty */",
"lineref5 : TAB TAB TAB TAB refdescriptor optionalless NEWLINE li
"lineref6 : /* empty */",
"lineref6 : TAB TAB TAB TAB TAB refdescriptor optionalless NEWLIN
"lineref7 : /* empty */",
"lineref7 : TAB TAB TAB TAB TAB TAB refdescriptor optionalless NE
"lineref8 : /* empty */",
"lineref8 : TAB TAB TAB TAB TAB TAB TAB refdescriptor optionalles
"refdescriptor : WORD",
"refdescriptor : WORD restofrefdes",
"restofrefdes : /* empty */",
"restofrefdes : SP",
"restofrefdes : SP restofrefdes",
"restofrefdes : DOLLAR",
"restofrefdes : DOLLAR restofrefdes",
"restofrefdes : TAB",
"restofrefdes : TAB restofrefdes",
"restofrefdes : WORD",
"restofrefdes : WORD restofrefdes",
"restofrefdes : DATE",
"restofrefdes : DATE restofrefdes",
"restofrefdes : COTEjk",
"restofrefdes : COTEjk restofrefdes",
```

```
"restofrefdes : COTEn",
"restofrefdes : COTEn restofrefdes",
"restofrefdes : CODE",
"restofrefdes : CODE restofrefdes",
"restofrefdes : NUMDOCj",
"restofrefdes : NUMDOCj restofrefdes",
"restofrefdes : NUMDOCk",
"restofrefdes : NUMDOCk restofrefdes",
"restofrefdes : NUMDOCn",
"restofrefdes : NUMDOCn restofrefdes",
"restofrefdes : DOCNUM",
"restofrefdes : DOCNUM restofrefdes",
"restofrefdes : PUBREF",
"restofrefdes : PUBREF restofrefdes",
"restofrefdes : KEYW1",
"restofrefdes : KEYW1 restofrefdes",
"restofrefdes : KEYW2",
"restofrefdes : KEYW2 restofrefdes",
"optionalless : /* empty */",
"optionalless : LESS",
"texte : NEWTEXTE NEWLINE",
"texte : NEWTEXTE NEWLINE contexte",
"texte : error",
"contexte : optnlspaces WORD restoftexte",
"restoftexte : /* empty */",
"restoftexte : restoftexte SP",
"restoftexte : restoftexte DOLLAR",
"restoftexte : restoftexte LESS",
"restoftexte : restoftexte TAB",
"restoftexte : restoftexte NEWLINE",
"restoftexte : restoftexte WORD",
"restoftexte : restoftexte NEWLINE NEWPAR NEWLINE",
"restoftexte : restoftexte DATE",
"restoftexte : restoftexte COTEjk",
"restoftexte : restoftexte COTEn",
"restoftexte : restoftexte CODE",
"restoftexte : restoftexte NUMDOCj",
"restoftexte : restoftexte NUMDOCk",
"restoftexte : restoftexte NUMDOCn",
"restoftexte : restoftexte DOCNUM",
"restoftexte : restoftexte PUBREF",
"restoftexte : restoftexte KEYW1",
"restoftexte : restoftexte KEYW2",
"conclusion : /* empty */",
"conclusion : NEWCONC NEWLINE",
"conclusion : NEWCONC NEWLINE contexte",
"conclusion : NEWCONC error",
"title2 : titlehead contitle2 titlehead",
"contitle2 : numdoc2 date cote12",
"numdoc2 : NEWNUMDOC NEWLINE",
"numdoc2 : NEWNUMDOC NEWLINE NUMDOCk",
"numdoc2 : NEWNUMDOC NEWLINE NUMDOCk NEWLINE",
"numdoc2 : error",
"body2 : titre references texte conclusion",
"title3 : titlehead contitle3 titlehead",
"contitle3 : numdoc3 date cote3 langue",
"numdoc3 : NEWNUMDOC NEWLINE",
"numdoc3 : NEWNUMDOC NEWLINE NUMDOCn",
"numdoc3 : NEWNUMDOC NEWLINE NUMDOCn NEWLINE",
"numdoc3 : error",
"cote3 : NEWCOTE NEWLINE",
"cote3 : NEWCOTE NEWLINE COTEn NEWLINE",
"cote3 : error",
"langue : NEWLANG NEWLINE",
"langue : NEWLANG NEWLINE conlangue NEWLINE",
"langue : error",
```

```
"conlangue : WORD",
"body3 : titre references texte conclusion",
"title4 : titlehead contitle4 titlehead",
"contitle4 : docnum dat pub_ref",
"contitle4 : docnum dat pub_ref type",
"contitle4 : docnum dat pub_ref type product",
"contitle4 : docnum dat pub_ref type product country",
"docnum : NEWDOCNUM NEWLINE",
"docnum : NEWDOCNUM NEWLINE DOCNUM",
"docnum : NEWDOCNUM NEWLINE DOCNUM NEWLINE",
"docnum : error",
"dat : NEWDAT NEWLINE",
"dat : NEWDAT NEWLINE DATE NEWLINE",
"dat : error",
"pub_ref : NEWPUB_REF NEWLINE",
"pub_ref : NEWPUB_REF NEWLINE PUBREF NEWLINE",
"pub_ref : error",
"type : NEWTYPE NEWLINE contype NEWLINE",
"type : error",
"contype : contype NEWLINE lignetype",
"contype : lignetype",
"lignetype : WORD",
"lignetype : WORD restoflntype optionalless",
"restoflntype : /* empty */",
"restoflntype : SP",
"restoflntype : SP restoflntype",
"restoflntype : TAB",
"restoflntype : TAB restoflntype",
"restoflntype : WORD",
"restoflntype : WORD restoflntype",
"product : NEWPRODUCT NEWLINE conproduct",
"product : error",
"conproduct : optnlspaces WORD restofproduct",
"restofproduct : /* empty */",
"restofproduct : restofproduct SP",
"restofproduct : restofproduct DOLLAR",
"restofproduct : restofproduct LESS",
"restofproduct : restofproduct TAB",
"restofproduct : restofproduct NEWLINE",
"restofproduct : restofproduct WORD",
"restofproduct : restofproduct NEWLINE NEWPAR NEWLINE",
"restofproduct : restofproduct DATE",
"restofproduct : restofproduct COTEjk",
"restofproduct : restofproduct COTEn",
"restofproduct : restofproduct CODE",
"restofproduct : restofproduct NUMDOCj",
"restofproduct : restofproduct NUMDOCk",
"restofproduct : restofproduct NUMDOCn",
"restofproduct : restofproduct DOCNUM",
"restofproduct : restofproduct PUBREF",
"restofproduct : restofproduct KEYW1",
"restofproduct : restofproduct KEYW2",
"country : NEWCOUNTRY NEWLINE concountry NEWLINE",
"country : error",
"concountry : countrycode restofcountry optionalless",
"countrycode : WORD",
"restofcountry : /* empty */",
"restofcountry : SP countrycode restofcountry",
"body4 : keywords",
"keywords : NEWKEYW NEWLINE",
"keywords : NEWKEYW NEWLINE conkeywords",
"keywords : error",
"conkeywords : keywordline",
"conkeywords : conkeywords NEWPAR NEWLINE keywordline",
"keywordline : KEYW1",
"keywordline : KEYW1 SP KEYW2",
```

```
        "keywordline : KEYW1 SP KEYW2 optionalless NEWLINE",
};
#endif /* YYDEBUG */
/*      @(#) yaccpar.src 1.2 88/10/25
 *
 *      Copyright (C) The Santa Cruz Operation, 1985.
 *      This Module contains Proprietary Information of
 *      The Santa Cruz Operation, Microsoft Corporation
 *      and AT&T, and should be treated as Confidential.
 */

/*
** Skeleton parser driver for yacc output
*/

/*
** yacc user known macros and defines
*/
#define YYERROR         goto yyerrlab
#define YYACCEPT        return(0)
#define YYABORT         return(1)
#define YYBACKUP( newtoken, newvalue )\
{\
        if ( yychar >= 0 || ( yyr2[ yytmp ] >> 1 ) != 1 )\
        {\
                yyerror( "syntax error - cannot backup" );\
                goto yyerrlab;\
        }\
        yychar = newtoken;\
        yystate = *yyps;\
        yylval = newvalue;\
        goto yynewstate;\
}
#define YYRECOVERING()  (!!yyerrflag)
#ifndef YYDEBUG
#       define YYDEBUG  1       /* make debugging available */
#endif

/*
** user known globals
*/
int yydebug;                    /* set to 1 to get debugging */

/*
** driver internal defines
*/
#define YYFLAG          (-1000)

/*
** global variables used by the parser
*/
YYSTYPE yyv[ YYMAXDEPTH ];      /* value stack */
int yys[ YYMAXDEPTH ];          /* state stack */

YYSTYPE *yypv;                  /* top of value stack */
int *yyps;                      /* top of state stack */

int yystate;                    /* current state */
int yytmp;                      /* extra var (lasts between blocks) */

int yynerrs;                    /* number of errors */
int yyerrflag;                  /* error recovery flag */
int yychar;                     /* current input token number */
```

```
/*
** yyparse - return 0 if worked, 1 if syntax error not recovered from
*/
int
yyparse()
{
        register YYSTYPE *yypvt;        /* top of value stack for $vars *

        /*
        ** Initialize externals - yyparse may be called more than once
        */
        yypv = &yyv[-1];
        yyps = &yys[-1];
        yystate = 0;
        yytmp = 0;
        yynerrs = 0;
        yyerrflag = 0;
        yychar = -1;

        goto yystack;
        {
                register YYSTYPE *yy_pv;        /* top of value stack */
                register int *yy_ps;            /* top of state stack */
                register int yy_state;          /* current state */
                register int  yy_n;             /* internal state number

                /*
                ** get globals into registers.
                ** branch to here only if YYBACKUP was called.
                */
        yynewstate:
                yy_pv = yypv;
                yy_ps = yyps;
                yy_state = yystate;
                goto yy_newstate;

                /*
                ** get globals into registers.
                ** either we just started, or we just finished a reductio
                */
        yystack:
                yy_pv = yypv;
                yy_ps = yyps;
                yy_state = yystate;

                /*
                ** top of for (;;) loop while no reductions done
                */
        yy_stack:
                /*
                ** put a state and value onto the stacks
                */
#if YYDEBUG
                /*
                ** if debugging, look up token value in list of value vs.
                ** name pairs.  0 and negative (-1) are special values.
                ** Note: linear search is used since time is not a real
                ** consideration while debugging.
                */
                if ( yydebug )
                {
                        register int yy_i;

                        printf( "State %d, token ", yy_state );
                        if ( yychar == 0 )
                                printf( "end-of-file\n" );
```

```
                        else if ( yychar < 0 )
                                printf( "-none-\n" );
                        else
                        {
                                for ( yy_i = 0; yytoks[yy_i].t_val >= 0;
                                        yy_i++ )
                                {
                                        if ( yytoks[yy_i].t_val == yychar
                                                break;
                                }
                                printf( "%s\n", yytoks[yy_i].t_name );
                        }
                }
#endif /* YYDEBUG */
                if ( ++yy_ps >= &yys[ YYMAXDEPTH ] )    /* room on stack?
                {
                        yyerror( "yacc stack overflow" );
                        YYABORT;
                }
                *yy_ps = yy_state;
                *++yy_pv = yyval;

                /*
                ** we have a new state - find out what to do
                */
        yy_newstate:
                if ( ( yy_n = yypact[ yy_state ] ) <= YYFLAG )
                        goto yydefault;         /* simple state */
#if YYDEBUG
                /*
                ** if debugging, need to mark whether new token grabbed
                */
                yytmp = yychar < 0;
#endif
                if ( ( yychar < 0 ) && ( ( yychar = yylex() ) < 0 ) )
                        yychar = 0;             /* reached EOF */
#if YYDEBUG
                if ( yydebug && yytmp )
                {
                        register int yy_i;

                        printf( "Received token " );
                        if ( yychar == 0 )
                                printf( "end-of-file\n" );
                        else if ( yychar < 0 )
                                printf( "-none-\n" );
                        else
                        {
                                for ( yy_i = 0; yytoks[yy_i].t_val >= 0;
                                        yy_i++ )
                                {
                                        if ( yytoks[yy_i].t_val == yychar
                                                break;
                                }
                                printf( "%s\n", yytoks[yy_i].t_name );
                        }
                }
#endif /* YYDEBUG */
                if ( ( ( yy_n += yychar ) < 0 ) || ( yy_n >= YYLAST ) )
                        goto yydefault;
                if ( yychk[ yy_n = yyact[ yy_n ] ] == yychar )  /*valid s
                {
                        yychar = -1;
                        yyval = yylval;
                        yy_state = yy_n;
                        if ( yyerrflag > 0 )
```

```
                        yyerrflag--;
                    goto yy_stack;
                }

        yydefault:
                if ( ( yy_n = yydef[ yy_state ] ) == -2 )
                {
#if YYDEBUG
                    yytmp = yychar < 0;
#endif
                    if ( ( yychar < 0 ) && ( ( yychar = yylex() ) < 0
                        yychar = 0;             /* reached EOF */
#if YYDEBUG
                    if ( yydebug && yytmp )
                    {
                        register int yy_i;

                        printf( "Received token " );
                        if ( yychar == 0 )
                            printf( "end-of-file\n" );
                        else if ( yychar < 0 )
                            printf( "-none-\n" );
                        else
                        {
                            for ( yy_i = 0;
                                yytoks[yy_i].t_val >= 0;
                                yy_i++ )
                            {
                                if ( yytoks[yy_i].t_val
                                    == yychar )
                                {
                                    break;
                                }
                            }
                            printf( "%s\n", yytoks[yy_i].t_na
                        }
                    }
#endif /* YYDEBUG */
                    /*
                    ** look through exception table
                    */
                    {
                        register int *yyxi = yyexca;

                        while ( ( *yyxi != -1 ) ||
                                ( yyxi[1] != yy_state ) )
                        {
                            yyxi += 2;
                        }
                        while ( ( *(yyxi += 2) >= 0 ) &&
                                ( *yyxi != yychar ) )
                                ;
                        if ( ( yy_n = yyxi[1] ) < 0 )
                            YYACCEPT;
                    }
                }

                /*
                ** check for syntax error
                */
                if ( yy_n == 0 )        /* have an error */
                {
                    /* no worry about speed here! */
                    switch ( yyerrflag )
                    {
                    case 0:         /* new error */
```

```
                        yyerror( "syntax error" );
                        goto skip_init;
                yyerrlab:
                        /*
                        ** get globals into registers.
                        ** we have a user generated syntax type e
                        */
                        yy_pv = yypv;
                        yy_ps = yyps;
                        yy_state = yystate;
                        yynerrs++;
                skip_init:
                case 1:
                case 2:         /* incompletely recovered error *
                                /* try again... */
                        yyerrflag = 3;
                        /*
                        ** find state where "error" is a legal
                        ** shift action
                        */
                        while ( yy_ps >= yys )
                        {
                                yy_n = yypact[ *yy_ps ] + YYERRCO
                                if ( yy_n >= 0 && yy_n < YYLAST &
                                        yychk[yyact[yy_n]] == YYE
                                        /*
                                        ** simulate shift of "err
                                        */
                                        yy_state = yyact[ yy_n ];
                                        goto yy_stack;
                                }
                                /*
                                ** current state has no shift on
                                ** "error", pop stack
                                */
#if YYDEBUG
#       define _POP_ "Error recovery pops state %d, uncovers state %d\n"
                                if ( yydebug )
                                        printf( _POP_, *yy_ps,
                                                yy_ps[-1] );
#       undef _POP_
#endif
                                yy_ps--;
                                yy_pv--;
                        }
                        /*
                        ** there is no state on stack with "error
                        ** a valid shift.  give up.
                        */
                        YYABORT;
                case 3:         /* no shift yet; eat a token */
#if YYDEBUG
                        /*
                        ** if debugging, look up token in list of
                        ** pairs.  0 and negative shouldn't occur
                        ** but since timing doesn't matter when
                        ** debugging, it doesn't hurt to leave th
                        ** tests here.
                        */
                        if ( yydebug )
                        {
                                register int yy_i;

                                printf( "Error recovery discards
                                if ( yychar == 0 )
                                        printf( "token end-of-fil
```

```
                                        else if ( yychar < 0 )
                                                printf( "token -none-\n"
                                        else
                                        {
                                                for ( yy_i = 0;
                                                        yytoks[yy_i].t_va
                                                        yy_i++ )
                                                {
                                                        if ( yytoks[yy_i]
                                                                == yychar
                                                        {
                                                                break;
                                                        }
                                                }
                                                printf( "token %s\n",
                                                        yytoks[yy_i].t_na
                                        }
                                }
#endif /* YYDEBUG */
                                if ( yychar == 0 )      /* reached EOF. q
                                        YYABORT;
                                yychar = -1;
                                goto yy_newstate;
                        }
                }/* end if ( yy_n == 0 ) */
                /*
                ** reduction by production yy_n
                ** put stack tops, etc. so things right after switch
                */
#if YYDEBUG
                /*
                ** if debugging, print the string that is the user's
                ** specification of the reduction which is just about
                ** to be done.
                */
                if ( yydebug )
                        printf( "Reduce by (%d) \"%s\"\n",
                                yy_n, yyreds[ yy_n ] );
#endif
                yytmp = yy_n;                   /* value to switch over *
                yypvt = yy_pv;                  /* $vars top of value sta
                /*
                ** Look in goto table for next state
                ** Sorry about using yy_state here as temporary
                ** register variable, but why not, if it works...
                ** If yyr2[ yy_n ] doesn't have the low order bit
                ** set, then there is no action to be done for
                ** this reduction.  So, no saving & unsaving of
                ** registers done.  The only difference between the
                ** code just after the if and the body of the if is
                ** the goto yy_stack in the body.  This way the test
                ** can be made before the choice of what to do is needed.
                */
                {
                        /* length of production doubled with extra bit */
                        register int yy_len = yyr2[ yy_n ];

                        if ( !( yy_len & 01 ) )
                        {
                                yy_len >>= 1;
                                yyval = ( yy_pv -= yy_len )[1]; /* $$ = $
                                yy_state = yypgo[ yy_n = yyr1[ yy_n ] ] +
                                        *( yy_ps -= yy_len ) + 1;
                                if ( yy_state >= YYLAST ||
                                        yychk[ yy_state =
                                        yyact[ yy_state ] ] != -yy_n )
```

```
                                {
                                        yy_state = yyact[ yypgo[ yy_n ] ]
                                }
                                goto yy_stack;
                        }
                        yy_len >>= 1;
                        yyval = ( yy_pv -= yy_len )[1]; /* $$ = $1 */
                        yy_state = yypgo[ yy_n = yyr1[ yy_n ] ] +
                                *( yy_ps -= yy_len ) + 1;
                        if ( yy_state >= YYLAST ||
                                yychk[ yy_state = yyact[ yy_state ] ] !=
                        {
                                yy_state = yyact[ yypgo[ yy_n ] ];
                        }
                }
                                        /* save until reenter driver code
                yystate = yy_state;
                yyps = yy_ps;
                yypv = yy_pv;
        }
        /*
        ** code supplied by user is placed in this switch
        */
        switch( yytmp )
        {

case 21:
# line 132 "jur.yacc"
{
                globnumsect = 0;
                globnumerr = 0;
                } break;
case 22:
# line 139 "jur.yacc"
{
                globnumsect = 1;
                globnumerr = 1;
                } break;
case 23:
# line 144 "jur.yacc"
{strcpy (numdoccour, yytext);} break;
case 26:
# line 151 "jur.yacc"
{
                globnumsect = 2;
                globnumerr = 1;
                } break;
case 29:
# line 161 "jur.yacc"
{
                globnumsect = 3;
                globnumerr = 1;
                } break;
case 32:
# line 171 "jur.yacc"
{
                globnumsect = 4;
                globnumerr = 1;
                } break;
case 35:
# line 184 "jur.yacc"
{
                globnumsect = 0;
                globnumerr = 0;
                } break;
case 36:
```

```
# line 191 "jur.yacc"
{
                globnumsect = 5;
                globnumerr = 3;
                } break;
case 62:
# line 272 "jur.yacc"
{
                globnumsect = 6;
                globnumerr = 1;
                checkref[0] = '\0';
                } break;
case 66:
# line 285 "jur.yacc"
{checkref[0] = '\0';} break;
case 68:
# line 292 "jur.yacc"
{
                int errinref;

                if (strlen (checkref) > 0)
                        checkref[strlen (checkref) - 1] = '\0';
                errinref = verifier (checkref, "REFERENCES", pasgrave);
                if (errinref != 0) erreur (6, errinref);
                } break;
case 83:
# line 344 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 84:
# line 348 "jur.yacc"
{
                strcat (checkref, ":");
                } break;
case 86:
# line 356 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 88:
# line 362 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 90:
# line 368 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 92:
# line 374 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 94:
# line 380 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 96:
# line 386 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 98:
```

```
# line 392 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 100:
# line 398 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 102:
# line 404 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 104:
# line 410 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 106:
# line 416 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 108:
# line 422 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 110:
# line 428 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 112:
# line 434 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 114:
# line 440 "jur.yacc"
{
                strcat (checkref, yytext);
                } break;
case 118:
# line 452 "jur.yacc"
{
                globnumsect = 7;
                globnumerr = 3;
                } break;
case 142:
# line 526 "jur.yacc"
{
                globnumsect = 8;
                globnumerr = 3;
                } break;
case 146:
# line 544 "jur.yacc"
{
                globnumsect = 0;
                globnumerr = 0;
                } break;
case 147:
# line 551 "jur.yacc"
{
                globnumsect = 1;
```

```
                globnumerr = 1;
                } break;
case 148:
# line 556 "jur.yacc"
{strcpy (numdoccour, yytext);} break;
case 151:
# line 566 "jur.yacc"
{
                globnumsect = 0;
                globnumerr = 0;
                } break;
case 153:
# line 581 "jur.yacc"
{
                globnumsect = 0;
                globnumerr = 0;
                } break;
case 154:
# line 588 "jur.yacc"
{
                globnumsect = 1;
                globnumerr = 1;
                } break;
case 155:
# line 593 "jur.yacc"
{strcpy (numdoccour, yytext);} break;
case 158:
# line 600 "jur.yacc"
{
                globnumsect = 3;
                globnumerr = 1;
                } break;
case 161:
# line 610 "jur.yacc"
{
                globnumsect = 9;
                globnumerr = 1;
                } break;
case 164:
# line 620 "jur.yacc"
{
                int errinlang;

                switch (strlen (yytext))
                        {
                        case 1 :
                                if ((yytext[0] < 'A') || (yytext[0] > 'Z'
                                        erreur (9, 1);
                                break;
                        case 2 :
                                if ((yytext[0] < 'A') || (yytext[0] > 'Z'
                                        erreur (9, 1);
                                else
                                if ((yytext[1] < 'A') || (yytext[1] > 'Z'
                                        erreur (9, 1);
                                break;
                        default :
                                erreur (9, 1);
                        }
                errinlang = verifier (yytext, "LANGUE", grave);
                if (errinlang != 0) erreur (9, errinlang);
                } break;
case 165:
# line 648 "jur.yacc"
{
                globnumsect = 0;
```

```
                globnumerr = 0;
                } break;
case 167:
# line 662 "jur.yacc"
{
                globnumsect = 13;
                globnumerr = 1;
                } break;
case 168:
# line 667 "jur.yacc"
{
                globnumsect = 14;
                globnumerr = 3;
                } break;
case 169:
# line 672 "jur.yacc"
{
                globnumsect = 15;
                globnumerr = 1;
                } break;
case 170:
# line 677 "jur.yacc"
{
                globnumsect = 0;
                globnumerr = 0;
                } break;
case 171:
# line 684 "jur.yacc"
{
                globnumsect = 10;
                globnumerr = 1;
                } break;
case 172:
# line 689 "jur.yacc"
{strcpy (numdoccour, yytext);} break;
case 175:
# line 696 "jur.yacc"
{
                globnumsect = 11;
                globnumerr = 1;
                } break;
case 178:
# line 706 "jur.yacc"
{
                globnumsect = 12;
                globnumerr = 1;
                } break;
case 185:
# line 728 "jur.yacc"
{
                strcpy (checktype, yytext);
                } break;
case 186:
# line 733 "jur.yacc"
{
                int errintype;

                errintype = verifier (checktype, "TYPE", pasgrave);
                if (errintype != 0) erreur (13, errintype);
                } break;
case 188:
# line 744 "jur.yacc"
{strcat (checktype, yytext);} break;
case 190:
# line 748 "jur.yacc"
{strcat (checktype, yytext);} break;
```

```
case 192:
# line 752 "jur.yacc"
{strcat (checktype, yytext);} break;
case 219:
# line 836 "jur.yacc"
{
                int errincountry;

                errincountry = verifier (yytext, "COUNTRY", grave);
                if (errincountry != 0) erreur (15, errincountry);
                } break;
case 222:
# line 852 "jur.yacc"
{
                globnumsect = 0;
                globnumerr = 0;
                } break;
case 223:
# line 859 "jur.yacc"
{
                globnumsect = 16;
                globnumerr = 1;
                strcpy (oldkeyw, '\0');
                } break;
case 228:
# line 877 "jur.yacc"
{
                /* les keywords doivent etre tries par ordre croissant */
                if (strcmp (oldkeyw, yytext) > 0) erreur (16, 13);
                strcpy (oldkeyw, yytext);
                } break;
case 229:
# line 884 "jur.yacc"
{
                int errinkeyw;

                errinkeyw = verifier (yytext, "KEYWORDS", pasgrave);
                if (errinkeyw != 0) erreur (16, 4);
                } break;
        }
        goto yystack;           /* reset registers in driver code */
}
```

```
state 0
        $accept : _textin $end

        error  shift 4
        NEWCHAP  shift 3
        .  error

        textin  goto 1
        chaphead  goto 2

state 1
        $accept :  textin_$end

        $end  accept
        .  error


state 2
        textin :  chaphead_chapter1 chaphead chapter2 chaphead chapter3 c
        chapter1 : _     (4)

        .  reduce 4

        chapter1  goto 5

state 3
        chaphead :  NEWCHAP_NEWLINE

        NEWLINE  shift 6
        .  error


state 4
        chaphead :  error_NEWCHAP NEWLINE

        NEWCHAP  shift 7
        .  error


state 5
        textin :  chaphead chapter1_chaphead chapter2 chaphead chapter3 c
        chapter1 :  chapter1_article1 artfooter

        error  shift 10
        NEWCHAP  shift 3
        TITLE  shift 13
        .  error

        chaphead  goto 8
        article1  goto 9
        title1  goto 11
        titlehead  goto 12

state 6
        chaphead :  NEWCHAP NEWLINE_     (2)

        .  reduce 2


state 7
        chaphead :  error NEWCHAP_NEWLINE

        NEWLINE  shift 14
        .  error
```

```
state 8
        textin :  chaphead chapter1 chaphead_chapter2 chaphead chapter3 c
        chapter2 : _     (6)

        .  reduce 6

        chapter2  goto 15

state 9
        chapter1 :  chapter1 article1_artfooter

        error  shift 18
        NEWART  shift 17
        .  error

        artfooter  goto 16

state 10
        chaphead :  error_NEWCHAP NEWLINE
        titlehead :  error_TITLE NEWLINE

        NEWCHAP  shift 7
        TITLE  shift 19
        .  error


state 11
        article1 :  title1_body1

        error  shift 23
        NEWTITRE  shift 22
        .  error

        body1  goto 20
        titre  goto 21

state 12
        title1 :  titlehead_contitle1 titlehead

        error  shift 27
        NEWNUMDOC  shift 26
        .  error

        contitle1  goto 24
        numdoc1  goto 25

state 13
        titlehead :  TITLE_NEWLINE

        NEWLINE  shift 28
        .  error


state 14
        chaphead :  error NEWCHAP NEWLINE_     (3)

        .  reduce 3


state 15
        textin :  chaphead chapter1 chaphead chapter2_chaphead chapter3 c
        chapter2 :  chapter2_article2 artfooter

        error  shift 10
        NEWCHAP  shift 3
```

```
        TITLE  shift 13
        .  error

        chaphead  goto 29
        article2  goto 30
        title2  goto 31
        titlehead  goto 32

state 16
        chapter1 :  chapter1 article1 artfooter_    (5)

        .  reduce 5


state 17
        artfooter :  NEWART_NEWLINE

        NEWLINE  shift 33
        .  error


state 18
        artfooter :  error_NEWART NEWLINE

        NEWART  shift 34
        .  error


state 19
        titlehead :  error TITLE_NEWLINE

        NEWLINE  shift 35
        .  error


state 20
        article1 :  title1 body1_    (14)

        .  reduce 14


state 21
        body1 :  titre_references texte conclusion

        error  shift 38
        NEWREF  shift 37
        .  error

        references  goto 36

state 22
        titre :  NEWTITRE_NEWLINE $$36 contitre

        NEWLINE  shift 39
        .  error


state 23
        titre :  error_    (38)

        .  reduce 38


state 24
        title1 :  titlehead contitle1_titlehead
```

```
        error  shift 41
        TITLE  shift 13
        .  error

        titlehead  goto 40

state 25
        contitle1 :  numdoc1_date cote12 code

        error  shift 44
        NEWDATE  shift 43
        .  error

        date  goto 42

state 26
        numdoc1 :  NEWNUMDOC_NEWLINE $$22 NUMDOCj $$23 NEWLINE

        NEWLINE  shift 45
        .  error


state 27
        numdoc1 :  error_    (25)

        .  reduce 25


state 28
        titlehead :  TITLE NEWLINE_    (19)

        .  reduce 19


state 29
        textin :  chaphead chapter1 chaphead chapter2 chaphead_chapter3 c
        chapter3 : _    (8)

        .  reduce 8

        chapter3  goto 46

state 30
        chapter2 :  chapter2 article2_artfooter

        error  shift 18
        NEWART  shift 17
        .  error

        artfooter  goto 47

state 31
        article2 :  title2_body2

        error  shift 23
        NEWTITRE  shift 22
        .  error

        body2  goto 48
        titre  goto 49

state 32
        title2 :  titlehead_contitle2 titlehead

        error  shift 53
        NEWNUMDOC  shift 52
```

```
state 290
        contitle4 :  docnum dat pub_ref $$167 type $$168_product $$169 co

        error  shift 302
        NEWPRODUCT  shift 301
        .  error

        product  goto 300

state 291
        type :  NEWTYPE NEWLINE_contype NEWLINE

        WORD  shift 305
        .  error

        contype  goto 303
        lignetype  goto 304

state 292
        pub_ref :  NEWPUB_REF NEWLINE $$178 PUBREF_NEWLINE

        NEWLINE  shift 306
        .  error


state 293
        dat :  NEWDAT NEWLINE $$175 DATE NEWLINE_    (176)

        .  reduce 176


state 294
        docnum :  NEWDOCNUM NEWLINE $$171 DOCNUM $$172 NEWLINE_    (173)

        .  reduce 173


state 295
        langue :  NEWLANG NEWLINE $$161 conlangue NEWLINE_    (162)

        .  reduce 162


state 296
        lineref2 :  TAB refdescriptor optionalless NEWLINE lineref3_    (

        .  reduce 70


state 297
        lineref3 :  TAB_TAB refdescriptor optionalless NEWLINE lineref4

        TAB  shift 307
        .  error


state 298
        conkeywords :  conkeywords NEWPAR NEWLINE keywordline_    (227)

        .  reduce 227


state 299
        keywordline :  KEYW1 $$228 SP KEYW2_$$229 optionalless NEWLINE
        $$229 : _    (229)
```

```
        .  reduce 229

        $$229  goto 308

state 300
        contitle4 :  docnum dat pub_ref $$167 type $$168 product_$$169 co
        $$169 : _    (169)

        .  reduce 169

        $$169  goto 309

state 301
        product :  NEWPRODUCT_NEWLINE conproduct

        NEWLINE  shift 310
        .  error


state 302
        product :  error_    (195)

        .  reduce 195


state 303
        type :  NEWTYPE NEWLINE contype_NEWLINE
        contype :  contype_NEWLINE lignetype

        NEWLINE  shift 311
        .  error


state 304
        contype :  lignetype_    (184)

        .  reduce 184


state 305
        lignetype :  WORD_$$185 restoflntype optionalless
        $$185 : _    (185)

        .  reduce 185

        $$185  goto 312

state 306
        pub_ref :  NEWPUB_REF NEWLINE $$178 PUBREF NEWLINE_    (179)

        .  reduce 179


state 307
        lineref3 :  TAB TAB_refdescriptor optionalless NEWLINE lineref4

        WORD  shift 102
        .  error

        refdescriptor  goto 313

state 308
        keywordline :  KEYW1 $$228 SP KEYW2 $$229_optionalless NEWLINE
        optionalless : _    (116)

        LESS  shift 124
```

```
        .  reduce 116

        optionalless  goto 314

state 309
        contitle4 :  docnum dat pub_ref $$167 type $$168 product $$169_co

        error  shift 317
        NEWCOUNTRY  shift 316
        .  error

        country  goto 315

state 310
        product :  NEWPRODUCT NEWLINE_conproduct
        optnlspaces : _     (40)

        .  reduce 40

        optnlspaces  goto 319
        conproduct  goto 318

state 311
        type :  NEWTYPE NEWLINE contype NEWLINE_     (181)
        contype :  contype NEWLINE_lignetype

        WORD  shift 305
        .  reduce 181

        lignetype  goto 320

state 312
        lignetype :  WORD $$185_restoflntype optionalless
        restoflntype : _     (187)

        TAB  shift 323
        SP  shift 322
        WORD  shift 324
        .  reduce 187

        restoflntype  goto 321

state 313
        lineref3 :  TAB TAB refdescriptor_optionalless NEWLINE lineref4
        optionalless : _     (116)

        LESS  shift 124
        .  reduce 116

        optionalless  goto 325

state 314
        keywordline :  KEYW1 $$228 SP KEYW2 $$229 optionalless_NEWLINE

        NEWLINE  shift 326
        .  error


state 315
        contitle4 :  docnum dat pub_ref $$167 type $$168 product $$169 co

        .  reduce 170


state 316
        country :  NEWCOUNTRY_NEWLINE concountry NEWLINE
```

```
        NEWLINE  shift 327
        .  error


state 317
        country :  error_    (217)

        .  reduce 217


state 318
        product :  NEWPRODUCT NEWLINE conproduct_    (194)

        .  reduce 194


state 319
        optnlspaces :  optnlspaces_SP
        optnlspaces :  optnlspaces_TAB
        conproduct :  optnlspaces_WORD restofproduct

        TAB  shift 105
        SP  shift 104
        WORD  shift 328
        .  error


state 320
        contype :  contype NEWLINE lignetype_    (183)

        .  reduce 183


state 321
        lignetype :  WORD $$185 restoflntype_optionalless
        optionalless : _    (116)

        LESS  shift 124
        .  reduce 116

        optionalless  goto 329

state 322
        restoflntype :  SP_$$188 restoflntype
        $$188 : _    (188)

        .  reduce 188

        $$188  goto 330

state 323
        restoflntype :  TAB_$$190 restoflntype
        $$190 : _    (190)

        .  reduce 190

        $$190  goto 331

state 324
        restoflntype :  WORD_$$192 restoflntype
        $$192 : _    (192)

        .  reduce 192

        $$192  goto 332
```

```
state 325
        lineref3 :  TAB TAB refdescriptor optionalless_NEWLINE lineref4

        NEWLINE  shift 333
        .  error


state 326
        keywordline :  KEYW1 $$228 SP KEYW2 $$229 optionalless NEWLINE_

        .  reduce 230


state 327
        country :  NEWCOUNTRY NEWLINE_concountry NEWLINE

        WORD  shift 336
        .  error

        concountry  goto 334
        countrycode  goto 335

state 328
        conproduct :  optnlspaces WORD_restofproduct
        restofproduct : _     (197)

        .  reduce 197

        restofproduct  goto 337

state 329
        lignetype :  WORD $$185 restoflntype optionalless_     (186)

        .  reduce 186


state 330
        restoflntype :  SP $$188_restoflntype
        restoflntype : _     (187)

        TAB  shift 323
        SP  shift 322
        WORD  shift 324
        .  reduce 187

        restoflntype  goto 338

state 331
        restoflntype :  TAB $$190_restoflntype
        restoflntype : _     (187)

        TAB  shift 323
        SP  shift 322
        WORD  shift 324
        .  reduce 187

        restoflntype  goto 339

state 332
        restoflntype :  WORD $$192_restoflntype
        restoflntype : _     (187)

        TAB  shift 323
        SP  shift 322
        WORD  shift 324
```

```
        .  reduce 187

        restoflntype  goto 340

state 333
        lineref3 :  TAB TAB refdescriptor optionalless NEWLINE_lineref4
        lineref4 : _     (73)

        TAB  shift 342
        .  reduce 73

        lineref4  goto 341

state 334
        country :  NEWCOUNTRY NEWLINE concountry_NEWLINE

        NEWLINE  shift 343
        .  error


state 335
        concountry :  countrycode_restofcountry optionalless
        restofcountry : _     (220)

        SP  shift 345
        .  reduce 220

        restofcountry  goto 344

state 336
        countrycode :  WORD_     (219)

        .  reduce 219


state 337
        conproduct :  optnlspaces WORD restofproduct_     (196)
        restofproduct :  restofproduct_SP
        restofproduct :  restofproduct_DOLLAR
        restofproduct :  restofproduct_LESS
        restofproduct :  restofproduct_TAB
        restofproduct :  restofproduct_NEWLINE
        restofproduct :  restofproduct_WORD
        restofproduct :  restofproduct_NEWLINE NEWPAR NEWLINE
        restofproduct :  restofproduct_DATE
        restofproduct :  restofproduct_COTEjk
        restofproduct :  restofproduct_COTEn
        restofproduct :  restofproduct_CODE
        restofproduct :  restofproduct_NUMDOCj
        restofproduct :  restofproduct_NUMDOCk
        restofproduct :  restofproduct_NUMDOCn
        restofproduct :  restofproduct_DOCNUM
        restofproduct :  restofproduct_PUBREF
        restofproduct :  restofproduct_KEYW1
        restofproduct :  restofproduct_KEYW2

        NEWLINE  shift 350
        TAB  shift 349
        SP  shift 346
        DATE  shift 352
        COTEjk  shift 353
        COTEn  shift 354
        CODE  shift 355
        NUMDOCj  shift 356
        NUMDOCk  shift 357
        NUMDOCn  shift 358
```

```
        DOCNUM  shift 359
        PUBREF  shift 360
        KEYW1  shift 361
        KEYW2  shift 362
        WORD  shift 351
        LESS  shift 348
        DOLLAR  shift 347
        .  reduce 196


state 338
        restoflntype :  SP $$188 restoflntype_    (189)

        .  reduce 189


state 339
        restoflntype :  TAB $$190 restoflntype_    (191)

        .  reduce 191


state 340
        restoflntype :  WORD $$192 restoflntype_    (193)

        .  reduce 193


state 341
        lineref3 :  TAB TAB refdescriptor optionalless NEWLINE lineref4_

        .  reduce 72


state 342
        lineref4 :  TAB_TAB TAB refdescriptor optionalless NEWLINE linere

        TAB  shift 363
        .  error


state 343
        country :  NEWCOUNTRY NEWLINE concountry NEWLINE_    (216)

        .  reduce 216


state 344
        concountry :  countrycode restofcountry_optionalless
        optionalless : _    (116)

        LESS  shift 124
        .  reduce 116

        optionalless  goto 364

state 345
        restofcountry :  SP_countrycode restofcountry

        WORD  shift 336
        .  error

        countrycode  goto 365

state 346
        restofproduct :  restofproduct SP_    (198)
```

```
        .  reduce 198

state 347
        restofproduct :  restofproduct DOLLAR_     (199)

        .  reduce 199

state 348
        restofproduct :  restofproduct LESS_     (200)

        .  reduce 200

state 349
        restofproduct :  restofproduct TAB_     (201)

        .  reduce 201

state 350
        restofproduct :  restofproduct NEWLINE_     (202)
        restofproduct :  restofproduct NEWLINE_NEWPAR NEWLINE

        NEWPAR  shift 366
        .  reduce 202

state 351
        restofproduct :  restofproduct WORD_     (203)

        .  reduce 203

state 352
        restofproduct :  restofproduct DATE_     (205)

        .  reduce 205

state 353
        restofproduct :  restofproduct COTEjk_     (206)

        .  reduce 206

state 354
        restofproduct :  restofproduct COTEn_     (207)

        .  reduce 207

state 355
        restofproduct :  restofproduct CODE_     (208)

        .  reduce 208

state 356
        restofproduct :  restofproduct NUMDOCj_     (209)

        .  reduce 209
```

```
state 357
        restofproduct :  restofproduct NUMDOCk_     (210)

        .  reduce 210


state 358
        restofproduct :  restofproduct NUMDOCn_     (211)

        .  reduce 211


state 359
        restofproduct :  restofproduct DOCNUM_     (212)

        .  reduce 212


state 360
        restofproduct :  restofproduct PUBREF_     (213)

        .  reduce 213


state 361
        restofproduct :  restofproduct KEYW1_     (214)

        .  reduce 214


state 362
        restofproduct :  restofproduct KEYW2_     (215)

        .  reduce 215


state 363
        lineref4 :  TAB TAB_TAB refdescriptor optionalless NEWLINE linere

        TAB  shift 367
        .  error


state 364
        concountry :  countrycode restofcountry optionalless_     (218)

        .  reduce 218


state 365
        restofcountry :  SP countrycode_restofcountry
        restofcountry : _     (220)

        SP  shift 345
        .  reduce 220

        restofcountry  goto 368

state 366
        restofproduct :  restofproduct NEWLINE NEWPAR_NEWLINE

        NEWLINE  shift 369
        .  error


state 367
```

```
        lineref4 :  TAB TAB TAB_refdescriptor optionalless NEWLINE linere

        WORD  shift 102
        .  error

        refdescriptor  goto 370

state 368
        restofcountry :  SP countrycode restofcountry_    (221)

        .  reduce 221


state 369
        restofproduct :  restofproduct NEWLINE NEWPAR NEWLINE_    (204)

        .  reduce 204


state 370
        lineref4 :  TAB TAB TAB refdescriptor_optionalless NEWLINE linere
        optionalless : _    (116)

        LESS  shift 124
        .  reduce 116

        optionalless  goto 371

state 371
        lineref4 :  TAB TAB TAB refdescriptor optionalless_NEWLINE linere

        NEWLINE  shift 372
        .  error


state 372
        lineref4 :  TAB TAB TAB refdescriptor optionalless NEWLINE_linere
        lineref5 : _    (75)

        TAB  shift 374
        .  reduce 75

        lineref5  goto 373

state 373
        lineref4 :  TAB TAB TAB refdescriptor optionalless NEWLINE linere

        .  reduce 74


state 374
        lineref5 :  TAB_TAB TAB TAB refdescriptor optionalless NEWLINE li

        TAB  shift 375
        .  error


state 375
        lineref5 :  TAB TAB_TAB TAB refdescriptor optionalless NEWLINE li

        TAB  shift 376
        .  error


state 376
        lineref5 :  TAB TAB TAB_TAB refdescriptor optionalless NEWLINE li
```

```
        TAB  shift 377
        .  error


state 377
        lineref5 :  TAB TAB TAB TAB_refdescriptor optionalless NEWLINE li

        WORD  shift 102
        .  error

        refdescriptor  goto 378

state 378
        lineref5 :  TAB TAB TAB TAB refdescriptor_optionalless NEWLINE li
        optionalless : _     (116)

        LESS  shift 124
        .  reduce 116

        optionalless  goto 379

state 379
        lineref5 :  TAB TAB TAB TAB refdescriptor optionalless_NEWLINE li

        NEWLINE  shift 380
        .  error


state 380
        lineref5 :  TAB TAB TAB TAB refdescriptor optionalless NEWLINE_li
        lineref6 : _     (77)

        TAB  shift 382
        .  reduce 77

        lineref6  goto 381

state 381
        lineref5 :  TAB TAB TAB TAB refdescriptor optionalless NEWLINE li

        .  reduce 76


state 382
        lineref6 :  TAB_TAB TAB TAB TAB refdescriptor optionalless NEWLIN

        TAB  shift 383
        .  error


state 383
        lineref6 :  TAB TAB_TAB TAB TAB refdescriptor optionalless NEWLIN

        TAB  shift 384
        .  error


state 384
        lineref6 :  TAB TAB TAB_TAB TAB refdescriptor optionalless NEWLIN

        TAB  shift 385
        .  error


state 385
```

```
        lineref6 :  TAB TAB TAB TAB_TAB refdescriptor optionalless NEWLIN

        TAB  shift 386
        .  error


state 386
        lineref6 :  TAB TAB TAB TAB TAB_refdescriptor optionalless NEWLIN

        WORD  shift 102
        .  error

        refdescriptor  goto 387

state 387
        lineref6 :  TAB TAB TAB TAB TAB refdescriptor_optionalless NEWLIN
        optionalless : _     (116)

        LESS  shift 124
        .  reduce 116

        optionalless  goto 388

state 388
        lineref6 :  TAB TAB TAB TAB TAB refdescriptor optionalless_NEWLIN

        NEWLINE  shift 389
        .  error


state 389
        lineref6 :  TAB TAB TAB TAB TAB refdescriptor optionalless NEWLIN
        lineref7 : _     (79)

        TAB  shift 391
        .  reduce 79

        lineref7  goto 390

state 390
        lineref6 :  TAB TAB TAB TAB TAB refdescriptor optionalless NEWLIN

        .  reduce 78


state 391
        lineref7 :  TAB_TAB TAB TAB TAB TAB refdescriptor optionalless NE

        TAB  shift 392
        .  error


state 392
        lineref7 :  TAB TAB_TAB TAB TAB TAB refdescriptor optionalless NE

        TAB  shift 393
        .  error


state 393
        lineref7 :  TAB TAB TAB_TAB TAB TAB refdescriptor optionalless NE

        TAB  shift 394
        .  error
```

```
state 394
        lineref7 :  TAB TAB TAB TAB_TAB TAB refdescriptor optionalless NE

        TAB  shift 395
        .  error


state 395
        lineref7 :  TAB TAB TAB TAB TAB_TAB refdescriptor optionalless NE

        TAB  shift 396
        .  error


state 396
        lineref7 :  TAB TAB TAB TAB TAB TAB_refdescriptor optionalless NE

        WORD  shift 102
        .  error

        refdescriptor  goto 397

state 397
        lineref7 :  TAB TAB TAB TAB TAB TAB refdescriptor_optionalless NE
        optionalless : _     (116)

        LESS  shift 124
        .  reduce 116

        optionalless  goto 398

state 398
        lineref7 :  TAB TAB TAB TAB TAB TAB refdescriptor optionalless_NE

        NEWLINE  shift 399
        .  error


state 399
        lineref7 :  TAB TAB TAB TAB TAB TAB refdescriptor optionalless NE
        lineref8 : _     (81)

        TAB  shift 401
        .  reduce 81

        lineref8  goto 400

state 400
        lineref7 :  TAB TAB TAB TAB TAB TAB refdescriptor optionalless NE

        .  reduce 80


state 401
        lineref8 :  TAB_TAB TAB TAB TAB TAB TAB refdescriptor optionalles

        TAB  shift 402
        .  error


state 402
        lineref8 :  TAB TAB_TAB TAB TAB TAB TAB refdescriptor optionalles

        TAB  shift 403
        .  error
```

```
state 403
        lineref8 :  TAB TAB TAB_TAB TAB TAB TAB refdescriptor optionalles

        TAB  shift 404
        .  error


state 404
        lineref8 :  TAB TAB TAB TAB_TAB TAB TAB refdescriptor optionalles

        TAB  shift 405
        .  error


state 405
        lineref8 :  TAB TAB TAB TAB TAB_TAB TAB refdescriptor optionalles

        TAB  shift 406
        .  error


state 406
        lineref8 :  TAB TAB TAB TAB TAB TAB_TAB refdescriptor optionalles

        TAB  shift 407
        .  error


state 407
        lineref8 :  TAB TAB TAB TAB TAB TAB TAB_refdescriptor optionalles

        WORD  shift 102
        .  error

        refdescriptor  goto 408

state 408
        lineref8 :  TAB TAB TAB TAB TAB TAB TAB refdescriptor_optionalles
        optionalless : _     (116)

        LESS  shift 124
        .  reduce 116

        optionalless  goto 409

state 409
        lineref8 :  TAB TAB TAB TAB TAB TAB TAB refdescriptor optionalles

        NEWLINE  shift 410
        .  error


state 410
        lineref8 :  TAB TAB TAB TAB TAB TAB TAB refdescriptor optionalles

        .  reduce 82


39/255 terminals, 118/200 nonterminals
1198/4000 charaters in id's and literals
232/300 grammar rules, 413/600 states
0 shift/reduce, 0 reduce/reduce conflicts reported
118/250 working sets used
memory: states,etc. 1913/5200, parser 221/4000
49/450 distinct lookahead sets
```

```
25 extra closures
530 shift entries, 3 exceptions
184 goto entries
2 entries saved by goto default
Optimizer space used: input 1313/5200, output 463/4000
463 table entries, 45 zero
maximum spread: 293, maximum offset: 408
```