A Machine for Information Retrieval

A. El Masri, J. Rohmer, D. Tusera
I.R.I.A.-LABORIA
B.P. 105
78150 LE CHESNAY
FRANCE

I. Introduction

The described machine was concieved especially for nonnumeric computation. The possible applications are :

- content and structures recognition in texts
- language translation
- modification of files format
- etc...

The first application of the machine will be "A library file query system". The library file is available on a 300 Mbyte disk. The proposed query language allows to ask questions which may be rather imprecise (with spelling mistakes or with missing words in sentences) and complicated with predicates containing logical expressions about the existence of words or words sequences.

II. Working principles

The machine is microprogrammed. It can read the entry data in sequential manner in its Input FIFO Memory and it can store the result data the same way in the Output FIFO Memory.

The microprogram memory is a read-write one. A compiler translates the user's problem in a data structure (a microprogram) that is then loaded in the machine memory.

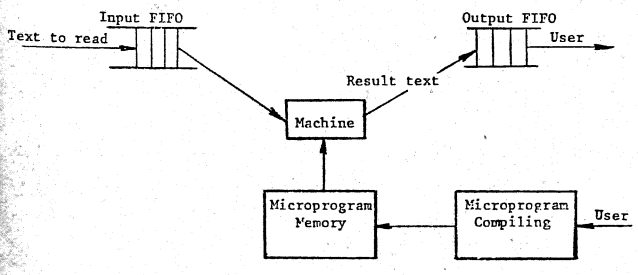


fig.1 Principle Scheme

So the realization of a user's problem can be divided in two stages :

a) machine initiation : microprogram compiling from user's question and its loading in the machine memory.
b) processing : input text reading and its transformation into the final text.

Physical resources of the machine are fixed (memory capacity, clock frequency). However its power can be easily raised by means of ordinary connection of several pipe-line wise stations.

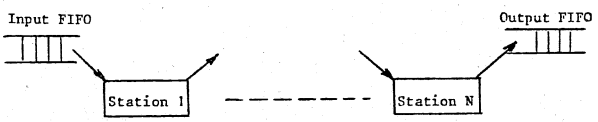


fig.2 Pipe-line stations

For instance it is planned for the first application (a library file query system-disk filled) to use a four stations pipe-line (1st station: words retrieval, 2nd station: expressions retrieval, 3rd station: fields retrieval, 4th station: inscription retrieval).

A microprogram is composed of microinstructions. A microinstruction contains two classes of fields :

1) Fields allowing to compute the machine state transitions-or the next microinstruction address.
2) Fields which determine the transcoding function-or the output result.

III. Sequencing algorithm

The mechanism of sequencing is that of a push down automata.

A microinstruction (its first class of fields) may be represented as a tree node (one microinstruction=one tree node, one microprogram=nodes tree).

The node structure is general

TAG	CAR-DEF
ALT	SUC

ALT, SUC, DEF - nodes addresses
CAR - node character
TAG - flag whose value is '1' or '0'
X - current node address
INSCR(I) - entry character from INPUT FIFO

The algorithm is simple :

```
Repeat
if CAR(X) = INSCR(I) then
          begin
        X:= SUC(X) - the next address
                     is SUC
        I:= I+1    - scan on the entry
                     character
          end
                   else
          begin
        X:= ALT(X) - the next address
                     is ALT
                   - don't scan on the
                     entry character
          end
```

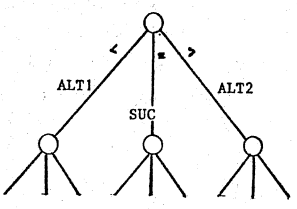
The addition of DEF and TAG fields in the node structure and of the push down stack for return addresses enables to deal with the recursive programs and subroutines.

The recursive description of the algorithm is :

```
PARSE X
A : if TAG(X) ≠ 0  then
              Y:=PARSE DEF(X)
                   else
                   begin
                   if CAR(X) = INSCR(I) then
                                  begin
                                  I:=I+1
                                  Y:=1
                                  end
                                       else
                                  Y:=0
                   end
        if Y = 1 then
                   X:=SUC(X)
                   else
                   X:=ALT(X)
        if X ≠ NIL  then go to A
        end
```

Here X=NIL means that the current node is a terminal one of the tree hence return from subroutine (if the stack is not empty) or program end.

In order to raise the number of possible comparisons for each character of Input FIFO with a given character frequency we have replaced ALT field by two ALT1 and ALT2 fields. This makes possible an alternative search by means of dichotomy.
The tree structure becomes ternary :



IV. Transcoding function

The sequencing algorithm is that of a syntactic analyser. It is able to determine whether the sentence at Input FIFO belongs or not to the given language. The given language may be any context-free language (Chomsky 2 type). The grammar rules of this language are then translated by the compiler into the tree structures (microprograms) for the machine.

Our aim was not only to recognize if the entry sentence belongs to a language but also to transcode its peculiarly interesting parts.

Each instruction contains two fields :
BASC - Character to output.
TR - Indicates whether to output BASC or not.

These two fields are interpreted by the machine only in case of positive comparison between the input and the microinstruction characters.

We also introduce an EMPTY character into the microinstruction with which the comparison is always positive.

These three devices: the TR and BASC fields and EMPTY character allow transcoding of characters, words, sentences or text structures.

There is a good number of user's problems which, if expressed by means of grammar would blow-up the microprogram memory.

For instance a question such as : "find every word containing the set of letters A, B, C " would be asking for a huge number of tree nodes.

In order to optimize the use of microprogram memory this kind of problems is not solved by the syntactical analyser but by means of an extra mechanism. It allows to evaluate logical expressions whose operators are logical "not", "or", "and" and operands are existences of chosen characters, words, structures etc...

In this case the transcoding is function of the logical expression evaluation.

V. The global architecture of an application of the machine.

To test the prototype of the machine whose principle scheme is given in fig.1 we are actually building a system around it. This system will enable us to realize the first application we have chosen which is "a library file query system".

The global architecture is shown by the block diagram (fig.3) and the main modules of the system are the following:

1- the disk
2- the Formatter
3- a pipe line assembly of 4 stages of the machine
4- the Buffer
5- the Controller
6- a terminal

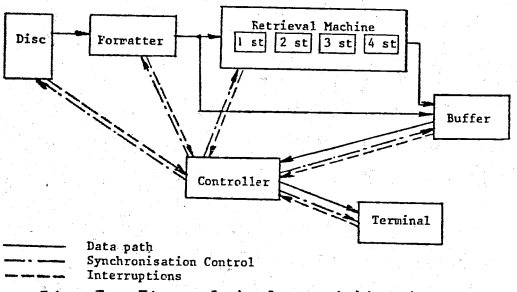


fig.3 The global architecture.

We shall now briefly describe these modules and their operation principles. We are talking of a "read only" mode i.e. the one that answers the user's queries, no mention is made of how the data base is stored on the disk or how updating is made.

V.1 The disk.

It is a 300 Mbytes unit supported by a CDC 9766 Storage Module Drive (SMD) it has an output data rate of approximately 10 Mbits/second. The library file is stored on the disk and every publication is represented by a variable length inscription. An inscription is a set of variable length fields (title, author, summary , keywords, etc...) these fields are not compulsory, i.e. some of them may or may not exist. Each field consists of expressions that are a succession of words.

V.2 The Formatter.

It recognizes the physical structure of the records by using the control signals sent by the SMD (start of track, sector mark, reading clock, etc...), recieves the bit stream and transforms them into 8 bits characters. It separates the information concerning the physical structure (cylinder, track and sector adresses , length of the data, etc...) from the inscriptions themselves which are sent character by character to the first stage's FIFO and to the Buffer.

V.3 The 4 stages pipe line.

This assembly holds in its different memories the user's query (or the set of the required answers) compiled in tree structures. They allow for the recognition of the inscriptions that more or less match the query. The last stage delivers a "mark" to each inscription indicating the degree of matching, which is sent to the Buffer.

V.4 The Buffer.

The Buffer is the "intelligent storage" of the system. In the query mode, it has two distinct functionning schemes (determined by the user at each query).

a)The fuzzy search in which the user expects the N "best" inscriptions to be retrieved (N is determined by the sizes of the memory and the inscriptions). The Buffer sorts the incoming inscriptions according to their "mark". When nearly full it discards the "worst" inscription stored to make room for the incoming ones, at the end of the search the Buffer thus contains the best inscriptions regardless of how "good" they are.

b)The exact search, here the user wants all the inscriptions that match the criteria given in the query so, the Buffer only stores the matching inscriptions and whenever full issues an interrupt to the Controller.

The Buffer also performs the projection(1) as instructed by the query.

Due to timing constraints, the logic of this module is hard wired. The prototype has a Buffer memory of 16K bytes.

V.5 The Controller.

This module plays a number of roles in the system:

a)It holds the software necessary to compile the queries that are entered via a terminal into data to be stored in the memories of the other modules. The algorithm of compilation has two steps:

1)the transformation of a query in a set of grammatical rules , this step is dependent on the application (in the present case the library file query system).

2)the transformation of the produced set of rules, describing the subset of the inscriptions we are looking for, in tree structures which are used by the stack automata of the machine. This step is common to all possible applications and is independent of them.

This solution was chosen in order to minimize the software that has to be modified when changing from an application to another.

b) The Controller then loads the result of the compilation in the various memories, initializes the different modules and sends them the necessary commands.

In the course of the processing the Controller handles the various interrupts and assumes the synchronization of the different modules.

We have chosen a classical system to implement the Controller, this system is built with a TMS 9900 microprocessor.

Conclusion

A prototype of this machine will be built up at the end of 1978.

Here are some of its technical characteristics :

TTL Shottky technology	
Instruction cycle	150 ns
Input FIFO	256 characters
Output FIFO	256 characters
Memory capacity	256 microinstructions

The complexity of this machine is such that its description would lead us beyond the matter of a short paper.

Here are a few interesting problems whose solution was not described :

a) hardware of the miss-spelled words detection algorithm.
b) hardware of the logical expressions evaluation algorithm.
c) hardware solution of the push down automata.
d) microprogram compiling algorithm :
   1) transformation of the set of grammar rules into a tree structure.
   2) finite state automata compiling for several patterns search.
e) actual examples of using the machine.

Acknowledgement

The authors are gratefull to V. Georges for his help in translating this paper.

References:

(1) E.F.Codd, A Relational Model of Data for Large Shared Data Banks, Communications of the A.C.M. Vol 13, N°6, June 1970