PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1984,
edited by ICOT. © ICOT, 1984

INTELLIGENT INFORMATION RETRIEVAL : AN INTERESTING APPLICATION AREA FOR THE NEW GENERATION COMPUTER SYSTEMS

Gian Piero ZARRI

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
Maison des Sciences de l'Homme
54, Boulevard Raspail
75270 PARIS Cedex 06
France

ABSTRACT

The aim of this paper is to provide some details about the inference procedures of RESEDA, an Intelligent Information Retrieval (IIR) system using techniques of an equivalent level to those now proposed by the Japanese project for Fifth Generation Computer Systems.

1 INTRODUCTION

An "intelligent" Information Retrieval system (IIR) - that is, one based on the systematic use of techniques borrowed from Knowledge Engineering - can be broken down into the three blocks indicated in figure 1. There is a fundamental difference between

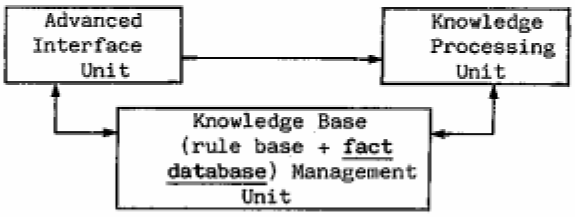
Advanced Interface Unit
Knowledge Processing Unit
Knowledge Base (rule base + fact database) Management Unit

figure 1

Intelligent Information Retrieval systems and ordinary expert systems for diagnostic applications, despite the analogies obvious from figure 1. This difference concerns the fact that - in accordance with the aims of any Information Retrieval system - the <u>fact database</u> of an IIR is <u>permanent and of considerable size</u> and is part of the knowledge base on an equal footing with the rule base. From this point of view, the architecture of an IIR is closer to that of a generalized information processing system of a "Fifth Generation" type, see, for example, Amamiya <u>et al.</u> (1982), than to a classical expert system "à la MYCIN".

Indeed, the theme of "Intelligent Information Retrieval" permeates many of the objectives of the Japanese Fifth Generation challenge; consider, for example, this definition of the role of one of the building blocks of a Fifth Generation Computer System: "The Knowledge Base Machine holds a large amount of knowledge data, which are structured well <u>for accessing each knowledge item effectively</u>: when the knowledge base machine receives a demand from the inference machine, <u>it searches and retrieves knowledge items effectively</u> and hands them to the inference machine; when the knowledge base machine receives knowledge items from the inference machine, <u>it compiles and integrates them into the knowledge base</u>" (Amamiya <u>et al.</u> 1982:181, my italics). But, pending the new tools which will be the offspring of the Fifth Generation project, we have to conclude that there probably exists, at the moment, no system, even at a prototype stage, which could truly be regarded as an IIR and which, at the same time, allows all the possibilities evoked by figure 1; in this context, the aim of this paper is to provide some details of an IIR system, the RESEDA system (Zarri 1981; 1983; 1984), which already includes a number of important IIR procedures foreseen in the Japanese project. I shall concentrate mainly here on the "inference procedures" aspect of RESEDA, whilst referring the reader for more details on the "knowledge representation" aspect to Zarri (1984).

RESEDA is an IIR system with "reasoning" capabilities in the field of complex biographical data management. The term "biographical data" must be understood in its broadest possible sense, i.e. referring to any elementary event, located by space-time coordinates, that it is possible to isolate within the life-cycle ("biography") of a given "personage". As regards the diagram in figure 1, RESEDA is at differing stages of practical implementation:

- A first version of the "Knowledge Processing Unit" and the "Knowledge Base Unit" is completely installed in the framework of a RESEDA prototype, written in VSAPL, which has been operational for over a year at the Centre Inter-Régional de Calcul Electronique (CIRCE), Orsay, France. This prototype has been developed thanks to grants from the Délégation Générale à la Recherche Scientifique et Technique (CNRS-DGRST contract n° 75.7.0456), the Institut de Recherche d'Informatique et d'Automatique (CNRS-IRIA contract n° 78.206) and the Centre National de la Recherche Scientifique (ATP n° 955045).

- The "Advanced Interface Unit" (AIU), on the other hand, is at an embryonic stage: querying the system, for example, is not done in natural language, but by using a coded format corresponding to the external form of the RESEDA's knowledge representation "metalanguage", see also section 3. Studies have been carried out, however, with a view to using an AIU for automatically converting an initial formulation in natural language of the information to be introduced into the fact database into its representation in the terms of RESEDA's knowledge representation metalanguage (Faribault et al. 1984).

The description of RESEDA's inference procedures will be based on examples referring to a, well known, application of the system to a knowledge base on the history of France. The system has been used in other fields, however, the military, for example (CNRS-CIMSA contrat n° 507602); other applications are being studied, concerning in particular the legal field.

2 FUNDAMENTAL CONCEPTS OF RESEDA'S KNOWLEDGE REPRESENTATION LANGUAGE

RESEDA's "knowledge base" is made up of the knowledge, in its entirety, represented in declarative form inside the system. This base is divided into two subsets which, from a strictly functional point of view, have quite different roles:

- the "fact database" contains the data, in the usual sense of the word, on which the system will operate, that is the true "biographical data" and the system's general information about the specific field it is meant to manipulate (extensional data);
- the "rule base" contains the reasoning schemata which allow the system to search for data in the fact database by instantiating particularly complex inference strategies (intensional data).

This differentiation disappears completely when one considers the knowledge representation language ("metalanguage", in our terminology) used to describe the information contained in the two parts of the knowledge base. This language is the same in both cases; the only difference being that extensional data do not have any quantified variables; they represent assertions of individual facts. The intensional data involve quantified variables because of the need to write inference rules applicable to large classes of events. This uniformity in the representations has considerable advantages from the point of view of both logical perspicuity and computational efficiency.

Space prevents me from describing, even briefly, the metalanguage used. Note, however, that, essentially, our metalanguage follows a "case grammar" conception as developed by AI researchers from the ideas of Fillmore (1968); for example, each elementary unit describing a particular event ("predicative plane") is coded in terms of one of five possible "predicates" (BE-AFFECTED-BY, BEHAVE, BE-PRESENT, MOVE, PRODUCE). One or more "modulators" may be attached to each predicate in order to inflect its basic semantic signification, allowing for the construction of several new "meanings" associated to the different combinations "modulator(s) + predicate". Each predicate is accompanied by "case slots" (SUBJect, OBJect, SOURCE, DESTination, MODALity, etc.) which introduce its arguments; dating and space location is also given within a predicative plane. These elementary units can be linked together using explicit links ("labelled pointers") of the type "CAUSALity", "GOAL", etc., which permit the construction of "frame" like data structures. An example of a very simple predicative plane is provided by the one marked "114" in figure 2, which is part of RESEDA's historical database. It is the representation of "Robert de Bonnay is appointed 'bailli' of Mâcon by the King's council on 27th september 1413"; the "bailli" was a high level civil servant who dispensed justice, administered finances, etc. for a particular area, the "bailliage", in the name of a king or lord. The code in capital letters indicates a predicate and its associated case slots. Each predicative plane is characterized by a pair of "time references" (date1-date2) which give the duration of the episode in question. In the plane 114, the second date slot (date2) is empty because the modulator "begin" specifies that the event described, in this case the nomination, is a punctual event which does not extend in time further than the single date explicitly indicated; for details of the representation and use of temporal data in the RESEDA system, see Zarri (1983). The presence of the modulator "soc" (social) and of a "SOURCE" conforms to the fact that Robert de Bonnay's appointment occurs in the context of his professional activity. "Robert de Bonnay" is a label (address) pointing to the group of planes available in the system which concern the corresponding personage; "bailli" (meaning the "post of bailli") and "king's-council" are entries in the RESEDA lexicon organized as a tree structure; "Mâcon" is obviously the "location of the object".

3 "LEVEL ZERO" OF RESEDA's INFERENCE PROCEDURES

From the point of view of the system's organization, the different inference procedures implemented in RESEDA have the following characteristics:

- They are all based on the same "kernel" of elementary functions from RESEDA's interpreter; this kernel is the system's "match machine". Thus, relative to the architecture illustrated in figure 1, the match machine forms the inner core of the Knowledge Processing Unit (KPU).
- Only high level inferences require the recourse to real "inference engines" of the type encountered in MYCIN or PROLOG, which necessarily make use of the complex information in the rule base. The low level ("level zero") does not require access to data in this base and is provided by using the "match machine" alone.

The example in figure 2 should make it clear what we mean by "level zero inference". The question at the top of the figure corresponds to the following natural language formulation: "Give me any information existing in the fact database concerning the posts that Robert de Bonnay may have had during 1414". In the present state of the system, it is up to the user to formulate his request in terms of RESEDA's metalanguage.

The original question is thus reduced to a "search pattern", which is one of RESEDA's fundamental concepts: a "search pattern" carries the essential elements, expressed in terms of RESEDA's metalanguage, which it is necessary to search for in the fact database; the aim is to deduce from this base all the planes which fit the pattern. A search pattern may originate from outside the system, if it is a direct translation of a query posed by the user. On the other hand, it may be automatically generated by the system, as will be clarified later, during the execution of a high-level inference procedure.

To recover information from the fact database, a search pattern, whether it originates internally or externally, calls upon the modules of the match machine in the KPU. The "match machine" is made up of three main modules; all three have a search pattern as starting point:

- The "PLANE-SELECTOR" has no other input, and produces a first list of labels (addresses) of planes contained in the fact database (see the list of eight addresses, "selected planes", indicated in figure 2).
- The "PARSER" receives as input the search pattern and a plane (the address of which is in the list produced by the plane selector). It compares the two to decide whether the plane matches the model or not. (In the case of figure 2, only planes 114 and 115 were retained by the module as truly corresponding to the entry pattern).
- The "VARIABLE-ASSIGNER" shares the inner core of the match routines with the PARSER; it has been developed according to the needs of the high level inference procedures - and thus does not come into play at level zero, it will be mentioned again in chapter 5.

The reason for using a preselection module ("PLANE-SELECTOR") working with the match module itself ("PARSER") is mainly linked to the fact that the syntactic structure of the predicative planes is normally far more complex, see for example plane 63 in figure 3, than it appears at a first glance when examining the very simple examples given in figure 2 (planes 114 and 115). Thus, it is an advantage to delay the match with the patterns until it is reasonably certain that a group of planes a priori relevant has been isolated (the eight "selected planes" in figure 2).

The criteria used in establishing this first sub-group are, in order, the following:

```
   query (search pattern)
[1-january-1414,31-december-1414]
   BE-AFFECTED-BY   SUBJ   Robert-de-Bonnay
                    OBJ    post
   answer
selected planes : 8
114 115 116 117 143 144 145 159
matched planes : 2
114 115
114) begin+soc+BE-AFFECTED-BY
                    SUBJ     Robert-de-Bonnay
                    OBJ      bailli:Mâcon
                    SOURCE   king's-council
                    date1 :  27-september-1413
                    date2 :
                    bibl. :  Demurger1,234
115) begin+soc+BE-AFFECTED-BY
                    SUBJ     Robert-de-Bonnay
                    OBJ      seneschal:Lyon
                    SOURCE   king's-council
                    date1 :  27-september-1413
                    date2 :
                    bibl. :  Demurger1,234
   Do you wish to transform the pattern
   (.yes/.no)
   .no
```

figure 2

- Select the planes where "personages" named in the search pattern appear.
- Select the planes built around the same predicate appearing in the search pattern.
- Select the planes where there is a correspondence between the temporal information encoded in the "search interval" of the pattern and the dates indicated in the planes. The search interval - see the information between square brackets associated with the pattern in figure 2 - is employed to limit the search to the slice of time which it is considered appropriate to explore in the fact database.

The search for a correspondence between search interval and dates is particularly complex, see Zarri (1983:100-106). There is no question of going into more detail here, and we must limit ourselves to a few remarks concerning the results of figure 2. The answer to the question asked, which concerned the posts held by Robert de Bonnay in 1414, is provided in the form "On the 27th of September 1413 (i.e. the year before), Robert de Bonnay was named (begin+BE-AFFECTED-BY) 'bailli' of Mâcon and seneschal of Lyon"; in the absence of any information to the contrary explicitly included in the base, the preselection algorithms were able to infer, despite the fact that the dates in the plane and the search interval do not coincide, that it is possible that in 1414 Robert de Bonnay still occupied those posts. Predicative planes like 114 and 115 are indeed registered in the fact database according to the "category" of dating known as "posteriority": the state of affairs represented by the events (in this case, for example, the fact of being in possession of the post of "bailli") is set up "as starting from" the only date (date1 : 17-september-1413) shown explictly in these planes (Zarri 1983:91-97). These simple temporal data level inferences are amongst the most interesting of RESEDA's level zero inference system.

The kind of match performed by the PARSER has certain characteristics which differentiate it from a normal unification even if the basic mechanism is, as in PROLOG (Colmerauer 1983), a comparison of labelled trees. These characteristics are as follows:

- The match routines do not operate on the temporal information contained in the plane and the pattern, since these have already been compared by the PLANE-SELECTOR module.
- A plane is considered compatible with the proposed pattern if it contains at least all the explicitly declared elements in that pattern.
- When entering the PARSER module, the pattern can only contain variables known as "implicit" variables. These are "zero = empty position" and "generic term = root of a sub-tree in the lexicon" variables.

The distinction between "explicit" and "implicit" variables is vital in RESEDA; "explicit" variables are used in high level inferences where the same variable is called a number of times during the procedure and must thus be individualized by an explicit "name". It should be noted, however, that in accordance with the principle of using the same basic building blocks throughout RESEDA's interpreter, explicit variables must be reduced to implicit variables when they are processed by the inner core of the match routines, which, as I have already mentioned, is shared between the PARSER and the VARIABLE-ASSIGNER modules; the latter deals with explicit variables. Returning now to the results in figure 2, one can see that the term "post" in the pattern, which is the head of the sub-tree of the same name in the lexicon, has been processed by the PARSER as an implicit variable, which enabled us to recover the two specific terms "bailli" and "seneschal" pertaining to this sub-tree in planes 114 and 115. A variable of type "zero" could have been used as a filler for the "SUBJ" case-slot in a pattern equivalent to the question "Who held the post of 'bailli' in Mâcon in 1414".

It is also important to point out that any element which can be used in a well-formed expression in RESEDA's metalanguage is a "tagged" element : in its internal representation, two bits are reserved to specify the "type", that is the category of the metalanguage to which it belongs (temporal information, location, predicates, modulators, cases, classes and sub-classes of the lexicon, variables, etc.). The advantages are obvious from the point of view of the simplification of coherence checks and, in particular, of being able to build a fast data type checking mechanism to speed up the match process. One might note that a similar decision was taken in the design of the Personal Sequential Inference Machine, PSI, which was constructed within the framework of the "Fifth Generation" project, see Yokota et al. (1983).

4 INFORMAL INTRODUCTION TO THE HIGH LEVEL INFERENCE PROCEDURES

The aim here is to provide an informal description of the two kinds of high level inference operation, relying on information in the rule base, that are implemented in the system: these are known as transformations and hypotheses. Section 5 will deal with the "inference machines" which, in practice, make these operations possible.

4.1 Transformations

Let us suppose that, when asking the system the question that corresponds to the search pattern in figure 2, we obtain no answer; or that having recovered planes 114 and 115, we would like to know more. It is then possible (by answering "yes" to RESEDA's question) to allow the system to automatically "transform" the initial search pattern by substituting it with another "semantically equivalent" pattern, if one exists, and then executing the corresponding program. "Semantically equivalent" means that the information eventually obtained with the new pattern should "imply" the information that we did obtain or would have obtained with the original pattern.

To keep to an extremely simple example, consider the transformation of figure 3, allowing us to change a search pattern formulated in terms of "end+BE-PRESENT" into a new one in terms of "MOVE", which can be submitted, in turn, to the usual preselection and match procedures. This formal rule translates the common sense rule "If someone goes from one place to another, he has certainly left his starting point": the justification of the use of substitution in figure 3 lies in the fact that any information about some personage $\underline{x}$ having moved from $\underline{k}$ to $\underline{l}$ is at the same time a response to any query about the possibility of his no longer being at place $\underline{k}$. Note that, in the terms of RESEDA's metalanguage, the movements of a personage are always expressed in the form of a subject $\underline{x}$ which moves itself as an object.

```
t1) end+BE-PRESENT   ──▶   MOVE   SUBJ x : k
       SUBJ x : k                 OBJ  x : l

    x = <personage>
    k,l = <location> ; k ≠ l
```

figure 3

On a conceptual level, it is worthwhile noting that the explicit "variables" which appear in the original search model ($\underline{x}$ and $\underline{k}$ in figure 3) must appear in the transformed model and/or in the "constraints" associated with the new variables ($\underline{l}$ in figure 3) introduced at the level of this transformed model, see rule t1. This ensures the logical coherence between the two parts of the transformation; as we said, the model on the right hand side must indeed "imply" the one on the left. The values which replace the variables in the retrieved plane (or planes) using the transformed model must obviously respect the constraints associated with all the explicit variables which appear in the transformation.

A second example of a transformation is that given in figure 4; the common sense rule underlying this formalism is: "To know if someone ($\underline{x}$) is employed by an organization ($\underline{p}$) which belongs (SPECIF) to a second person ($\underline{y}$), one might check, amongst other things, whether he regularly receives money ($\underline{q}$) from this organization".

```
t2) BE-AFFECTED-BY ◀────▶ BE-AFFECTED-BY
      SUBJ  p (SPECIF y)    SUBJ x
      OBJ   x (SPECIF q)    OBJ  r
                            SOURCE p (SPECIF y)

    x,y = <personages> ; p = <social-body>
    q = <permanent-post>
    r = <permanent-salary>
```

figure 4

Two points can be made about the transformation in figure 4:

- The first is that this transformation, contrarily to the one in figure 3, is "two way", i.e. it can be used not only for transforming the pattern on the left hand side into the one on the right, but also for transforming the pattern on the right into the one on the left. In this case, it is not, in fact, possible to state clearly that the right hand side should imply the left hand side rather than the contrary; this is echoed, at a formal level, by the equality between the number of variables which appear on both sides.

- In figure 4, the rule for creating well-formed transformations - which requires that any variable appearing in the pattern on the left hand side must also be found in the pattern on the right hand side - takes, for the variable $\underline{q}$, the form of an implicit link between those sub-trees of the lexicon whose terms can substitute $\underline{q}$ and $\underline{r}$; the restriction for $\underline{q}$ in the "permanent-post" zone of the "post" tree demands an analogous restriction in the "permanent-salary" zone of the "salary" tree.

4.2 Hypotheses

A second category of inference rules makes it possible to search for the hidden "causes", in the widest sense of the word, of an attested fact in the database. For example, after retrieving planes 114 and 115 in response to the question in figure 2, and if we assume that the "reasons" for the nominations are not explicitly recorded in the fact database, the user will now be able to ask the system to automatically produce a plausible explanation of these facts by using a second category of inference rules, the "hypotheses".

In order to give an initial idea, on an intuitive level, of the functioning of the hypotheses, figure 5 shows the formulation in natural language of two characteristic hypotheses of the RESEDA system. The first part of each of these rules corresponds to a particular class of confirmed facts (planes) for which one asks the "causes". For example, the planes in figure 2 are clearly an exemplification of the first part of the second hypothesis in figure 5. In RESEDA's terminology, the formal drafting of this first part is called a "premise". The second part (the "condition") gives instructions for searching the database for information which would be able to justify the fact which has been matched with the premise. That is, if planes matching the particular search patterns which can be obtained from the "condition" part of the hypothesis can be found in the database, it is considered that the facts represented by planes could constitute the justification for the plane-premise and are then returned as the response to the user's query.

Let us now look in some detail at the hypothesis h2 in figure 5. A whole family of inference rules expressed in RESEDA's metalanguage corresponds in reality to the natural language formulation given in h2; one of these realizations is shown in figure 6.

---

h1) ... one might take a particular attitude in an argument

BECAUSE

one has close links with one of the parties in a conflicting situation

h2) ... one might be chosen for a (official) post

BECAUSE

one is attached to a very important personage who has just taken power

figure 5

The meaning, in clear, of the formalism in figure 6 is as follows. To explain what brought the administration n to give post m to x, the hypothesis suggests we check in the system's database for the following two facts, which must be verified simultaneously (operator "Λ", "and"):

- At a date that is previous, but sufficiently close to the date of nomination d1, the administration n comes under the leadership of y (n starts to have y for chief (lid = leader)). b1 and b2 allow the system to automatically construct the two limits (bound1, bound2) of the search interval to be associated with the search pattern extracted from condition schema A.
- x was permanently employed by an important person y (the seigniorial administration p specific to y was "augmented" by x) during a sufficiently long period round the date of nomination (schema B).

```
premise : α

α) begin+soc+a+BE-AFFECTED-BY      SUBJ  x
                      OBJ     m
                      SOURCE  n
                      date1 : d1
                      date2 :

constraints on the variables of the premise :

   a ≠ neg,lid ; x = <personage>
   m = <monarchic-post>|<seigniorial-post>
   n = king's-council | lord's-council
if m = <monarchic-post>
                then n = king's-council
if m = <seigniorial-post>
                then n = lord's-council

condition : A Λ B

A) begin+lid+BE-AFFECTED-BY
       SUBJ     n
       OBJ      y
       bound1 : b1
       bound2 : b2

B) BE-AFFECTED-BY     SUBJ       p (SPECIF y)
                      OBJ        x (SPECIF q)
                      bound1 :  b3
                      bound2 :  b4

constraints on the variables of the condition
schemata :

y = <personage>; x ≠ y
p = <seigniorial-organization>
q = <permanent-post>

information for creating a search interval :

b1 = d1 - 1 month ; b2 = d1
b3 = d1 - 2 years ; b4 = d1
```

figure 6

Note also, in the formulation of the premise α, the introduction of a modulator variable "a" to explicitly exclude from the planes which will be explained by the hypothesis all those planes which involve the take-over of some organization (a = lid), or which relate an aborted nomination (a = neg).

Figure 7 shows planes (145 and 150) obtained by means of this hypothesis in the case of a query about the possible causes of the events related in planes 114 or 115 (Robert de Bonnay's nomination).

```
150) begin+lid+BE-AFFECTED-BY
        SUBJ     king's-council:Paris
        OBJ      (COORD Louis-d'Anjou
                 Charles-d'Orléans Jean-de-
                 Bourbon Dauphin-Louis
                 Jean-de-Berry
                 Bernard-d'Armagnac):Paris
        date1 :  1st-september-1413
        date2 :
        bibl. :  consensus
```

"On the 1st september 1413, the leaders (COORD = COORDination) of the faction favourable to the Duc d'Orléans took control ('lid') of the administration of the state.

```
145) BE-AFFECTED-BY
        SUBJ     prince's-court (SPECIF
                 Charles-d'Orléans) : Blois
        OBJ      Robert-de-Bonnay (SPECIF
                 chamberlain)
        date1 :  8-april-1409
        date2 :  (1415)
        bibl. :  Demurger1,234
```

"Robert de Bonnay held the post of chamberlain to the Duc d'Orléans (the court of Charles d'Orléans, whose residence was at Blois, was 'augmented' by Robert de Bonnay) from 8 april 1409 until 1415 (reconstituted date)"

figure 7

Let me finish by emphasizing that the system does not restrict itself to displaying the planes recovered from the database, but also explicitly displays the new "causality" relationships found in the form of a "parenthetic plane", [114 (CAUSAL 150 145)] for example. Parenthetic planes are, in the same way as predicative planes, elementary units of meaning accepted by the fact database, and are used to represent logical links (causality, goal, motivation, etc.) existing between predicative planes. A parenthetic plane such as the one given can thus be permanently stored in the fact database after being validated by the user; this second type of high-level inferences, the hypotheses, provides the system with an, albeit elementary, learning capability.

5 THE COMPUTATIONAL STRUCTURE OF RESEDA's INFERENCE ENGINE

The high level inferences introduced in the previous chapter are executed by an inference engine. The behaviour of this engine is defined by two machines in RESEDA's interpreter, the "transformation" and the "hypothesis" machines, which are part of the KPU, see figure 1, and which make use of the match machine. They can function independently, or in an integrated fashion, with the transformation machine being called in the context of the hypothesis machine. This means that - whenever all the possibilities of matching associated with a search pattern extracted from the condition schema "i" of a hypothesis have been exploited - the pattern can still be used by transforming it before "backtracking" to level i-1, thus retrieving new values for the variables which permit the "forward" processing of the hypothesis to continue.

5.1 Selection modules and execution modules

The "hypothesis machine" consists of two main modules:

- an "H-SELECTION" module which, from a predicative plane P existing in the base, provides a list of addresses of hypotheses liable to explain P ;
- an "H-EXECUTION" module which, given a predicative plane P and the address of an hypothesis H, displays all the planes offered by H as an explanation of P.

The execution module in turn consists of three sub-modules: a premise schema is processed by a sub-module EXECPREM; a condition schema is processed by a sub-module EXECCOND; EXECPREM and EXECCOND ensure the "forward traversal" in the "choice tree", see below; the "backtracking" is ensured, on the other hand, by the sole sub-module REEXEC.

In the same way, the "transformation machine" consists of two main modules:

- a "T-SELECTION" module which, from a search pattern R, provides a list of addresses of transformations liable to operate on R ;
- a "T-EXECUTION" module which, from a search pattern R and a transformation T, provides the model transformed from R by T.

This way of structuring the two machines is linked to the particular way in which a high-level inference rule is executed and which includes two distinct steps:

- a "bottom-up" phase which involves going from a particular expression compatible with RESEDA's metalanguage (a plane which is to be explained in the case of a hypothesis or a pattern for which a substitute must be found in the case of a transformation) and selecting in the rule base one or more "rule heads" (premises or left hand sides of a transformation) which define a general class encompassing the expression in question: this is the "selection" phase;
- a "top-down" phase, during which the program that corresponds to the rule or rules selected is executed generating

particular search patterns with which to explore the fact database (execution phase).

5.2 Exploration of the choice tree

The execution modules are rather more interesting than the selection modules from the point of view of their logical architecture; some details about selection modules can be found in Zarri (1984:102-103). I have already mentioned that the execution strategy implemented is "top-down"; in fact, any of RESEDA's high-level inference rules, hypothesis or transformation, can be defined as an implication of the type: X IF Y1 AND Y2 ... AND Yn, where "X" is a hypothesis premise or the left(right)-hand side of a transformation, whereas "Y1, ... Yn" are condition schemata (A and B in figure 6), or right(left)-hand sides of a transformation. The execution modules interpret each implication as a procedure which reduces problems of the form X to sub-problems Y1 AND ... AND Yn. Each of the sub-problems Yi in turn is interpreted as a procedure call which gives rise, at the end, to search patterns to be tested in the fact database. Note that, as in PROLOG, procedure calls Yi are executed left to right in the order in which they are written; thus, in the hypothesis in figure 6, condition schema A will be interpreted before schema B.

Two remarks of a general nature can be made immediately:

- The functioning of the execution modules of RESEDA's inference engine which we have just defined corresponds to the top-down functioning of PROLOG's interpreter (Kowalski 1982; Colmerauer 1983). It is probable that this similarity exists because PROLOG is permeated by a philosophy of "resolution by searching through a database" ; in which case a top down approach would seem more or less obligatory.
- In the long run, the execution of RESEDA's high level inference rules has identical characteristics whether in a "hypothesis" or "transformation" framework : only at the end of execution will the planes retrieved be interpreted as events explaining a certain fact (hypotheses) or as an indirect response to a question put to the system (transformations). This explains how the same blocks in the interpreter can be used in contexts which seem a priori quite different.

Let us take a look now at some technical details; for a more complete description, see Zarri et al. (1983:97-124).

The search for solutions within the fact database by means of high-level inference rules of the "hypothesis" type amounts to the exploration of a "choice tree". The branches of this tree originate at two different levels during the use of the match machine within the inference engine:

- The search patterns extracted from the condition schemata are first processed by the PLANE-SELECTOR module of the match machine as usual. A first source of branching arises from the fact that this module will normally provide in response a whole series of labels of planes, see figure 2; these labels will be stored in a Plane Address List (PAL).
- The planes that correspond to these addresses will then be examined one by one by the VARIABLE-ASSIGNER module of the match machine. As stated previously, see section 3, this module has much the same duties as the PARSER module; in other words, it checks in detail whether a preselected plane does, in fact, correspond to the search pattern; additionally, it gives all possible bindings ("plane values" = dates, locations, modulators, personages, terms of the lexicon) for each explicit variable needed by the originating condition schema. The fact that there can be a number of combinations of possible bindings for the set of explicit variables appearing in a given condition schema, which is the second source of branching, is due to the presence of "lists" in the planes of the base, see for example the COORDination list in plane 150 (figure 7).

The mechanism for progressively building the choice tree in the case, for example, of a hypothesis made-up by two condition schemata is illustrated in figure 8. The element marked "R" appearing at the top of the tree is the search pattern extracted from the premise schema S0. This search pattern must directly match plane P which relates the event that is to be explained, for example plane 114 or 115 in figure 2, see also section 4.2; thus, at this level, only the VARIABLE-ASSIGNER module of the math cmachine is used without recourse to the PLANE-SELECTOR. This match operation will give rise in general to a number of different combinations of bindings (B) for the variables of the premise schema; four of these are shown in figure 8. Each of these combinations will be used in turn to construct a search pattern R from the condition schema S1: this is the beginning of a "normal cycle" of the tree construction. Figure 8 presupposes that a call to the PLANE-SELECTOR module of the match machine leads to a failure for the first, third and fourth of these search patterns, whereas the second provides three preselected planes and thus a PAL of three plane labels; a PAL containing multiple labels, such as these, is the first source of branching associated with the processing of a condition schema. The planes thus obtained must now actually be compared, using the VARIABLE-ASSIGNER module, with the search pattern which gave rise to them; the match can lead to a failure or produce one or more combinations of bindings for the variables in condition schema S1 (second level of branching); etc.

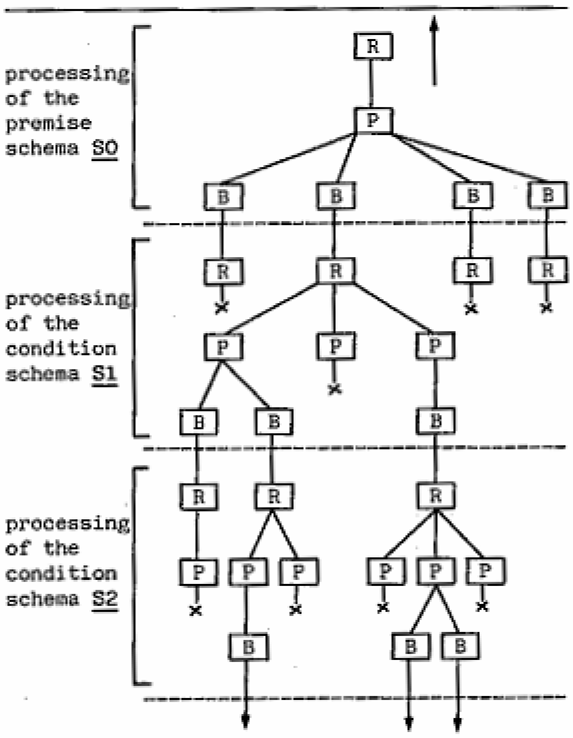
processing of the premise schema S0
processing of the condition schema S1
processing of the condition schema S2
R
P
B

figure 8

To refer now to a concrete example which has already been examined, the only possible search pattern extracted from the condition schema A in figure 6 (where the variable n will obviously already have been substituted by the value "king's-council", retrieved, by matching the premise, within plane (114 or 115) that is to be explained, see figure 2) will give rise, thanks to the PLANE-SELECTOR module, to a PAL containing 15 preselected planes; each plane of the PAL will therefore produce a branch in the choice tree.

When plane 150 (figure 7) which pertains to this PAL is examined, the six personages appearing in the COORD list filling the OBJ slot may be acceptable values for the new variable y introduced by schema A; each of these bindings is in turn substituted for y giving rise, in schema B, to six different combinations of values for the two variables already bound, x ("Robert de Bonnay") and y, and finally to six different search patterns extracted from schema B (second level of branching), etc.

It now becomes easier to explain the function of the sub-modules EXECPREM, EXECCOND and REEXEC. EXECCOND is called each time there exist conditions favourable for advancing in the hypothesis, in other words, for being able to process a new condition schema ; its function is to find a series of values which could be "acceptable" bindings for the variables introduced by the schema in question. "Acceptable" means that these bindings were retrieved from planes in the base which were syntactically comparable with the search patterns extracted from the condition schema, and that they satisfy the semantic constraints associated with the variables. EXECPREM processes the premise schema with the same aims as EXECCOND, but is called in a situation where no bindings for the variables of the hypotheses exist so far. To reach their goals, EXECCOND and EXECPREM must carry out a fairly complex sequence of operations (instantiating the variables of the schema being processed with the values which may already have been retrieved from previous schemata, constructing search intervals and search patterns, determining the combinations of binding variables, etc.). For example, to verify that a certain candidate value satisfies the constraints associated with a variable requires that one has firstly solved the possible "conditional constraints" of the variable, see the constraints of the type "if/then" in the example of figure 6.

EXECPREM and EXECCOND perform the forward traversal of the choice tree; if the attempt to find a combination of values valid for the new variables of a condition schema fails, or if it succeeds, and the condition schema examined was the last one in the hypothesis, then the inference engine is forced to backtrack. This is done by the REEXEC sub-module, which backs up to the level of the schema Si-1 preceeding the one, Si, were the dead-end was found; an attempt by REEXEC to reach a level higher than the premise indicates that the traversal of the choice tree is finished. REEXEC would not function correctly if, at every level "i" encountered during a hypothesis, the "environment" of the process was not stored. The information which defines the environment is contained in a "Hypothesis Structure" (HS); for more on the use and implementation of the notion of environment in a top-down execution framework, see Zarri et al. (1983:97-124) and Bruynooghe (1983).

It should be obvious from the above that, in the execution modules of RESEDA's inference engine, the forward traversing of the choice tree is of the depth-first type: the backtracking, at least in the present prototype, is however systematic. In other words, REEXEC is called even if the exploration of a branch of the tree is terminated and an acceptable solution is found. This method enables us to have a full panorama of the successes and failures associated with the different branches of the choice tree and is particularly well suited to the aims of an Information Retrieval system; obviously, this can be computationally expensive.

In the case of RESEDA, two fundamental architectural characteristics allow the inference engine to operate with a quite remarkable level of efficiency while maintaining exhaustive backtracking; these are:

- the use of the PLANE-SELECTOR module of the match machine within the inference engine, which allows inferencing to be carried out on a much-reduced sub-set of the fact database;
- the notion of "type" mentioned in section 3, which speeds up considerably the match operations themselves (VARIABLE-ASSIGNER module).

This obviously does not mean that we refuse to consider the possibility of adding to RESEDA the option of superposing, when necessary, a "best first" approach onto the basic "depth first" mechanism. Studies are being carried out at the moment which may lead to the introduction in RESEDA of an "intelligent" control option of the kind described in Gallaire and Lasserre (1979) or Pereira and Porto (1982).

I shall end this section with a few remarks about the "integrated mode" of functioning of the transformation and hypothesis machines. The interest from a practical point of view of this possibility is obvious: <u>it enables a considerable increase of the explanatory power of the hypotheses</u>. As mentioned above, using the transformation machine inside the hypothesis machine means that - whenever all the possibilities associated with a search pattern extracted from a condition schema "<u>i</u>" have been exploited (that is, <u>PALi</u> has been completely explored) - <u>the pattern can still be employed by transforming it before backtracking to level i-1</u>.

Suppose for example that the <u>PAL</u> associated with the search pattern that can be derived from the condition schema B of figure 6 after having assigned to <u>y</u> the value "Charles d'Orléans" has been completely explored. The pattern can then be transformed using transformation t2 of figure 4 from left to right, checking now whether Robert de Bonnay received any form of permanent salary (<u>r</u>) from an organization (<u>p</u>) belonging to Charles d'Orléans (variables <u>x</u> and <u>y</u> have already been bound in the stages that preceded the processing of schema B in figure 6; obviously, the use of the same variable names for this schema and the transformation in figure 4 is merely a didactic artifice serving to clarify the operations performed). One may expect in this way to retrieve plane 151 of figure 9, relating the fact that Robert de Bonnay, in 1409, started to receive a pension from Charles d'Orléans, which confirms the existence of a strong link between the two personages.

```
151) begin+BE-AFFECTED-BY
         SUBJ     Robert-de-Bonnay
         OBJ      pension
         SOURCE   prince's-court (SPECIF
                  Charles-d'Orléans) : Blois
         date1 :  1409
         date2 :
         bibl. :  Demurger1,999
```

figure 9

6 CONCLUSION

In this paper, I have concentrated on a description of the inference mechanisms realized by the "machines" which are part of the Knowledge Processing Unit (KPU) of RESEDA, an Intelligent Information System (IIR) dealing with complex "biographical" data which is compatible with the aims of the Japanese Fifth Generation project. In particular, I have tried to evidentiate two fundamental principles underlying the system's architecture:

- allow the execution of inference procedures of a sufficiently general nature to be something of a quantum jump when compared with usual Information Retrieval techniques;
- maintain criteria of computational economy and efficiency: the use of real "inference engines" only in high-level inference operations; systematic multiple use of basic logical building blocks; the use of "tagged" elements and of "preselection" operations; etc.

I would like to add that the choice of APL for the programming of RESEDA's interpreter (about 150 KBytes) is to a large extent responsible for the indisputable efficiency of the system. A high level, function oriented interpreted language, APL provides a programming environment comparable to that of many LISP dialects, with the advantage of a memory management which is both more efficient and less greedy than that imposed by the "list-oriented" philosophy of LISP. Its concision, and the power of its "assembler type" operators (for the manipulation of bit strings, for example, or for the execution of complex arithmetic or algebraic operations) have, on the other hand, no equivalent in other more usual AI languages; its array oriented philosophy has proved indispensable for the realization of some of RESEDA's fundamental tools, such as the PLANE-SELECTÓR of the match-machine or the selection modules of the inference engine.

Pending the successful outcome of research into the "ideal logic programming language", see Robinson (1983), APL should not be left on the sidelines when considering the shorter term objectives of the Fifth Generation project.

REFERENCES

Amamiya, M., Hakozaki, K., Yokoi, T., Fusaoka, A., and Tanaka, Y., New Architecture for Knowledge Base Mechanisms, in Fifth Generation Computer Systems, Moto-oka, T., ed. North-Holland, Amsterdam, 1982.

Bruynooghe, M., The Memory Management of PROLOG Implementations, in Logic Programming, Clarck, K.L., and Tärnlund, S.A., eds. Academic Press, New York, 1982.

Colmerauer, A., PROLOG in 10 Figures, in Proceedings of the 8th International Joint Conference on Artificial Intelligence - IJCAI/8. William Kaufmann, Los Altos, 1983.

Faribault, Marthe, Léon, Jacqueline, Meissonnier, V., Memmi, D., and Zarri, G.P., From Natural Language to a Canonical Representation of the Corresponding Semantic Relationships, in Cybernetics and Systems Research 2, Trappl, R., ed. North-Holland, Amsterdam, 1984.

Fillmore, C.J., The Case for Case, in Universals in Linguistic Theory, Bach, E., and Harms, R.T., eds. Holt, Rinehart and Winston, New York, 1968.

Gallaire, H., and Lasserre, Claudine, Controlling Knowledge Deduction in a Declarative Approach, in Proceedings of the IJCAI/6. William Kaufmann, Los Altos, 1979.

Kowalski, R., Logic as a Computer Language, in Logic Programming, Clarck, K.L., and Tärnlund, S.A., eds. Academic Press, New York, 1982.

Moto-Oka, T., ed., Fifth Generation Computers Systems. North-Holland, Amsterdam, 1982.

Pereira, L.M., and Porto, A., Selective Backtracking, in Logic Programming, Clarck, K.L., and Tärnlund, S.A., eds. Academic Press, New York, 1982.

Robinson, J.A., Logic Programming - Past, Present and Future, New Generation Computing, I, 107-124, 1983.

Yokota, M., Yamamoto, A., Taki, K., Nishikawa, H., and Uchida, S., The Design and Implementation of a Personal Sequential Inference Machine, New Generation Computing, I, 125-144, 1983.

Zarri, G.P., Building the Inference Component of an Historical Information Retrieval System, in Proceedings of the IJCAI/7. William Kaufmann, Los Altos, 1981.

Zarri, G.P., An Outline of the Representation and Use of Temporal Data in the RESEDA System, Information Technology : Research and Development, II, 89-108, 1983.

Zarri, G.P., Expert Systems and Information Retrieval : An Experiment in the Domain of Biographical Data Management, International Journal of Man-Machine Studies, XX, 87-106, 1984.

Zarri, G.P., Lelouche, R., Ornato, Monique, Faribault, Marthe, Meissonnier, V., Lee, G., et Léon, Jacqueline, Réalisations pratiques et développements théoriques dans le cadre d'un système I.A. pour bases de données biographiques : Rapport final (Rapp. LISH/363 - ATP n° 955045). CNRS, Paris, 1983.