ANALYSIS OF DIAGNOSIS PROBLEM OVER A NETWORK OF OBJECTS

AUTHOR : Olivier CROISSANT

DATE : 10.10.1985

I - DATABASE - RUN TIME OBJECTS - ACTIVE KNOWLEDGE

First, let us examine the representation problem of the passive knowledge of a network of object.

Each object (ex : DSSS, MPR, LCN, ...)

owns its proper characteristics

Name : (specific identifier of the object)
Type :
Power :
.
.
.

and relationship with other objects :

essentially 3 :
System to sub-system :
Functional link :
Physical proximity :

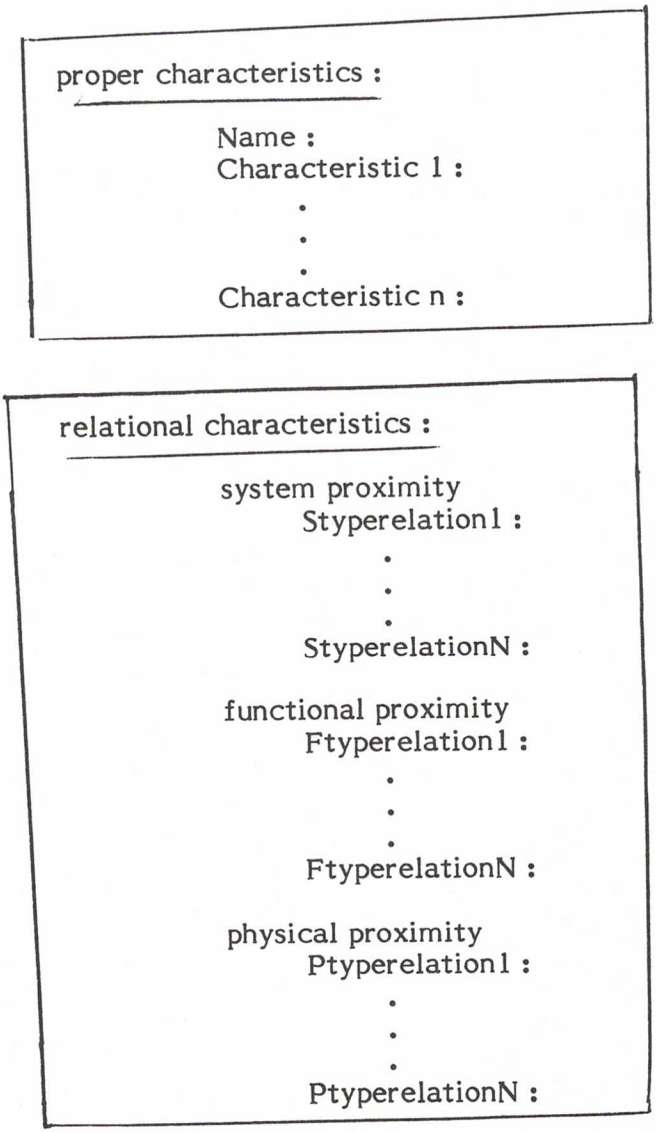
Actually, all the characteristics associated to an object can be seen as slots that we fill with names of other objects.

As we are going to use the contents of each slots for the diagnosis, it is possible that the fact for an object to be referenced in a relational slot of another object be not sufficient or too brutal.

(ex : physical proximity - which distance threshold ? or other criteria ?)

There is a solution for coding all this information inside the object : it is to structure these slots and to introduce the notion of link's types that we can also imagine as a sub-slots or as a structuration of the value of these slots (cf : value hierarchies in S.1). The idea that the physical proximity relational slots is structured to speak about different types of physical proximity (same rack, same card, adjacent rack, adjacent card, ...) can be extended to other relational slots. This allows to separate different types of functional relationship. This will be very useful to limit and control the running of deep models based on these objects. In the same way, the link system sub-system can be differenciated. It is obvious that the information contained in all these slots are redundant. But considering the fact that this will be used to do checkings, that we are not worrying at this stage of the study about memory place problems (if it remains sensible, for example, a factor 2 or 3) and that we can diminish the complexity of inference mechanism, we are going to adopt this way of seeing.

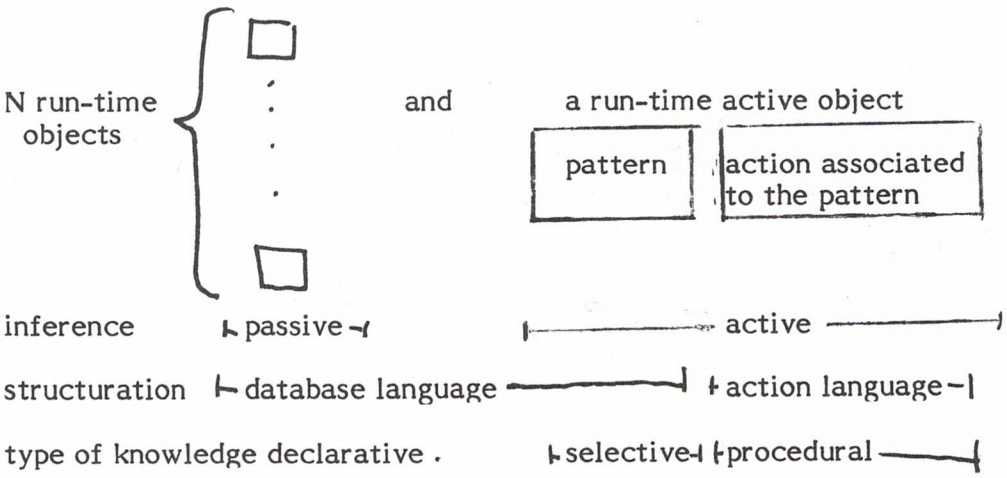
This gives us a basic object having the following shape :



The set of all these objects describes the network and forms a database for the diagnoser.

This is not the only database because other database will intervene later. The fact that it is specified to be a database is important because this means that the so-described objects are not run-time objects for the objects oriented languages. An instanciation phase is necessary because of the number of these objects and the basic mechanism of automatic reasoning : the instanciation of a pattern.

As a rule, a step of reasoning corresponds to an attempt of link between :



The instanciation mechanism is the mechanism which produces a run-time object from objects extracted from a database and in a certain context.

The necessity of this mechanism comes from the difference between the number of objects that could compose the passive knowledge in a reasoning act and the total number of object. The total set of objects constitutes the global knowledge of the system about the network it has to analyse. This mechanism is not simply a selection or a call to objects because a run-time objects should have the following characteristics :

It should include empty slots that will be variables to determine and that will characterize the precise case to deal with.

Example of such slots :

State :
Fault :
Load :
Out/in :
.
.
.

We must add intermediate slots allowing the guidance of the reasoning. These slots are added at the time of the instanciation following a scheme which depends on the type of objects (notion of instanciation type). This can be implemented as an additional slot in the basic object of the database and in an additional object which describes the slots to add.

Example of such additional object :

NAME : type II

SLOTS TO ADD : state, fault, load

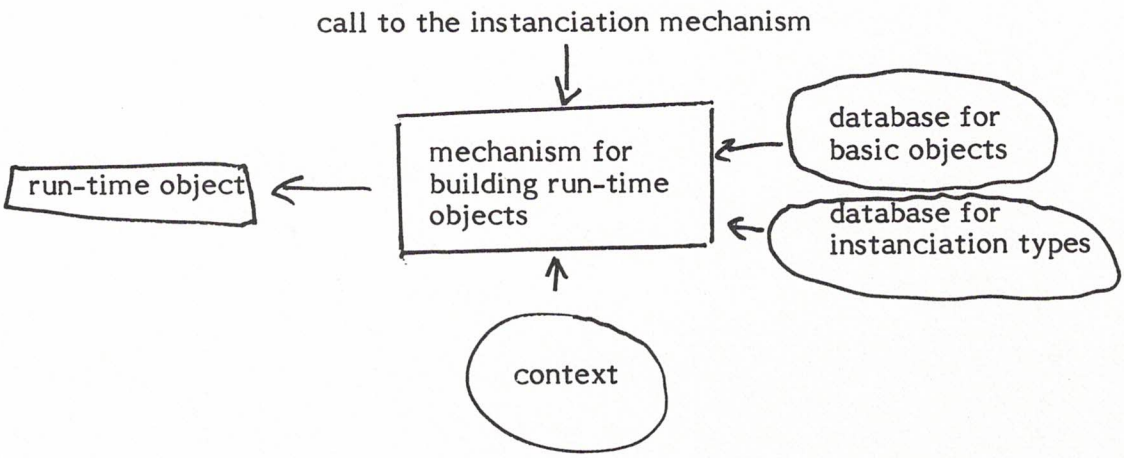
The instanciation can also depend on the context. That means that following the phases of the diagnosis or the state of the consultation, different types of instanciation could be used. We can think for the explicit instanciation done inside a procedure of a syntax of the following type :

A← INSTANCIATE (obj1, typeI)

database for the instanciation types

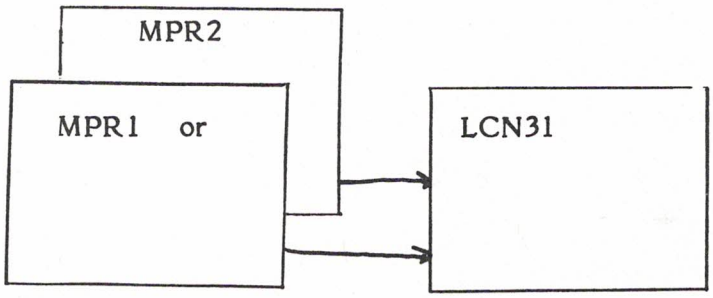
database for the basic objects

This corresponds to the following instanciation mechanism :



To illustrate the notion of typed link, a set of three objects with such links are showed on the next page.

Example of typed functional relationship :



Name : Function.MPR

Relations :

sub-systems : (OR↓: MPR1, MPR2)

functional link : (control : LCN31)

These two slots are representing in fact a single relation.

Name : MPR1

Relations :

over-systems : (OR↑: Function.MPR)

functional link : (control : LCN31)

Redundance allowing control

Name : MPR2

Relations :

over-systems : (OR : Function.MPR)

functional link : (control : LCN31)

II - INFERENCE MECHANISMS

I) INTRODUCTION : Three levels of solution

Having solved the problem of passive knowledge representation, we have to worry now about the active knowledge representation and the inference mechanisms which are attached to it.

This is the most complex part and we distinguish in the analysis three levels of solution.

1 - A first level which doesn't worry about the shape that will take the solution (deamons, rules, chaining, selection, ...) but only the logical feasability of the inference. This is the **LOGICAL LEVEL** (one discern in it logical functions).

2 - A second level where basing on temporel considerations, memory space considerations and knowledge base structure considerations, we determine the necessity of using such or such technics (forward rules, inheritances, deep inferences, ...). This is the level of functional cutting of the knowledge base or **FUNCTIONAL LEVEL** (one discern in it functional entities).

3 - A last level where we detail the syntax of the logical entities after practical considerations and by adapting strongly to the application field. This is the level of the knowledge engineering tool and structural limitations decided for each entity thanks to a compromise universality/speed adapted to each case. We will call it the **SYNTAXIC LEVEL** (one discern in it syntaxic elements of languages).

These three levels induce an approach of the problem that we must follow for efficiency reasons.

II) GENERAL FUNCTIONING

We are examining now the logical level of the inference mechanism for a knowledge based system able to achieve the following performances.

1) REAL-TIME LINK with the external world in input :

Messages reporting faults occuring on the network arrive permanantly and the inference mechanism should take it into account in the reasonning and for that purpose, they have to integrate it and to process it dynamically in the factual base in evolution.

2) INTERMITTENT LINK in input and output :

The user must be able to guide the global strategy of the diagnosis and the system have to report the conclusions of the diagnosis process.

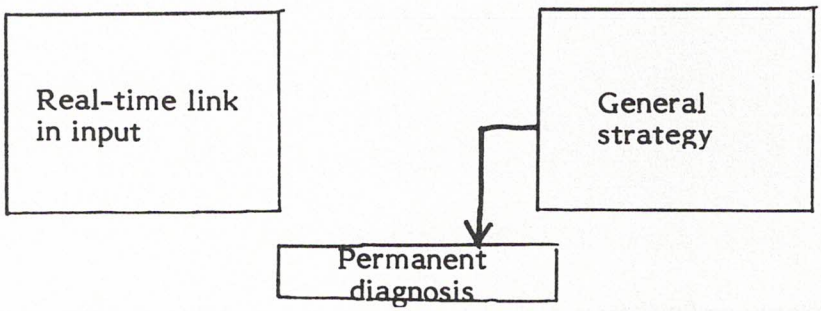
3) STATE OF PERMANENT DIAGNOSIS of the problems occuring on the network :

The system should be able to deal with the problem of N faults occuring simultaneously at different stages of solving.

These three functions are relatively independant, that means that the system where these three functions are thought and solved separately is possible. Anyway, some interactions exist.

The existence of a general strategy for a system dynamically adjustable by the user involves that, at the level of diagnosis functions, the system is working in a way that supports these changes.

Then we have the following functional scheme :



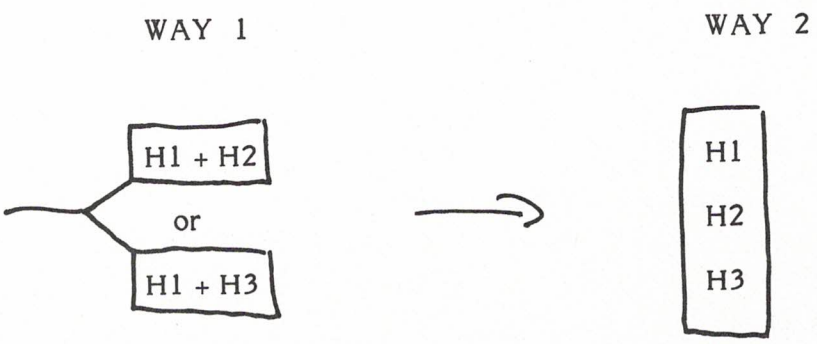
In order to determine the intensity of the different constraints involved by these three functions, we have to worry about the heart of the system and look at the current case (generic one).

Because we have decided that the system should be able to deal with several faults simultaneously, this involves the existence of several contexts of handling, each of them referenced by a working hypothesis. This hypothesis will be refined everytime fault reports arrive, this reports confirm or invalidate this hypothesis. It is obvious that a fault report can result of the interaction of several defective elements and can be explained by several different sets of defective elements. It is why two solutions of processing exist :

1) We can deal with the different possibilities of explaination for a fault by a system of hypothetical worlds. This system has the inconveniences of being extremely greedy for computing power. This can be catastrophic despite its attractive logical look (combinatory explosion of the tree).

2) We can try to make hypotheses of different level coexist. Considering that these hypotheses may have an independant life if we can minimize the importance of belonging to different branches of the hypothetical tree.

Graphically :



The justification for the choice of the second solution comes from the following fact :

It is a simplifying method which allows to concentrate on the real difficulties of the diagnosis. The domain of application may or may not present the characteristic of combinatory explosion of the arborescence. And if it happens that it is useful, we can come back to a system of hypothetical worlds when it is required.

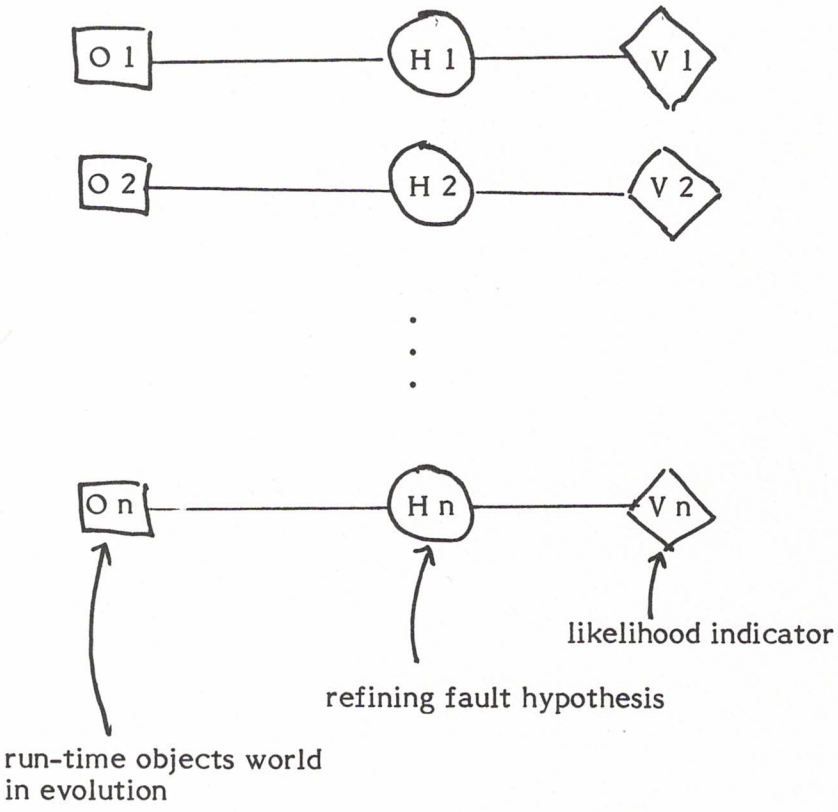
This choice done, we mustn't forget the fact that this disparity of hypotheses may have an influence on the mechanism of diagnosis, but we will mention it when it happens.

The existence of a fault hypothesis in refining is equivalent to the existence of a process. In this process, we have a window on the passive knowledge related to this hypothesis. This window consists of run-time objects progressively selected by the diagnoser and instancied in this world.

In another hand, it is natural that this hypotheses live, that means they appear and disappear. The mechanism of birth is external to the intrinsic dynamism of a process associated to an hypothesis. Anyway, the fact that an hypothesis may die is essential because it limits the increase of the number of such hypotheses and must be adjusted to make the life of system coincide with the life of the network. Nevertheless, it is likely that a shutdown of the network will involve an explosion of the number of fault messages which will involve a shutdown of the system, even if it is not hoped. Our aim is that the system's life lasts at least the life of the network.

The death of an hypothesis may intervene in two ways, either by a brutal invalidation coming from the diagnoser, or by an overstepping of a threshold showing a certain accumulation of partial invalidation coming from the diagnoser. The use of these two modes depends on the context and it is important to keep the choice for the knowledge engineer. This involves such an indicator associated to each hypothesis.

We can then summerize the current state of the diagnoser graphically by :



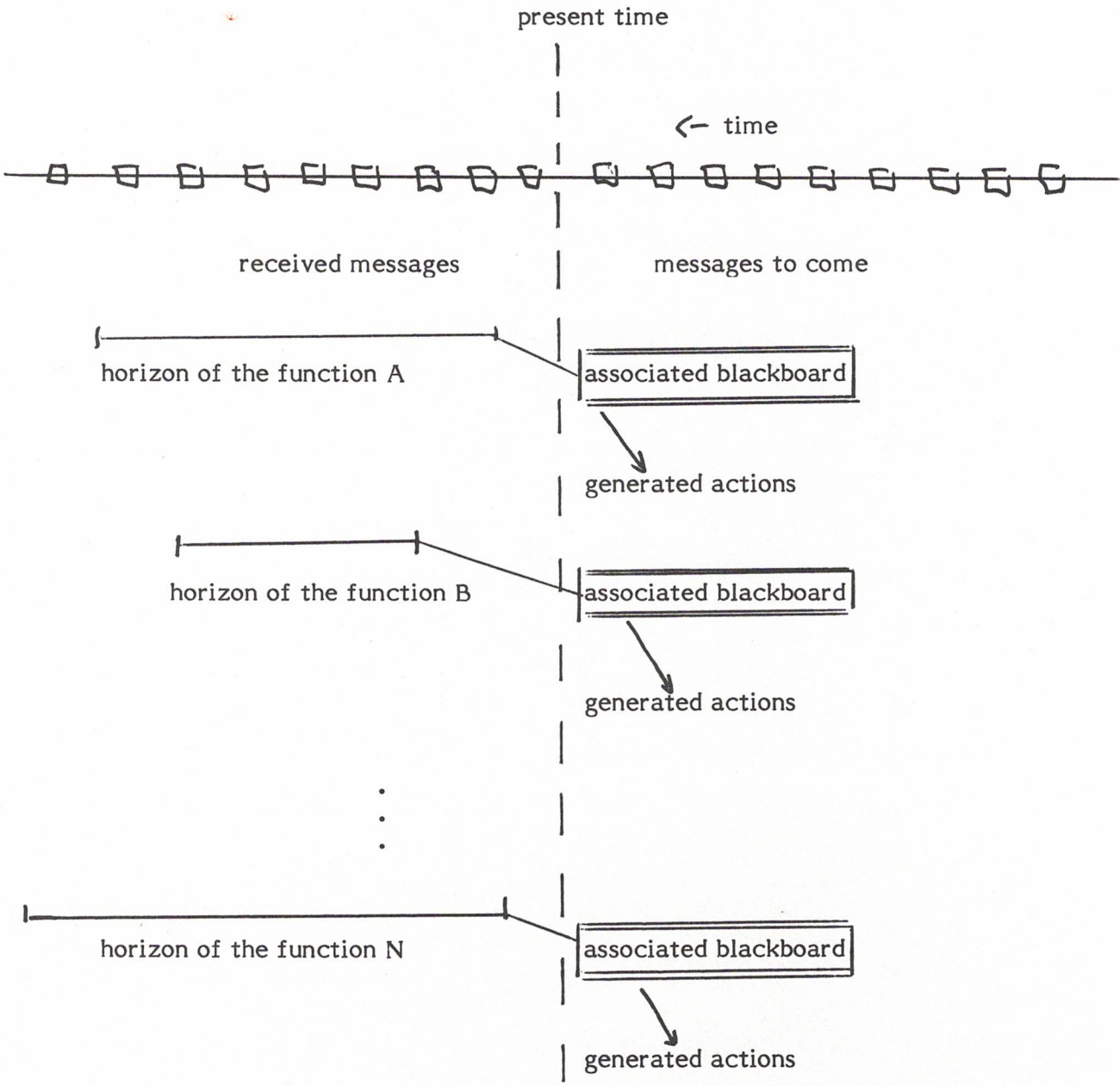
The second part of the current state is the set of fault reports arriving in an aleatory way to the diagnoser and the processing of this message. The system, outside the memory of these messages that it should have, must have a global vision where the horizon is necessary limited and centred on the present time.

Two consequences :

- the messages that arrive progressively to the diagnoser are memorized and organized in a chronological pile.

- the horizon of the diagnoser on this pile depends on the process and the function which is executed. This is analogue to a private blackboard associated to each function and looking with a finite horizon. This must involve a forgetting function (in fact long term memorization) if we want to avoid the saturation !

DISPATCHING OF THE FAULT REPORTS INSIDE THE SYSTEM



Once having reviewed the two elements which form the action field for the diagnoser (flow of fault reports and current hypotheses), we are going to examine the link between both in which consists the heart of the diagnoser. To make easy the analysis, we are going to distinguish the functions or know-how which will be reviewed in details. Every functions can be seen as a link between the flow of fault reports and the hypotheses in evolution. These functions are built in such a way that they may be considerated as independant even synchronized. Some of them will be specialized in basic functions (diagnosis evolution, hypotheses evolution, ...) ; others may be added to speed the evolution of hypotheses or to allow the choice in the diagnosis strategy.

- One of the useful functions is the loading of the run-time objects used for each hypothesis. This function is controled by a certain know-how that has to be structured in a rule-based system, as we are going to see.

- Another function will be the management of the hypotheses : birth and death of a new hypothesis.

- We can mention the management of the likelihood indicators as another function.

III) DIAGNOSIS OF A SINGLE FAULT

1) METHOD USING A DEEP MODEL

The principle of this method consists of infering an hypothesis for a fault report from a model which returns the fault reports that we have to wait for, comparing these messages with the effectively received messages and deducing the validation or the invalidation of the hypothesis. In order to generate correctly the messages, the model should examine the link between the faulty supposed elements and the other elements. The real state of the circuit and the type of fault supposed have also to be taken into account. This can be structured very simply in an object oriented language for one level of relationship.

The faulty element A is in a state $<e_i>$. One supposes a fault of type $[P_j]$ and it has links with elements B_k. These relationships of functional type should be differenciated (master of power, synchronization,...).

They have therefore to be typed : the links $[A - B_k]$ will be typed : l .

Then the message which arrives to the object B_{k_n}, being commissioned to answer will be :

$$(A , <e_i>, [P_j] , l_m)$$

It is obvious that a lot of answers will be generated after several steps of this type. The idea is to control the deepth by the type of link.

At the step N, the message which arrives to the object B will be of type :

$$(A , <e_i>, [P_j], l_{m_1}, B_{k_1}, \ldots, l_{m_{n-1}}, B_{k_{n-1}}, l_{m_n})$$

We assume that it exists a function $z (l_{m_1}, l_{m_2}, \ldots, l_{m_n})$ which allows to decide wether or not we stop the transmission of the messages. That means that the object in question is terminal node or not.

The set of concerned objects returns a list of possible fault reports. This list is compared with the list effectively received, a fuzzy matching is realized which decides wether or not the hypothesis should be kept.

2) METHOD USING TESTS ASSOCIATED TO FUNCTIONAL ENTITIES

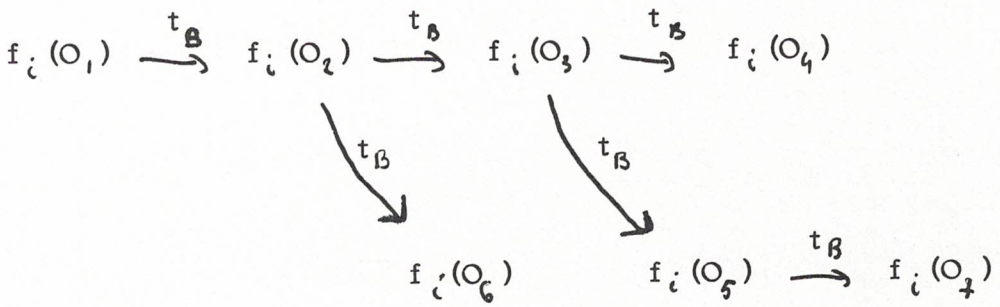
The idea that a positive test (a test whose result is that a functionality is working) can reveal that a chain of functional entities is working. We just have to code explicitly these chains. Every possible test is a pointer to a certain entity in a chain. Every fault report can be considered as a negative result of a test.

A same physical entity can intervene in several functional chains. The diagnosis process runs according to the following manner :

> After having identified a chain involved by a fault report, some elements of this chain can be tested. Progressively, this process restricts the possibilities ; then, thanks to a test hierarchy, the final fault is identified (one of the functionalities of the physical element).

A successful test proves the good running of a whole branch of a functional tree. A more or less general physical entity is associated to each functional entity. Then, this is not naturally represented by a set of objects, even in a network. The adequate representation is a tree of functional entities.

Every physical object has with other objects functional relations thanks to functional aspects of this object O. One object may have several associated functional entities f which may correspond to several elements of functional chains. A chain is defined by an identificator t_β .



3) RELATIONSHIP BETWEEN THE METHOD 1 AND 2

The method 2 can express itself in the terms of the method 1 because the existence of such functional chains (built thanks to the positive result tests) is represented with the formalism 1 thanks to a typed link defined by :

$$l = l (t_x , f_i)$$

That means that a link is defined by providing a chain and binding the objects concerned by the elements of the chain. The dependence on the entities intervene in the definition of the messages retransmitted to other objects or by return.

Given the message $(l_{m_1}, B_{k_1}, \ldots, B_{k_{n-1}}, l_{m_n})$

where $l_{m_1} = \ldots = l_{m_n} = l_{mm}$

$l_{mm} = l (t_{mm})$

if a entity $f_i (B_{k_n})$ is so that

$f_i (B_{k_{n-1}})$ and $f_i (B_{k_n})$ are related by t_{mm}

then a message is sent to B_{k_n}

otherwise the chain of messages stops.

Therefore, the formalism 2 can be reduced to the formalism 1. So, the formalism 1 has a representation power more universal than the formalism 1. But the formalism 2 has the advantage of enabling to build links and messages explicitly through an interpretation of these links thanks to chains of successful tests.

However, there is a difference between the exploitation of the representation 1 and the exploitation 2. The representation 1 enables to confirm an hypothesis whereas the representation 2 enables to refine the hypothesis. This fact is due to the possibility in the representation 2 to associate tests to chains in an easy way. The centring of the diagnosis on an element which is not at the end of a chain is done thanks to positive result tests. This induces to return a sequence of tests to isolate the faulty element. A complete diagnosis can be done by mixing model 1 and model 2. The association of tests to functional entities can be done on the representation 2 and then have to be transposed on the representation 1.

The interest of working on the model 1 is a notion of object. If we work on the formalism of chains, the delocalisation of the different facets concerning a same physical object or a same function makes the maintenance difficult and the process of windowing logically intricate.

Nevertheless, the formalism 2 is formally required to build the links between objects through the chains and very useful to reason.

IV) PARTICULAR TREATMENTS

1) THE PROBLEM OF MULTIPLE FAULTS

The problem of multiple faults appears in the formalism 2 according to the following manner :

A set of fault reports orients the search towards a specific element which, as soon as it has been identified, enables to deduce a sub-set of fault reports, which are not explained by this fault. These fault reports concern another fault which will be tracked down in the same way.

In the structure of a set of simultaneous processing hypotheses, the idea is to resend this sub-set of fault reports to a function which is going to create another hypothesis or to dispatch these reports to the other hypotheses.

2) CREATION OF HYPOTHESES

A priori, the experience shows that the fault reports concerning a same fault generally arrive at the same time, and are regrouped in the time. This will be used to create a new hypothesis which will begin to exist and to be handled by the whole set of functions of the system.

A particular function is to receive through the diagnoser from each hypothesis H a set of tests S_i to perform. This function realizes

$$\text{the set } S = \bigcup_{i=1}^{n} S$$

and suggests all these tests to the user independently of every hypothesis processing. The user can answer to some of them. The result of these tests is sent to every concerned hypotheses independently of the tests. This enables every hypothesis to profit of all the result of the tests if these ones concern it (certain tests involve several chains).

V) DESCRIPTION OF THE DIAGNOSIS MECHANISM

1) GENERAL DESCRIPTION

The fault reports first go through a waiting file working like a buffer. This waiting file is scrutinized by a rule-based system which take account of the input time associated to each fault report and all the necessary informations (can be context dependent). This system attaches to each fault report a tag whose meanning is the belonging to a fault reports group (cluster). All these fault reports probably concern the same fault.

These reports are then transformed in fault hypotheses which are pointer into arborescences. These arborescences enable to recognize wether a description of a fault belongs or is included inside another or not. The set of all these fault hypotheses is logically structured by these arborescences (indeed reduced).

Then, each of these fault hypotheses, because of its relationship to objects whose it is only one facet, asks them an answer to the question : which fault reports must we wait for with such a fault. This answer is collected through the set of typed relations existing between the objects concerned by the fault.

Each of these fault reports (the collected ones) is compared with the set of fault reports which arrived to the system and which are associated by the mean of the tag which follows the whole processing. This is achieved by a rule-based system. The goal of this system is to know wether or not the messages may be included in the list of the ones which have been received. The basic mechanism should be a simple unification.

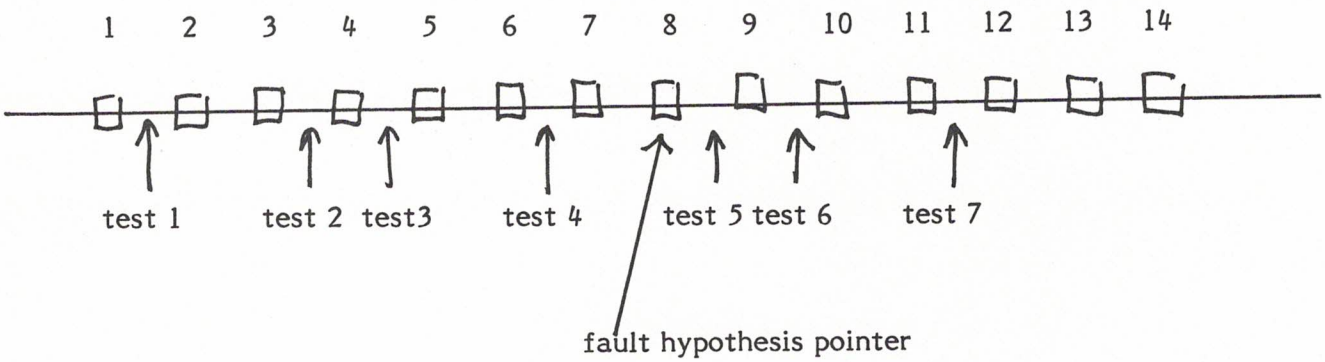
If certain inclusions are not verified after a checking by a rule system, the related fault hypothesis may be removed.

If a fault hypothesis go through all these steps, it starts a refinement process and/or a localisation process thanks to tests. This is achieved by a structured rule-based system which determines the tests to suggest and which are associated to chains. All these tests are suggested to the user in cascade without waiting for the results as far as it is possible. As soon as the user has the result of a test, he entred it inside the system and the system takes account of the results and all the concerned fault hypotheses may take advantage of it.

2) DICHOTOMY PRINCIPLE

The logic for test determination, given an hypothesis, can follow the principle of dichotomy. That means to follow a strategy which minimizes the number of tests to suggest or any other more complex strategy : for example, minimizing the cost over a mastered deepth.

Graphical explaination :



Let us assume that the fault report had pointed to the element 8, that means that one of the 8 first elements of the chain is faulty. Which test should we suggest ? We are going to determine it by economical considerations.

If we suggest the test 4 and if the result of the test is positive, that will mean that all the elements from 1 to 6 are running correctly, which is equivalent to conclude that the faulty element is the element 7. This is very brutal and little likely. If the result is negative which is very likely, we are not going to learn much : namely that one of the elements from 1 to 6 is faulty.

However, if we suggest the test 2 or 3, whatever the answer will be, the chances are that we shall learn something interesting. That is dichotomy.

3) REMARKS

If the diagnosis runs correctly, a certain number of faults will be determined. Therefore, it will be possible to repear these faults, and the system will have to be kept informed. In case of reparation of a fault, the system removes the corresponding fault hypothesis. This will be managed by an independent rule-based system.

This dichotomy rule or a cost analyse is not the only control we can have on the test to suggest. Certain materials are often known sensitive because they are at the crossing of information path (for example : computers...). By experience, these elements are to be tested at first if it is simple. Therefore, it is important to be able to modify and build the reasoning which deals with this function. It is the reason for the rule-based choice.

VI) THE DIFFERENT TYPES OF TASKS INVOLVED IN THE DIAGNOSIS

1) INTRODUCTION

This description of a diagnosis process shows a certain number of functions that the system should be able to achieve. Because of the difficult task of extracting and structuring the knowledge and because of the fast evolution of the hardware, the software and the associated know-how, we must think of an easy programmation and reprogrammation. The first consequence is the generalisation throughout the system of the use of the rule paradigm. We have to develop software concepts adapted to the diagnosis purpose.

2) THE F E E (ELEMENTARY FUNCTIONAL ENTITY)

We have one fundamental concept which is the list (or chain), (or type) of functional dependence (order).

The other fundamental concept is the concept of object of structural proximity (in a formal sense) (no order).

The link between these two concepts is the notion of FEE which is the elementary brick from which is built the whole knowledge of the system.

The duality between the chains and the objects is the heart of the diagnosis process.

A FEE is written with lower case letters ex. : e_i , f_i

The relationship of a FEE with the whole knowledge of the system can be written down as :

$$\left[e \right] \begin{matrix} j_1|B_1 & j_2|B_2 & \cdots & j_n|B_n \\ i_1|C_1 & i_2|C_2 & \cdots & i_p|C_p \end{matrix}$$

j_k is the order or the numbering of the FEE inside the object B.

i_k is the order or the numbering of the FEE inside the chain C.

REMARK :

The physical proximity
the system sub-system proximity
the functional proximity

generate chains of dependence, where the functionality used in the functional proximity is related to the functional reality of diagnosis world, whereas the functionality of the functional dependence of the chains is related to data-processing functionality.

LIST OF DIFFERENT KNOW-HOW

1 - **Parsing selection :** different FEE representing the different functionalities pointed by a fault report are generated.

FR $\longrightarrow$ $[e_1 \dots e_n]$

2 - **Logical addition :** by comparing with the place of the other FEE in the chains, we decide to create a new FEE or to include the description of what is pointed by a fault report inside an existing FEE (redundancy).

$(e, [e_1, \dots, e_n] \longrightarrow [e_1, \dots, e_{n'}]$

3 - **Deep inferencing** (use of model) : by examining the typed link in the object formalism, the objects involved in a fault of type e sent back fault report patterns that we would wait for in the case of reality of the hypothesis e.

e $\longrightarrow$ $[f_1, \dots, f_n]$

4 - **Formal unification :** by comparing with the messages FR accumulated in the horizon H, it is determined wether f is instanciable in one of these messages. Example :

f = ((fault of type 1 or 2) and (object of type 37))

FR = fault of type 1 in an object of type 37

If it is not true, the know-how 5 will be involved.

$$(f , [FR_1 , ..., FR_n]) \longrightarrow \text{YES or NO}$$

5 - **Logical ~~removal~~ :** by examining the current hypothesis set, the fault e as an hypothesis is removed.

$$(e , [e_1 , ..., e_n]) \longrightarrow [e_1 , ..., e_{n'}]$$

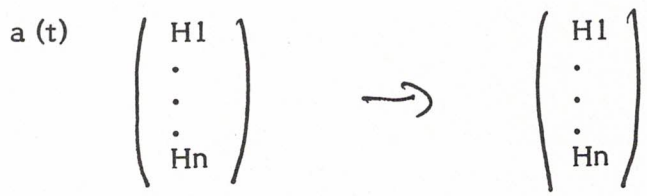
6 - **Tests phase :** a process is created which try to diagnose the fault by refining the hypothesis e through a sequence of tests.

$$[e_1 , ..., e_n] \begin{pmatrix} H = e'_1 \\ \cdot \\ \cdot \\ \cdot \\ H = e'_p \end{pmatrix} \longrightarrow [e_1 , ..., e_{n+1}] \begin{pmatrix} H = e'_1 \\ \cdot \\ \cdot \\ \cdot \\ H = e'_{p+1} = e_n \end{pmatrix}$$

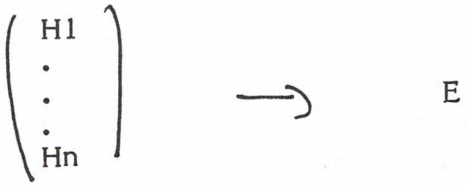
7 - **Tests selection :** by examining the chains involved in each fault hypothesis, one generate a cunning test.

$$H = e \longrightarrow t$$

8 - **Integration of a test answer :** the system tries to integrate the test answer by making every hypothesis take advantage of the answer whatever is the hypothesis which has generated it.

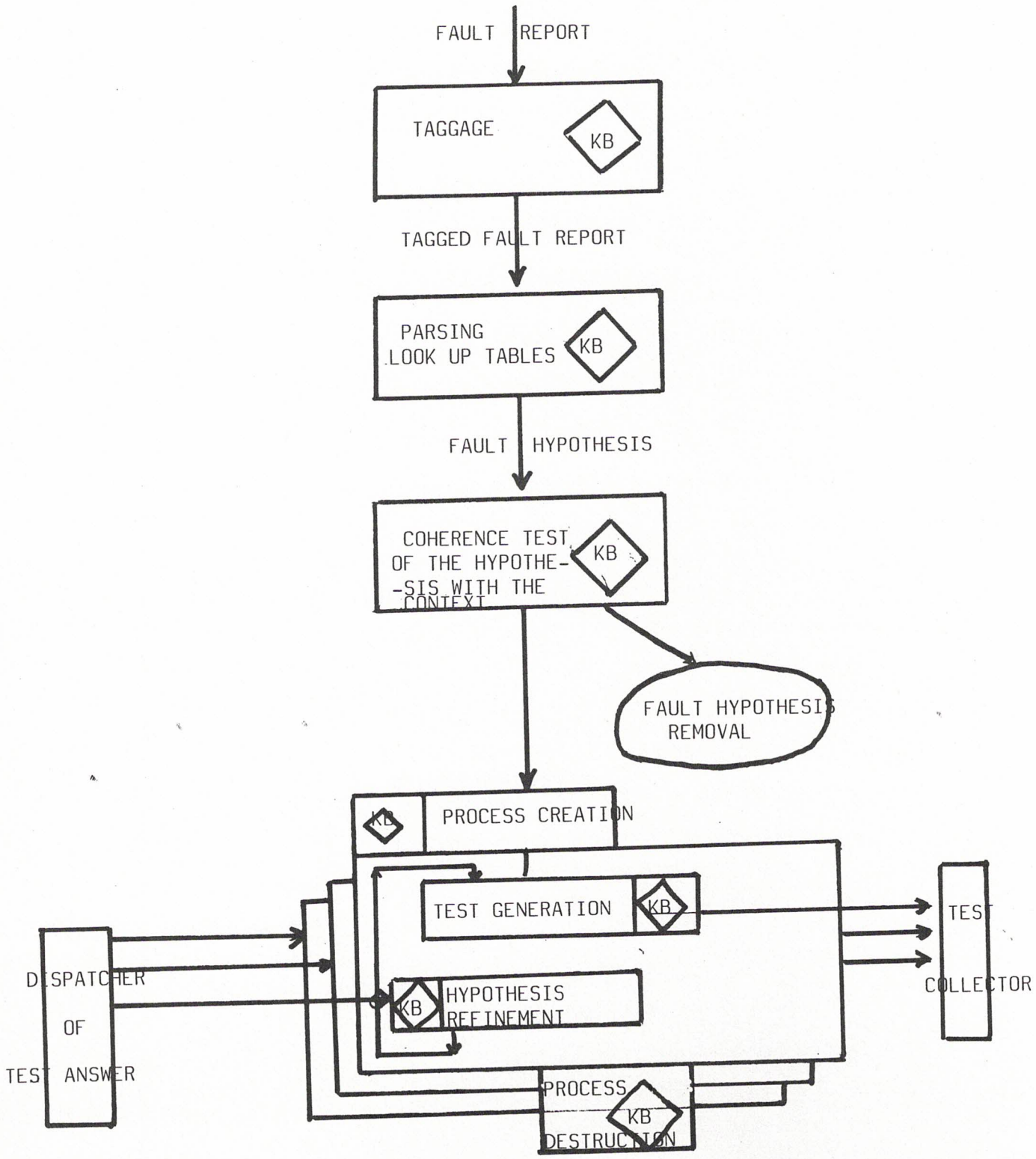


9 - **Regular output of the diagnosis :** Regularly, a state of the current hypotheses is output.



10 - **External intervention :** when an updatage, a modification or a reparation is achieved inside the diagnosed network, the system has to be informed.

FUNCTIONAL MAP



III - THE FOLLOWING STEP IN THE ANALYSE

The following step is to overstep the separation which exists between the logical level and the functional level.

In this phase, we have to solve the following problem : what are the kinds of paradigm and what should be the different types of software concepts that ought to be used in order to facilitate the writing of the 10 functions explained in the former paragraph.

TECHNICAL ANNEX 1

COMPUTING OF THE SIZE OF THE ARBORESCENCE IN THE CASE OF HYPOTHETICAL WORLDS LOGIC

P is the average number of simultaneous faults.

Every fault report is a constraint on the possible faults. Generally, several sets of simultaneous faults can explain this anomaly. Let's assume G, the average number of such sets. Let's assume L, the average life duration of an hypothesis (average number of fault reports to make disappear an average newly born hypothesis).

By a very simple reasoning, we deduce that the average size of the arborescence can be evaluated as :

$$N = P\,G^{L}$$

for $P = 3, G = 3, L = 10$ which are sensible numbers

$N = 177\,147$

Then, the system should be able to deal with 200 000 parallel worlds.

EXPLANATION OF THE COEFFICIENT G

REPORT 1 either the circuit C1, either the circuit C2, or both

3 parallel worlds

REPORT 2 either C1, either C2, or C3

where C1 includes C1 and

C2 includes C2

3 new parallel worlds

We must maintain all these parallel worlds because the validation or the invalidation will come from the accumulation of certitude.

LAST REMARK ON THE PARALLEL WORLDS

In fact, each fault report is a constraint and creating the corresponding parallel worlds is propagating this constraint inside the system.

The forgetting of parallel worlds is the volontary forgetting of a certain number of implications of the constraint between the different possibilities for the fault (logical relation).

Nevertheless, this implication will be able to be taken into account in a controled manner by another part of the system : the parts involved in the management of the hypothesis life.