LABORATOIRE d'INFORMATIQUE THEORIQUE
& APPLICATIONS DE MARSEILLE
L.I.T.A.M.

Faculté des Sciences Economiques
Université d'Aix-Marseille II

ISSN
0291-5413

INFORMATIQUE FONDAMENTALE & APPLICATIONS

BULLETIN N° 33

SOMMAIRE

Décembre 1992

Adresse postale:
Faculté des Sciences Economiques
LITAM
14 rue Puvis de Chavannes 13001 Marseille
Tel. 91 13 96 29 - (20, 21)

EDITORIAL

E. Bianco

Informatique et évolution de société.

Notre société semble évoluer à la manière des papillons. D'abord la chrysalide se terre longtemps dans le fumier, l'humus ou le bois pourri, puis la chenille naît, laborieuse, qui va engloutir des quantités de verdure. Elle en profite pour dégrader sérieusement l'environnement . Tout-à-fait comme la période industrielle, mais en plus édulcoré. Et enfin jaillit le papillon étincelant de mille couleurs vivantes, aboutissement éclatant d'un long grouillement sordide et rampant. Mais il est guetté par une mort rapide à l'issue de la pariade.

Est-ce à dire que notre époque, en train de sortir péniblement de la gangue industrielle, pour avoir l'air de prendre son essor, ressemble au papillon ? On pourrait parfaitement bien le croire si l'on s'en tenait à une bonne partie des images dont nous abreuvent les "médias". Ces yachts de rêve que leurs propriétaires en général inconnus du public utilisent quelques jours par ans occupés qu'ils sont à entasser leurs milliards, ces châteaux issus d'un autre âge mais aménagés comme dans les contes de fées, aux frais du contribuable, astuce suprême, car on les fait classer. Et toutes autres sortes de choses et d'activités brillantes rendues possibles par l'accumulation fantastique de richesses subtilement détournées par des règles et des lois qu'on ne trouve dans aucun code officiel.

Et puis l'envers du décor. Les guerres sordides autour du plus ou moins un dollar par baril de pétrole. Ces soldats qu'on enterre tout vifs à coup de chars-bulldozers pour qu'un émir puisse faire pousser une oasis artificielle au milieu du désert, à la gloire du saint nom d'Allah, bien entendu.

Et puis ces régimes qui se sont complètement trompés de support. On s'est basé sur une fausse idée du Peuple. On a pendu un Autocrate, et des millions de petits Tsars se sont partagés le pouvoir, mais avec leurs pauvres moyens basés sur leur pauvre expérience. Il n'avaient vu et subi que brutalité, rapine, exploitation, mépris. Pour eux la liberté qu'on leur faisait miroiter c'était avant tout essayer de vivre comme ceux qu'ils abattaient. Ils ont appliqué chacun dans leur minuscule domaine le pouvoir tel qu'ils le concevaient: brutalité, rapine, exploitation, mépris, mais ces attitudes exacerbées, minutieusement perfectionnées, et portées à l'état de dogme par un manque de pratique du pouvoir et une absence de culture. Un éventuel moyen devenu un but par enfermement de la pensée dans une sorte de cycle infernal: plus ça va mal, plus on durcit le pouvoir qui aggrave la situation.

Là encore signe de la richesse des temps, des personnages savent utiliser les circonstances pour décider du sort des masses en utilisant des arguments extrêmement efficaces sinon de goût douteux. La puissance des moyens disponibles sert de levier à des individus sans scrupules pour agir sur des mentalités fragilisées par une incompétence douloureusement ressentie face à la complexité croissante de la société, et aussi par nombre de facteurs tels que des "conditions économiques" désastreuses ou présentées comme telles, car elles ne le sont jamais pour tout le monde.

Cette sorte de "war game" à l'échelle de la planète, voila donc le grand essor du papillon. Une poignée d'individus manipulant des gigatonnes, comme des gamins une poignée de pétard de quatorze juillet, jonglent avec des milliards d'individus éberlués, subjugués, fascinés, leurs pauvres cervelles traversées par le brasillement imaginaire de ces luttes de titans, tandisque que leur tripaille perturbée par la trouille les laisse à la merci de leurs polices respectives. L'un des deux papillons gigantesques s'est effondré avant d'avoir pu achever sa pariade, ses œufs empoisonnés vont-ils se déssécher avant d'avoir pu ensemencer le monde d'un rayonnement meurtrier?

Le malheureux petit tas de pourriture, reste de l'insecte étincelant, augure-t-il de l'avenir de notre civilisation ? Je m'en vais interroger mon ordinateur, et peut-être pourrais-je vous apporter une réponse.

STATUT DES VARIABLES

E. Bianco

Subject Classification Informatics: D3, D4, E2, E4.

Résumé.

L'ère des heuristiques a mis en évidence une richesse d'expression qui ne peut se comparer qu'à l'art du romancier. Mais autant dans ce dernier domaine débrider largement l'imagination mène au dépaysement, à l'exotisme, bref à l'enrichissement de l'imaginaire, autant en informatique, passer un certain point revient un peu à tourner en rond, les langages artificiels n'ont pas le pouvoir évocateur des langages naturels. Et bien des théories pourraient paraître spécieuses si, précisément elles n'étaient pas, de temps en temps reliées au fondamental par ce qu'elles ont en commun avec les autres théories. Le travail ici présenté appartient à ce genre de tentatives ardues, ingrates, mais, me semble-t-il indispensables pour éviter de partir dans une sorte de spirale qui consiste en glissant un peu chaque fois à perpétuellement retrouver les mêmes choses sous des noms différents.

STATUT DES VARIABLES

NOTION DE STATUT.

Les débuts du calcul automatique se caractérisaient par le traitement d'informations, essentiellement des nombres, dont la représentation en mémoire se faisait à géométrie constante. Il s'entend par là que l'élément de mémoire était, et reste encore d'ailleurs, constitué par des sortes de cases à dimension déterminée à l'avance. La théorie de la machine de Nolin montre bien quelle est en fait l'idée ainsi matérialisée. Depuis, les choses ont en apparence fortement changé. La représentation d'alphabets de plus en plus complets, qui a permis le traitement automatique de véritables textes, a largement poussé à élargir la question. On a conçu des machines très spécialisées, qui utilisaient des marques de mots capables de traiter d'un coup des suites de cases de longueur à peu près quelconques. Puis, la puissance des machines s'accroissant, on les a dotées de mémoires à segmentation multiple, on pouvait travailler sur l'octet, le mot, le double mot, etc. selon la catégorie d'instructions employée. Tous ces artefacts, plus ou moins perfectionnés, plus ou moins abandonnés selon l'évolution de la technologies des composants électroniques ne recouvraient en aucune manière la généralité du problème posé.

Comme tous les problèmes de cette sorte, il devenait nécessaire de le poser sur le plan théorique, à partir d'une expérience suffisante, afin de le maîtriser correctement. Mais la concurrence aidant, on n'avait pas de temps à perdre ...

Ce que je voudrais exposer ici c'est l'approche d'une prise en compte la plus complète possible du problème fondamental de la représentation de l'objet dans une mémoire électronique. Je me place dans une optique que je qualifierai de classique, dans la mesure où la notion d'élément de mémoire support de l'information est fondée sur le "bit".

Je fais également l'hypothèse qu'il existe une préstructuration de la mémoire en cases, désignées chacune par un nom différent des autres, et qui ne dépendent les unes des autres que par une loi reliant leurs noms entre eux. Ces cases seraient-elles identiques ou différentes les unes des autres du point de vue de leur contenu possible. Je vais toutefois en un premier temps les supposer toutes identiques selon ce point de vue. Mais ce n'est là qu'une simple hypothèse de travail nullement fondamentale.

HYPOTHESE 1.

Il existe une préstructuration de la mémoire en cases, désignées chacune par un nom différent des autres, et qui ne dépendent les unes des autres que par une loi reliant leurs noms

entre eux. Ces cases seraient-elles identiques ou différentes les unes des autres du point de vue de leur contenu possible.

HYPOTHESE 2.

Ces cases seront toutes identiques entre elles, bien que ce ne soit pas fondamental, à l'exception des cases spécialisées et uniques dans leur genre. On pourra d'ailleurs, aussi bien supposer que la totalité des cases sera subdivisée en plusieurs grandes catégories de cases identiques entre elles, mais différentes d'une catégorie à l'autre.

Il est indispensable de préciser que je me place dans un contexte de machine et de langage machine. Actes et repères sont évidemment définis directement par rapport aux organes de la machine, ce qui n'est jamais le cas, ou ne devrait jamais l'être, quand il s'agit de langages symboliques.

REPRESENTATION.

J'imagine un ensemble quelconque d'éléments sur lequel on décide d'appliquer du calcul automatique. La première opération qui s'impose de toute évidence, est une opération de représentation des éléments en terme de contenus de mémoire. Dans tous les cas, à deux éléments différents, il est clair qu'on va faire correspondre deux entiers différents qu'on puisse noter chacun dans une ou plusieurs cases. Mais là où les choses se compliquent c'est que, précisément la construction de l'entier image de l'élément, doit être choisie pour permettre l'application plus ou moins aisée de fonctions calculables censées représenter les traitements envisagés.

Il est en général commode, selon la complexité de structure de l'élément, d'utiliser en fait plusieurs entiers séparés, analysables séparément, ou encore des groupes de paquets d'entiers. Et il faudra bien se souvenir du découpage de ces jeux de nombres pour pouvoir les prendre en compte séparément afin de les soumettre au calcul. C'est ici qu'apparaît la notion de statut. En effet, pour se souvenir de la structure de la représentation on va définir un type nouveau de variable. Cette variable de statut prend des valeurs qui permettent sans ambiguïté de retrouver la structure de la représentation. A titre d'exemple, Je représente une variable de langage symbolique lors d'une compilation. La variable est elle-même désignée dans la phrase symbolique par un identificateur, suite de lettres et de chiffres qui commence par une lettre. J'ajoute à cette indication un numéro de correspondance dans l'espace de la phrase compilée, puis j'ajoute un type pour rappeler la nature de la variable, et enfin un volume qui sert à distinguer variables simples et tableaux. Ainsi pour un objet déclaré:

Réèl XM3GRA,

Je construis l'image:

Valim : XM3GRA, 21, 'réèl', 1

On constate que l'identificateur exige 6 cases, et chacune des autres valeurs 1 case pour être représentée. De plus, les identificateurs peuvent être à nombre de lettres quelconque. Je définis donc une variable dite de "statut", dont la valeur serait en l'occurence:

S1 = 6, 1, 1, 1

Si j'ai besoin, maintenant de représenter à son tour la valeur de cette nouvelle variable, il m'est possible de choisir le code:

code S1 : 5, 21, 6, 1, 1, 1

Le 5 signifie que cette variable se code sur 5 cases, le 21 est une référence à la variable à laquelle ce statut est rattaché. On peut faire alors plusieurs remarques.

Il saute aux yeux que si chaque identificateur est susceptible de comporter un nombre quelconque de lettres, alors à chaque variable sera attaché un statut particulier. Mais on peut concevoir des ensembles d'éléments qui auraient un statut unique en commun, bien entendu.

Le couple constitué par le code de la variable et celui de son statut va simplifier la description de certains traitements d'ensemble applicables à la variable. Ainsi, le transfert du code entier n'exige pas la désignation de chacun des éléments. Le rattachement au code de son statut suffit.

Je rappelle simplement que nous travaillons dans un contexte machine ce qui implique notamment que les organes de la machine sont directement visibles au travers de la programmation. Diverses questions se soulèvent alors quant à la désignation des paquets de codes censés représenter l'information, soit du point de vue de leur structure, soit du point de vue de leur accession par les commandes d'actes. L'organisation matérielle d'un circuit de machine exige certaines limitations, aussi je poserai le problème en termes de problème pratique. On peut faire une théorie générale de "l'empaquetage des représentations", mais un tel travail serait plutôt intéressant pour des langages symboliques, encore que même à ce niveau des limitations s'imposent forcément. Il en est de même, par exemple pour les insertions récursives (récurrentes) de procédures, en principe on n'est pas limité, c'est d'ailleurs un problème du fini-illimité, en réalité on sature très rapidement les plus grandes mémoires par étalement des configurations, lors de chacune des insertions successives.

Une autre difficulté surgit à l'introduction de la notion de statut, en effet à partir du moment où l'on connaît la valeur du statut attachée à une représentation de variable, on atteint facilement à chacun des éléments de cette représentation. Mais qu'en est-il du code du statut lui-même ? La variable est a priori quelconque, aussi le statut peut-il être très compliqué.

L'interprétation de son code risque alors de devenir parfaitement ambiguë. Il devient nécessaire de choisir une structure de statut standard pour le code du statut. On retombe bien entendu sur une question de langage formel, et les solutions sont bien connues, il suffit d'alterner des objets d'interprétation différente: des nombres pour donner des dimensions, et des marques de séparation. Mais il est bon de forcer sur des conventions implicites afin d'éviter au dérouleur qui tombe sur un code de statut d'avoir à réaliser la véritable compilation d'un mot de Chomski.

STRUCTURES DE FILES.

J'appellerai "file" une suite organisée de groupes de représentations d'objets. On est amenés à placer en mémoire des représentations de suites d'objets hétéroclites. J'appelle "structure" l'information qui permet de retrouver les types de chacun des objets, dans la file qui contient une suite de telles représentations. Par définition d'abord et construction ensuite, c'est le type de l'objet qui permet d'en assurer le bon traitement. En effet toutes les représentations se font au moyen des entiers, mais les traitements diffèreront selon la nature de l'objet représenté. On n'appliquera pas les mêmes algorithmes sur la représentation d'un entier et sur celle d'un réèl, et encore moins sur celle d'un objet alphabétique.

Attachés à une même file, ce sont deux cas sémantiques distincts que nous allons envisager. En effet, une même file est normalement sujette à des traitements provenant de deux origines différentes.

A) Quand on compile un programme, quel que soit le langage utilisé, on doit trouver dans le texte des descriptifs, explicites ou implicites, déclaratifs ou non, qui permettent de retrouver la nature des variables traitées. Traduire ceci en termes de phrases compilées montre que le dérouleur du programme va trouver des indications concernant précisément ces variables, à la fois dans le code du programme et dans le code qui correspond à la représentation des données.

B) Quand un système par exemple travaille sur un stock de données à des fins d'échanges, il ne dispose pas de l'information contenue dans le code d'un quelconque programme qui s'appliquerait sur elles. Pour que l'information soit exploitable, il devient nécessaire que le code contienne davantage d'information que dans le cas précédent.

Exemple 1.

Imaginons une machine à mots, chaque mot étant considéré uniquement comme un entier. Quelle que soit la nature du programme qui lit un tel mot, il n'y aura aucune ambiguïté à en interpréter le contenu. En effet on sait qu'on aura toujours affaire à

une case de même nature, et que son contenu est d'un seul et même type. Si le programme qui traite et la donnée traitée sont indépendants l'un de l'autre, alors il faut au programme fournir en plus l'adresse d'accès et le volume en nombre de mots.
Exemple 2.

Prenons alors un exemple en PFS. Imaginons un programme qui travaille sur une "file dynamique" déclarée avec 2 statuts. Tant que le programme n'a pas calculé, il est impossible de connaître la structure de la file, car la suite des éléments qui vont y être construits, le seront selon l'un ou l'autre des statuts au gré des évènements. Chaque élément doit donc être répéré par son numéro (qui permet une localisation équivalente à l'adresse), et aussi par le statut qui lui a été attribué (volume et géométrie de la représentation). Le couple formé par le programme et sa configuration n'exige pas de noter dans la configuration, d'autres informations attachées à l'élément. En PFS, il existe plusieurs sortes de files, mais lorsqu'une instruction s'adresse à une file, ou un élément de file, elle contient sous la forme d'une référence, le moyen de savoir à laquelle des files on s'adresse. La raison en est qu'une référence est toujours attachée à une file et une seule. Or, cette information est naturellement portée dans le code du programme.

Un système qui lirait une telle file pourrait effectivement en retrouver la structure, mais à la condition stricte de savoir de quel type de file il s'agit. On constate alors qu'il n'est pas nécessaire de noter son type dans la configuration de la file, tant que c'est le programme dans le contexte duquel elle a été déclarée qui travaille dessus. A partir du moment où un second programme, construit à part du premier, doit explorer une telle file alors qu'elle n'a pas été déclarée dans son propre contexte, il devient nécessaire que cet autre programme puisse en trouver le type quelque part .

Pour établir ce raisonnement, je suppose de toute évidence que tous les programmes que je cite sont écrits dans un même langage, et que ce langage est conçu spécialement pour ce genre d'opérations, ce qui, entre autres, est le cas du PFS.

STATUTS EXPLICITES ET IMPLICITES.

Je me place dans l'hypothèse selon laquelle toute la charpente du logiciel de l'ordinateur est construite en PFS. Imaginons une configuration qui aurait été traitée par un programme, et supposons que le système lui-même, ait à intervenir sur cette configuration.

Quand le programme considéré fait partie des compilateurs intégrés au système, au lieu d'être un simple programme d'utilisateur, bien que les propriétés demeurent les mêmes, les

choses se présentent un peu différemment. Dans un système ouvert, de nouveaux sous-systèmes peuvent être ajoutés à l'aide de l'opération spéciale d'intégration. Les compilateurs en sont donc construits d'abord comme de simples programmes utilisateurs et ce n'est que lorsqu'ils sont considérés comme parfaitement au point qu'on peut envisager de les intégrer. Ils ne seront d'ailleurs généralement intégrés que comme composants d'un sous-système complet.

Je reprends les définitions du PFS, la "file" se définit comme une suite de "lignes" et chaque ligne, comme une suite "d'éléments", en toute généralité il est nécessaire de disposer de ce que j'appellerai le "type" de l'élément pour pouvoir effectivement le traiter. En effet, non seulement il est indispensable de savoir sur combien de cases la valeur de l'élément est étalée, mais en plus il faut pouvoir déterminer comment interpréter chaque contenu de cases. Le simple entier binaire, l'entier représenté sur une base décimale, hexadécimale ou autre en utilisant des caractères, des éléments purement alphanumériques sont des exemples parmi mille autres de ce qu'on peut rencontrer. Je définirai donc le type d'un élément comme un moyen de le rattacher à un ou à un ensemble d'algorithmes. Par exemple tout nombre représenté en "flottant" ou "virgulé, cadré" sera rattaché à tous les algorithmes susceptibles d'appliquer l'arithmétique dessus. On s'aperçoit alors que dire simplement qu'un élément est un nombre flottant est une indication insuffisante, car un tel nombre peut être représenté d'autant de manières différentes qu'on veut. Seule la référence à un algorithme déterminé peut lever l'ambiguïté. Là encore on va rencontrer deux cas, ou l'algorithme est intrinsèquement défini dans le langage, et il est implicite, ou il est construit par utilisation du langage, et il est alors explicite. Rendre la relation à ce dernier implicite, pour une utilisation indirecte par un programme étranger, oblige à prévoir des moyens linguistiques adéquats.

Pour l'instant, je pars d'un simple exemple de file dynamique au sens du PFS, qui se déclare selon la structure:

x, y dynamique (liste d'index) :
(liste de statuts) : (liste de références) ;

x est un entier qui représente le volume total accordé à la file, et y en est le nombre de lignes.
Une valeur de statut se décompose ainsi:

(Nb d'éléments) (type E1) (dim E1) (val E1) ,
(type E2) (dim E2) (val E2) , ... ,
(type En) (dim En) (val En)

Le type de l'élément Ei est l'indication de rattachement à un algorithme d'interprétation. Le code val Ei est la dimension de la

représentation de l'élément en nombre de cases nécessaires. Quant à la dim Ei, il s'agit d'un entier différent de 0 si l'élément est répétitif, dont il est alors la valeur de répétition, et il est à 0 si l'élément est non répétitif.

Le code de file, lui, pour être complet, doit comporter un minimum d'informations, par exemple:

(type de file)
(accès à la "liste de:") (accès à la file de valeurs)
(accès à la fin de la file de valeurs)
(Nb de statuts) (accès aux statuts) (valeurs des statuts)
{liste de: [(accès à l'élément) (statut de l'élément) (état)] }
{file de valeurs: [(accès case libre) (état de la file) (valeurs des éléments)] }

Les accès sont des entiers ayant le sens de numéro d'ordre, pour les objets à atteindre, par rapport à une origine arbitraire définie.

La "liste de:" contient "y" éléments, chacun d'entre eux est la description d'une ligne à partir du moment où cette ligne est définie, c'est-à-dire quand une valeur lui est affectée.

Comme on peut le constater, le volume du code est connu à partir du moment où la déclaration de la file est compilée. En effet, si j'appelle "série" la suite des objets représentée par un libellé entre-parenthèse, comme "(accès aux statuts)" , par exemple, les quatre premières séries occupent une place déterminée dès qu'on connaît les statuts. C'est "y" , le nombre de lignes qui sert à calculer le volume total des séries de "liste de :". La dernière série, celle des valeurs de la file, a son volume défini par l'entier "x", et ceci par définition.

Remarque.

Cet entier "x" est évalué par le programmeur lui-même, une évaluation approximative pourrait en être faite par le produit du nombre de lignes et du volume de l'élément de plus gros statut. Mais ceci est une évaluation dans tous les cas approximative qui ne tient pas compte des valeurs calculées des variables de répétition, et qui, dans le cas général risque d'être le plus souvent excessive. La valeur de "x" pourrait être interprétée de deux façons différentes. Car on pourrait l'envisager soit comme le volume exclusif des valeurs des éléments que contient la file, soit encore comme le volume total qui compterait également toutes les séries descriptives.

Exemple.

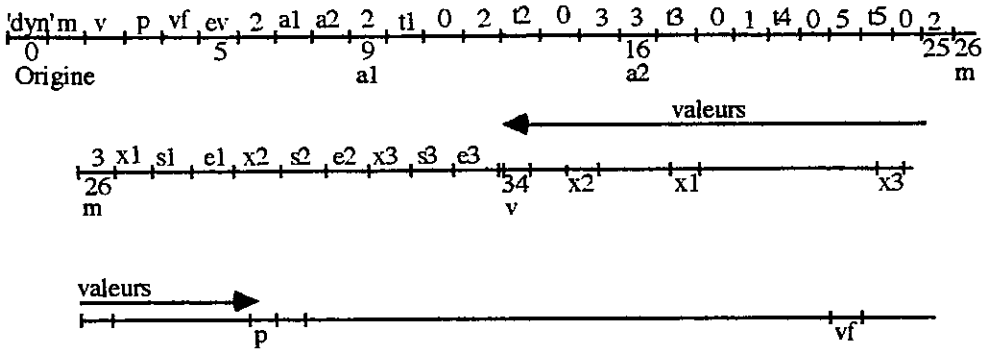
Imaginons une file dynamique ainsi déclarée:

24 , 3 dynamique h1,h2 .

1 (2,3) ,

2(1,5,2) : R1 , R2 ;

les index et les références ne serviront pas ici pour le code de la file, qui pourrait se représenter:



L'index "p" est un accès à la première case de libre dans la configuration des valeurs, tenu à jour au fur et à mesure que des valeurs de lignes sont ajoutées. La comparaison de cet index avec la valeur "vf" permet le contrôle de débordement de la file. C'est l'état "ev" dont les valeurs peuvent être: "normal", "extension", "réduction", "contraction" qui suit l'évolution de cette configuration des valeurs. En cas de débordement, c'est "extension" qui correspond à une demande de place supplémentaire au système. En fin de traitement, "réduction" marque la demande, toujours au système, d'ajuster le volume de la configuration à la place effectivement occupée. Quand à "contraction" c'est la marque d'une demande de suppression des éléments périmés pour récupérer de la place.

Dans l'ordre on rencontre: le type de la file, les accès m, v, p, vf respectivement à la descriptive des lignes, à la configuration des valeurs, à la fin de cette configuration. Ensuite on trouve le nombre de statuts, 2 en l'occurence, les accès aux codes de chacun d'entre eux, ici a1 et a2 c'est-à-dire 9 et 16 calculés par rapport à l'origine. En a1 et a2 se trouvent les descriptifs des statuts. Pour le premier d'entre eux, on a 2 éléments de dimension 2 et 3 et de type t1 et t2. En PFS les types ne sont pas déclarés mais en toute généralité je dirai que cette information existe. L'autre comporte 3 éléments de type t3, t4 et t5. En m c'est-à-dire à partir de la case 24, on trouve le code de la "liste de:" qui est en fait, le descriptif de la file d'accès aux valeurs. En tête il y a la valeur de y , ici 3, donc 3 lignes, et pour chaque ligne un triplet xi, si, ei qui donne l'accès xi à la valeur, si le numéro du statut attribué à la ligne, et ei son état qui peut être: "affecté", "non affecté", "annulé", etc. Enfin, à partir de 34, s'étalent les valeurs des lignes dans l'ordre où elles sont affectées.

Lequel n'est pas forcément l'ordre de classement qui lui, est imposé par la "liste de".

A partir de cet exemple on imagine facilement le travail de la machine universelle qui déroule un code d'opérande tel que:

$[_{R1,h2}$

le descriptif attaché à R1 doit contenir le numéro d'ordre 'li' de la ligne à laquelle elle a été liée par calcul, ainsi que l'accès 'af' au code de la file dont elle fait partie. Si je désigne par Aj toute adresse, et $[_{Aj}$ la valeur contenue à cette adresse, le calcul se décrit alors ainsi:

$A1 = af + 1$

$A2 = [_{A1} + af$ (m+af)

en A2 se trouve le nombre de lignes,

si $li > [_{A2}$ alors erreur.

$A3 = A2 + 1 + (li - 1) \times 3$

A3 est l'adresse du triplet attaché à la ligne.

Les modalités du calcul vont dépendre en partie de la valeur de l'état lue en A3 + 2: "affectation", "remplacement", "annulation".

$A4 = [_{A3} + af$

L'adresse A4 permet d'accéder directement à la valeur de la ligne.

$A5 = [_{A3+1}$ (si)

$A6 = af + 6 + A5$ (af+si)

L'adresse A6 désigne l'accès à l'adresse relative du code du statut si cependant la condition $A5 \leq [_{af+6}$ est vérifiée.

$A7 = [_{A6} + af$

C'est par l'adresse A7 qu'on accède au code du statut qui permet d'analyser la ligne d'adresse A4.

Remarque du point de vue du câblage.

On peut parfaitement imaginer que les informations utilisées dans ce calcul soient ainsi réparties, que l'ensemble des valeurs d'adresses puisse être obtenu en un minimum de tops d'horloge. Il suffirait par exemple, que dans le code de l'opérande, où se trouvent normalement les accès à la file, à la ligne, et au statut, on puisse accéder simultanément à trois tranches de mémoire spécialisées. Certaines de ces cases pourraient agir comme des registres, c'est le cas de la ligne et du statut. Il serait bon, alors, que dans le code de l'instruction, selon qu'elle est à un, deux ou trois opérandes, une indication permette le chargement immédiat des indicateurs correspondants dans ces registres spécialisés.

Je me suis placé dans un cas particulier de gestion des éléments périmés. En fait ce problème est un des plus complexes qui existent en informatique c'est une des raisons pour lesquelles un grand nombre de solutions peuvent être mises en avant, mais aucune n'est satisfaisante en toute généralité. Deux moyens de base

devraient permettre de prendre en compte à peu près tous les cas dans la mesure où le système intervient en temps opportun. D'abord il faut pouvoir noter l'oblitération d'une valeur. C'est ce qu'on peut faire en utilisant l'état de l'élément en file des triplets attachés à la ligne. Ensuite on dispose d'un prédicat de débordement qui indique au système quand la file est pleine. On peut, dans le même sens prévoir un indicateur de deuxième modification du même élément, car là, on n'a plus rien pour noter le nouvel élément périmé.

CHAMP FICTIF ET CHAMP REEL.

Quand on construit un programme, on fournit au compilateur traducteur une suite de lettres qui sont transformées en une suites de codes interprétables par une machine universelle. Le produit obtenu constitue un champ fictif. En effet l'algorithme est conçu pour opérer une transformation déterminée applicable à une certaine configuration, mais pour l'instant, seule l'image de l'algorithme existe. Pour passer au champ réel, il faut implanter cette image d'algorithme, et également lui fournir, à savoir implanter, une configuration compatible c'est-à-dire essentiellement fournir à la machine universelle l'accès au code à interpréter et l'accès à la configuration à traiter.

Du point de vue du système, l'organe qui se déroule en permanence pour le contrôle des opérations, un programme utilisateur prend successivement les deux aspects. Il sera d'abord champ fictif lorsqu'il est construit, et ensuite il devient champ réel quand à la suite d'une demande, on en prépare le déroulement. Il existe ainsi deux sortes de champs réels toujours du même point de vue du système. Lorqu'il travaille avec un compilateur intégré, le système a un accès direct à son champ réel. Quand on déroule un programme utilisateur, le champ réel est indirect car code et configuration sont inclus obligatoirement dans la configuration du compilateur dérouleur. On peut alors se poser la question de savoir dans quelles conditions dérouler un programme directement avec le processeur. Il s'agit là d'un problème de politique de gestion de l'ordinateur. C'est possible à partir du moment où le programme est "intégré" dans le système, opération qui n'est à tenter que si l'on est absolument sûr de ses propriétés.

Le problème important surgit quand le système est amené à agir directement sur une configuration de programme utilisateur. Or, si l'on considère le phénomène tel qu'il est pris en compte en PFS, on s'aperçoit qu'on va avoir à traiter une information qui se trouve étalée dans des lignes de file support. Le système peut atteindre en bloc à de telles lignes car il possède des références pour y accéder. Puis il possède un autre type de références pour pouvoir atteindre à l'intérieur de la ligne. Mais les statuts

indispensables pour aller fouiller le détail des objets, devront être pris - non pas dans le descriptif de la ligne de la file support comme c'est normal dans le cas direct - mais dans le contenu de la ligne elle-même comme indiqué plus haut.

REMARQUE.

En PFS il est prévu que le statut déclaré en contexte, soit calculable d'une certaine manière. C'est ainsi que la notion de variable de répétition permet, par calcul de sa valeur, de définir non pas une structure, mais une variété de structure. Ainsi, si je prends le statut:

S = 3, n(4)

selon que n = 0, 1, 2, 3, ...
donne les statuts:

s0 = 3 ; s1 = 3, 4 ; s2 = 3, 4, 4 ; s3 = 3, 4, 4, 4 ; etc.

PROPRIETE.

La notion de système implique une notion de statut calculable. Entendons par là qu'il doit être capable de travailler avec des valeurs de statuts quelconques, plus précisément qui n'ont pas été déclarées dans son contexte.

C'est-à-dire en particulier, qu'un système qui travaille sur une ligne de sa file support, à partir d'une référence qui repère cette ligne calcule une référence-nom, des instructions permettent de récupérer la valeur du statut qui n'est pas connue implicitement, mais qui est récupérable dans le contenu de la ligne, si celle-ci est codée de façon standard. C'est le rôle du programme de s'adapter à la structure du statut, a priori inconnue.

Un système peut avoir à extraire par exemple un contenu de ligne en traitant différemment la forme des éléments selon leur type. Je ne vais pas tout de suite prendre en compte le traitement des types qui est un chapitre à lui tout seul, mais seulement montrer les conditions pour qu'un système puisse le faire. Le traitement complet des types fait partie de ces genres de calculs qui se situent à la limite du fini-borné et du fini-illimité. En effet, en mémoire centrale, quel que soit le type attribué à un élément, ce sera toujours un entier binaire, par exemple, qu'on rencontrera dans la case, en vertu des hypothèses 1 et 2. Le type intervient obligatoirement quand on passe de la mémoire à un dispositif d'échange. Dans un sens, de l'extérieur vers l'intérieur, ce sont des mécanismes spécialisés qui fournissent la valeur binaire, et il peut être fait appel à un pré-traitement pour obtenir la représentation définitive, sur demande de l'utilisateur. Dans l'autre sens, il faut bien trouver une indication pour savoir sur quel instrument diriger l'information, et sous quelle forme. Ainsi une même valeur binaire,

contenu de case, peut être, à la demande, condidérée comme un ou plusieurs codes alphanumériques, ou encore comme une valeur numérique à représenter dans une base déterminée, ou encore comme un picsel etc. Au sein d'un sous-système, c'est dans le rôle du langage d'échanges de permettre ce genre de communication. L'information descriptive est transportée par la machine universelle, via le VIIF.

La démonstration de la propriété est simple, tenant compte de toutes ces remarques. En effet, les lignes d'une file support sont destinées à contenir en particulier, les productions des compilateurs des sous-systèmes. Par construction le contexte de ce programme qu'on appelle système ne peut pas contenir de statuts qui appartiennent à des programmes conçus ultérieurement, il est donc absolument nécessaire que le système qui aura à travailler dans ces lignes se serve de statuts calculables dont les valeurs sont récupérées dans les lignes mêmes à traiter. Ce qui est possible, comme nous venons de le voir.

CALCUL IMPLICITE, CALCUL EXPLICITE.

Pour ces parties délicates, on peut construire le système de deux façons, soit en programmant explicitement le détail de traitement des éléments soumis au calcul, soit en les programmant implicitement. J'imagine une file support, dans cette file une ligne contient le code complet d'une file construite par un compilateur de sous-système, par exemple un code de file dynamique. Il s'agit donc de pouvoir atteindre n'importe quelle ligne de la file dynamique en question, dans le but d'en imprimer tout ou partie toujours par exemple. Donc, dans le premier cas, celui du traitement explicite, après avoir calculé une référence de la file support qui permet d'atteindre à la ligne en cause, on calcule à partir de cette référence une référence-nom qui, elle, permet d'atteindre à l'intérieur de la ligne le détail de la file qui y est enregistrée. Pour chacune de ces lignes internes il faut pouvoir disposer d'une instruction qui assure la récupération des valeurs du statut qu'il faut alors dépecer. C'est ainsi qu'un aiguillage contrôlé par une variable dont la valeur est calculée à partir du nombre d'éléments que comporte la ligne, traite individuellement de chacun des éléments. Il faut noter que cette valeur de la variable de contrôle de l'aiguillage, dépend des valeurs des éventuelles variables de répétition de la ligne. Le cas de chaque élément est évidemment fonction du type qui lui est attaché, et qu'on trouve dans le code de la file. Il doit donc exister une liste de "traitements de types", tables et algorithmes, fournis au départ ou construits après, auquel doit se référer le système pour transformer le code.

Le cas du calcul implicite ne laisse au système que l'appréciation globale du code à traiter. C'est dans les commandes

de pré-échanges que vont s'opérer automatiquement les transformations du binaire intérieur en code externe, et réciproquement selon les types. Mais, de même qu'il faut une file spéciale: le cartouche pour contenir le descriptif de l'échange, il faut disposer d'une file spéciale qui sert de tampon, et qui, elle, contient l'information sous une forme prête à être échangée et qui dépend précisément du type qui lui est affecté, c'est-à-dire de son mode d'interprétation.

Study of a thesaurus creation and development system by means of Petri nets. Medical decision aid: case of Kent's repertory

J.M.KNIPPEL M.T.LASKRI

Computer classification: H.3.1, F.1.1, I.2.4

Keywords: Petri nets, modelling of complex biological systems, thesaurus, organizational systems, prototyping.

Topics: Application of nets to the design and performance evaluation of systems, case studies

SUMMARY

The frame of the selected application is the medical decision aid through the study of KENT's repertory of medical matter (1). According to a process based on the study of local, common, characteristic, seldom, personal or general symptoms, specific to one individual, KENT proposes the research of the Simillimum or medecine to be prescribed.

We will describe the creation of a man-machine communication system, adapted to the creation of a Thesaurus, based on the essential principles of the computer system theory : "autojectivité", commutation and opening (6). The creation, development and operation of the Thesaurus are formalized thanks to the Petri nets (5). This will enable us to detect inconsistencies in the knowledge base. The purpose of such process is the homoeopathic decision aid through quantitative empiric models.

I. INTRODUCTION

J.M. KNIPPEL first studied the performance of the man-machine system of dialysis, and then the representation of nephrological knowledge. At the beginning, the approach to this question was mainly numerical ; later, it was completed by an approach integrating subjective, objective, lesional or body specific signs (5).

The knowledge accumulated since the last ten years leads us to take stock of the situation about the possible representation of knowledge in the present application, i.e. the medical decision aid through the thorough study of Kent's repertory (1).

N.T. LASKRI started from the description of a man-machine communication system adapted to the creation of a thesaurus, which enables to define a structure and a content by means of natural language.
On the one hand, he presented a method of text analysis by putting it back in its general context which is the creation of a thesaurus.
On the other hand, he integrated the development of a detector-corrector system of the main groups of mistakes which may be introduced in human textual datas (2).

So, our works have lead towards a complementary approach to Kent's repertory of homoeopathic matter, taken as a thesaurus. Our study is based on a representation of medical concepts able to help Users in using the formalism of the Petri nets.
The Petri nets are used for representing the connections between concepts, and finding a way to carry out research into the Thesaurus. With the research, a detection of inconsistency in this knowledge basis may be put forward (8). This "progression" can be organized by taking into account essential concepts of open computer systems (6).

II. NOTION OF THESAURUS

It would be good to define suitable methodologies for the description and the characterization of datas handled by means of concepts representation. These principles built a regular basis of rules allowing an indexation adapted to a system for data research in a documentary context.
The indexation consists in transcribing the concepts of a document into a documentary language and therefore leads to the registration of these concepts in an organized and easy-to-access form called "Thesaurus".

The guiding lines for the creation and the development of Thesaurus amount to two main steps :

- the identification of the concepts representing the piece of information contained in the datas to be indexed ;

- the organization of these concepts to represent them into the documentary language.

Therefore, the Thesaurus is made of a group of terms, or descriptors, connected together by semantic relations. It is defined as a means to carry out this indexation with the aid of a vocabulary of these descriptors. Then, the Thesaurus enables to translate any concept that may go into or out of a documentary system into an indexation term (5).

The representation of the relations between concepts shows the connections between the terms used. This is one of the main functions of a Thesaurus. The value of the documentary tool appears as much in the selection of terms as in the choice of relations between concepts.

The relations to be considered are :

- relations of association
- relations of equivalence
- hierarchical relations

The Picture 1 shows the type of organization that we have described above (10). The modelization of these three types of relation will be further developped in the Paragraph V.

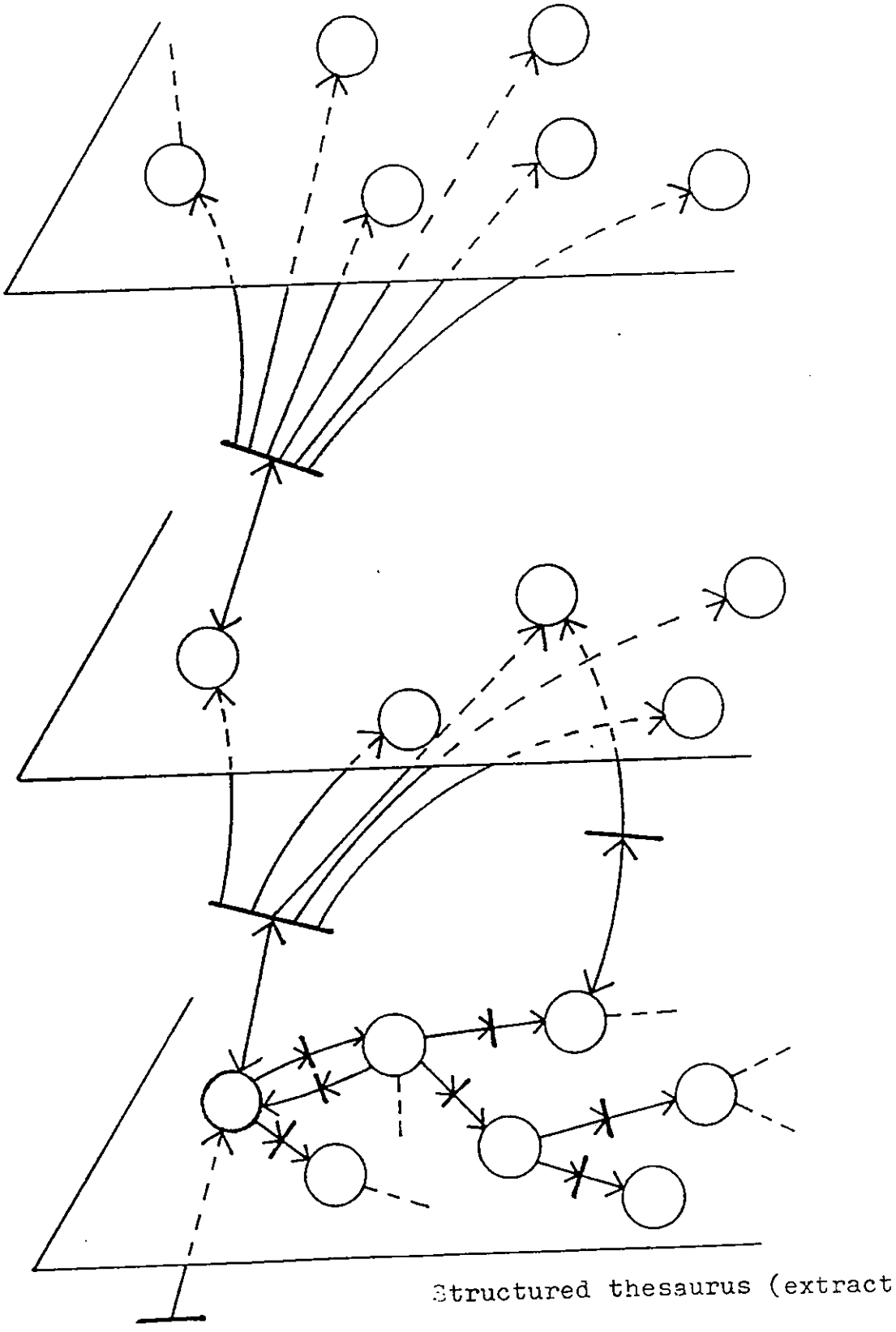


Figure I. General organization of a structured thesaurus : case of Kent's repertory

III. KENT'S REPERTORY OF MEDICAL MATTER

The choice of Kent's repertory as reference document is an open question (1). Two reflections have guided this initial choice :

- on the one hand, it is a source of rich information ;
- on the other hand, its structure of index and Thesaurus mades it directly usable in a data research system.

A homoeopathic doctor must know the symptoms generated by active substances (for ex. PHOSPHORUS) on the healthy individual. The whole of these symptoms built what we call the "Pathogenesy" of a substance. The collection of these pathogenesies makes up the homoeopathic medical matter. There are many medical matters ; one of the most famous is that of Hering. Significant efforts have been made since the beginning of this century to make out more accessible medical matters. Kent made out a repertory which enables, for a specific symptom, under specific conditions, to find out the corresponding medicines or medicine, as far as the simillimum.

The homoeopathic repertory is a standard tool for data research, i.e. the index.

Each rubric of the repertory is a term to index documents which are medicines. The degree of a medicine in a rubric shows the representativeness of the medicine for this term.

We usually take into account :
- relations of synonymy (equivalence)
  for ex. : hatred, misanthropy

- relations of proximity (association)
  for ex. : company, fear

- relations of genericity and specificity
  for ex. : man is generic of Phosphorus and specific to Atred.

The Picture 2 gives an extract of the rubric organization in the Chapter "Psyche".

The various connections of the rubric "Atred" are developped. The whole connections are not fully presented in order to keep somewhat legibility.

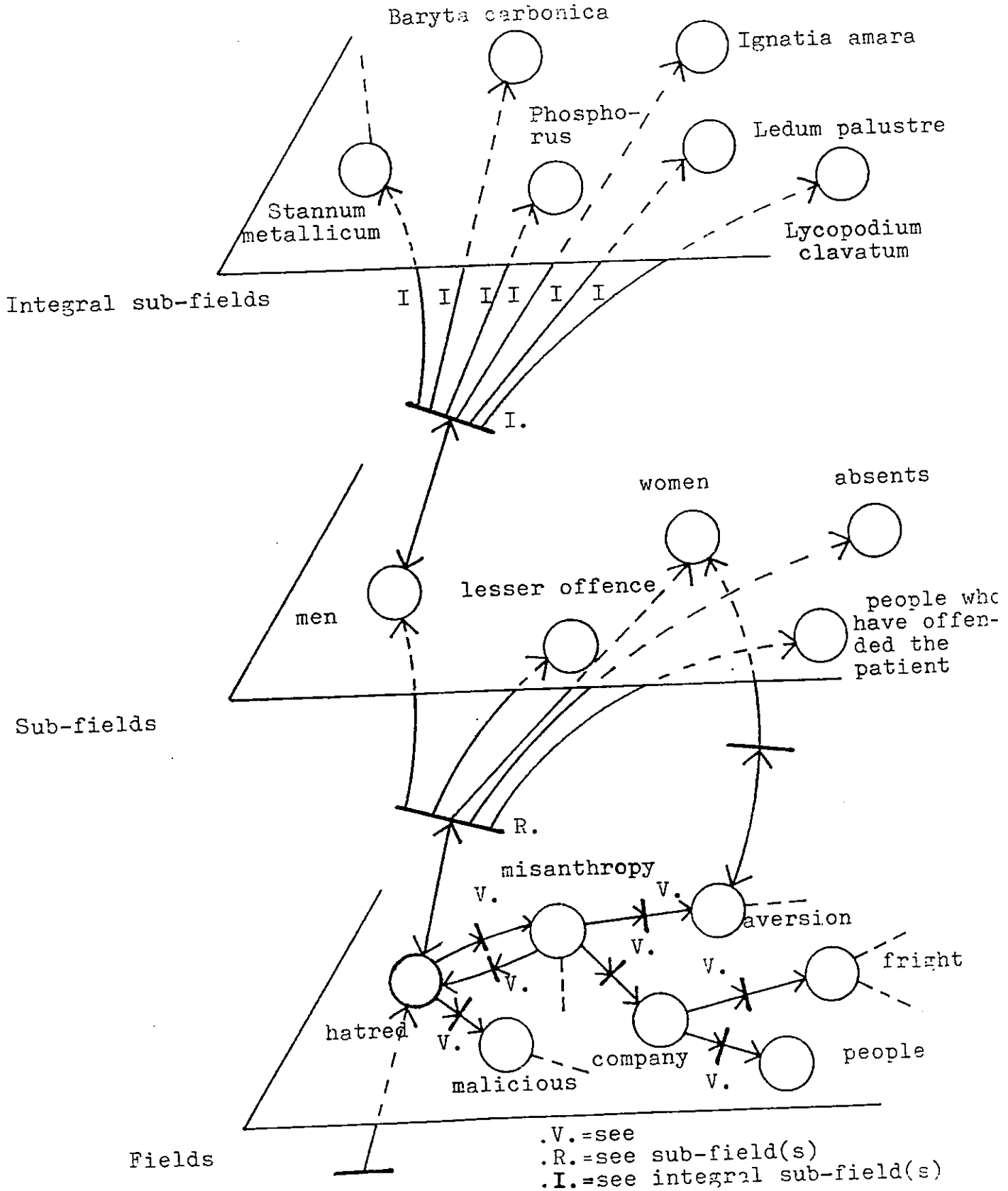


Figure 2. Extract from the organization of a structured thesaurus: case of Kent's repertory

IV: METHODOLOGY : TAKING INTO ACCOUNT OF OPEN COMPUTER SYSTEMS

It is convenient, if not necessary, to name a group of rubrics which is meaningful for a specific concept, in order to be able to find it out again thanks to this name. The computer support allow such achievement with two advantages to be underlined :

- the first one is a quick access ;
- the second one is that the system is open to a continuous enrichment.

A printed Thesaurus is set rigidly and must be considered as definitive until the next edition.
A well-designed computer system is open, i.e. it allows changes or addition of concepts (9).

This consideration leads us to the formalization of open computer systems. First of all, we share the "finish-limited" from the "finish-unlimited". The first one is the field of compilers (presently, translater of symptoms into medicines) ; the latter is the one of the system.
The compiler works on the repertory, i.e. a "finish-limited" field essentially.
The system will ensure a conversational connection and the feeding of the existing Thesaurus by the access to other essential documents :
the pure medical matter of Hahnemann, the medical matters of Allen and Hering, and the Synoptic Key of Boger (9).

The works of the system and those of the compilers become fully shared. Therefore, the compilers are pre-structured so that the system operates as a program into which all sorts of procedures fit. The calculation time of these procedures, key point of this method, is strictly limited. The complete work of this task is achieved by the insertion of as many procedures as necessary. The system can no longer be blocked by some data flow, as it remains in control of the situation (6).
This is the method which is developped by the Researchers of "the Laboratory of theoretical computer science and applications" in Marseille.

This methodology is based on two main ideas :

- to structure algorithms so that the work quanta are predetermined and nothing is left to chance ;

- to design a computer structure adequate to add a new compiler repertory in our system later.

A software structure only allows the complete description of our computer system considering the previously set conditions : "autojectivité" (4).

This structure has a single input and a single output. It is composed of a group of parallel ways which execute the whole operations. In addition, we must have the following properties :

- Each of these ways is finished and limited in time. Whatever operation each of these ways achieve, it will need less time than a pre-set limit ;

- Each of these ways is finished and limited in space. Whatever operation is executed, each way can only use a memory volume smaller or equal to a pre-set volume limit.

The drawing of the Picture 4 gives a very simplified view of how to cover the repertory rubrics leading to the medicine research. To reach such result, we will need to have a complex group of tools, including a suitable "navigationnel" programming language.

A technique to build "autojectif" algorithms has been developped (6). Besides, a method to convert any algorithm (type A (ej. higher) picture 3) ; (type B - picture 3) into a "autojective" form (type A (ej. lower) picture 3) has been defined.

This method can be compared with the reachability tree analysis.

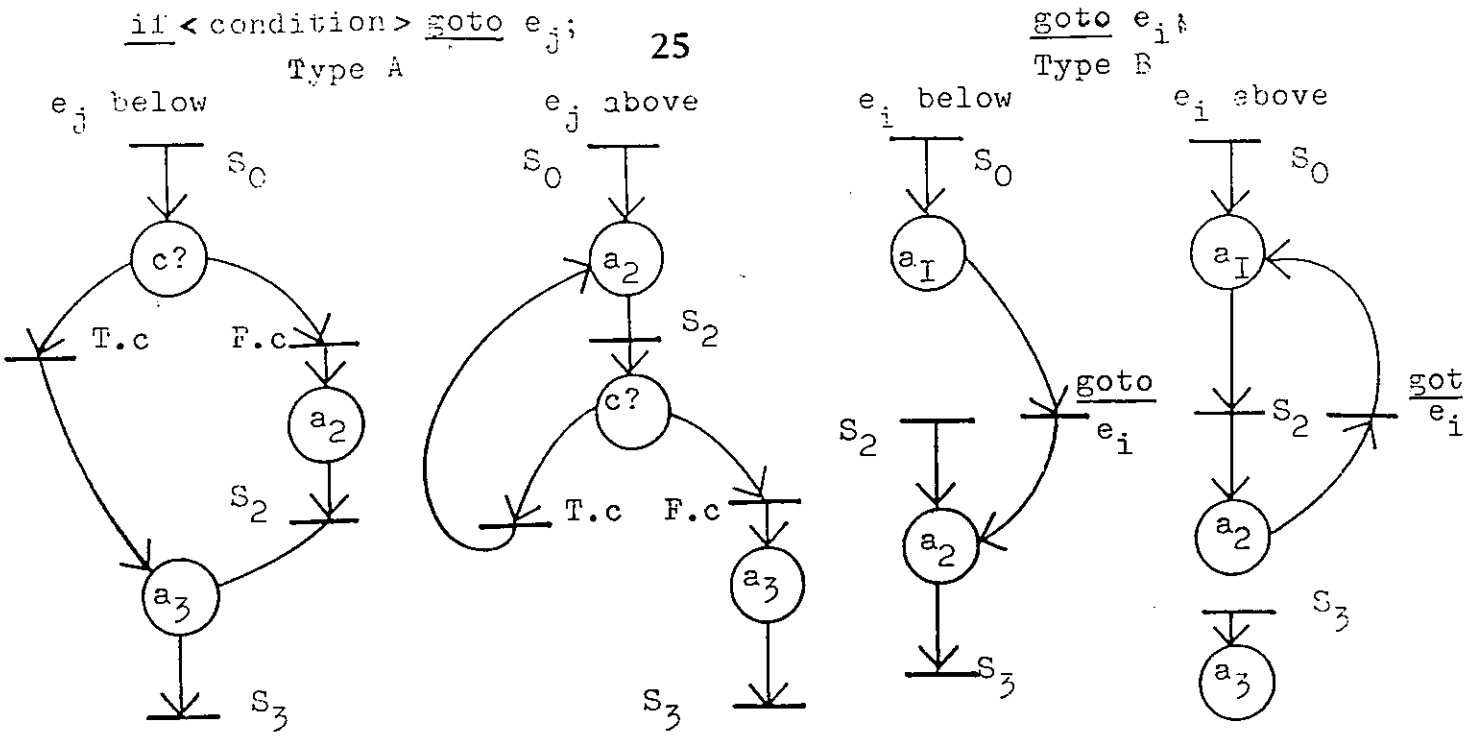


Figure 3. *Autojective* commutation

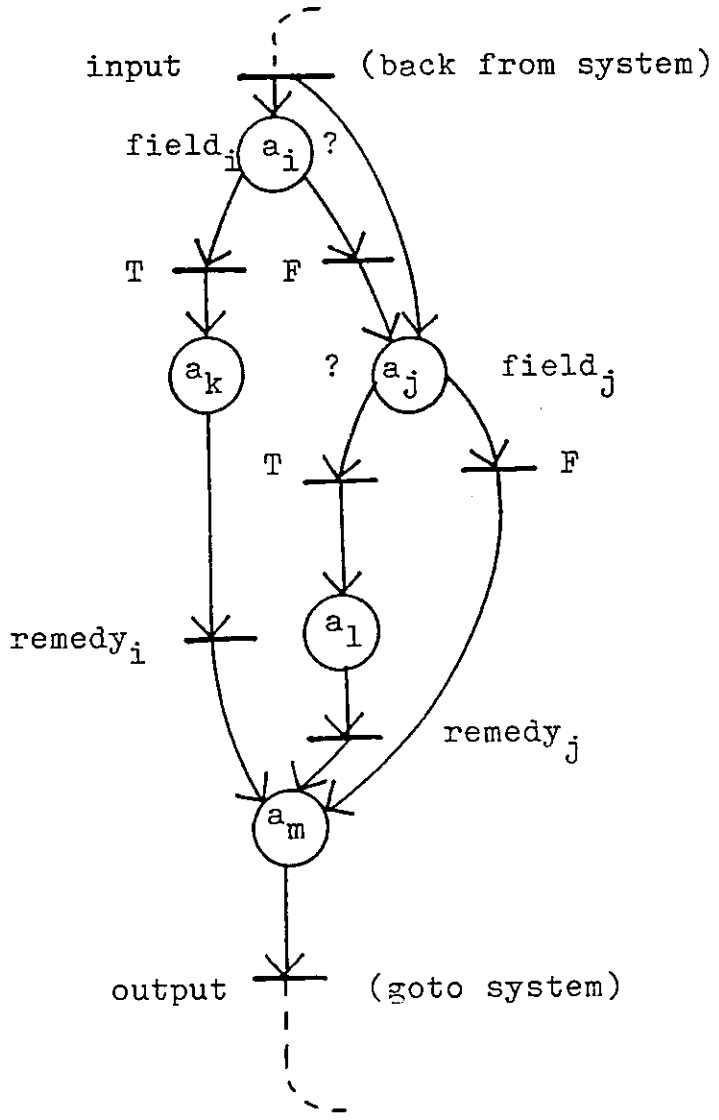


Figure 4. Principle representation of a simplified *autjective* structure

V. MODELIZATION OF THE SYSTEM WITH THE AID OF PETRI NETS

The first step consists in submitting an assertive sentence expressed in natural language and in bringing out the different parts of the sentence. This is the first approach analysis. The second approach analysis, applied to a generated intermediate diagramm, enables to attribute to each word or group of words of the sentence the sole corresponding concept.

Thus, any problem of synonymy or equivalence between notions is solved. Then, the semantic way or ways corresponding to the intermediate diagramm are generated. The identification of the different concepts is ensured by the secondary Thesaurus which collects all the relations of synonymy, hierarchy or equivalence existing between the natural language terms in the specific application field, i.e. presently the medical matter.

We have decided to formalize the secondary Thesaurus with the aid of Petri nets. The modelization of the representations of the relations between concepts will be achieved by unary predicate / transitions nets (8). Now, let us present the modelization of the various relations to be considered :

- the relation of equivalence (picture 5.b) which determines the sole descriptor of the various synonyms of a concept used in the documentary language. This enables to find out all the things associated with the equivalence group ;

- the hierarchical relation (picture 5.c), which is the skeletal structure of the Thesaurus and ensures the internal consistency, this is the repertory's rubric. It expresses the relations of insertion between notions ; thus, the broadest notion represents a whole in which the specific notion is one part or a special case ;

- the relation of association (picture 5.a) enables to give the meaning analogies between terms, which are generally mentally associated by the relevant specialists. These relations, also called "relations of proximity" , exclude the relations of equivalence and the hierarchical relations.

At least, the third approach analysis enables to check whether the semantic way generated by the second approach analysis is consistent, if it has a meaning or not.

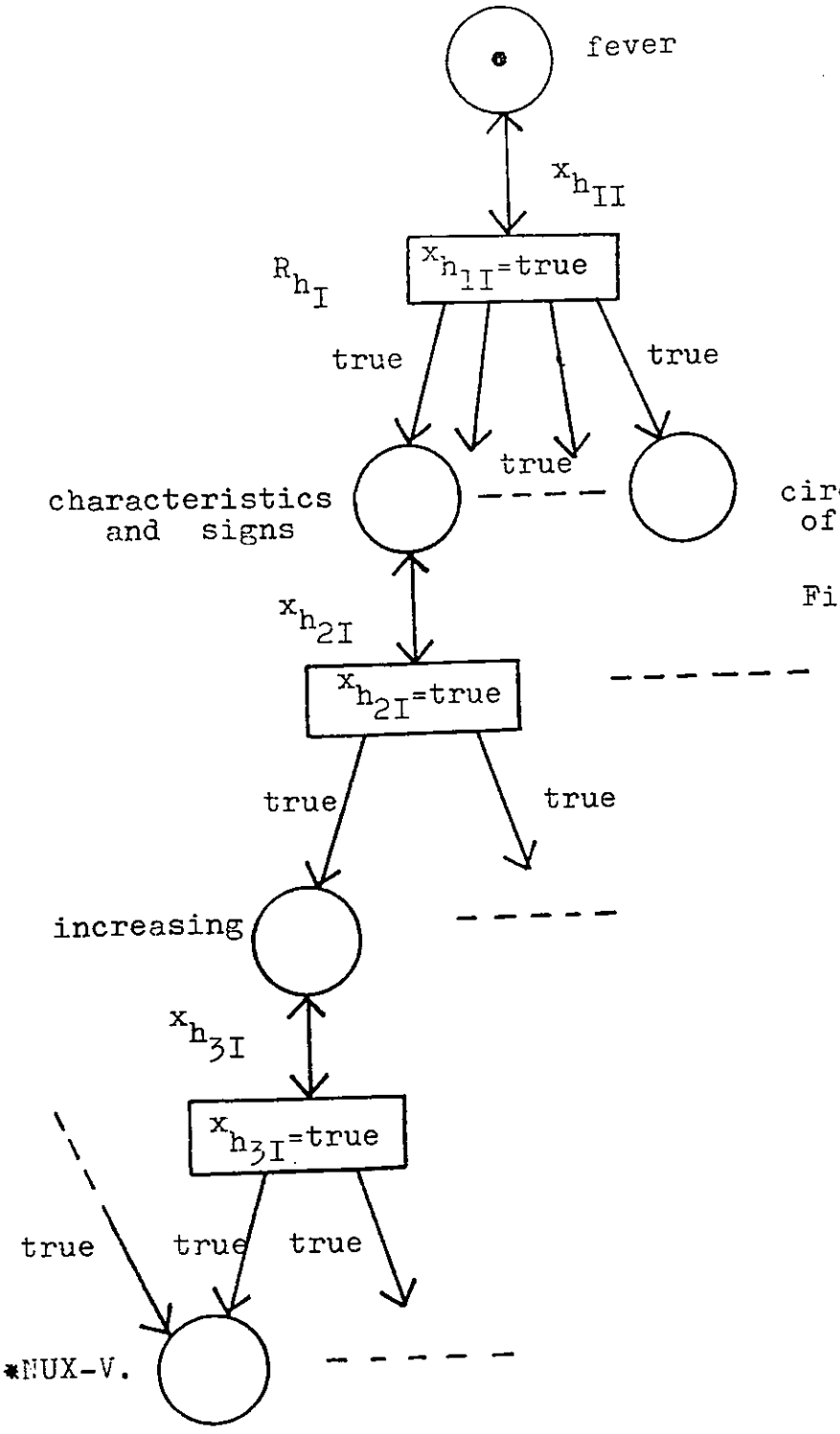


Figure 5c. relation of hierarchy

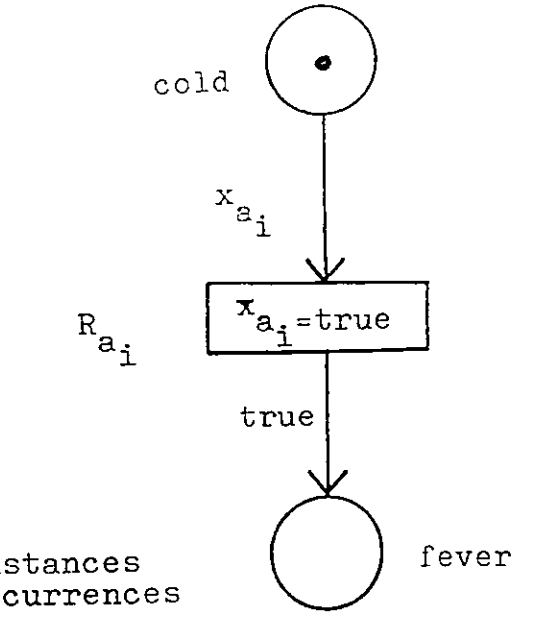


Figure 5a. relation of associatio

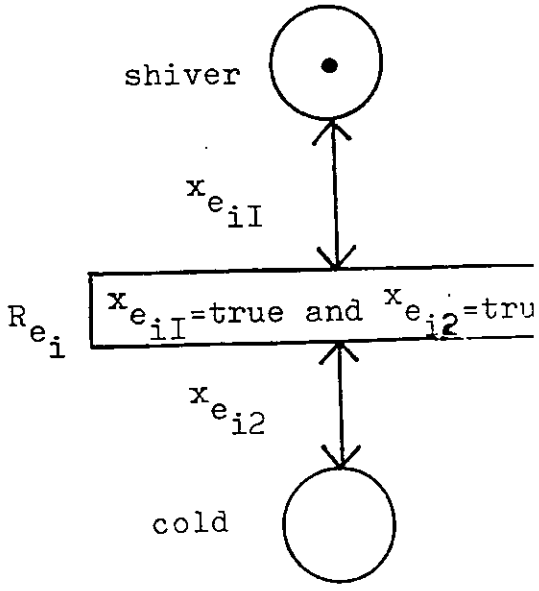


Figure 5b. relation of equivalenc

Figure 5. Examples of representation of links between concepts

VI. FINALITY : MEDICAL DECISION AID

An attempted clarification in this field is necessary as the confusion in the (functions, operations) that will ensure the decision aid systems is partly responsible for their very varying fortunes. Globally, three types of systems can be distinguished according to the mode of intervention into the decision process :

- systems of indirect aid to the decision
  (ex. results from laboratories)

- systems of automatic (return, reminder)
  (ex. watch-dog systems)

- consulting systems(10).

The consulting systems tend to give a specialist's advice faced with a specific medical situation (case).
In our case, applied to documents/medicines, we face the problem of researching the medicine "simillimum".

A first step consists in suggesting a criteria for a rubric selection, which uses the rubrics and the medicine names of the repertory : this is the "request" language, very similar to the "P.F.S." language (language of symbolic formal procedure) intended to describe systems and "autojectifs" compilers (6). We have decided to show an exemple (Picture 6) : how to operate the composition of request items with the aid of Petri nets with data flow. Indeed, three types of operators can be distinguished in a program with data flow : execution (Ot), connection (Ol) and control (Oc).
This can be compared to the three language types of the open systems : programming language, exchanging language and control language.

A second step consists in naming a rubric group to keep it as a concept. This original mechanism, as illustrated in the Picture 6b, enables to enrich the "request" language as per the User's requirements.

In the paragraphs V and VI, we have seen the introduction of a calculation possibility by a system of knowledge representation, datas and occurences, based on the Petri nets. Therefore, the dynamics is naturally used (7) to study the properties of this knowledge base, i.e. Kent's repertory.

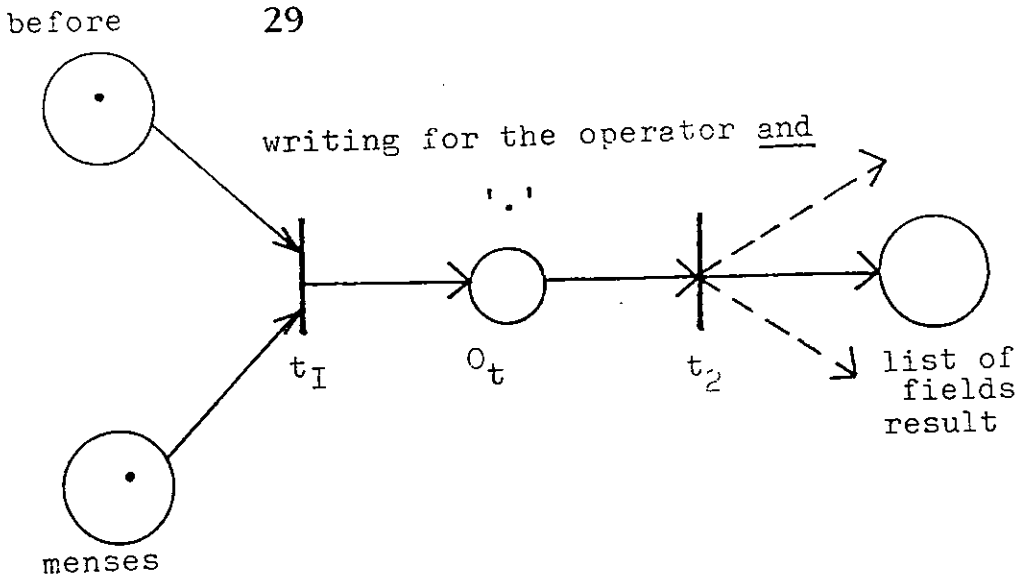


Figure 6a. Calculation of before and menses request

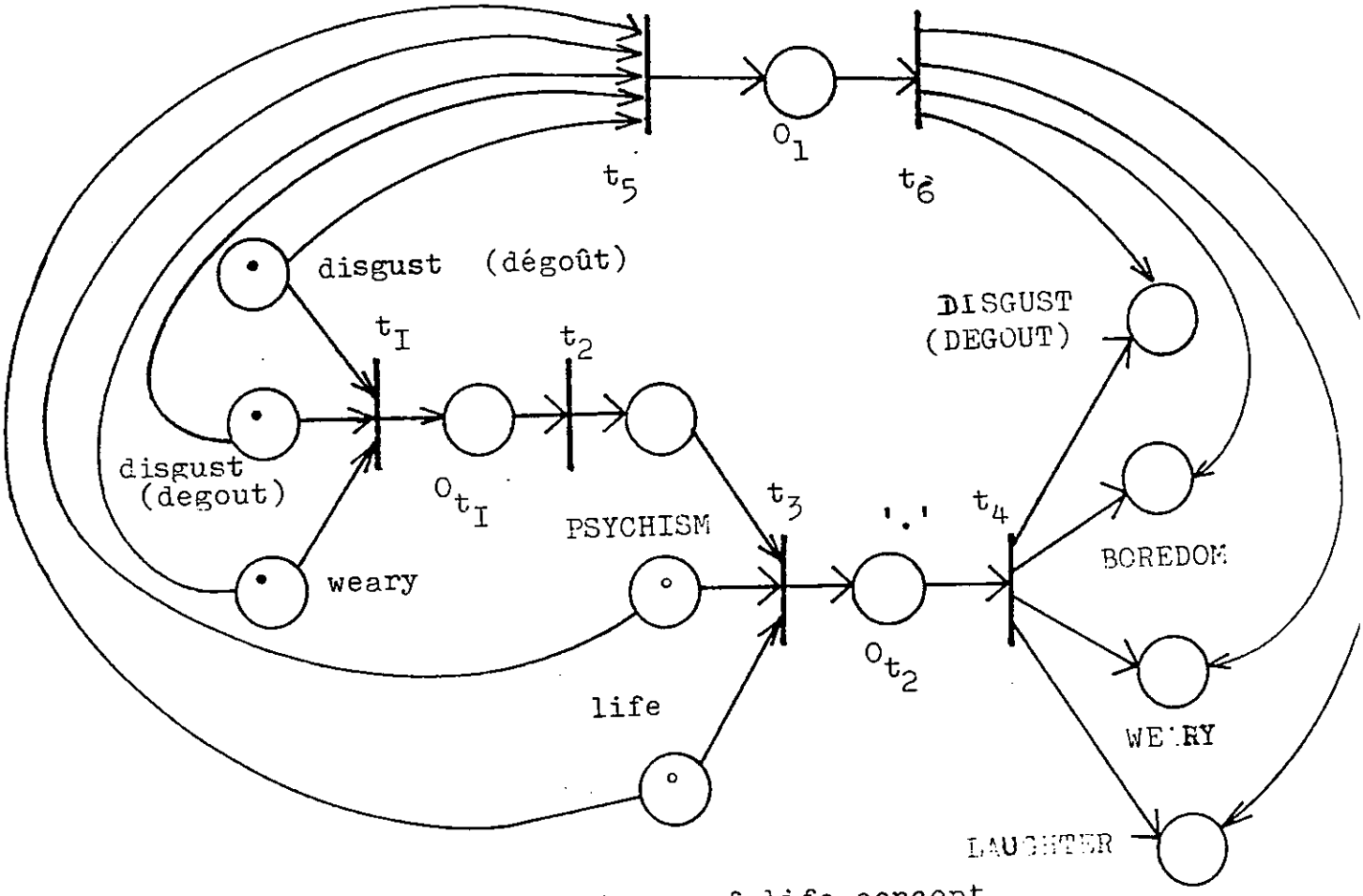


Figure 6b. Weariness of life concept

Figure 6. Operations in request language

VII. DISCUSSION

We have found it desirable to design and create a system which balances simplicity and power of expression enabling to have a system powerful in therory, and realizable in practice (3). Such is our process with the choice of Petri nets.

Our base has been the existing software SAPHIR (9), software for data research in the homoeopathic medical matter using Kent's repertory in the new, complete, French translation made by the Doctor Edouard Broussalian (1).

We have just proposed the representation of knowledge and semantic relations for the creation of a Thesaurus for homoeopathic signs, what is a long-term job. Indeed, it appears as relevant to keep 1000 to 2000 concepts to represent the essential features of medicines. This leads us to determine major concepts of a Thesaurus for homoeopathic signs (9).

Then, the connection of these concepts with the 65 000 rubrics of Kent' repertory to lead to the relevant medicines requires a "navigationnel" language in order to assist the practician in researching the simillimum. We have quoted a few features and properties relating to the notion of "autojectivité" which enables to check easily the consistency of the described system.

The "open" computer system enables to receive new additional repertories : Hahnemann, Allen and Hering, etc... It will allow, thanks to " a vector of datas of split insertion " to convey the necessary peace of information between the different processes / repertories that have to communicate. These necessary datas will be researched in the same way in each repertory.

The best hardware and software settling is researched to bear the Existing and integrate the environment of specification and validation of the distributed applications, i.e. the different repertories.

VIII. REFERENCES

(1) E.BROUSSALIAN
Répertoire de la Matière Médicale Homoeopathique de James T. Kent
Editions Roger Jollois. 1992. Limoges

(2) J.M.KNIPPEL, N.T.LASKRI
First study of a thesaurus creation and development system by means of Petri nets
International Conference CECOIA3/CEMIT92. Tokyo
Edited by JASMIN/IFORS. 1992. Tokyo

(3) M.RØTHEMUND
Modelling medical organizational systems with nets
Methods of information in medicine. 25. 1986
F.K. Schauttauer-Verlag Gmbh. 1986. FRG

(4) P.ISOARDI
Structures de commutation
Thèse pour obtenir le titre de docteur de spécialité en intelligence artificielle
Université d'Aix-Marseille II. 1978. Marseille

(5) J.M.KNIPPEL, N.T.LASKRI
Première étude d'un système d'aide à la conception de thésaurus: cas du répertoire de Kent
Bulletin d'informatique approfondie et applications
ISSN 0291-5413. N°26. 1990. Marseille

(6) E.BIANCO
Essai sur la compléxité des systèmes informatiques
Revue internationale de Systémique. Vol.4, N°2
AFCET. Gauthier-Villars. 1990. Paris

(7) C.SIBERTIN-BLANC
Le prototypage des applications interactives à l'aide des réseaux de Petri
Séminaire "Les réseaux de Pétri".
AFCET-IEEE-CS. 1988. Evry

(8) E.PIPARD
INDE:un système de détection d'inconsistances et d'incomplétudes
dans les bases de connaissances
Thèse pour obtenir le titre de docteur 3°cycle en informatique
Université de Paris-Sud. 1988. Orsay

(9) M.SIMONET
Du mot au concept
Congrès international d'Homoeopathie.1990. Liège

(10) P.DEGOULET, M.FIESCHI
Nouvelles technologies et traitement de l'information en médecine
Informatique et Santé. Tome 4
Springer-Verlag. 1991. Paris

Le drame d'Arsène Lapin.

Il y a des gens qui portent des noms de fleurs, c'est bien souvent charmant, Lupin, Rose, Dahlia, mais cela fait un peu mièvre. D'autres portent des noms de lieux ou d'accidents géographiques, Garonne, Montalenvers, Entraïgue, Val, cela peut faire très sérieux, voire passer inaperçu. Mais d'autres encore portent, allez savoir pourquoi, des noms d'animaux, Bœuf, Mouton, Loir, et cela peut devenir extrêmement pénible. Tant qu'il s'agit d'animaux discrets, passe encore. Mais certains d'entre eux, font tout pour se faire remarquer, pensez au lion qui porte une exécrable réputation de dominateur. Allez porter un nom pareil si vous êtes un individu calme et qui recherche la paix, imaginez les quolibets en permanence ... Ce n'est encor rien, essayez de vous mettre dans la peau de quelqu'un qui s'appelle Lapin, avec tout le passé lourd de couardise de ce pauvre animal. Et plus grave encor, sa réputation d'infatigable amoureux.

Avec leur férocité de jeunes mâles, les petits copains de notre héro Arsène Lapin, s'en étaient donné à cœur-joie. Et notre blondinet tout rose et timide, un peu grassouillet, avait beaucoup souffert. Il s'était juré in petto de se venger quand il serait devenu grand. Qui d'entre nous tous n'a jamais fait ça à un moment ou à un autre ? Qui, en bute à une dérision, n'a jamais ruminé une vengeance terrible ? et plus le malheureux coincé entre sa fierté et la malignité publique se sent ridicule, et plus il a tendance à se refermer sur un univers intérieur chargé des foudres de la représaille. En général une bonne nuit de sommeil et tout est oublié. Mais il est des exceptions gravissimes chez qui la perturbation si menue, si négligeable qu'elle puisse apparaître tout sang-froid revenu, engendre des troubles graves dont les conséquences ultérieures sont incalculables.

C'est ainsi que ce qui allait devenir chez lui comme une sorte de vocation était né le jour où un copain l'avait appelé Arsène Lupin. Tiens, pourquoi Lupin ? C'est quoi Lupin, une fleur ? Vexé il n'avait rien répondu mais dans le feu couvant de son indignation, sa curiosité s'était allumée. C'était plus tard qu'il avait enfin découvert l'œuvre de Maurice Leblanc. Il avait alors décidé de se forger une mentalité à la mesure de son presqu'homonyme. Bien sur il avait fait ça avec les moyens dont il disposait. Et la bonne tenue morale était rapidement devenue de la rigidité morale. D'une morale, bien entendu qui avait légèrement tendance à glisser au cours du temps pour finir par composer avec ses sentiments nettement parfumés à l'aigre, sur ce vieux fond de rancœur. Il passait tout son temps à s'observer dans ses actes et ses réactions et il classait le tout en

deux parties, celle de tout le monde, et celle qui caractérisait son génie personnel. Il se délectait de ses bonnes répliques, il se rengorgeait de ses bonnes réactions. Quant à ce qu'il classait dans la première partie, les attitudes quelconques, les petites lâchetés, les réactions plates et peu brillantes, elles étaient pour lui comme une sorte de masque pour mieux passer inaperçu. Les déguisements de génie du grand Arsène Lupin, en quelque sorte.

Et puis vint le temps des amours, il se découvrit la Pierrette de sa vie avec laquelle il tenta d'expérimenter la vigueur qu'il sentait assortie à son nom. Ce fut plus ou moins bien réussi. Et puis tout lasse, surtout les réussites moyennes. Que font alors les supermâles dont la virilité se révèle normale, voire médiocre ? Tout simplement ils remplissent leur maison d'enfants, pour montrer que ça marche bien, ou alors ils recherchent le pouvoir, ou les deux à la fois, car la deuxième partie est une excellente échappatoire à la première. Pour notre homme, la première voie se révéla une voie de garage. Au bout de plusieurs années les méthodes Ogino à l'envers, les pilules de fertilité et autres impositions de mains de mages célèbres, demeurèrent tout autant de vaines tentatives. Pierrette demeurait stérile, malgré toutes les visites des plus grands spécialistes qui la déclaraient normale. Arsène ne pouvait imaginer un instant que sa propre robuste constitution pouvait être en cause. Alors il se jeta dans la politique. Son élocution facile et populaire fit merveille. Il savait avec adresse reprendre tous les poncifs de bistrot, et jeter l'anathème sur les politicards pourris qui, il faut bien le reconnaître ne brillaient pas spécialement par leur efficacité dans les affaires publiques, en tout cas bien moins que dans la réussite de leurs petites affaires personnelles.

De réunion de quartier, en tournées de popotes, car il était réserviste d'une vieille guerre coloniale dans laquelle il s'était brillamment illustré, il s'était hissé au plus haut niveau de la pratique des médias. Il n'était plus de problème grave qui surgissait sans qu'on ne l'interviouve, les télévisions et les radios se l'arrachaient. Les présentateurs arboraient toujours un petit air d'ironie, pour montrer qu'on prenait ses distances, mais son bon gros humour mode marteau pilon faisait bien souvent mouche, surtout dans le peuple excédé. La presse passait des photos de lui, souvent ridicules, il attaquait en justice et ne perdait pas toujours. Mais son drame personnel était toujours là, présent. Et même il s'aggravait. Emule de Céline qui prétendait que la race blanche était forcée de périr par mélange avec de la couleur, il prétendait, lui, que les étrangers trop prolifiques allaient bientôt carrément remplacer les français sur leur propre sol, et il prônait la démographie galopante de la noble race pour vaincre. Donc il fallait donner l'exemple. Mais, bon chétien, il ne pouvait pas comme ces

musulmans maudits prendre plusieurs femmes. Ah Dieu que tu ne nous rends pas la vie facile!

Et c'est ainsi qu'un jour il entendit la voix des Sirènes. On lui expliqua les miracles de l'insémination artificielle. Il lui fallut réfléchir longtemps. Il se teignait en blond pour avoir davantage l'air d'appartenir à la race pure des seigneurs qui, comme l'on sait depuis qu'un grand mage trop tôt disparu nous l'a bien expliqué, sont les véritables maîtres naturels de la planète. Foin de tous les petits trapus, des grands secs, et autres tignasses sombres et crépues sur des faciès étranges dont le répugnant est aggravé par des ongles crochus, qui voudraient répandre leurs ignominies sordides d'humanoïdes mal réussis sur tous les continents, dans un permanent et grouillant travail de sape .

Il voulait être absolument sûr d'être l'heureux papa d'un délicieux bébé rose et blond, un garçon, comme lui dans le temps, bref un futur guerrier.

On recueillit de sa précieuse semence en plusieurs fois, mais là encore ce fut un échec. Lorsqu'avec les plus grandes précautions oratoires et les métaphores les plus alambiquées le grand Professeur tenta de lui suggérer une provenance extraconjugale, il mit longtemps à comprendre. Le pauvre médecin en transpirait de chercher des images anodines et suggestives tout à la fois. Quand, soudain, son front se plissant sous l'effort il comprit enfin, Arsène faillit lui foutre son poing sur la gueule. Il fit un demi-tour brutal et partit furieux. C'est dans un violent bruit de verre brisé qu'il franchit d'un bond la porte transparente peu visible dans la pénombre du couloir. C'est à peine s'il sentit le choc.

Le temps pressait, car depuis les derniers mois, quand il tonnait en public contre la dénatalité, il lui semblait en percevoir qui commençaient à ricaner, dans le fond. Il tenta une deuxième épreuve d'insémination artificielle, et derechef on recueillit sa semence. Il ne put s'empêcher de faire une prière pendant cet instant délicat, bien que l'acte ne s'y prête guère, et Pierrette officiante en fut presque choquée.

Dieu, sans doute, devait être ce jour-là un peu plus miséricordieux que d'habitude, car Pierrette ne tarda pas à s'arrondir joliment. Désormais ils apparaissaient tous les deux en public, la main dans la main. Pudiquement, sans un mot pour souligner l'évidence, avec de faux airs de modestie touchante. Le miracle annoncé prit son temps, on l'attendit le temps qu'il faut, et puis un jour, ce fut la délivrance qui s'annonça. Elle fut annoncée dans un faste d'une impériale simplicité.

On eût pu relever dans la salle d'attente un florilège de la quintessence des personnalités pensantes du Parti. L'heureux futur père arborait le front soucieux qui s'impose, et la gravité qui irradiait de la puissance du menton turgescent atténuait à peine

l'éclat des yeux. Fierté noble du père primipare, certitude de l'arrivée du fils tant souhaité, victoire enfin sur l'envahisseur. Bref un orgueil empreint de dignité, tout-à-fait compréhensible. Aucun cœur un tant soit peu patriote ne pouvait demeurer insensible.

Quand soudain dans le silence profond que ne troublait même pas les chuchotis des média venus en force, une infirmière entra pour annoncer l'heureux évènement : « Le bébé est très beau .. dit-elle, il se porte à merveille ...» Arsène ressentit comme un léger trouble, quelque chose sonnait mal dans les mots de la fille.
« Alors! » gronda-t-il. « Je veux le voir » Ajouta-t-il plus doucement. L'infirmière battit en retraite, la porte se referma sur de longues secondes puis elle revient avec un délicieux bébé sur les bras, tout nu sur une courtepointe. Les yeux de notre héro s'agrandirent brusquement, sa face se convulsa pendant que crépitaient les flashes des reporters, et que les invités saisis retenaient leur souffle: l'enfant était une délicieuse petite fille juste un peu très sombre de peau, de plus, fait plutôt rare à cet âge-là, elle était ornée d'une très belle chevelure, mais noire comme le jais et plutôt frisée, disons même crépue. Arsène devint tout pâle, il sentit son sang refluer, il sentit sa tête se mettre à tourner, des images éclatèrent devant ses yeux, dans un vertige, un éclair fugitif fit étinceler des tubes à essais qu'on intervertissait, puis d'autres scènes s'imposèrent, des hordes de Maures déferlant sur la Gaule envahirent son esprit en tourbillonnant, tandisque ses gardes du corps qui l'avaient vu défaillir le soutinrent fermement par les bras. A travers une sorte de brume, il distingua alors une femme, une gauloise blonde comme les blés, il la vit collaborer avec l'ennemi, avec un Ennemi sombre et crépu qui semblait pourtant lui plaire, c'était une lointaine ancêtre ... son ancêtre...