

## QUELQUES REMARQUES SUR LE CADRE PSYCHOLOGIQUE DE LA PROGRAMMATION

Harald WERTZ

Université Paris VIII  
Dépt. Informatique  
2 rue de la Liberté  
93526 St.-Denis Cédex 02

C.N.R.S. LA 248 / L.I.T.P.  
2 Place Jussieu  
75005 Paris

*The best way to live with our limitations is to know them.<sup>1</sup>*

Avant d'entrer dans une description détaillée des capacités des environnements de programmation et des langages des ordinateurs, nous devons situer notre travail dans un cadre cognitif adéquat. Les environnements et les langages de programmation sont un ensemble d'outils supposés d'aider le programmeur lors de la conception, de la réalisation, de la mise au point et de l'observation de programmes. Comment pouvons nous déterminer si un outil, une caractéristique d'un système, est une aide effective? Bien évidemment, le choix d'un environnement de programmation suppose résolu au préalable deux problèmes :

1. Elle suppose résolu le problème de savoir *en quoi consiste* l'activité de la programmation : quelles sont ses difficultés, ses pièges? Quel est le rapport entre représentation interne (à l'intérieur d'une machine) d'un programme et la représentation que le programmeur en possède? Bref, le choix d'un langage et d'un environnement de programmation suppose une théorie vérifiable sur l'activité de la programmation.
2. Etant donné qu'un environnement de programmation est supposé aider le *programmeur*, le constructeur humain d'un programme, nous devons également avoir résolu au préalable le problème de savoir quels processus un programmeur met en jeu pendant qu'il conçoit, implémente, met au point, observe ou lit un programme. Ceci suppose que nous avons une théorie cognitive de son

1. [Dijkstra72]

---

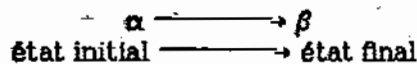
fonctionnement, que nous connaissons ses faiblesses et forces inhérentes. Ce sera seulement à partir de là que nous pourrons décider des outils nécessaires, efficaces et compréhensibles.

Le reste de ce papier sera consacré à l'ébauche d'une réponse à ces questions.

## 1. Qu'est-ce qu'un 'Problème'?

Puisqu'une des hypothèses de départ de tout notre travail est que la programmation est une activité de *résolution de problèmes*<sup>2</sup> nous aimerions, afin de préciser notre sujet, donner une courte définition de la notions de *problème*.

Nous dirons que nous sommes confrontés à un *problème*, si nous avons un état initial  $\alpha$  que nous voulons transformer en un état final  $\beta$ , et nous ne savons pas immédiatement quelle série d'actions peut réaliser cette transformation.



Autrement dit : un *problème* est la recherche d'une suite d'*opérateurs* qui transforment un état initial  $\alpha$  donne dans un état final  $\beta$  désiré. Chacun des *opérateurs* est soit un *opérateur primitif*, soit une *combinaison* d'*opérateurs primitifs*. Les objets définissant l'état initial et l'état final peuvent être - en général - de n'importe quel type (des symboles mathématiques, des figures du jeu d'échec, les briques pour construire une maison, etc). La *représentation interne*, que l'humain (ou la machine) qui résoud le problème en possède, est construite à partir d'*objets primitifs*. D'une certaine manière nous pouvons considérer les *objets primitifs* et les *opérateurs primitifs* comme identiques aux *structures profondes* de la linguistique générative.<sup>3</sup> Tant que nous nous limitons à l'activité de la programmation, les *opérateurs primitifs* sont évidemment les opérateurs primitifs du langage de programmation utilisé, et les *objets primitifs* sont les objets primitifs de ce langage (registres, nombres, listes, etc).<sup>4</sup>

La résolution d'un problème présuppose donc des connaissances préliminaires : une connaissance des opérateurs et des objets primitifs. Par exemple, si l'on trouve l'énoncé suivant :

2. [Kilpatrick69, Lompscher76, Wertz81]

3. [Chomsky57, Chomsky65]

4. Cette comparaison entre *opérateurs et objets primitifs* et les *structures profondes* des linguistes génératifs est partiellement à la base de la *psychologie cognitive* et de l'*intelligence artificielle*. Sans un approfondissement de cette analogie, il serait difficile de prendre au sérieux les efforts d'expliquer le comportement cognitif humain par le comportement d'un programme.

<b>Données :</b>	un triangle ABC
<b>Demontrez :</b>	les trois lignes bisectrices de ses angles ont une intersection en un point unique.

c'est évidemment l'énoncé d'un problème. Pour l'appréhender il faut avoir une compréhension (au moins implicite) de la définition d'un triangle, de la bisection etc (les objets primitifs de ce problème exemple). Il faut avoir une compréhension préliminaire des opérateurs algébriques et géométriques de base, nécessaires à la recherche d'une solution, et il faut, de plus, avoir des interprétations appropriées sur les mots *Données* et *Demontrez*.

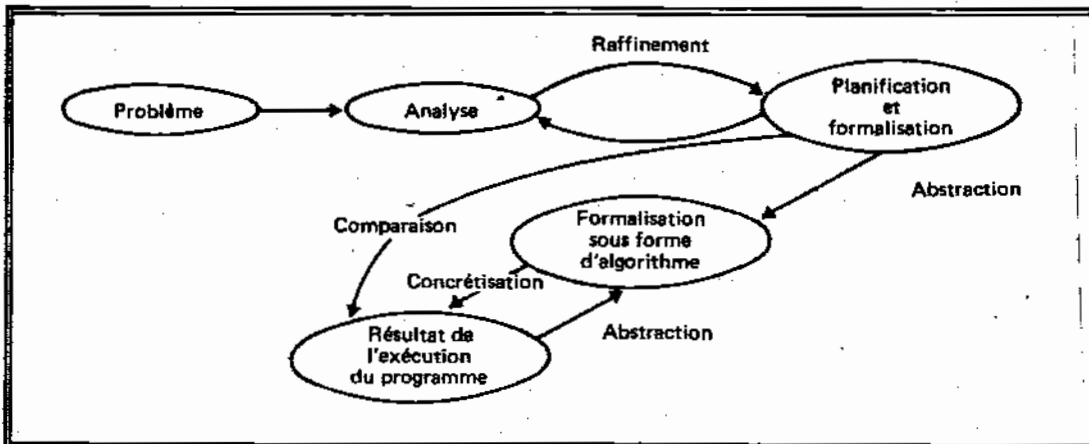
Naturellement, afin de réellement pouvoir parler d'un problème, il faut qu'il soit *bien défini*.<sup>5</sup> Par cela nous entendons qu'une méthode de *test adéquat* doit exister, déterminant si une solution proposée résout *réellement* le problème. Etant donné que nous nous situons à l'intérieur du domaine de la programmation, *adéquat* veut dire que le calcul de cette procédure de test peut se faire dans un temps *raisonnable*.<sup>6</sup>

5. [McCarthy56]

6. Evidemment, sous cette condition, un problème tel que *trouvez une stratégie gagnante pour le jeu d'échecs* ne peut pas être considéré comme un problème *bien défini*, étant donné que pour générer l'espace des solutions possibles il faut générer approximativement  $10^{120}$  états, aussi la procédure de test prendrait donc un temps énorme.

## 2. Courte Analyse de l'Activité de Programmation

Comme nous l'avons déjà dit ci-dessus, la programmation peut être considérée comme une activité de *résolution de problèmes*<sup>7</sup> de même que la démonstration de théorèmes, les échecs, l'architecture, etc, et peut donc, comme tout autre activité de résolution de problèmes, être décomposée en différentes unités conceptuelles, ou cognitives :



1. *Analyse du problème* : C'est le premier pas dans la résolution de problèmes. Il s'agit de comprendre le *domaine* du problème : ce qui implique une compréhension de son énoncé, des conditions particulières inhérentes, de ce qui est désiré (l'état final) et des conditions secondaires.

Cette phase d'analyse comprend également la décomposition du problème en sous-problèmes, l'abstraction de détails insignifiants et la reconnaissance des parties principales, la classification, éventuellement le travail par analogie, l'utilisation de diagrammes, le développement d'un langage d'analyse, etc.

2. *Planification d'une solution* : Une fois l'analyse terminée, il faut trouver une possibilité pour reconstruire les différentes parties du problème en un tout, il faut trouver une définition pour un ordre dans lequel les différents sous-problèmes doivent être résolus et éventuellement il faut développer une stratégie. C'est probablement la partie la plus créatrice dans le processus de résolution de problème.
3. *Formalisation de la solution* : C'est la transposition du plan et des résultats de l'analyse dans un langage formel (en l'occurrence dans un langage de programmation). Ceci inclut une formalisation de la représentation des données du problème. Ce n'est nullement limité

7. [Newell72, Polya57]

au *codage* proprement dit : très souvent même, la formalisation fait apparaître des erreurs dans le raisonnement précédent et conduit à des révisions (au moins partielles) de la stratégie choisie. Une esquisse d'analyse de ce stade se trouve dans [Newell72] ou [Wertz78].

4. *Exécution du programme* : La formalisation ayant abouti à la construction d'un programme exécutable, il faut le tester, i.e.: l'exécuter avec des données concrètes judicieusement choisies. *Judicieusement* signifie que, dans le cas idéal, les données sur lesquelles le programme sera testé devraient couvrir l'ensemble de toutes les classes de données possibles pour ce programme.
5. *Analyse des résultats* : Même pour des informaticiens confirmés, il est rare qu'un programme ne contienne pas d'erreurs, d'*irrégularités* (des *bugs*). Afin de détecter ces *bugs* il faut d'abord comparer le résultat obtenu (si on en a obtenu un!) avec celui désiré, en trouver les différences, les *raisons* possibles ayant engendré ces différences afin d'imaginer des solutions alternatives.
6. *Correction du programme* : Dépendant de l'analyse, le programmeur sera amené à modifier le programme afin de le faire mieux converger vers l'état final désiré. Une correction implique naturellement une répétition des quelques derniers points, avec le programme ainsi modifié.

Le cas particulier de la résolution de problèmes par la programmation nécessite trois stades supplémentaires :

- A. *Optimisation du programme* : Très souvent la première version *correcte* d'un programme contient des faiblesses concernant l'espace de travail nécessité ou la vitesse d'exécution trop lente. Des travaux de modification, de transformation du programme en vue d'une amélioration de ces critères suivent le développement propre du programme.
- B. *Maintenance du programme* : Une fois le développement et l'implémentation d'un programme terminé, il reste toujours des petits *bugs* qui ne se montrent que dans une utilisation intensive du programme. Nous nommons la mise au point ultérieure de ces bugs et l'adaptation du programme à des conditions matérielles changeantes (l'achat d'un nouvel ordinateur ou d'un nouveau compilateur, etc) la *maintenance* de programmes.
- C. *Documentation du programme* : Une fois le programme terminé et prêt à être utilisé, il faut développer une documentation indiquant "comment utiliser ce programme" et "comment le programme est implémenté", afin de permettre aux utilisateurs futurs et aux

---

programmeurs chargés de la maintenance du système, d'en profiter au maximum.

Chacun de ces stades présente ses propres problèmes. Un *environnement de programmation* est un ensemble de programmes capable d'*assister* le programmeur efficacement dans chacun de ces stades différents du processus de résolution de problème, ou, autrement dit, dans chacun des stades de la conception, le développement, la mise au point et la maintenance de système.

### 3. Les Processus de la Pensée

Jusqu'à maintenant nous nous sommes limités à exposer le domaine de la programmation. Chaque programmation nécessite des programmeurs. Dans ce paragraphe nous allons exposer une limitation très importante des programmeurs (et de tout autre humain) : la *quantité* des informations, qui peuvent être traitées à un instant donné, est très limitée. Aucun moyen n'existe pour augmenter ce nombre. Par contre aucune limite concernant la *qualité* des informations semble imposée. Ceci a un effet immédiat sur la conception de tout environnement de programmation : l'environnement de programmation doit, d'une part, limiter le nombre d'informations et, d'autre part, il doit donner des informations sur un niveau le plus élevé (le plus conceptuel) possible.

#### 3.1 la mémoire

Dans le contexte de cette étude, le problème principal sera de déterminer combien d'informations un humain peut traiter à un instant donné. Evidemment, ceci dépend des limitations quantitatives de la mémoire humaine. Jetons donc d'abord un bref regard sur les résultats des recherches psychologiques sur la mémoire :

Actuellement les théories psychologiques proposent trois ou quatre niveaux de mémoires différents : la mémoire *sensorielle*, la *mémoire à court terme*, la *mémoire à long terme* et éventuellement une *mémoire à terme intermédiaire*.<sup>8</sup> La mémoire à long terme est essentiellement un mécanisme de stockage, bien qu'elle utilise apparemment des stratégies d'accès assez sophistiquées et peut traiter des demandes dans un mode correspondant au *temps-partagé* des systèmes opératoires,<sup>9</sup> i.e.: elle peut exécuter différents processus en même temps. La mémoire sensorielle n'a qu'un rôle mineur dans les processus humains de traitement de

8. cf.: [Cofer76, Anderson73, Kintsch74, Norman75] Remarquons que cette classification des mémoires en mémoires à court et à long terme n'a aucune relation avec des géographies neurophysiologiques : elle est une conceptualisation d'observations expérimentales. Nous nous imaginons les processus neurophysiologiques sous-jacents comme des états d'excitation de neurones à des degrés variés. Ainsi, par exemple, la mémoire à court terme peut correspondre à un ensemble de neurones interconnectés qui se trouvent dans un état d'excitation très élevé, de manière telle, que l'arrivée d'un unique signal suffit pour *lancer* un (ou un groupe) des neurones. Cette théorie rendrait bien compte des temps d'accès extrêmement rapide de la mémoire à court terme.

La mémoire sensorielle par contre correspond bien à une réalité neurophysiologique connue. Une description de l'état des recherches neurophysiologiques sur les mémoires se trouve dans [Sperry82] et [Platt85]. Un résumé se trouve dans [Sperry82a].

9. [Norman76]

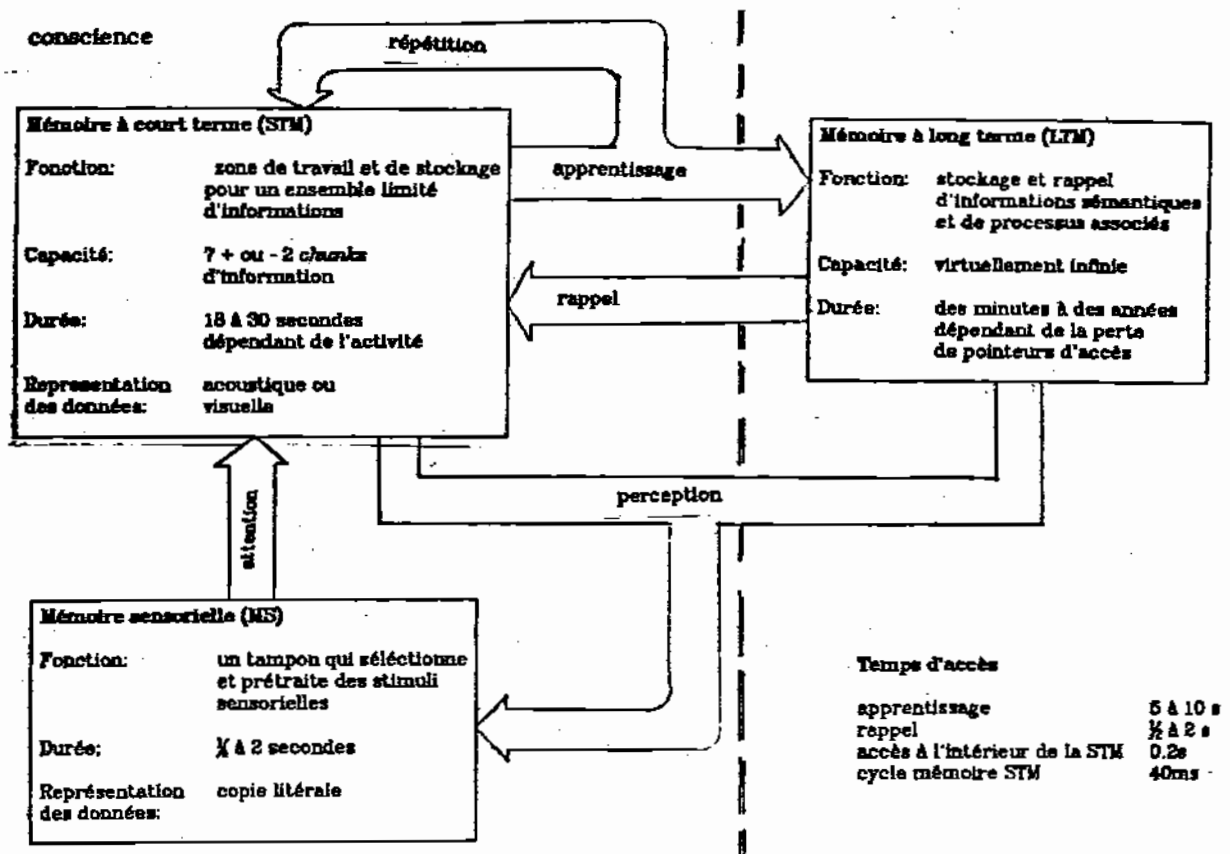


Figure 1. organisation et flux d'information dans la mémoire

l'information : c'est essentiellement une sorte de *tampon* connectant les organes de perception et le cerveau. La majorité de l'activité mentale se fait à travers la mémoire à court terme. C'est cette mémoire qui nous intéresse ici particulièrement. Si la mémoire intermédiaire est mentionnée, c'est principalement en terme de mémoire transitoire où les informations des dernières minutes à dernières heures sont traitées. C'est une sorte de *préprocesseur* du mémoire à long terme.

Pour Broadbent<sup>10</sup> le système nerveux est une structure à un canal d'entrée, qui applique un filtre pour sélectionner de l'information entrant, celle qui semble relevante pour l'activité cognitive en cours. D'après lui, la sélection est basée exclusivement sur les caractéristiques des entrées sensorielles. La mémoire à court terme est alors nécessaire pour garder temporairement les entrées sensorielles qui ne sont pas encore passées à travers les filtres. Bien que cette théorie de Broadbent soit maintenant très critiquée, à cause de la supposition d'un canal unique, elle représente historiquement le départ de toute la recherche de la psychologie cognitive.

10. [Broadbent58]

Naturellement, la performance de chacun de ces différents types de mémoires n'est pas la seule caractéristique significative. D'autres questions se posent : 'quel est le type d'informations gardées?', 'quelles sont les modes de représentation des informations?', 'quels sont les modes d'accès?' et 'quels sont les modes de stockage?'. Avec le progrès des recherches dans ces domaines, il devient de plus en plus évident que les mécanismes sensoriels sont dépendants du contenu et des besoins des processus actifs du cerveau. C'est en contraste claire avec la vue populaire que la perception est un processus passif qui ne fait que recevoir les données entrant du monde extérieur. La reconnaissance de cette caractéristique a des implications profondes sur les théories cognitives et les recherches sur les mécanismes physiologiques et psychologiques sous-jacents.<sup>11</sup>

La recherche psychologique sur la mémoire se centrait originalement sur les mécanismes de *rappel*, ignorant des processus mentaux plus complexes. Le model sous-jacent aux expérimentations considérait la mémoire à court terme comme la structure psycho-physiologique lieu central de la plupart du traitement mental des informations. Elle était également considérée comme le médiateur entre mémoire sensorielle et mémoire à long terme : elle organise et sélectionne les informations venant de la première et demandées pour stockage par la dernière.<sup>12</sup> Ces théories expriment un image relativement simple des activités de la mémoire sensorielle et de la mémoire à long terme, qui servent alors principalement à donner du support à l'activité de la mémoire à court terme. Un exemple de ce rôle de support est la manière par laquelle une demande de rappel associative et l'activité en *temps partagé*, participent à l'activité cognitive principale, ainsi quand on se rappelle un nom ou quand on resoud un problème longtemps après l'instant auquel on avait emis la demande, à un temps où on a déjà abandonner l'effort de résolution. Ce schema correspond bien à la conception intuitive de la conscience ou de la *pensée* en tant que flux des perceptions, des idées et des souvenirs traversant le cerveau.

11. [Gregory66, Norman76a]

12. Ceci implique une théorie cognitive où la mémoire à long terme joue un rôle tout à fait actif : elle ne sera plus qu'un simple endroit de stockage d'informations, mais elle sera aussi un processeur indépendant, interagissant activement avec la mémoire à court terme et la mémoire sensorielle. Ce point de vue qui considère les différentes mémoires comme des *acteurs* communiquant entre eux, est défendu, par exemple, dans [Minsky82].

### 3.2 limitations de la capacité de la mémoire à court terme

La recherche contemporaine en psychologie cognitive a élargi le domaine restreint de la psychologie traditionnelle en s'intéressant plus au *traitement* de l'information qu'à son *rappel* : la capacité de la mémoire à court terme va beaucoup plus loin que le simple rappel d'informations immédiates. Le *rappel* concerne des questions d'absorption, de rétention et de reproduction d'informations à l'intérieur d'un laps de temps très court. Parfois on a besoin de retenir une donnée un très court instant, comme par exemple, lorsqu'on regarde un numéro de téléphone pour appeler quelqu'un. L'activité pourtant de la mémoire à court terme dépasse de très loin cette simple rétention de la donnée : l'information qu'elle retient livre le matériel de base pour l'activité cognitive centrale du cerveau. La fonction de rétention d'une information donne le matériel de base pour d'autres processus et sert de stockage temporaire pour des résultats intermédiaires.

Une bonne analogie pour l'activité cognitive de l'humain est le fonctionnement d'un ordinateur : l'équivalent de la mémoire à court terme serait l'ensemble des registres de l'ordinateur, qui contiennent soit les opérandes d'une opération, soit des résultats partiels supposés nécessaires dans une courte période de temps. La grande masse de données qui se trouve dans la mémoire à long terme correspond à la masse de données des mémoires périphériques (disques, bandes, cassettes). La mémoire centrale de l'ordinateur correspond à la mémoire intermédiaire de l'humain, et à la mémoire sensorielle correspondent les *tamppons* qui font l'interface entre les périphériques d'entrées et la mémoire centrale. Le point essentiel de cette analogie est qu'il existe un endroit central où la plus grande part de l'activité a lieu (les registres) et que

1. cet endroit est limité dans la grandeur (à cause des considérations techniques et économiques)
2. les données sur lesquelles ces registres travaillent viennent soit des périphériques d'entrées, soit de la mémoire centrale, soit qu'elles sont des résultats de manipulations d'autres données qui se trouvent à cet endroit.

Les registres sont faits d'une technologie de mémoire à accès extrêmement rapide. A cause de leur prix, leur consommation de puissance ainsi que leurs caractéristiques thermiques et électromagnétiques, le nombre de tels registres est très limité. On trouve typiquement de l'ordre de 8 à 16 registres, plus quelques registres cachés, tel que le registre adresse mémoire, le registre données mémoires ou le compteur ordinal. Par contre, l'architecture des ordinateurs est grandement simplifiée par le fait que tout calcul n'a lieu que dans ces registres.

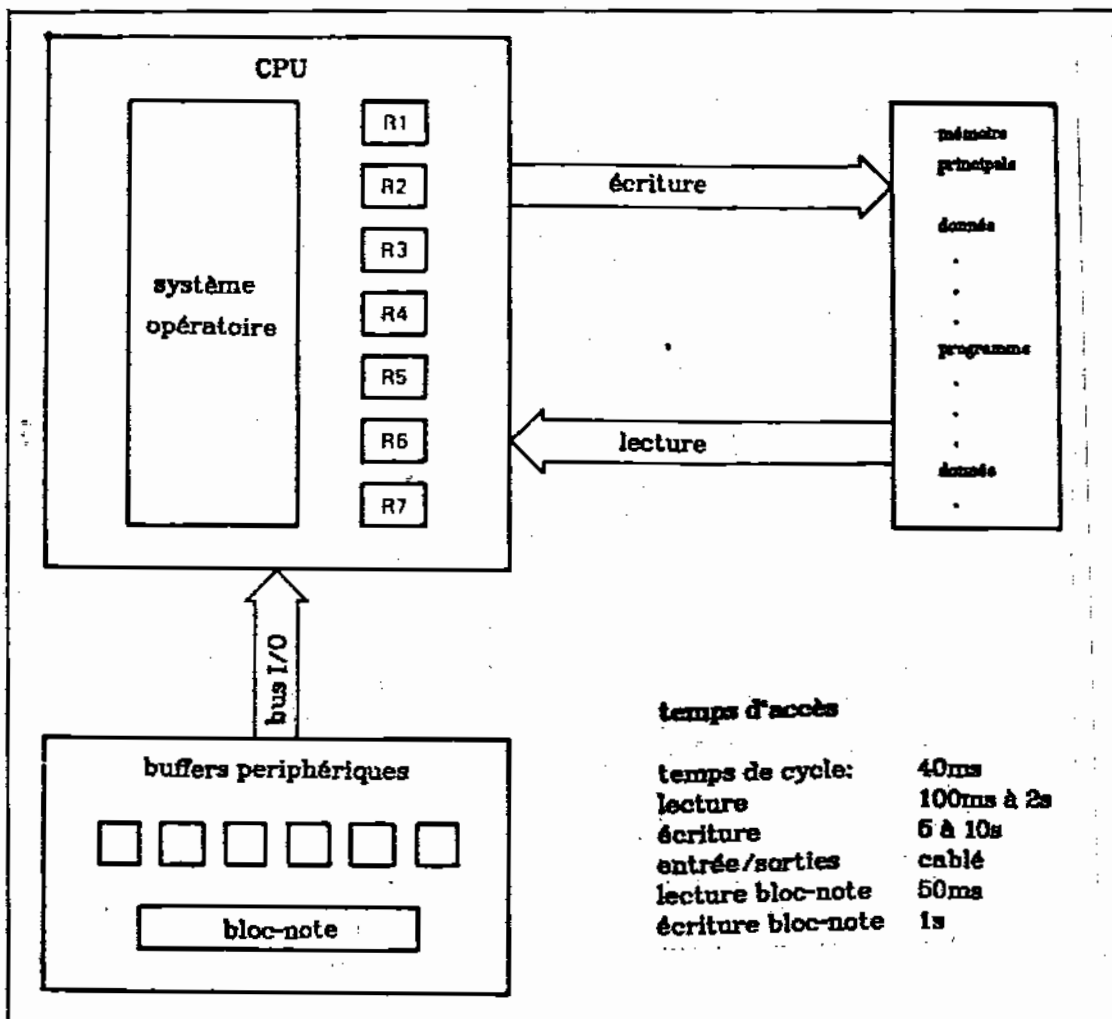


Figure 2. l'ordinateur cognitif

Toute activité de l'ordinateur utilise ces registres : les opérations arithmétiques et logiques, les transferts d'informations d'un mot de la mémoire centrale vers un autre, les transferts d'informations de la mémoire centrale vers des périphériques et vice versa. Evidemment, l'efficacité de la machine dépend du *temps d'accès* de ces registres.

Cette analogie peut nous aider à éclaircir notre discussion sur les limitations de la mémoire à court terme : si, en moyenne, un ordinateur a autour de 15 registres, il n'existe évidemment pas une possibilité de garder simultanément 50 nombres dans les registres. Mais il n'est pas plus coûteux de garder un 'petit' entier dans un registre qu'il en est d'y garder un grand nombre (pourvu qu'il ne produit pas de débordement), et il ne sera pas plus difficile de stocker un pointeur vers un caractère qu'il sera d'y stocker un pointeur vers un mot de 5 caractères. Bien que ceci a l'air trivial et évident, les conséquences vont très loin : George Miller<sup>13</sup> a déterminé la limitation quantitative de notre mémoire à court

terme à environ sept symboles. Il a également découvert qu'en combinant plusieurs symboles dans un item d'ordre supérieur (on peut par exemple représenté un nombre binaire de 4 digits binaires comme un simple digit hexadecimal) on peut augmenter la masse d'information totale de la mémoire à court terme, mais on ne peut jamais augmenter le nombre de symboles individuels possibles à stocker dans cette mémoire.

Ce qui est important ici n'est pas le nombre exact de symboles possibles à l'intérieur de cette mémoire, mais qu'il existe une limite très basse concernant ce nombre. Sans combiner des informations de niveau élémentaire en des informations compactes de niveau supérieur, nous ne sommes pas capable de traiter des grandes masses d'informations complexes (tel que nous le faisons tous les jours). Prenons un exemple : pratiquement personne peut correctement répéter la séquence 925634329462516. Cette séquence n'a que 15 chiffres, et pourtant, à l'instant où on entend la dernière, on a oublié la première. Par contre, les séquences 12345678987654321 et 111222333444555666777888, bien qu'ils se composent de plus de chiffres, sont aisées à retenir. Ce phénomène pourrait être expliqué informellement en disant que les chiffres peuvent être groupés simplement en deux structures de style : "de 1 à 9 et ensuite à l'invers", ou "de 1 à 8, chacun trois fois". Ces structures plus compactes ne prendraient que deux emplacements à l'intérieur de la mémoire à court terme, pendant que la première structure, où il n'y a pas de possibilité de description compacte, prendraient 15 emplacements, ce qui, d'après nos connaissances, est un nombre de symboles largement trop haut pour notre mémoire limitée.

Cette limitation quantitative conditionne évidemment la manière dont notre cerveau fonctionne : en permanence nous devons trouver des stratégies utilisant de manière effective le nombre limité de *registres*, et nous avons en permanence besoin d'interaction entre mémoire à court terme et mémoire à long terme. Toute réflexion sur un langage et un environnement de programmation doit considérer le problème de la limitation quantitative des ressources comme un critère constant de l'utilité de tel ou tel mécanisme d'un environnement de programmation.

Notons finalement, que cette capacité de compacter des informations élémentaires dans des structures conceptuelles complexes, qui se fait (chez nous) de manière spontanée à partir d'une certaine limite quantitative des informations recues, doit obligatoirement se faire également dans les outils de construction, d'observations et d'analyse de programmes : un environnement de programmation est *puissant* et peut constituer une *aide* efficace, dans la mesure où il pré-complexifie (ou aide à pré-complexifier) l'information de (pour) l'utilisateur, tel qu'il lui permette de garder dans sa mémoire à court terme un image complet de ce qui se passe.

---

#### 4. REFERENCES

- [Anderson73] J. R. Anderson et G. H. Bower, *Human Associative Memory*, Winston, Washington (1973).
- [Broadbent58] D. E. Broadbent, *Perception and Communication*, Pergamon Press, London (1958).
- [Chomsky57] N. Chomsky, *Syntactic Structures*, Mouton, The Hague (1957).
- [Chomsky65] N. Chomsky, *Aspects of the Theory of Syntax*, M.I.T. Press, Cambridge, Mass (1965).
- [Cofer76] C. Cofer, *The Structure of Human Memory*, Freeman, San Francisco (1976).
- [Dijkstra72] E. W. Dijkstra, "The humble programmer," *Communications of the ACM* 15(10) pp. 859-866 (Oct. 1972).
- [Gregory66] R. L. Gregory, *Eye and Brain*, McGraw-Hill Book Company, New York (1966).
- [Kilpatrick69] J. Kilpatrick et I. Wirszup, *Soviet Studies in the Psychology of Learning and Teaching Mathematics*, Vol. 1-4, University of Chicago Press (1969).
- [Kintsch74] W. Kintsch, *The Representation of Meaning in Memory*, Lawrence Erlbaum Associates, Hillsdale, N.J. (1974).
- [Lompscher76] J. Lompscher, *Verlaufsqualitäten der geistigen Tätigkeit*, Volk+Wissen, Volkseigener Verlag, Berlin (1976).
- [McCarthy56] J. McCarthy, "The Inversion of functions defined by Turing Machines," *Automata Studies, Annals of Mathematical Studies*, (34) pp. 177-181 Princeton University, (1956).
- [Miller56] G. A. Miller, "The Magical Number Seven, Plus or Minus Two : Some Limits on Our Capacity for Processing Information," *Psychological Review*, (63) pp. 81-97 (1956).
- [Minsky82] M. Minsky, *Learning Meaning*, M.I.T. Artificial Intelligence Laboratory, Cambridge, Mass. (July 1982). Draft
- [Newell72] A. Newell et H. Simon, *Human Problem Solving*, Prentice Hall, Inc., Englewood Cliff, N.J. (1972).

- [Norman75] D. A. Norman, D. E. Rumelhart, et the LNR Research Group, *Explorations in Cognition*, Freeman, San Francisco (1975).
- [Norman76] D. A. Norman et D. G. Bobrow, *On the Role of Active Memory Processes in Perception and Cognition*, Freeman, San Francisco (1976).
- [Norman76a] D. A. Norman, *Memory and Attention*, John Wiley and Sons, New York (1976). 2nd edition
- [Platt65] J. R. Platt, *New Views of the Nature of Man*, University of Chicago Press (1965).
- [Polya57] G. Polya, *How To Solve It*, Doubleday-Anchor, Garden City, L.I. (1957).
- [Sperry82] R. W. Sperry, *Science and Moral Priority : Merging Mind, Brain, and Human Values*, Columbia University Press, New York, NY (1982).
- [Sperry82a] R. W. Sperry, "Some Effects of Disconnecting the Cerebral Hemispheres," *Science* 217(36) pp. 1223-1226 (Sept. 1982). Nobel Lecture 8. Dec. 1981
- [Wertz78] H. Wertz, F. Mathieu, et D. Perolat, *L'Experience d'Arc et Senans*, Département Informatique, Université Paris 8 - Vincennes (Novembre 1978).
- [Wertz81] H. Wertz, "l'Utilisation de l'Outil Informatique - Pilotage et Programmation," pp. 177-194 dans *l'éducation et l'informatique dans la société, rapport au Président de la République*, Vol. annexes 1 : les voies de développement, ed. J. C. Simon, La Documentation Française, (1981).

\$LIST\$