ACQUISITION DE STRUCTURES CONDITIONNELLES :

effet des prérequis logiques et des représentations du dispositif informatique.

J. ROGALSKI

This paper presents some issues concerning the interaction between prerequisites in logic and mental representations of the "informatical device" in acquisition of conditional structures in programming. A study was conduced with 10th grade students (15-16 years old), in the phase of "computer literacy". It is shown that knownledge in logic is necessary but not suficient to insure acquisition. A subject's model based on a specific form of "presupposition" (the PRES-model) may explain why students get difficulties in managing the communication with an informatical device and dont use efficiently their logical knowledge.

INTRODUCTION

De nombreux éléments interagissent dans le processus d'acquisition de concepts informatiques. Tout d'abord les connaissances préalables que les étudiants ou élèves ont en d'autres domaines peuvent servir de **précurseurs** pour des notions informatiques. Ces précurseurs sont des notions générales, comme celles de la logique, ou des connaissances mathématiques, ou des connaissances "métacognitives", sur l'approche dans la résolution d'un problème, ou la planification de l'action.

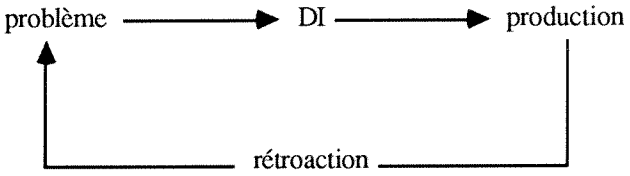
Ces précurseurs jouent un rôle producteur de sens pour de nouvelles acquisitions ; ils jouent aussi éventuellement un rôle réducteur dans la mesure où les élèves risquent de transposer dans le nouveau domaine de connaissance des propriétés de ces précurseurs qui sont en fait typiques de leur domaine d'origine.

© *Annales de Didactique et de Sciences Cognitives*
1 (1988) (p.131-152) IREM de Strasbourg

De tels effets producteurs et réducteurs dépendent également des acquis des élèves et de leur cursus (Rogalski, 1987a; 1987b).
Les caractères intrinsèques au champ de savoir de l'informatique jouent leur rôle propre dans les difficultés voire les obstacles épistémologiques dans l'acquisition. Les concepts informatiques, leurs représentations symboliques, leurs relations constituent en effet un nouveau champ conceptuel ; les élèves doivent construire de nouvelles représentations mentales adaptées à ce champ conceptuel (Rogalski et Samurçay, 1986).
Enfin, les représentations que les élèves (étudiants) se construisent sur le dispositif informatique lui-même jouent un rôle particulier dans les acquisitions: en effet un programme informatique se réalise toujours à travers un dispositif qui d'une part a des propriétés quant à son fonctionement lors de l'exécution du programme et d'autre part possède des propriétés quant à la communication avec l'utilisateur du programme (et le programmeur lui-même). L'utilisation de diverses réalisations d'un dispositif programmable a été étudiée par Cohors-Fresenborg (Cohors-Fresenborg, 1984; 1986). (On peut trouver une revue des recherches sur cette question dans Rogalski, 1986).
Notre hypothèse initiale est qu'il existe une forte interaction entre les acquis antérieurs, les éléments du champ conceptuel informatique considérés et les représentations mentales sur le dispositif informatique.
Nous avons centré plus précisément certaines de nos recherches sur les aspects de l'interaction due à l'existence de ce dispositif informatique. Nous avons travaillé avec des élèves en phase d'alphabétisation, c'est à dire la phase des 30 à 50 premières heures d'enseignement. Nous allons présenter des observations qui concernent les 10-15 premières heures: on observe que des élèves "décrochent" complètement dans cette phase d'initiation. Si on veut se préoccuper de savoir comment organiser un enseignement qui ne sélectionne pas excessivement il est important de maîtriser les phénomènes didactiques de cette première période.
Les résultats qui suivent concernent des élèves de seconde des lycées qui suivent un enseignement d'informatique dans un cadre scolaire normal. Il s'agit de l'option informatique dans le second cycle des lycées, qui est un cours institutionnel comme les autres options, avec les contraintes de présence d'un programme, et à partir de 1988 d'une épreuve facultative au baccalauréat. Cet enseignement comporte en particulier une initiation aux concepts de la programmation, avec les structures conditionnelles et les structures itératives lors de la première année. (Nous avons également observé des adultes au-delà de la formation secondaire et vérifié que cette période d'alphabétisation était un moment crucial pour certains).

1. Intervention du Dispositif Informatique: DI

On peut schématiser comme suit le traitement d'un problème quelconque au travers d'un dispositif informatique:

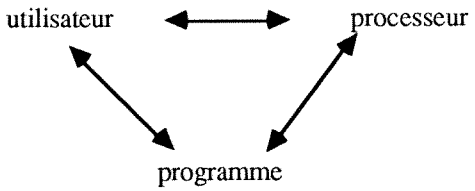


La solution du problème doit être réalisée au travers du DI (dispositif matériel, système d'exploitation, logiciels utilisés, interfaces : clavier, écran , souris etc..).
Lors de l'exécution l'élève reçoit des informations sur sa production: effets attendus et divergences ou échecs. Ces informations contiennent d'une part des éléments tenant à la qualité "logique" de la solution qu'il a proposée pour le problème, et d'autre part des éléments qui relèvent de l'adaptation de la forme de la solution aux contraintes du dispositf informatique.
Lors de l'exécution d'un programme l'élève doit donc mettre en relation les effets obtenus avec la signification qu'il attribue au texte du programme, en utilisant les représentations qu'il se fait (même implicitement) sur la manière dont le dispositif informatique "traite" le texte du programme.
Une conséquence majeure est qu'il faut introduire dans les analyses une différence entre la formalisation (ou la modélisation) informatique et la formalisation mathématique qui peuvent correspondre à un même problème. L'existence du dispositif informatique oblige en effet à considérer une unité constituée par l'algorithme (mathématique) et sa réalisation effective dans un environnement informatique donné.

2. Quelques résultats sur les représentations du dispositif informatique.

Les représentations sur le dispositif informatique peuvent concerner des niveaux multiples: des propriétés fonctionnelles de la communication avec le dispositif informatique aux caractères particuliers des représentations de données en mémoire..

Nous allons ici donner des résultats sur les représentations qu'ont les élèves d'une part de l'exécution d'un programme par un dispositif informatique et d'autre part de la place qu'occupent respectivement l'utilisateur et le processeur lors de cette exécution.



Nous avons analysé des productions d'élèves (écriture et complétion de programmes) après une quinzaine d'heures de programmation (élèves de seconde). Trois grands niveaux de représentation du DI peuvent être dégagé à l'analyse des observations:

- Au niveau 0 un programme n'est pas conçu par l'élève comme une séquence d'instructions destinée à faire produire par un dispositif une classe de résultats pour une classe de données (avec les langages dits impératifs). Cela se traduit en particulier par des difficultés importantes à assimiler les contraintes syntaxiques nécessaires à la communication avec le DI. Par exemple on peut trouver à ce niveau des expressions comme "quand a>0 écrire racine de a", sans aucune prise en compte de l'existence d'une syntaxe spécifique. Ce niveau semble très fugace ; toutefois nous avons trouvé dans une de nos recherches 4 cas de ce niveau sur 44 élèves après 15 heures de programmation.

- Le niveau 1 correspond à une confusion entre les instructions tournées vers l'ordinateur et les actions de l'utilisateur: par exemple, au lieu d'écrire en LSE[1] l'instruction TERMINER qui indique la fin de l'exécution, des élèves écrivent *AFFICHER 'TERMINER'* qui produit une sortie sur écran à destination de l'utilisateur. Autre exemple, au lieu de ALLER EN 10 (où 10 est un numéro de ligne) on trouvera *AFFICHER 'ALLER EN 10'*, ce qui indique que l'élève délègue à l'utilisateur la fonction d'aller se placer au bon endroit dans le texte du programme au lieu de donner l'instruction à l'ordinateur.

- Le niveau 2 correspond à l'attribution à l'ordinateur des propriétés de compréhension de l'utilisateur. Un exemple classique relevant de ce niveau est l'écriture d'une instruction du

type *AFFICHER 'le carré de x est x*x'* : la lecture que fera l'utilisateur et le calcul d'une expression qu'exécutera l'ordinateur sont mis sur le même plan.

Nous allons donner un exemple de comportement observé de niveau 2. On présente aux élèves le texte d'un programme dont les dernières instructions sont les suivantes:

```
AFFICHER 'voulez-vous continuer ? oui/non'
LIRE rep
SI rep='oui' ALORS ALLER EN 3
TERMINER
```

et on leur demande "ce qui se passe si l'utilisateur répond BOF".

Avec des représentations de niveau 2 les élèves attribuent à l'ordinateur des propriétés de compréhension de l'opérateur humain. Ils peuvent alors considérer que les seules réponses attendues par l'ordinateur sont OUI ou NON. Sur 56 élèves de seconde, testés après environ 15 heures de programmation, 28 donnent des réponses du type "l'ordinateur va se bloquer", ou "il va afficher un ?"; certains justifient explicitement leur réponse par "il attend OUI ou NON, comme il reçoit autre chose il ne sait pas quoi faire".

Des résultats analogues ont été relevés par Laborde et al, dans une étude expérimentale de l'acquisition de l'itération (Laborde, Balacheff et Méjias, 1985).
Tous les élèves qui ont produit, dans une autre tâche, des erreurs relevant de représentations de niveau 1 ont également fait ce type d'erreurs de niveau 2. Cela nous conduit à supposer qu'il y a une hiérarchie de niveau de représentation. (Nous avons retrouvé des erreurs de type 2 avec des adultes après plus de 50 heures de programmation, sur des problèmes de programmation plus difficiles: l'analyse des niveaux de représentation doit donc prendre en compte la complexité propre des programmes à produire).

3. Structures conditionnelles et représentations du dispositif informatique.

Nous allons aborder maintenant le point plus spécifique des effets produits par ces représentations du dispositif informatique sur l'acquisition des structures conditionnelles. Nous nous intéresserons essentiellement aux représentations de la séquentialité d'un programme. Nous mettrons ces représentations des élèves en relation avec leurs connaissances logiques.

En effet l'acquisition des structures conditionnelles pose d'une part le problème des rapports avec les acquis en logique (combinatoire et négation), d'autre part le problème de traitements qui lors de l'exécution ne coïncident pas avec l'ordre du texte du programme, puisque selon le cas effectivement rencontré lors de l'exécution une branche seulement sera parcourue.

La première situation conditionnelle est celle de l'alternative simple complète:

SI <condition> ALORS <TRAITEMENT 1> SINON <TRAITEMENT 2>.

Lorsqu'il y a plus de deux cas à traiter, une seule alternative complète simple n'est pas suffisante. Trois types de structures sont possibles:

- *emboîtement* de SI..ALORS..SINON..
- utilisation du *saut conditionnel* (GOTO ou ALLER EN)
- structure de *cas*.

Nous allons donner un exemple de réalisation de ces trois structures pour le traitement de l'équation du second degré selon le signe du discriminant D. Nous notons T1 le traitement: calcul et affichage de deux racines distinctes réelles; T2 calcul et affichage d'une racine double; T3 affichage de la non-existence de racines réelles.

Emboîtement:

```
SI D>=0 ALORS SI D>0 ALORS T1
                    SINON T2
SINON T3
```

Saut conditionnel:

```
20 SI D<0 ALORS ALLER EN 50
30 SI D>0 ALORS T1 SINON T2
40 ALLER EN 60
50 T3
60 TERMINER.
```

- *Structure de cas :*

SI D>0 ALORS T1
SI D=0 ALORS T2
SI D<0 ALORS T3

Nous coderons par la suite ses différents styles respectivement par EMB, SAUT, CAS. Les difficultés relatives d'apprentissage et d'utilisation de ces structures ont été observées avec des adultes (Green, 1980; 1985; Sime, Green et Guest, 1973); on peut trouver dans Rogalski (Rogalski, à paraître) une brève revue sur les recherches concernant les structures conditionnelles.
Il faut ici rappeler une caractéristique commune au LSE et au BASIC: ce sont les langages comportant des lignes de programme, auxquelles on peut faire référence par leur numéro. Cela a des effets sur la manière dont on peut structurer les instructions pour gérer des conditions. Ces langages se prêtent particulièrement bien à l'étude des questions de séquentialité: en effet, la gestion des ALLER EN (ou des GOTO) est étroitement liée à la représentation de l'exécution.
Certains langages posent en termes différents la question de la séquentialité et des structures conditionnelles; ainsi dans un langage comme PASCAL le texte des programmes n'est pas organisé en lignes, et la structuration du langage permet d'éviter l'usage du GOTO. La rencontre avec la séquentialité de l'exécution dans l'alphabétisation se pose a priori de manière différente qu'avec BASIC ou LSE. Les langages de type "applicatif" (comme LOGO, LISP) ou "logique" (PROLOG) proposent un traitement notablement différent des conditions; en particulier la question des définitions conditionnelles de fonctions diffère conceptuellement de celle des instructions conditionnelles; il en est de même de la condition "implicative" en PROLOG. Les résultats et les hypothèses qui suivent ne sont donc pas directement transposables pour de tels langages; les méthodes d'étude ne sont pas non plus directement utilisables.

4.Quelques problèmes de méthode.

Nous nous sommes placées en situation de "recherche de terrain" au sens suivant: toutes nos observations ont été faites sur des problèmes posés par des enseignants dans le cadre de leur enseignement: ces problèmes ont donc été définis en fonction des objectifs pédagogiques des enseignants. Par ailleurs, nous ne sommes pas intervenues sur cet enseignement

mais nous avons essayé d'en tenir une chronique pour définir un cadre général de réactions d'élèves face à un ensemble de situations d'enseignement.
L'observation de phénomènes de classe permet de repérer l'existence d'indices significatifs: il s'agit de comportements d'élèves qu'on voit apparaître dans telle occasion, qu'on peut interpréter dans les termes de notre problèmatique. Il peut s'agir d'erreurs, de questions, de discussions entre élèves: on relève dans la mesure du possible les conditions dans lesquelles ces indices sont apparus. Ce sont des éléments pour construire ensuite des expériences (dans la mesure où celles-ci sont compatibles avec un déroulement normal de la classe). Soulignons qu'il n'est pas toujours possible de provoquer certains de ces comportements dans des situations d'interrogation de type expérimental et que les observations "in situ" ne peuvent pas toujours se ramener à une expérimentation.
Une question ouverte est de déterminer la dépendance des observations par rapport à la séquence didactique dans laquelle elles ont eu lieu (par séquence didactique nous entendons l'organisation de situations didactiques sur le moyen et le long terme).
Nous avons toutefois effectué aussi des constructions de problèmes à visée expérimentale. Cela est évidemment l'objet d'une négociation avec les enseignants, parce que de tels problèmes n'apparaissent pas toujours directement pertinents par rapport à l'enseignement; leur complexité ne répond pas toujours au contrat entre les enseignants et les élèves (problèmes trop "simples", ou au contraire trop peu "familiers" dans leur présentation).
Dans le cas de l'étude des structures conditionnelles nous avons utilisé plusieurs types de situations-problèmes: complétion de programmes dans lesquels il manque soit des lignes complètes isolées soit un morceau significatif; écriture d'un programme complet; explicitation de la classification des cas à traiter par des structures conditionnelles.
Il faut souligner que les problèmes d'écriture de programmes sont difficiles à analyser. Lorsque les élèves ne produisent pas de texte de programme on n'a plus aucune information sur leurs difficultés et leurs acquis. De plus un programme de quelques lignes, avec une seule structure de contrôle et une structure de données simple, peut être difficile pour des débutants : sur une production brève il peut s'avérer difficile d'inférer ce qui produit un programme erroné. Des tâches de complétion où l'élève doit remplir un "blanc" avec un test, un traitement, une instruction conditionnelle complète constituent un moyen d'exploration des acquis fournissant davantage d'informations directes. L'expérimentateur choisit la place du fragment de programme que les élèves doivent produire: cela limite le contenu et la forme possible des réponses, et facilite l'analyse de difficultés "locales". Toutefois de telles tâches permettent plus difficilement d'étudier toute la gamme des difficultés conceptuelles auxquelles on peut s'attendre dans les acquisitions en programmation (2). Les observations

"cliniques " des processus mêmes de réponse de élèves sont d'autant plus indispensables qu'on cherche à analyser des acquisitions complexes.

5.Les élèves et les situations-problèmes.

Les élèves sont en seconde de l'option informatique des lycées (2 classes). Ils ont d'abord rencontré dans l'enseignement l'alternative simple complète comme premier exemple de structure conditionnelle. L'enseignant leur a présenté des exemples de problèmes résolus avec la forme de conditionnelle:

SI <condition> ALORS <TRAITEMENT 1> SINON <TRAITEMENT 2>

Ils ont fait des exercices en classe sur cette forme de conditionnelle, avec un passage effectif de leur programme sur l'ordinateur.
D'autre part ils ont vu la conditionnelle incomplète:

SI <CONDITION> ALORS <TRAITEMENT 1>

dans laquelle il faut tenir compte de la séquentialité pour contrôler le traitement du cas où la condition n'est pas remplie.
Les problèmes choisis devaient permettre de voir des interactions entre acquisitions logiques, séquentialité et structures conditionnelles: il faut pour cela avoir plus de deux cas à traiter, sinon il s'agit d'une application pure et simple des structures dans leur forme apprise; dans ce cas l'usage de l'alternative conditionnelle conduit à des solutions correctes même si les élèves ne maîtrisent pas certains problèmes logiques, ou certains problèmes de représentation de la séquentialité.
Par ailleurs pour rencontrer des problèmes de précurseurs de combinatoire et/ou de logique, il faut avoir un minimum de cas à combiner, et avoir à effectuer des opérations logiques. Toutes les situations-problèmes posées aux élèves comportent donc au moins trois cas à traiter.

Ces situations sont les suivantes:

- **problème 1** : il s'agit de traiter trois cas avec trois traitements associés. Les cas correspondent à des moyennes à un examen, et les traitements sont les affichages appropriés selon les cas ("reçu", "collé", "oral"). La situation est simple, sauf le fait qu'il faut traiter les relations >= et <= et leur négation ; or on peut anticiper des problèmes de négation portant sur le traitement des inégalités. En effet une inégalité large comme >= peut être considérée comme une relation de type disjonctif (non exclusif); or on sait par ailleurs que des difficultés existent dans la négation de relations conjonctives ou disjonctives.

- **problème 2** : il s'agit d'écrire un programme qui affiche si l'utilisateur peut ou non conduire une moto, en fonction de la donnée de son année de naissance et du fait que son anniversaire est passé ou non. Soulignons que les élèves connaissent tous la condition à remplir: avoir 16 ans révolus. Ils savent évidemment déterminer correctement si quelqu'un remplit ou non la condition: c'est un prérequis, relevant des connaissances familières, nécessaire pour que l'analyse suivante ait tout son sens.
  Il n'est pas inutile de s'assurer de tels prérequis: nous avons pu observer par exemple que les programmes avec des pourcentages étaient mal traités par de nombreux élèves qui ne connaissaient pas le passage de l'expression en % au calcul numérique..
  La situation 2 est plus complexe que la première puisque il y a a priori 6 cas possibles, ou 4 si on fait les tests les plus "économiques", et seulement 2 traitements. Il y a donc des cas à regrouper. Dans cette situation les questions de traitement des égalités et inégalités peuvent également se poser.
  Les élèves qui ont eu ce problème à traiter ont moins de difficultés en mathématique et en français que ceux auxquels a été posé le premier problème, plus simple. On peut espérer avoir ainsi partiellement compensé les différences de difficulté.

- **le troisième problème** aborde le problème de la coordination entre des activités de classement (relevant de la logique et des représentations symboliques déjà disponibles), des tâches de programmation portant sur les structures conditionnelles; il est accompagné d'une tâche de programmation destinée à apprécier relativement "directement" l'existence de problèmes de maîtrise de la séquentialité par les élèves.
  La dernière situation (utilisée pour étudier les rapports entre logique et programmation) porte sur les états de l'eau selon la température: les élèves connaissent globalement le fait qu'au-dessous de 0° l'eau est à l'état de glace, qu'entre 0° et 100° l'eau est à l'état de li-

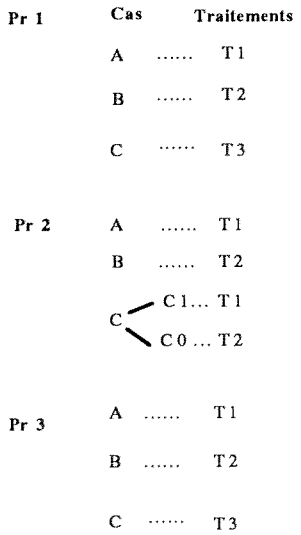
quide et qu'au-dessus de 100° l'eau est à l'état de gaz. On leur précise qu'à 0° l'état est la glace et à 100° la vapeur. Un des buts de cette précision est de marquer dans l'énoncé même du problème l'existence des cas d'égalité.
La tâche des élèves est de classer les cas possibles et de compléter le programme suivant:

```
10  LIRE T
20  ALORS AFFICHER 'liquide'
30  SI T > 100 ALORS AFFICHER 'vapeur'
40  ...
50  TERMINER
```

Ils ont donc d'une part à compléter une condition dans une ligne d'instruction connaissant le traitement (l'état de l'eau) et d'autre part à compléter toute une ligne de programme, correspondant à un traitement aisément déductible des informations précédentes sur les états de l'eau (puisqu'on en fournit 2 sur 3).

6.Analyse comparative a priori des traitements des situations-problèmes

Le problème 1 se prête très bien à un traitement sous forme de cas, tout comme le problème 3. Le problème 2 se prête mieux à une structure d'emboîtement. Les schémas ci-dessous résument la structure des trois situations et les structures conditionnelles qui leur correspondent le plus directement en LSE.



Dans le problème 2 il est possible mais peu économique de traiter les 6 cas a priori.

7. Résultats.

Le tableau qui suit (tableau 1) donne la répartition en pourcentage de la structure conditionnelle utilisée. Le nombre de sujets est de 20 pour le problème 1 et de 24 pour le problème 2, et les pourcentages ne sont donnés que pour faciliter les comparaisons.
Le codage est le suivant:

EMB: emboîtement de conditions
CAS: structure de cas
SIGN: problème de signification d'un texte de programme

TABLEAU 1

	Pr 1	Pr 2	Total
CAS	35 (25)	46 (17)	41 (20)
EMB	25 (5)	36 (8)	32 (7)
SIGN	10	8	9
∅	30	8	18

(Les nombres entre parenthèses donnent les pourcentages de réponses correctes au sens suivant: la logique des programme est correcte même s'il peut y avoir des erreurs de syntaxe).(3)
La faiblesse des élèves du premier groupe se manifeste surtout par le nombre relativement important de ceux qui ne produisent pas de programme. On constate par ailleurs que quelques élèves n'attribuent pas de signification à la notion de programme (ligne SIGN).
On remarque qu'il n'y a pas une différence significative entre les problèmes quant à l'utilisation des structures de cas et d'emboîtement: les élèves adaptent assez peu la structure du programme à la nature des situations traitées, et à l'organisation des cas.
On observe toutefois une différence entre le traitement par cas et le traitement par emboîtement: les réussites dans l'écriture des programmes sont liées essentiellement à l'utilisation de la structure de cas. Il semble plus facile de traiter les situations de cas, qui ne nécessitent pas la maîtrise de la séquentialité que l'emboîtement de conditions.
Nous retrouverons des résultats obtenus par des auteurs travaillant avec des adultes sur le

fait que la structure de cas a des propriétés facilitatrices; c'est le cas en particulier lorsqu'il existe une relation univoque entre les cas et les traitements (cette relation permet en effet de retrouver aussi bien les informations séquentielles que les inforamtions conditionnelles) (Sime et al; Green, op. cit.).

8. Relations entre logique et conditionnelles.

Une première analyse concerne le problème 1: nous avons déjà signalé plus haut que les élèves doivent travailler sur les relations d'égalité et d'inégalité avec les 3 cas: *M>=10, M<10 et M>=8, M<8*. Leur expression ne pose pas de problème majeur aux élèves : tous trouvent rapidement que *8≤M<10* doit se formuler *M>=8 et M<10* pour être compatible avec la syntaxe de LSE (les précurseurs d'écritures algébriques symboliques sont bien utilisés). En revanche lorsqu'il s'agit de traiter le cas *nonA* (où *A* est *M<8* par exemple) on constate des difficultés à faire apparaître l'égalité. Beaucoup d'élèves prennent comme négation de la relation d'inégalité stricte l'autre relation d'inégalité stricte: le cas d'égalité n'est pas pris en compte. Dans les tests du problème 1 nous avons ainsi relevé que les 3/4 des élèves "oubliaient" les cas d'égalité.
S'agit-il seulement d'un problème de négation ou doit-on aussi considérer le problème des statuts respectifs de la relation d'inégalité et de la relation d'égalité? Notre hypothèse est qu'il existe un problème de fond sur la façon dont sont utilisées dans l'enseignement ces deux relations : les relations d'ordre n'y jouent pas le même rôle que les relations d'égalité. En effet les élèves rencontrent l'égalité très fréquemment et cela dans des situations de résolution d'équations; les situations de comparaison, d'encadrement ou d'approximation dans lesquelles interviennent les relations d'ordre sont beaucoup plus rares. Cela doit probablement contribuer à donner un statut très hétérogène aux égalités et inégalités; ces notions appartiendraient, par le type de problèmes dans lesquels elles interviennent, à des champs conceptuels peu mis en relation. Il n' y a toutefois pas, à notre connaissance, de recherche didactique sur cette question, et l'hypothèse faite est donc une conjecture ouverte. Nous avons également étudié les problèmes spécifiques de négation dans le problème 2. On peut trouver des élèves produisant des négations de type "non(A et B)" = *"nonA et nonB"*, où A et B sont des booléens obtenus à partir de comparaison d'âges et d'information sur l'anniversaire.

Nous avons également noté les erreurs de logique du programme proposé: combinatoire des cas fausse (non exhaustive ou non exclusive, indépendamment du traitement particulier éventuel de l'égalité). Le tableau II donne la répartition de ces deux types d'erreurs dans les deux problèmes. Les codages sont les suivants:

LOG: erreurs de logique (combinatoire non exhaustive ou non exclusive)
NEG: problèmes de négation de relations
Total désigne le nombre d'élèves ayant produit effectivement un texte de programme.

TABLEAU II

	LOG	NEG	Total
CAS	4	3	11
EMB	4	2	9
N	8	5	20

Dans une situation où la combinatoire des cas possibles reste relativement simple, on peut relever que pour beaucoup d'élèves (parmi ceux qui rédigent un programme complet) les prérequis logiques ne sont pas assurés: alors que les élèves connaissent le traitement de la situation réelle évoquée, l'interaction des opérations logiques avec la tâche nouvelle d'écriture de programme conduit à de nombreuses erreurs de négation ou de combinatoire.
Cela nous conduit à des commentaires sur la didactique des structures conditionnelles; quand les enseignants font des corrigés de programme ou donnent aux élèves des méthodes de "vérification" fondées sur des jeux d'essais, ils utilisent les cas bien écrits et bien regroupés si nécessaire; or les élèves vont regarder dans un jeu d'essais les cas sur la base desquels ils ont écrit leur programme: celui-ci ne va donc pas être mis en défaut. (Dans le problème 1 les élèves ont fait tourner leur programme avec différentes valeurs, aucun n'a choisi les cas d'égalité..). Il faut donc fournir aux élèves des jeux de cas qui puissent effectivement mettre leur programme en défaut, et à partir desquels ils obtiennent des informations productives en retour de l'exécution.
Il est également utile de s'interroger plus précisément sur la manière dont les élèves passent d'un savoir implicite ou même explicite à une formulation adéquate, avec un système de représentation qui assure exhaustivité et exclusivité. Cela permettrait de dégager dans les dif-

ficultés d'écriture de programme ce qui relève des prérequis et ce qui relève des acquisitions conceptuelles propres au champ informatique. Des recherches sur des adultes (étudiants non scientifiques) ont confirmé l'existence d'interaction entre les acquis en programmation et les prérequis mathématiques (van der Veer, van Beek et Gruts, 1986).

8. Le modèle PRES

Nous avons dégagé dans la manière dont les programmes étaient conçus, l'existence possible d'un "modèle" (ou un théorème en acte), que nous avons appelé le modèle PRES (comme présupposition) et qui est le suivant.
"Quand j'ai effectué une instruction de type
SI <condition A> ALORS <traitement 1>
je suis dans la situation <non A>. Réciproquement tant que je n'ai pas terminé le traitement de la condition A les instructions écrites continuent à s'y rapporter".
C'est en gros le fonctionnement des conditionnelles dans la communication courante en langue naturelle dans laquelle la présupposition joue un rôle très important.
Nous avons attribué à ce type de modèles des réponses du genre suivant:

10 Si *A* ALORS T1
20 T2

où T2 est le traitement qui doit être associé à *nonA*.(4)

et du type plus complexe suivant:

10 SI A ALORS T1 SINON I
20 SI B ALORS T2

où I est une instruction qui appelle une donnée complémentaire (sur l'anniversaire dans le problème 2) et où B est une condition sur cette donnée, conduisant au traitement T2 lorsqu'on est toujours dans la situation nonA (c'est à dire dans la portée du SINON).

L'existence de ce modèle PRES n'est pas repérable avec toutes les écritures de programmes; en particulier la structure de cas permet rarement de le voir apparaitre (sauf si la structure est incomplète), en revanche ce modèle est repérable dans les structures d'emboîtement. L'existence d'un tel modèle n'est pas repérable avec tous les langages. Ainsi, la possibilité de mettre un ensemble complexe d'instructions, avec des imbrications de IF..THEN..ELSE, après le THEN et le ELSE de la structure alternative en PASCAL, limite le recours éventuel au modèle PRES. C'est sans doute à un autre niveau de complexité que les problèmes de séquentialité peuvent se repérer en PASCAL.
Le tableau III donne la répartition (en nombre de sujets) des écritures de programme compatibles avec l'existence de ce modèle PRES (dans les textes de programmes où on pouvait les voir apparaître, c'est à dire ici essentiellement ceux qui utilisent des emboîtements).
Le codage est le suivant :

PRES : réponses compatibles avec l'existence du modèle PRES
CORR : réponses correctes.

TABLEAU III

	Pr 1	Pr 2	Total
PRES	5	6	11
CORR	1	2	3
Autres		1	1
N	6	9	15

Il n'y a pas de différences majeures entre les problèmes 1 et 2, dans lesquels on observe une présence importante d'erreurs compatibles avec l'existence du modèle PRES.
Dans le troisième problème, la nature des éléments à compléter permet d'observer des erreurs compatibles avec le modèle PRES. Ainsi le fait de mettre dans la condition de la ligne 20 seulement l'inégalité T>0, peut s'interpréter par le modèle PRES: le test serait correct si on était bien dans une situation excluant le cas traité dans la ligne 30. Le fait de ne pas indiquer de condition dans la complétion de la ligne 40 est également compatible avec la présence du modèle PRES.
Si on regroupe les deux types d'erreurs compatibles avec le modèle PRES on observe environ 40% des élèves pour lesquels on peut faire l'hypothèse de l'utilisation de ce modèle; il faut remarquer que cette utilisation n'est pas systématique: seul 1 élève fait des erreurs à la

fois pour la complètion des lignes 20 et 40, ce qui est compatible avec une utilisation permanente de ce modèle.
Cette remarque ne nous semble pas invalider l'existence du modèle PRES: pour de très nombreux contenus n'appartenant pas au domaine informatique on a pu en effet observer une instabilité des modèles erronés. Dans les questions de structures conditionnelles, il y a peu de raisons qu'un modèle de représentation s'instaure comme modèle stable, quand rien ne le renforce dans les éléments apportés par l'enseignement. Cela n'empêche pas pour autant un tel modèle de jouer un rôle d'obstacle à l'intériorisation d'un modèle stable conforme à la séquentialité d'exécution des programmes.
Une tâche de programmation posée aux mêmes élèves nous a permis d'observer d'autres effets de non représentation de la séquentialité, ou du moins d'une absence de maîtrise de la représentation appropriée. Les élèves ont eu à construire une partie de programme qui stoppe l'exécution si l'utilisateur répond NON à la question qui lui est posée par le programme et affichée à l'écran: "voulez-vous continuer ?".
Le tableau IV donne la répartition des réponses selon les catégories suivantes:
CONF dénote les confusions entre le rôle de l'ordinateur et celui de l'utilisateur dans l'exécution du programme,
Par exemple au lieu d'écrire SI rep='NON' ALORS TERMINER, avec une instruction à l'ordinateur, l'élève écrit SI rep='NON' ALORS AFFICHER 'terminer', avec un message à l'utilisateur;
SEQU dénote une absence de maîtrise de la séquentialité, ainsi l'élève peut écrire un texte du type suivant :

110 SI rep='NON' ALORS TERMINER
120 SI rep='OUI' ALORS AFFICHER 'on continue'; TERMINER

avec l'instruction de fin d'exécution juste après le traitement du cas dans lequel l'exécution est dite continuer..

SIGN dénote l'absence d'acquisition de la notion de programme
CORR dénote des réponses correctes
NEG caractérise ici des textes de programme pour lesquels le contraire de la réponse NON conduisant à l'arrêt de l'exécution est la réponse OUI (alors que la logique de fonctionnement correspondant au texte de problème est d'arrêter en cas de reponse négative et de continuer sinon, c'est à dire pour toute autre réponse). Il faut remarquer que ce type d'erreur re-

coupe partiellement des problèmes de représentation sur les rapports entre utilisateur et ordinateur: l'utilisateur étant implicitement supposé répondre comme on l'attend: NON pour arrêter et OUI pour continuer. La possibilité d'autres réponses n'est pas envisagée, ce qui correspond aux présupposés dans le langage usuel.
Cette tâche pose les problèmes méthodologiques généraux des écritures de programmes: si les élèves utilisent le schéma appris de l'alternative complète on ne peut voir apparaître aucune erreur de séquentialité.

TABLEAU IV

	CONF	SEQU	SIGN	CORR	Total
CORR	2	2	1	7	12
NEG	2	1		3	6
N	4	3	1	10	18

Enfin, la classification des cas proposés aux élèves comme tâche préalable à la complétion du programme sur les états de l'eau, est correctement résolue par la plupart des élèves. On retrouve dans ces tâches le problème du statut de l'égalité: 2 élèves ne la prennent pas en compte ni dans la classification, ni dans l'écriture du programme, tandis que 4 "l'oublient" dans la seule écriture du programme. Il faut ajouter que 3 élèves ne donnent pas une formulation synthétique pour la classification mais exhibent un ensemble de cas représentatifs des différentes situations, égalité comprise. La plupart des formulations proposées pour la classification sont des formulations en langage naturel, ou des mélanges de formulations "algébriques" telles que T<=100 et de formulations en langage naturel. Le support de représentations graphiques de type tableau ou arbre, ne semble pas disponible chez les élèves, alors qu'il constituerait une aide pour structurer le programme avec les conditionnelles appropriées.

CONCLUSION

L'ensemble des résultats obtenus conforte l'hypothèse initiale d'interaction entre des prérequis logiques (la maîtrise des négations et de la combinatoire des cas en ce qui concerne les structures conditionnelles), des représentations sur le fonctionnement du dispositif informatique (en particulier de la séquentialité de l'exécution) et l'acquisition des concepts informatiques (ici les structures conditionnelles).

Il est possible de supposer que des difficultés à maîtriser la séquentialité sont une des raisons des difficultés observées dans l'usage des structures conditionnelles de type GOTO; cependant que les structures de type CAS (le CASE OF en PASCAL par exemple) sont plus faciles à utiliser correctement dans la mesure où elles ne font pas appel à la même représentation de la séquentialité. Il faut toutefois souligner que les élèves doivent maîtriser la négation pour traiter si nécessaire le complément des cas exprimés dans le CASE OF.

De plus nous pouvons, au vu de certaines difficultés à transposer dans l'écriture d'un programme des connaissances par ailleurs acquises, faire l'hypothèse d'une difficulté spécifique de "formalisation", au sens où il faut pour écrire un programme s'astreindre à une formulation respectant des contraintes particulières de syntaxe et de sémantique dues à la nécessité de la communication avec un dispositif informatique.

Les observations faites dans les séances d'enseignement nous conduisent à l'hypothèse que les résultats obtenus pour les structures conditionnelles pourraient se retrouver avec d'autres notions informatiques, comme les stuctures itératives. De plus les autres éléments que la séquentialité d'exécution qui interviennent dans les représentations sur le dispositif informatique interagissent certainement aussi avec l'acquisition des concepts informatiques; cela nous apparaît tout particulièrement important pour les acquisitions concernant la structuration des données sur laquelle existent peu de recherches à l'heure actuelle. Nous pouvons citer ici l'effet des types de variables et de leur statut fonctionnel sur des activités de programmation (Rogalski et Samurçay, 1986).

L'enseignement des notions informatiques ne nous semble en conséquence pas pouvoir s'appuyer seulement sur l'enseignement de l'algorithmique, hors de l'interaction avec les contraintes propres d'un langage de programmation et de la confrontation avec le fonctionnement réel d'un dispositif informatique.

Notes

(1) Le langage LSE a été élaboré dans un but didactique. Les mots réservés du langage sont "français" (AFFICHER, LIRE, TERMINER; ALLER EN..); LSE permet l'écriture de procédures indépendantes.

(2) *Question: vous n'avez pas demandé d'écrire l'exécution d'un programme ?*
C'est une question qu'on s'est posée; la méthode consistant à faire exécuter "à la main" un programme a été utilisée par d'autres chercheurs; toutefois nous nous sommes aperçues (Colette Laborde et Béatrix Méjias avec un travail sur l'itération, moi-même au niveau de l'observation avec des élèves dans un contexte analogue) que les élèves exécutent le programme qu'ils voulaient écrire, et non pas toutjours celui qui est effectivement rédigé. Ainsi, au lieu d'exécuter ce qu'ils ont écrit, ils exécutent ce qu'ils voulaient avoir écrit
Question: mais s'ils n'ont pas écrit eux-mêmes le programme?
S'ils ne l'ont pas écrit c'est éventuellement plus facile mais ils peuvent aussi le lire de la manière dont ils l'auraient écrit (lorsqu'ils comprennent le programme) c'est à dire en lui donnant une autre signification, et ils peuvent donc donner une exécution "correcte" d'un programme "faux" parce qu'ils vont avoir des présupposés sur la signification des lignes écrites. Les observations doivent donc être de type "clinique" pour tirer des informations sur une telle tâche. Des chercheurs ont utilisé une technique voisine à celle que vous proposez dans l'étude des structures conditionnelles, en demandant au sujet soit de dire quel traitement vient après telle condition (information séquentielle) soit d'identifier quelle condition est remplie lorsqu'un traitement donné est effectué (information conditionnelle). Toutefois ils ont travaillé avec des adultes, étudiants en université de sciences humaines, dont on peut raisonnablement penser qu'ils n'ont pas de problèmes de formulation. La technique utilisée ne portait pas sur l'exécution proprement dite, et les auteurs ne s'interrogeaient pas sur des questions de représentations.
Certaines de nos tâches de complétion de programme ont des similitudes avec ces dernières tâches: lorsqu'on fait compléter un traitement on est dans la première situation (recherche d'information séquentielle); lorsqu'on fait compléter une condition on est dans la seconde situation (recherche d'information conditionnelle).

(3) L'étude de ces erreurs n'est pas notre propos; de plus des études effectuées avec des programmeurs débutants et des programmaeurs avertis montrent que les erreurs de syntaxe sont , en pourcentage ,assez faibles et du même ordre dans les deux populations; on peut supposer qu'il ne s'agit donc pas fondamentalement d'erreurs de nature conceptuelle.

(4)Il existe des langages où ce traitement d'une condition ne met pas en défaut le modèle PRES: en LOGO par exemple.

REFERENCES

COHORS-FRESENBORG E. (1984) On the representation of algorithmic concept. *Osnabrücker Schriften zue Mathematik.* Heft 76.

COHORS-FRESENBORG E. (1986) Empirische untersuchungen über verschiedene kognitive Struckturen bei algorithmischen Begriffs bildungsprocessen. *Actes du colloque fran co-allemand de didactique des mathématqiues et de l'informatique.* CIRM, Marseille-Luminy.

GREEN T.G.R. (1980) Ifs and Then : is nesting just for birds ? *Software pratice and experience.* 10, 371-381.

GREEN T.G.R. (1985) Design and use of programming languages (draft).

LABORDE C, BALACHEFF N, MEJIAS B (1985) Genèse des concepts d'itération : une approche expérimentale. *Enfance* 2-3, 223-239.

ROGALSKI J. (1986) Les représentations mentales du dispositif informatique. *Actes du colloque franco-allemand de didactique des mathématiques et de l'informatique.* CIRM Marseille-Luminy

ROGALSKI J. (1987a) Epistémologie génétique et didactique : pour une théorie de l'acquisition des connaissances complexes. *Actes du colloque SFP : "les apprentissages : perspectives actuelles"*. Université de Saint-Denis.

ROGALSKI J. (1987b) Acquisition de savoirs et savoir-faire en informatique. *Cahier de didactique des mathématiques* , 43, IREM, Université Paris VII.

ROGALSKI J. (à paraître) Acquisition et didactique des structures conditionnelles en programmation informatique. *Psychologie Française* (numéro spécial sur l'informatique dans l'enseignement).

ROGALSKI J., HÉ Y. (1987) Logic and mental representations of the informatical device in acquisition of conditional structures by 15-16 years old students. Draft.

ROGALSKI J., SAMURÇAY R. (1986) Les problèmes cognitifs rencontrés par des élèves de l'enseignement secondaire dans l'enseignement de l'informatique. *Journal Européen de Psychologie de l'Education* 1, 2, 97-110.

SAMURÇAY R. (1985) Signification et fonctionnement du concept de variables informatique chez des élèves débutants. *Educational Sudies in Mathematics* . 16, 143-161.

SIME M, GREEN T.G.R., GUEST G.A.N. (1973) Psychological evaluation of two conditional constructions used in computer languages. *Int. J. Man-Machine Studies* , 5, 102-113.

Van der VEER G, van BEEK J, GRUTS G.A.N. (1986) Learning structured diagrams. Effects of mathematical background, instruction and problem semantics. *Colloque : visual aids in programming*. Passau.