REPRÉSENTATIONS MENTALES DES DONNÉES INFORMATIQUES ET DIFFICULTÉS D'ACQUISITION CHEZ DES DÉBUTANTS EN PROGRAMMATION (Première partie : Hypothèses)

Jean-Baptiste LAGRANGE

Le travail présenté dans cet article concerne l'apprentissage de la programmation par des débutants dans le cadre d'un enseignement de l'informatique ouvert à tout public. Dix années d'existence d'une "Option Informatique" s'adressant, en classe de Seconde des Lycées, à un très vaste public, ont montré l'intérêt d'un enseignement de l'informatique, conçu comme une contribution au développement d'aptitudes intellectuelles et répondant aux enjeux culturels posés par l'intervention de l'informatique dans les différents champs du savoir. Elles ont mis en évidence également les difficultés rencontrées par une majorité d'élèves au cours de cet apprentissage, ainsi que l'insuffisance du cadre didactique permettant l'analyse de ces difficultés et la mise en place de moyens pour y remédier. Mon but est de contribuer à la construction de ce cadre didactique.

Je remercie l'EPI de me permettre de faire connaître ce travail en particulier aux enseignants de l'Option Informatique dont j'attends les remarques et critiques. La longueur de l'article impose sa parution sur deux numéros du bulletin. La première partie, publiée dans ce bulletin présente des hypothèses sur les conduites du débutant face à un problème de programmation et sur le rôle que peuvent jouer les objets dans les premiers apprentissages. La seconde partie, à paraître dans le prochain numéro, rapporte les résultats d'une étude expérimentale auprès d'élèves de Seconde et de Première.

1. INTERROGATIONS ET CHOIX INITIAUX

1.1. Difficultés d'acquisition chez les débutants

Dans les cursus s'adressant à des débutants "tout public", on constate de façon assez générale l'abandon d'une partie notable des étudiants ou élèves à l'issue d'une première phase d'apprentissage : concernant l'option informatique des Lycées, (BARON, 1988) souligne que, si l'hétérogénéité est effective en classe de Seconde, par contre, les élèves qui persistent en classe de Première sont dans leur grande majorité, des élèves de sexe masculin des classes scientifiques. Ce phénomène de non-continuation est souvent considéré comme l'indice de difficultés importantes rencontrées principalement par les filles et les élèves non-scientifiques. Néanmoins, on pourrait aussi supposer des causes non directement liées à la discipline informatique : difficulté, pour ces élèves, de suivre plusieurs enseignements optionnels, organisation des établissements... D'autre part, cette non-continuation, dans la mesure où elle serait liée à une situation d'échec des élèves concernés, n'apporte pas d'élément explicatif des difficultés propres à la discipline qui conduisent à cet échec.

1.2. Dépendance des acquisitions par rapport au contexte.

Il est nécessaire d'y "aller voir de plus près", et donc d'observer des élèves débutants face à un problème de programmation. Cette observation, comme toute étude expérimentale suppose des hypothèses, c'est-à-dire ici des directions dans lesquelles chercher l'origine des difficultés rencontrées par le programmeur débutant. On cherche souvent à observer ces difficultés à partir de problèmes de mise en oeuvre de traitements (itération, alternatives imbriquées...) et on situe leur origine dans une insuffisante prise de conscience des capacités de contrôle du dispositif. A la suite de plusieurs expériences d'enseignement de la programmation à des débutants, j'avais pour ma part constaté que les acquisitions de ces élèves étaient fragiles parce que trop liées au contexte des objets sur lesquels porte la programmation : par exemple, des élèves utilisant la récursivité en LOGO pour la programmation de graphismes éprouvent des difficultés importantes à transférer cette connaissance de la récursivité à la programmation d'objets numériques [1]. Le sujet mettrait donc en jeu dans l'activité de programmation une conception des objets informatiques plus ou moins adéquate, et intégrerait en fait les

1 Voir aussi les observations rapportées par (SAMURCAY ROUCHIER, 1990).

traitements de façon dépendante de cette conception. J'ai donc essayé d'étudier, en amont de l'acquisition des traitements, les conceptions chez le débutant des objets informatiques, et l'influence de ces conceptions sur les premiers traitements complexes qu'il a à produire.

1.3. Place des objets dans les premiers apprentissages

De la même façon, alors que, plus loin dans les cursus d'informatique, la structuration des données joue un rôle essentiel, en interaction avec les algorithmes, l'enseignement à des débutants est quant à lui, le plus souvent centré sur l'acquisition des seules structures algorithmiques : dans cet apprentissage, les objets intervenant sont de type numérique, ou, le langage étant un "pseudo-langage" non exécutable, il opère en fait sur les objets du problème [2]. Si ces approches ont le mérite d'apporter une aide au débutant dans la planification des traitements, elles se heurtent cependant aux difficultés que j'ai soulignées ci-dessus quand il s'agit d'"appliquer" les connaissances supposées acquises sur les traitements à des objets informatiques tels que chaînes de caractères, booléens... C'est pourquoi, il m'a semblé important de construire des problèmes et d'explorer des stratégies pédagogiques permettant de mieux insérer les objets dans les premiers apprentissages.

2. L'ACQUISITION D'UN TYPE COMME INTÉGRATION D'UNE STRUCTURE FORMELLE

2.1. Un exemple de problème

Les problèmes de programmation dont j'ai étudié la résolution ne comportent pas de construction algorithmique d'ampleur, ni de structuration complexe des données (tableaux, enregistrements...). Ils présentent cependant pour le débutant un caractère de nouveauté fondamental : il s'agit de passer du monde des objets familiers (les "situations connues") à un monde d'objets formels (les "données codifiées"). Dans les "situations connues", le sujet investit dans la tâche proposée tout un ensemble de connaissances où la signification qu'il donne aux objets est fondamentale ; il en résulte que, dans les problèmes étudiés, si l'on se situe au niveau des "situations connues", la tâche est généralement immédiate. Au contraire, les "données codifiées" que traite

2 Voir comme exemple le "robot peleur de patates" de (DUCHATEAU 1991).

l'informatique sont utilisées de façon formelle, c'est-à-dire en considérant seulement les fonctions opérant sur ces données (leur "type") et les propriétés de ces fonctions (la "définition du type"), ce qui implique, pour le sujet, des contraintes non encore rencontrées dans d'autres activités cognitives.

L'exemple de la figure n° 1 illustre cette différence entre les deux mondes. Dans le monde des objets familiers, le sujet isole sans difficulté les chiffres représentant l'année de naissance dans un numéro à 13 chiffres, et les interprète sous forme numérique de façon immédiate. Au contraire, avec les objets informatiques, il lui faut en premier lieu choisir un type pour la donnée : le choix d'un type numérique qui pourrait paraître le plus adapté dans le cas d'un numéro se heurte aux limitations des représentations des nombres dans les langages, et rend le calcul du résultat plutôt complexe. Le choix du type "chaîne de caractères" permet le calcul d'une sous-chaîne représentative de l'année de naissance ; portant sur des données non-numériques, ce calcul n'est pas trivial pour des débutants, et le calcul (numérique) du résultat final impose de convertir ce résultat intermédiaire sous forme numérique.

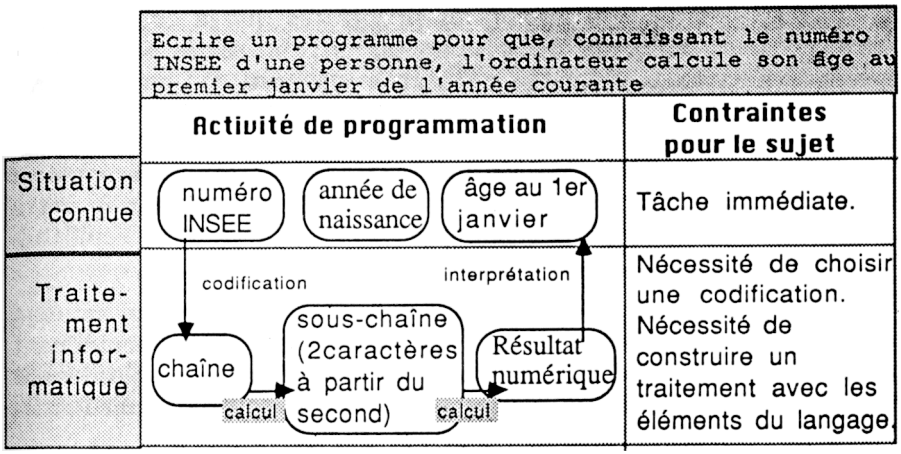

Figure 1

2.2. Difficultés d'acquisition et stratégie pédagogique

On peut faire un parallèle éclairant entre l'acquisition d'un type de données et l'acquisition d'une théorie mathématique s'appuyant sur une connaissance du réel, comme par exemple la géométrie élémentaire

(Figure 2). Il s'agit en effet dans les deux cas de dégager l'idée d'une structure formelle à partir d'une connaissance intuitive, et des difficultés du même ordre sont prévisibles : conception des objets influencée par leurs propriétés intuitives, utilisation naïve du langage...

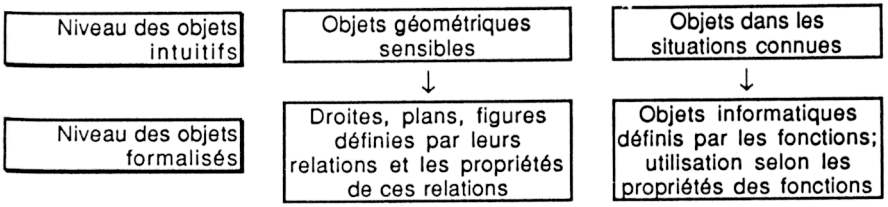

Figure 2

Poursuivons le parallèle dans le domaine des stratégies pédagogiques : l'enseignement d'une théorie mathématique ne se fait pas en énonçant "en vrac" les propriétés qui la constituent ; l'enseignement utilise la déduction pour structurer le domaine et donner un sens aux éléments qui y interviennent. On part de propriétés initiales, d'autres sont déduites, par exemple en réponse à un problème ; elles acquièrent ainsi d'abord un statut local, puis quand l'élève est capable de reconnaître le caractère général d'une propriété ainsi déduite, l'enseignant peut donner à cette propriété le statut de théorème (ce qu'en didactique des mathématiques on appelle l'institutionnalisation).

Cette description évidemment très rapide, donne un schéma possible d'enseignement d'un type de donnée, en considérant qu'en informatique les "fonctions" définissant un type sont l'équivalent des "propriétés" de la théorie, et que la programmation est l'équivalent de la déduction (Figure 3).

Par exemple, dans le cas du type "chaîne de caractères", je me propose de limiter le nombre des fonctions primitives à trois fonctions présentes dans beaucoup de langages de programmation (*Sous-chaîne, Longueur, Concaténation*). La fonction *Position* (qui donne pour résultat le rang du premier élément d'une chaîne donnée, comme sous-chaîne d'une autre chaîne donnée) pourra, quant à elle, d'abord être programmée en tant qu'outil dans des problèmes (nous verrons plus loin la classe des problèmes de "recherche translation" où elle intervient). Quand les élèves auront reconnu la portée générale de cette fonction, il sera possible de la nommer, ou même d'utiliser de façon consciente la fonction pré-définie, si elle existe dans le langage de programmation.

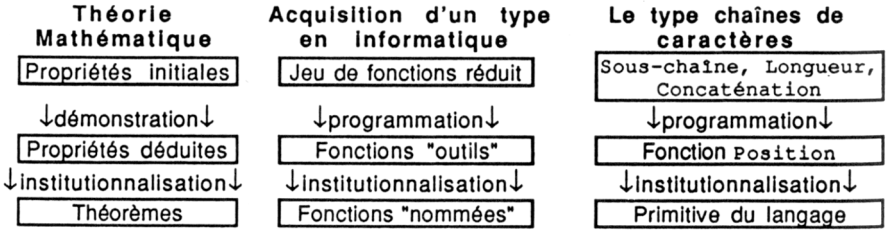

Figure 3

3. LE FONCTIONNEMENT MENTAL DU DÉBUTANT EN PROGRAMMATION

Mener une étude expérimentale suppose de disposer de résultats en psychologie, de façon à interpréter les conduites observées dans l'activité de programmation. J'ai utilisé, pour l'essentiel, le cadre établi par J.M. HOC à partir de l'observation de programmeurs en situation professionnelle.

3.1. Les Systèmes de Représentation et de Traitement

Le sujet, face à une tâche impliquant un dispositif, met en jeu des structures mentales "reflets" plus ou moins compatibles de ce dispositif, que J.M. HOC (HOC, 1987) appelle des Systèmes de Représentation et de Traitement (S.R.T) ; ceux-ci comprennent de façon liée d'une part des connaissances déclaratives (régles relatives au dispositif, au langage, à des méthodes, des heuristiques...) et d'autre part des éléments procéduraux, véritables machines mentales qu'il fait fonctionner afin d'anticiper les réponses du dispositif. La résolution d'un problème de programmation non trivial impliquant par exemple une décomposition en modules, la coordination de structures algorithmiques complexes, la construction d'une structuration des données et son articulation avec la structure algorithmique, suppose chez le programmeur un S.R.T. construit sur plusieurs niveaux et directement compatible avec l'écriture d'un programme (c'est-à-dire un S.R.T. informatique). En effet, le sujet doit être capable, pour mener à bien sa décomposition, pour coordonner les différentes structures, modules, sous-tâches, de se les représenter de façon directement adéquate au dispositif, et doit en même temps disposer d'une vue d'ensemble permettant de les coordonner.

La résolution de problèmes sur les objets telle que je l'envisage ici se donne quant à elle, pour but la constitution chez le sujet débutant, des premiers éléments d'un S.R.T. informatique : il y a "situation connue" à partir du moment où le sujet possède un S.R.T. (non informatique, naïf, ou relatif à un domaine comme les mathématiques...) lui permettant de se représenter données et résultats dans leur généralité, et d'établir un lien permettant de passer des données au résultat. Le S.R.T. permettant de résoudre nécessite au contraire une compréhension des objets adéquate au dispositif (jeu de fonctions sur un type, propriétés de ces fonctions...) et une conception du traitement comme calcul sur ces objets. Dans ce passage à un S.R.T. informatique, les "plans" jouent un rôle essentiel.

3.2. Les plans et l'analogie dans la construction des S.R.T.

Les plans sont, à l'intérieur des S.R.T., des représentations schématiques servant de guide pour l'action : ne prenant pas en compte les détails d'implémentation, ces représentations sont applicables à une classe de problèmes, et peuvent être communes à des S.R.T. distincts : par exemple, devant rechercher une information dans une chaîne de caractères, un programmeur pourra avoir l'"idée" d'un parcours itératif de la chaîne en s'appuyant sur son expérience naïve de recherche d'une information dans un texte. Ce "plan" itératif fait partie du S.R.T. naïf. Il a cependant un sens dans le S.R.T. informatique, et même, il se révèle plus adapté que d'autres plans qui pourraient être déduits du S.R.T. naïf. Le sujet peut donc l'"évoquer" s'il ne dispose pas dans son S.R.T. informatique d'un plan directement applicable. C'est ce que J.M. HOC appelle l'évocation par "analogie".

La résolution ne s'arrête cependant pas à cette évocation ; à partir du "plan", le sujet doit construire un programme. Or, les différents éléments de ce plan, lorsqu'ils sont suffisamment précisés peuvent se révéler incompatibles avec une programmation sur le dispositif. Dans l'exemple ci-dessus, le sujet peut, sans en avoir au départ clairement conscience, itérer à certains moments sur des mots, à d'autres sur des caractères ; le parcours itératif tel qu'il se le représente est inutilement complexe et dépasse ses capacités de programmeur : il y a donc nécessité d'adapter ce plan, en fait ici, de le simplifier en une lecture séquentielle des caractères. D'autres éléments du plan peuvent nécessiter des adaptations pour satisfaire aux contraintes de la programmation : toujours dans l'exemple d'un parcours de chaîne en vue de la recherche d'une information, les objets mis en jeu dans l'itération (index, condition

de sortie...) ont, dans le contexte naïf, un statut lié à ce contexte (l'index est un élément matériel, par exemple le doigt du même nom ; la condition de sortie est une forme verbale...). Le sujet doit par conséquent adapter le plan aux objets informatiques du dispositif ; statut numérique de l'index, statut booléen de la condition de sortie...

En passant du plan au programme, le programmeur débutant doit donc remettre en cause certains aspects non informatiques, en adapter d'autres pour les rendre compatible avec la programmation pour le dispositif. Par différenciation, adaptation de plans issus de S.R.T. non informatiques, il intègre des contraintes du langage et leur donne une signification, et ainsi construit les premiers éléments d'un S.R.T. informatique.

Il me semble donc important que, dans des problèmes sur les objets posés à des débutants, le contexte intuitif permette l'évocation de plans par analogie, mais que les contraintes du dispositif informatique conduisent, lors de l'écriture effective du programme soit à la remise en cause du plan s'il est inadapté au traitement, soit à son adaptation partielle, en donnant en particulier un statut informatique aux objets sur lesquels il porte.

3.3. Les plans des objets et les plans des traitements

Quand, par exemple, un sujet confronté à la construction d'un graphisme en "géométrie-tortue", doit calculer l'angle de rotation nécessaire pour "positionner" la tortue dans la direction qui lui permettra de terminer le dessin, il prend en compte la position atteinte, l'orientation de la tortue, les propriétés géométriques des dessins élémentaires... Il met donc en oeuvre des éléments de représentation mentale de la tortue et du dessin élémentaire à réaliser et des procédures mentales, qui constituent pour moi un plan de l'objet "tortue". De façon plus générale, le S.R.T. utilisé par un programmeur lui permet, éventuellement, de distinguer des objets d'un type particulier, d'intégrer la variabilité des objets, la possibilité de les nommer, de faire changer leur valeur, de disposer d'un langage sur ces objets permettant d'exprimer et de combiner les fonctions propres à ces objets. Pour moi, ces éléments s'organisent, au sein du S.R.T., en plan(s) des objets.

En reprenant l'exemple ci-dessus, le sujet confronté au tracé d'un graphisme en "géométrie-tortue" ne doit pas seulement imaginer comme nous l'avons vu, des actions élémentaires comme les rotations de la tortue, et ses déplacements. En s'appuyant sur sa conception de l'objet

"tortue", il doit également trouver dans son S.R.T. les plans lui permettant d'imaginer et de coordonner les déplacements de la tortue. Le S.R.T. comprend donc aussi des plans permettant au programmeur de se représenter, de prévoir, de faire fonctionner mentalement des traitements complexes, en articulant entre elles les actions élémentaires sur les objets. Ces plans des traitements comprennent des représentations des capacités de traitement (réelles ou supposées) du dispositif, des significations que le sujet attribue aux divers éléments syntaxiques marquant le traitement, des procédures d'exécution mentale des traitements complexes.

3.4. Les plans des objets chez le débutant

Les dispositifs les plus abstraits (par exemple ceux que constituent les langages de programmation "évolués"), permettent de considérer toute manipulation sur des objets comme un calcul (c'est à dire un processus données -> résultats, entièrement explicitable à l'aide de règles formelles). Au contraire, dans le S.R.T. d'un sujet débutant, les plans des objets sont marqués par une conception pré-calculatoire où le sujet prête à des éléments du langage la capacité d'agir sur des objets internes au dispositif (par exemple, la concaténation peut être comprise comme un déplacement effectif de caractères).

Par ailleurs, l'informatique considère les objets d'une part sous leur définition abstraite et d'autre part par les représentations [3] qui relient les types entre eux, en premier lieu la(les) représentation(s) des objets du problème avec les types pré-définis du langage de programmation. En programmation informatique, le travail sur les objets consiste donc à spécifier des définitions abstraites et à construire des représentations dans des types connus. Pour le débutant, cette distinction abstraction-représentation est à construire. En premier lieu, le sujet débutant ne distingue pas l'objet informatique de sa forme externe (par exemple, il assimile une entité numérique à la chaîne de ses chiffres en base 10), et éprouve des difficultés à donner une signification à des objets informatiques n'ayant pas de forme externe, ou une forme externe peu manipulable, tels que les booléens et plus généralement les types énumérés...

3 Il s'agit ici de représentations au sens informatique, et non des représentations mentales constituantes des S.R.T.

3.5. L'interaction des traitements et des objets chez le débutant

Les langages informatiques permettent des traitements généraux indépendants des objets sur lesquels ils opèrent : c'est ainsi que par exemple, la concaténation d'un caractère à une chaîne étant un calcul isomorphe à la multiplication de deux nombres, le "retournement d'une chaîne de caractères" est une itération isomorphe à l'exponentiation obtenue comme multiplication itérée.

Au contraire, par le mécanisme d'évocation par analogie décrit ci-dessus, le sujet débutant "importe" à partir de S.R.T. non informatiques, de façon liée un plan du traitement qu'il projette de transformer en programme, et des plans des objets, qui sont, dans ces S.R.T., compatibles entre eux. La solution qu'un débutant produit pour un problème de programmation mettant en jeu des objets et un traitement, doit donc pouvoir s'analyser comme une conjonction de plans du traitement, et de plans des objets. La mise en évidence et l'analyse de cette conjonction de plans est un des aspects importants de l'observation qui est rapportée ci-dessous.

4. QUELLE UTILISATION DES TYPES SIMPLES ?

J'explique ici pourquoi les types numériques se prêtent mal à la problématique que je viens de définir, en particulier en quoi les difficultés rencontrées avec ces types n'ont pas de caractère de généralité et sont donc peu productives. J'examine aussi les conditions dans lesquelles les types chaînes de caractères et booléens sont utilisables pour cette problématique.

4.1. Singularité du type numérique

Si l'on utilise un type numérique sans tenir compte des limitations inhérentes à leur représentation informatique, en ne différenciant pas les entiers des réels, ni les cardinaux des ordinaux, on utilise en fait la structure numérique abstraite des mathématiques. Les objets numériques sont ainsi "déjà codifiés" ; ils ne sont plus porteurs des propriétés des objets familiers ; on perd donc la "profondeur", la distance entre le niveau des "situations connues" et celui des objets informatiques : s'il y a modélisation d'une situation, elle consiste plus en la recherche de l'outil mathématique pertinent qu'en un véritable travail en informatique. En fait, le travail de construction de représentations adéquates des objets a déjà été effectué dans l'enseignement des

mathématiques, (ou est ressenti par les sujets comme faisant partie de cet enseignement), et donc on ne pourra l'observer.

D'autres situations purement numériques mettent en jeu des S.R.T. distincts : ainsi (Samurcay, 1985) des élèves de seconde peuvent se représenter l'exponentiation pour le calcul explicite d'une puissance par multiplications dans un contexte mathématique, sans pour autant que cette représentation soit utilisable pour construire un algorithme exécutable par un dispositif informatique. En effet, la représentation mathématique est statique, elle permet une multiplicité de calculs pour un but donné, et n'est pas la représentation d'un calcul organisé qui serait nécessaire pour programmer. On a bien un S.R.T. mathématique permettant les calculs d'expressions numériques dans le contexte de cette discipline, et un autre, qui serait nécessaire pour "faire faire" ce calcul par le dispositif. Pour le débutant, ce dernier S.R.T. est à construire. Il serait certes intéressant d'étudier la construction de ce S.R.T. informatique, mais cela supposerait de caractériser au préalable les différents S.R.T. installés par l'enseignement des mathématiques.

4.2. Les chaînes et les entités numériques dans les expressions sur les chaînes

Les chaînes permettent la codification de données intuitives, sans passer par la recherche d'outils mathématiques. Comme souligné ci-dessus, un apprentissage sur un type d'objets ne saurait se concevoir comme l'acquisition du jeu de fonctions dans un langage donné. Il s'agit au contraire à travers la résolution de problèmes portant sur des "situations connues", d'enrichir un jeu de fonctions, au départ très limité. Dans le cas des chaînes trois fonctions suffisent : *Sous-chaîne, Longueur, Concaténation*. Le calcul sur des chaînes que permet ce jeu de fonctions, et dont nous avons vu un exemple au &2 n'est en rien spontané pour le débutant. En particulier, la fonction sous-chaîne est une fonction à trois arguments de type hétérogène (chaîne, ordinal, cardinal) et utilise la notation fonctionnelle préfixée. En mathématiques, dans la scolarité obligatoire, les expressions algébriques formées sur les types cardinaux ont été rencontrées, et le sujet a pu s'en construire une signification. Par contre, les expressions formées à l'aide des fonctions chaînes sont nouvelles, et mettent en jeu les notions de variable et de fonction de façon plus générale.

Les cardinaux et les ordinaux interviennent dans la structure de chaîne comme arguments et comme résultats de fonctions sur les

chaînes. De ce fait, ils n'ont pas le statut de "déjà codifié" des entités intervenant dans des problèmes numériques : particulièrement l'ordinal, rang d'un caractère dans une chaîne, est la codification d'un index qui, dans les "situations connues" est un élément matériel. Dans le cadre de l'enseignement obligatoire, les questions portant sur les ordinaux et la coordination entre ordinaux et cardinaux sont abordées seulement au tout début de cet enseignement, et l'apprentissage du calcul, en particulier le passage au cadre algébrique, porte sur les seuls cardinaux. Pour des sujets ayant suivi seulement cet enseignement, l'utilisation d'ordinaux pour la codification d'index ou de positions, et les calculs faisant intervenir des ordinaux et des cardinaux sont donc nouveaux. Certains cardinaux apparaissent même sous un jour nouveau : c'est le cas de la longueur d'une chaîne, qui peut intervenir dans une expression, conjointement avec la chaîne elle-même : en mathématiques, il est rare qu'une expression fasse intervenir simultanément l'objet et sa "mesure" [4].

4.3. Les booléens

Étudier l'acquisition du type booléen permet de montrer que la problématique énoncée dans cet article s'applique à d'autres types que les chaînes, et de savoir dans quelle mesure les résultats obtenus sont cohérents d'un type à l'autre. Les booléens présentent en effet des caractéristiques bien différentes des chaînes : on n'a pas la lourdeur de la syntaxe, par contre, on rencontre les restrictions d'entrée/sortie qui en font un premier exemple de type énuméré.

En tant que type, les booléens se définissent comme des objets à deux valeurs. Les fonctions sont les "connecteurs propositionnels" et leurs propriétés, qui constituent la logique du premier ordre. Les conditions, constituantes des instructions conditionnelles (alternatives, itérations...)

4 Le jeu de fonctions (Sous-chaîne, Longueur, Concaténation) constitue une définition des chaînes comme objets simples. D'autres définitions seraient possibles. Par exemple, PASCAL considère les chaînes comme des tableaux de caractères et certains problèmes se résolvent plus simplement avec cette définition. Cependant, les chaînes comme objets simples, ont des propriétés que n'ont pas les tableaux : on peut les comparer, les obtenir comme résultat de fonction, définir des constantes. Nous avons choisi cette définition pour mettre en évidence plus facilement et plus rapidement les conceptions des objets présentes chez les élèves en faisant fonctionner les expressions, et les éléments généraux du langage sur ces expressions.... Les chaînes comme tableaux de caractères (affectation individuelle de chaque caractère, comparaison caractère à caractère...) mettent en effet en jeu une organisation des données plus générale, et il sera intéressant (mais dans un second temps), de mettre en relation la structure de tableau de caractères avec la structure d'objet simple, de façon à choisir en connaissance de cause une structuration adaptée à chaque problème.

sont en fait, dans les langages de programmation, des objets de type booléen, et la possibilité de définir des variables de ce type, et des fonctions à valeur dans ce type permet le calcul de ces conditions. L'enjeu de l'apprentissage des booléens que je considère est donc la construction d'une conception des conditions comme objets informatiques c'est-à-dire comme valeurs booléennes calculables.

L'enjeu de l'apprentissage est donc différent de celui d'un enseignement de la logique qui serait par exemple destiné à développer des capacités en raisonnement. Il s'apparente par contre à l'apprentissage de la logique propositionnelle du premier ordre dans des disciplines technologiques qui considèrent des objets à deux états et des combinaisons de ces objets, généralement des circuits électriques, ou électroniques, ou des systèmes pneumatiques.

Dans le prochain numéro : Observation d'élèves de l'option informatique et interprétation.

Jean-Baptiste LAGRANGE