Calculer avec des points pour comprendre la machine (Collège/Lycée Maths/ISN)

Guillaume CONNAN*- IREM de Nantes

JA de l'IREM de Nantes - Jeudi 9 avril 2015

Résumé La géométrie tend à disparaître des programmes...c'est bien dommage. L'enseignement de l'informatique tend à prendre une importance croissante mais de manière assez floue et parfois peu scientifique. L'avenir de l'éducation est aux « projets transversaux » : faisons d'une pierre deux coups et apprenons le fonctionnement de la machine avec une règle et un compas. Faisons également découvrir aux élèves que l'informatique peut être théorique voire uniquement cela.

1 Machine de construction géométrique

En 1989, Ülrich HUCKENBECK [a] décrit une machine dont les primitives sont les opérations géométriques usuelles que l'on peut effectuer à la règle et au compas.
Il introduit un *automate* (GCM : Geometrical Construction Machine) et donc des *états* et des *transitions*. La notion d'automate est extrêmement importante en informatique et est souvent introduite par les automates à états finis sur des langages ou plus généralement à partir de machines de TURING.
Il ne s'agit pas ici d'introduire une quelconque théorie des automates mais de faire découvrir comment en informatique le papier et le crayon sont des préalables (avec parfois des dizaines d'années d'avance) à une mise en pratique sur machine, que l'informatique ne peut être d'ailleurs que théorique, que la géométrie est un champ d'investigation informatique et qu'il est bien dommage qu'elle disparaisse des programmes.
Nous allons construire une « machine » équipée de registres comme en vrai dont on peut définir la configuration à chaque instant.
On a des registres pour stocker :

- des points (appelons-les p_i) qui contiennet des données de type $p(abs, ord)$;
- des droites (appelons-les d_i) qui contiennent des données de type $d(p_1, p_2)$;
- des cercles (appelons-les c_i) qui contiennent des données de type $c(p_1; p_2, p_3)$.

On considèrera que les registres sont initialisés et contiennent au départ :

- le point $(0,0)$ dans les registres p_{3i}
- le point $(1,0)$ dans les registres p_{3i+1}
- le point $(0,1)$ dans les registres p_{3i+2}
- l'axe des abscisses dans les registres d_{2i}
- l'axe des ordonnées dans les registres d_{2i+1}
- le cercle unité centré en $(0,0)$ dns les registres c_i

La machine peut être dans trois états :

- `Non terminé` : la machine n'est pas dans un état terminal ;
- `Fini` : la machine n'attend plus d'instruction ;
- `Erreur` : il y a une erreur et la machine s'arrête.

Les deux derniers états sont dits *terminaux*.
On peut effectuer des opérations sur les registres selon ce qu'ils contiennent et l'état de la machine :

*guillaume.connan@univ-nantes.fr
a. Huckenbeck [1989] et Huckenbeck [1988]

— copier une valeur d'un registre vers la sortie standard ;
— mettre dans un registre l'intersection de deux droites ;
— mettre dans un registre une intersection d'une droite et d'un cercle avec la possibilité de la distinguer d'un point indiqué ;
— mettre dans un registre une intersection de deux cercles avec la possibilité de la distinguer d'un point indiqué ;
— mettre dans un registre la droite passant par deux points ;
— mettre dans un registre le cercle de centre un point et de rayon la distance entre deux points ;
— copier le contenu d'un registre dans un autre ;
— indiquer que « c'est fini ».

On voit déjà poindre plusieurs problèmes :la gestion des erreurs et le non-déterminisme.
Généralement, l'informatique n'est abordée que sous l'angle déterministe.
Ici, on peut dérouler un *arbre* d'exécution qui aura éventuellement des embranchements.
S'il n'y a aucune *feuille* `Erreur` et que toutes les feuilles sont identiques, alors le *programme* termine et la sortie est celle des feuilles

2 Premiers exemples

On considère deux points distincts A et B : on voudrait calculer leur milieu.

1 $c_1 \leftarrow c(A; A, B)$
2 $c_2 \leftarrow c(B; A, B)$
3 $p_1 \leftarrow$ `intersection (`c_1, c_2`)`
4 $p_2 \leftarrow$ `intersection (`c_1, c_2`) différente de` p_1
5 $d_1 \leftarrow d(p_1, p_2)$
6 $d_2 \leftarrow d(A, B)$
7 $p_3 \leftarrow$ `intersection (`d_1, d_2`)`
8 `Écrire` p_3 `(dans un registre ou vers la sortie standard)`
9 `Fini`

Construisez l'arbre correspondant. Est-ce que ce programme termine ? Est-ce que ce programme effectue la tâche prévue ?
Adresse, étiquette, contenu ?
Quels sont les états de la machine ?
Donnez une séquence d'instructions qui construit un triangle équilatéral de base $[A, B]$. Des remarques ? Spécification ?
Étant donné une droite Δ, un point A de cette droite et un point B différent de A, est-ce que la droite parallèle à Δ passant par B est *constructible* par un GCM ?
Imaginez un problème avec des perpendiculaires.
Constructible/calculable ?

3 Opérations algébriques et théorème de Thalès

Étant donné un point A de coordonnées (x, y), renvoyer le point de coordonnées $(x, 0)$ et le point de coordonnées $(0, y)$.
Étant donné un point $(x_1, 0)$ et un point $(x_2, 0)$, quelle construction associer à $(x_1 + x_2, 0)$? Êtes-vous sûr(e) ?

1 $p_1 \leftarrow p(x_1, 0)$
2 $p_2 \leftarrow p(x_2, 0)$
3 $p_3 \leftarrow \texttt{milieu}(p_0, p_5)$
4 $d_3 \leftarrow \perp(d_1, p_3)$
5 $d_4 \leftarrow \perp(d_1, p_5)$
6 $d_5 \leftarrow d(p_1, p_3)$
7 $p_6 \leftarrow \cap(d_4, d_5)$
8 $d_6 \leftarrow d(p_0, p_6)$
9 $p_7 \leftarrow \cap(d_3, d_6)$
10 $d_7 \leftarrow d(p_2, p_7)$
11 $p_8 \leftarrow \cap(d_4, d_7)$
12 $d_8 \leftarrow d(p_8, p_3)$
13 $p_9 \leftarrow \cap(d_0, d_8)$
14 Écrire p_9
15 Fini

Quels automates associer à $(x_1 - x_2, 0)$? $(x_1 \times x_2, 0)$? $(x_1/x_2, 0)$?

4 Un autre type d'automate

Enlevons les opérations sur le cercle mais rajoutons l'emploi d'une équerre : on va donc maintenant construire à la règle et à l'équerre : quelle propriété « informatique » importante gagne-t-on ?

5 Informatique sans machine

Une grande part de l'informatique s'étudie « sans machine » mais son enseignement peut également se faire de manière ludique et sans recours aux ordinateurs. On pourra se référer à l'ouvrage Néo-Zélandais [b] qui introduit le programme d'informatique de Licence...à l'école primaire mais de manière ludique.
Pour voir au contraire des automates géométriques beaucoup plus riches et pour l'instant plutôt théoriques mais dont les implications pourraient être très importantes, voir Durand-Lose [2015].

6 Mathématique discrète et « computer science »

Nous avons du mal en français à trouver des termes pour désigner les domaines gravitant autour de l'informatique. Ce dernier mot est souvent remplacé par TICE (avec un T comme *technologie*...cela restreint le domaine et permet de comprendre pourquoi pendant si longtemps informatique rimait avec bureautique) ou bien par *numérique* et la dernière mode est de parler de *code*. Les anglo-saxons parlent de *Computer Science*, désignation qui a l'avantage de comporter le mot *science*. *Computer Science* est **en relation avec** les domaines *Scientific Computing* et *Computer Engineering* mais n'est pas du tout réduit à ces deux composantes proches.
Le département de *Computer Science* de l'Université de Boston s'adresse ainsi à ses futur(e)s étudiant(e)s [c] :

> « *Computer Science is about problem solving. Thus, the qualities of a good computer scientist include a passion for finding elegant solutions, an ability to use mathematical analysis and logical rigor to evaluate such solutions, creativity in modeling complex problems through the use of abstractions, attention to details and hidden assumptions, an ability to recognize variants of the same problem in different settings, and being able to retarget known efficient solutions to problems in new settings. If you like to solve puzzles, then computer science is for you !* »

Cela ressemble beaucoup à ce que l'on attend d'élèves étudiant les mathématiques !...

b. Bell et coll. [2015]
c. http://www.cs.bu.edu/AboutCS/WhatIsCS.pdf

Références

T. Bell, I. H. Witten et M. Fellows,
CS Unplugged : An enrichment and extension programme for primary-aged students,
adresse : http://csunplugged.org/, mars 2015.

J. Durand-Lose,
« Construire et calculer dans un monde 2d »,
Dans Informatique mathématique : une photographie en 2015, N. Ollinger (coordinateur), p. 135–176, CNRS, CNRS Éditions, mars 2015, ISBN 978-2-271-08791-1, adresse : http://www.univ-orleans.fr/lifo/evenements/EJCIM2015/.

U. Huckenbeck,
« Euclidian geometry in terms of automata theory »,
Theoretical Computer Science, vol. 68, n° 1, p. 71–87, oct. 1989.

U. Huckenbeck,
« Geometrical abstract automata »,
Dans Proceedings on International Workshop on Computational Geometry on Computational Geometry and Its Applications, p. 217–231, New York, NY, USA, Springer-Verlag New York, Inc., 1988, ISBN 0-387-50335-8.