CONFÉRENCES INVITÉES

Pierre Dillenbourg

École Polytechnique Fédérale de Lausanne, Suisse
pierre.dillenbourg@epfl.ch

Pensée computationnelle : Pour un néopapertisme durable car sceptique

« *On a vu souvent rejaillir le feu d'un ancien volcan qu'on croyait trop vieux* » chantait le Grand Jacques. L'intérêt actuel pour la pensée computationnelle n'a-t-il pas un goût de déjà vu pour ceux qui ont connu les années Logo ? L'histoire de l'éducation est peuplée d'exemples dans lesquels une idée ne s'impose pas à sa première apparition mais à sa deuxième, ou à sa troisième… Le contexte actuel explique la résurgence de ces idées brillantes. On peut décrire le contexte en termes d'offre et de demande. Du côté de la demande, l'explosion des performances de l'intelligence artificielle et de la robotique a fait prendre conscience de l'urgence de considérer la pensée computationnelle comme compétence fondamentale de tout citoyen confronté aux algorithmes qui nous influencent. Le climat est cependant différent : alors que les tentatives des années 80 reposaient sur un formidable optimisme, les efforts actuels baignent dans un certain climat d'inquiétude, notamment quant à l'emploi. Du côté de l'offre, il n'existe plus une approche unique de la question, comme Logo, mais une multitude de plateformes logicielles et robotiques pour l'apprentissage de la programmation et/ou de la pensée computationnelle. Je vous parlerai évidemment de Thymio dont mon collègue Francesco Mondada a déjà vendu plus de 30 000 exemplaires et pour lequel il a formé 900 enseignants. Je vous présenterai également les projets de robotique développés dans mon laboratoire, tels que Cellulo[1]. Mais, bien que l'offre et la demande convergent magiquement, la situation n'est pas sans embûches. Il est bon de regarder à la fois en arrière, apprendre des erreurs commises avec Logo, et vers l'avant, pour anticiper les écueils que l'on voit poindre à l'horizon.

1 <https://chili.epfl.ch/cellulo>

Dans le rétroviseur, on s'aperçoit que le potentiel pédagogique de Logo fut en quelque sorte victime du niveau d'attentes suscité par le discours de Papert, discours certes brillant et charismatique, mais promettant des effets qu'aucune approche pédagogique ne pouvait atteindre. Si on vous promet de gagner 1 million, vous serez déçu d'en recevoir un demi. Aujourd'hui, soyons modestes dans nos promesses de résultats. Du côté futur, anticipons que l'engouement actuel, le *hype* autour de ce thème, s'affaiblira à terme parmi les décideurs de nos systèmes éducatifs. Il convient dès lors d'inscrire nos plans d'actions dans le long terme, de conduire des projets qui continueront à fonctionner quand les médias auront détourné leur attention.

Dans le rétroviseur, on peut dire qu'il régnait un certain abus de confiance dans les propriétés éducatives intrinsèques dont un outil technologique serait pourvu. Un élève peut passer des heures avec un langage ou un robot sans rien apprendre. Ce qu'il apprend dépend de l'activité qu'il fait avec ce robot : copie-t-il du code fourni par un (mauvais) enseignant ou explore-t-il, va-t-il utiliser des variables ou la récursion ? Aucun outil n'a des propriétés pédagogiques intrinsèques ; un outil possède certes un certain potentiel (« affordance ») mais ce potentiel ne devient effet réel qu'en fonction des activités que l'élève réalisera. Or, qui prépare ces activités ? L'enseignant ! Ne commettons donc pas pour la seconde fois l'erreur de négliger le rôle des enseignants dans l'effet des technologies utilisées. Dès lors, du côté futur, préparons les enseignants à la pensée computationnelle et, en toute urgence, formons les formateurs d'enseignants.

Dans le rétroviseur, l'approche Logo postulait le développement des capacités de résolution de problèmes qui soient indépendantes du domaine d'application. Or, l'existence même de ces compétences a été remise ensuite en cause par les travaux sur la cognition située. Pourquoi suis-je capable de décomposer un problème complexe en problèmes simples lorsqu'il s'agit de programmation et non pas lorsque j'essaie vainement de réparer la plomberie de ma salle de bains ? Parce que notre système cognitif n'est pas composé de strates, certaines génériques, d'autres spécifiques. Le transfert a toujours été le talon d'Achille de l'éducation. Pour qu'un élève soit capable de transférer une compétence d'un domaine à l'autre, les cours doivent inclure des activités de transfert entre plusieurs domaines. C'est pourquoi il ne faut pas dédier un nouveau cours à la pensée computationnelle, probablement assigné aux professeurs de maths, mais au contraire parler de ces principes dans les cours d'histoire, d'allemand, d'économie, etc. Le défi en termes de formation des enseignants n'en est que décuplé.

Dans le rétroviseur, certains ont confondu la pensée algorithmique avec l'histoire ou la sociologie des médias. Dans le futur, il faut certes parler des *fake news* et des réseaux sociaux, mais une vraie compréhension des médias actuels ne limite pas à identifier les mauvaises intentions d'une inéluctable partie de notre société, mais à identifier par exemple les propriétés des graphes sous-jacents aux réseaux, propriétés qui expliquent certaines failles et les biais du système.

Dans le rétroviseur, on revit les combats idéologiques autour de Logo, qui ressemblent assez à ceux que j'observe aujourd'hui sur la relation entre pensée computationnelle et programmation. Certains considèrent la première comme un pas intermédiaire et la seconde comme l'objectif principal. D'autres craignent que les activités de codage nuisent à l'acquisition de la pensée computationnelle, en exagérant les contraintes formelles, notamment la syntaxe. Mais dans ce cas, on peut craindre que la pensée computationnelle soit enseignée comme des maths. Aujourd'hui, les outils de programmation permettent de libérer les élèves des contraintes syntaxiques et donc aux enseignants de leur faire découvrir les algorithmes de manière exploratoire, grâce à la programmation. La programmation permet de rendre un problème croustillant ; l'élève peut commettre autant d'erreurs qu'il veut et recommencer souvent sans s'exposer au feedback d'un enseignant. Je considère que la programmation est au service de la pensée computationnelle et non l'inverse.

Papert était un scientifique très inventif et un communicateur de génie, et les efforts actuels ne sont pas dénués d'un certain militantisme sympathique. Il n'y a certes pas d'éducation sans système de valeurs, mais gardons un regard relativement objectif et scientifique sur la conception des programmes de formation à la pensée computationnelle.

Marina Umaschi Bers

Tufts University, Massachusetts, États-Unis
marina.bers@tufts.edu

La programmation en tant que place de jeu développementale : la pensée informatique et la robotique dans la petite enfance

Marina Umaschi Bers est professeure au département Eliot-Pearson de l'étude de l'enfant et du développement humain et au département d'informatique de la Tufts University. Elle dirige le groupe de recherche interdisciplinaire des technologies développementales. Ses recherches comprennent la conception et l'étude de technologies d'apprentissage qui favorisent le développement positif des enfants. Elle a créé l'application de programmation bien connue ScratchJr et le kit robotique KIBO, destinés aux enfants de 4 à 7 ans.

Dans ma conférence, je présenterai un survol de mon programme de recherche interdisciplinaire en utilisant la métaphore de l'opposition entre place de jeu et parc pour enfants pour comprendre le rôle de la programmation dans l'apprentissage de l'informatique que font les enfants. Les places de jeu sont des espaces prisés des jeunes enfants pour jouer et apprendre. Les enfants peuvent y être autonomes tout en développant différentes séries de compétences. Les parcs pour enfants, en revanche, « parquent » les enfants dans un espace confiné et sûr. Même si les risques y sont limités, il ne s'y passe que peu d'exploration, de résolution de problèmes ou de jeu imaginatif.

Ma présentation utilisera la métaphore « place de jeu/parc pour enfants » pour investiguer le rôle de la programmation et de la pensée informatique chez les jeunes enfants. Je présenterai un cadre de réflexion autour des idées fortes de l'informatique et de l'ingénierie adaptées à de jeunes enfants et montrerai des exemples qui mettent en œuvre les deux environnements que j'ai créés, le langage de programmation ScratchJr et le kit robotique KIBO.

La conférence reprendra des idées de mon récent livre *Coding as a Playground*, où je propose que la programmation soit considérée non

seulement comme une compétence technique, mais comme une nouvelle littératie – un nouveau moyen pour les enfants d'exprimer et de partager leurs idées. En tant que littératie, la programmation a le pouvoir de changer le monde. Je présenterai enfin mes nouvelles recherches, qui s'intéressent aux bases cognitives et neuronales de l'apprentissage de l'informatique dans la petite enfance.

Ivan Kalaš

Université Comenius de Bratislava, Slovaquie
UCL Institute of Education, Londres, Royaume-Uni
ivan.kalas@fmph.uniba.sk

La programmation à l'école primaire : De Papert à la nouvelle informatique

Conférence originale en anglais

Ces dernières années, nous avons été témoins d'un regain d'intérêt sans précédent pour la programmation éducative, non seulement à l'école secondaire, mais aussi à l'école primaire et même à des degrés préscolaires. Si la tendance actuelle n'est pas la première de l'« ère numérique », c'est certainement la plus forte, la plus complexe et la plus substantielle de toutes les précédentes. Les « pères fondateurs » qui, dans les années 70 et 80, ont reconnu le potentiel des ordinateurs pour les apprentissages chez les enfants par l'exploration, l'expression et la création (voir Papert, 1980, ainsi que les travaux précurseurs de Feurzeig, Kay, Clayson, Goldenberg et d'autres) ont été récemment propulsés sur le devant de la scène et retravaillés au travers de nouvelles publications d'influence comme, entre autres, Wing (2006), le rapport révolutionnaire de la British Royal Society (2012) et la transformation qui s'en est suivie en septembre 2014 de la traditionnelle « éducation aux TIC » en une nouvelle discipline « Informatique » (*Computing*) au Royaume-Uni[1], soutenue par un ambitieux plan d'études national (voir DfE, 2013).

Actif dans le domaine de la programmation et de l'informatique éducatives depuis le milieu des années 80, j'essaierai dans ma conférence de tirer profit de mes expériences passées et présentes de développement d'environnements logiciels pour enseigner la programmation au sein de mon groupe à l'Université Comenius, de mon récent rôle dans la création de contenus pour l'apprentissage de la programmation dans les écoles

1 Dès l'âge de 5 ans et jusqu'au baccalauréat.

primaires du Royaume-Uni (*UCL ScratchMaths*[2]) et de mon implication à l'UNESCO, à l'IFIP (*International Federation for Information Processing*) et dans le programme *Partners in Learning* de Microsoft. En partant de cette perspective multiple, je poserai la question de savoir si la poussée d'intérêt actuel envers la programmation éducative diffère des précédentes initiatives et, dans l'affirmative, j'interrogerai nos chances de réussir une implantation durable.

Comme nous travaillons avec des concepts parfois peu clairs comme TIC, *computing*, littératie numérique ou informatique[3], je commencerai par définir la programmation éducative en me concentrant exclusivement sur le milieu scolaire, principalement au primaire[4], où toutes les matières ou presque sont traitées par un seul enseignant généraliste. Si cela est parfois vu comme un obstacle à l'enseignement de l'informatique à ces degrés, je me ferai l'avocat du contraire : l'atmosphère et la culture d'apprentissage de l'école primaire offrent un potentiel extraordinaire pour tirer parti de l'informatique et de la programmation au bénéfice des jeunes apprenants – en tant que nouvel et puissant instrument d'exploration, d'apprentissage, de création et de communication.

J'essaierai aussi de formuler un cadre conceptuel pour la programmation éducative au primaire ainsi que d'identifier certains principes qui, je pense, devraient être respectés si l'on souhaite concevoir et mettre en œuvre de manière pérenne des contenus pour l'apprentissage de la programmation ainsi que la pédagogie qui les accompagne. Je mentionnerai enfin les défis et les risques potentiels dans le futur proche. J'illustrerai ceci par trois projets que nous avons développés récemment ou qui sont en cours : les débuts de la programmation avec Thomas le Clown en première primaire, notre dernier projet de construction systématique d'éléments de programmation pour les degrés 3 et 4, et notre cursus ScratchMaths pour les degrés 5 et 6 (de 9 à 11 ans).

Mots clés : programmation au primaire, programmation, informatique, computing, pensée informatique, pertinence de développement

2 Voir <http://ucl.ac.uk/scratchmaths>.

3 En tant que domaine d'études aux degrés primaires et secondaires.

4 Au sens plus large selon le schéma ISCED de l'UNESCO, avec des durées typiques de 4 à 6 voire 7 ans dès l'âge de 5 ou 6 ans.

Références

Benton, L, Saunders, P., Kalaš, I., Hoyles, C., Noss, R. (2017). Designing for learning mathematics through programming : a case study of pupils engaging with place value. To appear in *The International Journal of Child-Computer Interaction*.

Blackwell, A. F. (2002). What is Programming ? In *14th Workshop of the Psychology of Programming Interest Group*. 204–218.

DfE. (2013). *Computing Programmes of Study : Key Stages 1 and 2*, DfE. Available at : <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study>.

Gujberova, M., & Kalaš, I. (2013). Designing productive gradations of tasks in primary programming education. In *Proceedings of WiPSCE 2013*, 108–117. ACM Digital Library DOI 10.1145/2532748.2532750.

Kabatova, M., Kalaš, I., Tomcsanyiova, M. (2016). Programming in Primary Slovak Schools. *Olympiads in Informatics*, Vol 10 (2016), 125–158.

Papert, S. (1980). Mindstorms : Children, Computers, and Powerful Ideas. New York : Basic Books, Inc.

The Royal Society. (2012). *Shut down or restart ? The way forward for computing in UK schools.* Available at : <https://royalsociety.org/topics-policy/projects/computing-in-schools/report/>.

Wing, J. (2006). Computational Thinking. *Communication of the ACM*, Vol 49, No. 3, 33–35.