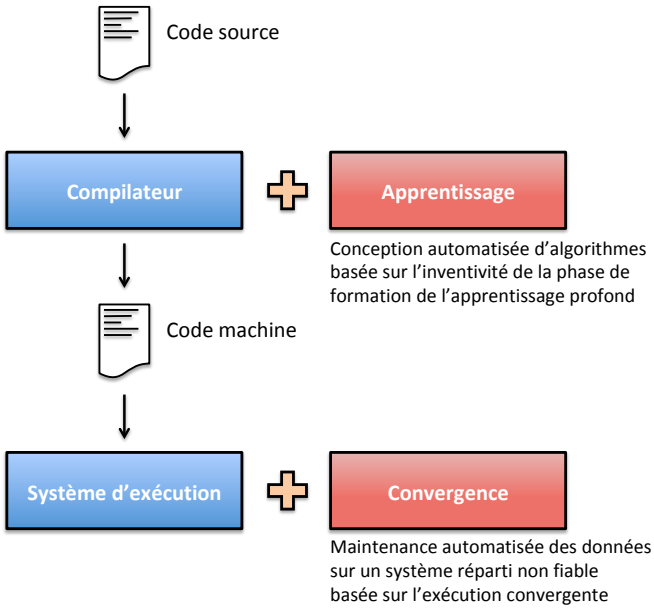
L'étincelle #18: journal de la création à l'IRCAM
Avril 2018

La programmation du futur

Peter Van Roy
Université catholique de Louvain

Les systèmes de programmation subissent actuellement deux changements profonds que je propose d'appeler la *convergence* et l'*apprentissage*. A la différence des améliorations incrémentales que l'on peut observer dans les différents langages de programmation (langages multi-agents, programmation réactive, langages fonctionnels...), ces changements constituent des véritables révolutions qui permettront aux langages du futur d'exprimer facilement des algorithmes beaucoup plus complexes pouvant s'exécuter sur des infrastructures matérielles ayant à leur tour une plus grande complexité.



La convergence ou la maintenance automatisée des données

Il a toujours été difficile pour un logiciel de représenter fidèlement le monde réel à travers des éléments d'une base de données, celles-ci n'offrant qu'une perspective statique et limitée de la vérité changeante du monde réel. Le logiciel doit à tout moment laborieusement corriger ces données pour qu'elles restent valables. Dans l'avenir, cela changera complètement grâce à une correction automatiquement des données et de leur validité. Il s'agit d'un nouveau type de programmation que l'on peut appeler la programmation *convergente*.

La programmation convergente est le résultat de la complexité croissante des fonctionnalités des logiciels qui sont de plus en plus répartis, c'est-à-dire étalés sur de nombreux ordinateurs liés par un réseau de communication. Le passage à la programmation convergente implique un changement paradigmatique dans notre rapport avec l'ordinateur, surtout en ce qui concerne la véracité de ses données. Loin de représenter des anomalies, les défaillances d'un système réparti deviendront des événements normaux qui font partie de la vie du logiciel et qui pourront même avoir une utilité non négligeable. La propriété de

« cohérence convergente », généralisant celle de « cohérence à terme » (*eventual consistency*) [1] dans un environnement de programmation convergente, exprime la possibilité de faire converger de façon continue les résultats des calculs vers des résultats corrects par rapport aux données vraies, à savoir celles qui sont les plus proches du monde réel : elles ne sont pas dans la base de données, mais au bord du système, dans des dispositifs de l'Internet des Objets, chez les clients et à l'aide tout type de capteurs. Dans un logiciel de type convergent, le contenu de la base de données converge continuellement et automatiquement vers les données vraies du monde réel [2]. Il s'agit donc non seulement d'une programmation qui tolère naturellement les défaillances mais dans laquelle les pertes de messages et les partitionnements de réseau ont pour seul effet de ralentir la convergence et pas de produire des erreurs. Ceci est vrai si l'on suppose que l'état d'exécution existe toujours quelque part dans le système.

Depuis quelques années, la convergence est devenue un sujet de recherche dans la communauté des systèmes répartis. A titre d'exemple, on peut citer notre projet de recherche « LightKone » centré sur la construction d'un système convergent appelé Lasp (*Lattice Programming*) [3,4]. Il est basé sur une structure de données convergente qui s'appelle un CRDT (« Type de Données Répliqué sans Conflit » ou *Conflict-free Replicated Data Type*) [5]. Dans Lasp, on peut combiner des CRDTs pour faire une exécution complète tout en gardant la propriété de convergence. Nous développons Lasp pour faire des applications qui marchent bien sur des infrastructures où les défaillances sont nombreuses, en particulier dans l'Internet des Objets. Notre système actuel s'exécute avec succès sur 1024 ordinateurs en réseau [6].

L'apprentissage ou la conception automatisée des algorithmes

L'apprentissage automatique constitue la deuxième révolution au sein des langages de programmation. Parmi les formes les plus populaires d'apprentissage automatique on peut citer l'« apprentissage profond » (*deep learning*) qui a connu une série de succès majeurs à partir de 2011 et cela dans plusieurs domaines tels la classification d'image (AlexNet), les langues naturelles (IBM Watson) ou encore le jeu de Go (AlphaGo). L'apprentissage profond a la capacité de reconnaître des formes hiérarchiques dans toutes sortes de données brutes (textes, dessins, images, vidéos, son, etc.), avec un niveau de qualité comparable aux êtres humains. Son formalisme se base sur des réseaux neuronaux multi-couches qui utilisent des fonctions non-linéaires dérivables. Les paramètres de ces réseaux sont déterminés lors d'une phase de formation qui utilise des techniques de rétropropagation [7]. La phase de formation demande une grande puissance de calcul qui est responsable de la très grande inventivité dans les solutions trouvées.

Grâce à la disponibilité de la puissance des calculs dans les GPUs et dans les smartphones, on assiste à une utilisation massive de techniques issues de l'apprentissage profond dans toute tâche de reconnaissance des formes. L'apprentissage profond joue également un rôle important dans le développement des logiciels, au point qu'il est en train de devenir une partie du génie logiciel. Dans les années à venir on assistera à une complexification des logiciels par l'ajout de composants intelligents à tous les niveaux, composants qui marcheront à cause précisément de l'inventivité de la phase de formation. Celle-ci n'est rien d'autre qu'une forme de *compilation*, comme la compilation du code source d'un logiciel. Dans l'avenir, l'apprentissage profond deviendra une partie du langage de programmation et sera utilisé de façon automatique comme c'est le cas pour la compilation au sein d'un logiciel.

Des résultats récents montrent que cette évolution améliorera grandement les capacités des

logiciels tout en réduisant l'effort de leur développement, l'apprentissage profond pouvant trouver des algorithmes parfois assez pointus - comme les algorithmes récursifs grâce à des couches récursives dans les réseaux neuronaux multi-couches - et cela dans plusieurs domaines du calcul informatique. Comme l'observe Thomas Breuel [8], l'apprentissage profond a montré sa puissance dans le domaine de l'algorithmique traditionnelle (par exemple dans des problèmes consistant à trouver le chemin le plus court dans un graphe), de la vision par ordinateur (par exemple dans la détection des contours de John Canny), la reconnaissance de formes, le traitement de signal, la théorie des jeux et la théorie de la décision. Dans tous ces domaines, un réseau neuronal profond peut trouver automatiquement une bonne solution lors de la phase de formation. Cela est dû premièrement à la structure du réseau neuronal multi-couches, qui donne un espace de solutions expressif, et deuxièmement, à la grande puissance de calcul utilisée lors de la phase de formation.

Conclusions

Les langages de programmation du futur permettront d'exprimer une complexité logicielle bien au-delà de ce que l'on peut faire aujourd'hui. Ils soutiendront une relation avec le monde réel basée sur la *convergence*, qui permet de gérer les données dans une infrastructure matérielle où les défaillances sont acceptées comme normales. Ils trouveront eux-mêmes des algorithmes complexes sans besoin de les programmer, en se basant sur l'inventivité apportée par l'*apprentissage*. Les preuves de concept pour la convergence et l'apprentissage existent déjà et on peut raisonnablement prévoir que dans les années à venir ces deux techniques feront de plus en plus partie des langages de programmation. C'est ainsi que la programmation passera véritablement à la vitesse supérieure !

Bibliographie

[1] Sebastian Burckhardt, *Principles of Eventual Consistency*, Foundations and Trends in Programming Languages, Now Publishers, octobre 2014.

[2] Christopher Meiklejohn, « A Certain Tendency of the Database Community », salon des Refusés (workshop colocalisé avec le congrès <Programming> 2017), Bruxelles, Belgique, 3-6 avril 2017.

[3] LightKone (Lightweight computations on the edge), Projet H2020, janvier 2017 - décembre 2019. Voir lightkone.eu.

[4] Christopher Meiklejohn et Peter Van Roy, « Lasp: A Language for Distributed, Coordination-free Programming », in *Proceedings of the 17th ACM International Symposium on Principles and Practice of Declarative Programming* (PPDP 2015), Sienne, Italie, juillet 2015.

[5] Marc Shapiro, Nuno Preguiça, Carlos Baquero et Marek Zawirski, *Conflict-free Replicated Data Types*, INRIA Rapport de Recherche RR7687, juillet 2011.

[6] Christopher Meiklejohn, Vitor Enes, Junghun Yoo, Carlos Baquero, Peter Van Roy et Annette Bieniusa, « Practical Evaluation of the Lasp Programming Model at Large Scale », in *Proceedings of the 19th ACM International Symposium on Principles and Practice of Declarative Programming* (PPDP 2017), Namur, Belgique, 9-12 octobre 2017.

[7] Ian Goodfellow, Yoshua Bengio et Aaron Courville, *Deep Learning*, MIT Press, 2016.

[8] Thomas Breuel. « Different Views of Deep Learning », International Summer School on Deep Learning, Université de Deusto et Université Rovira i Virgili, Bilbao, Espagne, 17-21 juillet 2017.