Joseph Sifakis : « Nous avons rendu l'informatique plus sûre »

Les systèmes informatiques sont devenus indispensables à nos activités quotidiennes comme à l'industrie. Mais peut-on toujours leur faire confiance ? La mise au point de méthodes qui garantissent leur fiabilité a été récompensée par le dernier prix Türing, la plus haute distinction en science de l'information.

LA RECHERCHE : Vous avez reçu le prix Türing pour la mise au point du *Model Checking*. De quoi s'agit-il ?
JOSEPH SIFAKIS : Prenons le cas du logiciel d'un contrôleur dans un avion. Il doit effectuer de nombreuses tâches indispensables. Par exemple, lorsque l'avion est en descente et atteint une altitude déterminée, il faut sortir le train d'atterrissage. Il est essentiel que ce logiciel soit parfaitement sûr, et il faut donc le tester de façon aussi complète que possible. Sans les méthodes de vérification comme le *Model Checking*, la seule solution était de tester ce programme dans toutes les situations possibles. Il s'agissait d'une vérification fastidieuse, empirique, et surtout peu sûre, car il n'est pas possible de parcourir par test exhaustivement toutes les combinaisons de valeurs des variables du programme. Leur nombre augmente exponentiellement avec le nombre des variables. D'où l'idée du *Model Checking :* on construit un modèle mathématique du logiciel, et on vérifie ce modèle plutôt que de tester le logiciel lui-même. Cela permet une exploration plus efficace de l'ensemble des configurations, et dans certains cas même lorsque leur nombre est infini.

Joseph Sifakis est directeur de recherche au CNRS. Il a fondé en 1993 le laboratoire Verimag, qu'il a dirigé jusqu'en 2006. Il a reçu en 2007 le prix Türing décerné par l'Association for Computing Machinery, en compagnie des Américains Edmund Clarke et Allen Emerson.

Est-ce le même principe que les modèles en ingénierie traditionnelle, utilisés par exemple pour la construction des ponts ?
JOSEPH SIFAKIS : Non, car l'ingénierie traditionnelle est fondée sur des lois physiques que l'on comprend bien, des équations générales… Ce n'est pas le cas en informatique : même si nous disposons de lois théoriques, on ne sait pas les résoudre, y compris pour des systèmes très simples. L'informatique est une discipline jeune qui n'est pas aussi prédictive que des disciplines comme le génie électrique ou la mécanique, fondées sur la physique. Le développement de systèmes informatiques complexes reste très empirique. Seuls 30 % des projets de grands systèmes informatiques atteignent leurs objectifs, 30 % échouent totalement, et 40 % échouent partiellement. Il n'existe pas toujours de moyen de prévoir, avant de le construire, si un système fonctionnera. Il faut donc le vérifier *a posteriori*.

Comment le *Model Checking* est-il né ?
JOSEPH SIFAKIS : Ma thèse, soutenue en 1979, portait sur des méthodes de vérification de systèmes finis. À l'époque, cette approche était qualifiée de « naïve » car on ne disposait ni d'ordinateurs assez puissants, ni de techniques permettant de vérifier des systèmes complexes. J'ai néanmoins développé un outil de vérification avec mon étudiant Jean-Pierre Queille, puis j'ai bénéficié de contrats de France Télécom car cette technologie s'appliquait bien aux protocoles de communication (les règles et procédures permettant aux machines de communiquer avec un langage commun). Durant tout ce temps, j'étais en concurrence avec Edmund Clarke et Allen Emerson, qui

L'ENTRETIEN DU MOIS
→ Joseph Sifakis

avaient commencé à travailler ensemble à Harvard, et avec lesquels je partage ce prix Türing. Mais nous avions beaucoup de difficultés pour faire accepter chacun nos articles, car nous nous heurtions au scepticisme quant à la faisabilité du *Model Checking*. Le seul moyen de publier était de créer notre propre conférence sur ce thème, ce que nous avons fait avec Edmund Clarke en 1989, à Grenoble. Nous avons réussi à faire venir des sociétés comme Intel, IBM et AT & T. Les premiers investissements des industriels, notamment d'IBM et d'Intel, dans les années 1990, ont montré que cette technologie était viable.

Nos algorithmes sécurisent les métros automatiques ou l'aéronautique

Que signifie exactement « vérifier un système informatique » ?
JOSEPH SIFAKIS : Il faut tout d'abord construire le modèle, puis détailler toutes les exigences auxquelles ce modèle doit répondre, et enfin construire un algorithme permettant de vérifier que le modèle satisfait ces exigences. Prenons l'exemple de la ligne 14 du métro, qui est automatique. Pour piloter chaque rame, il faut définir les variables à contrôler, tels la position de la rame sur le parcours, sa vitesse, son accélération, l'état des portes (ouvert ou fermé)… Il existe un très grand nombre de configurations (par exemple, la rame au kilomètre 10, roulant à 20 kilomètres à l'heure, portes fermées, est une configuration parmi d'autres), mais seules certaines sont autorisées. Les exigences opérationnelles préciseront, par exemple, qu'une porte doit être fermée dès que la vitesse de la rame est non nulle. Il faut aussi prendre en compte l'environnement du programme : la manière ⇨

⇨ dont on freine la rame ne dépend pas que des commandes informatiques, mais aussi de l'état des freins et du poids de la rame, plus ou moins chargée. Un modèle est une version simplifiée de ce système informatique, mais néanmoins fidèle. Vérifier le modèle, c'est appliquer un algorithme qui parcourt toutes les configurations possibles des variables du programme, et s'assure qu'elles répondent aux exigences opérationnelles.

Quelles sont les difficultés ?

JOSEPH SIFAKIS : Vérifier un logiciel est difficile, car c'est un objet mathématique, le nombre de calculs à effectuer pour sa vérification peut devenir infini. Imaginons un système fictif de boutons poussoir. Chaque bouton a deux états différents : allumé ou éteint. Si l'on a 300 boutons poussoirs, il existe 2^{300} états différents, soit plus de 10^{90} états ! C'est un chiffre qui dépasse le nombre d'atomes dans l'Univers connu. Il est impossible d'explorer chacun de ces états séparément.

Quelles sont les principales applications du *Model Checking* ?

JOSEPH SIFAKIS : Les premières applications ont concerné le matériel *(hardware)*. Il est plus simple à vérifier car c'est un système physique, donc ayant un nombre d'états fini, contrairement aux logiciels. De plus la vérification est rentable car le développement des puces électroniques est très coûteux. Intel et IBM, notamment, ont beaucoup utilisé le *Model Checking*. En France, de nombreux systèmes informatiques sont vérifiés par le *Model Checking*. On peut citer de nombreux protocoles de France Télécom, des systèmes de vol d'Airbus, ou encore le système de vol d'Ariane 5, après que le premier vol a échoué à cause d'un défaut dans un programme. Vérifier un système informatique coûte cher. Pour un ordinateur de bureau que l'on peut redémarrer à volonté il n'est pas nécessaire d'alourdir le coût de développement. En revanche, la vérification est indispensable pour les systèmes embarqués, c'est-à-dire les logiciels et composants électroniques « enfouis » dans les appareils, par exemple les avions, les téléphones portables, les voitures, les appareils ménagers ou médicaux… Aujourd'hui, plus de 95 % des puces produites sont embarquées, et ne peuvent pas être facilement changées en cas de défaillance, d'où la nécessité d'augmenter leur fiabilité. Ces systèmes embarqués sont très importants pour l'Europe, qui est forte dans les domaines des transports, de l'espace, des télécommunications, des technologies médicales…

Le *Model Checking* est crucial pour l'informatique embarquée

Quels sont les défis à relever pour cette informatique embarquée ?

JOSEPH SIFAKIS : Ces systèmes doivent être plus réactifs, plus autonomes et plus robustes que l'informatique « visible ». Et surtout vérifiables, notamment pour les systèmes dits « critiques », dont la défaillance peut avoir des conséquences catastrophiques. La mise au point de ces systèmes critiques suit une approche différente de celles des logiciels plus traditionnels, plus rigoureuse, car si l'on ne comprend pas comment ils sont faits, il est impossible de les vérifier. On ne peut donc pas utiliser des logiciels « propriétaires », dont le contenu des programmes reste un secret commercial. Bien sûr, concevoir un logiciel critique a un coût, entre cent et mille fois plus élevé que celui d'un logiciel non critique. Ce coût est d'ailleurs un frein pour certaines applications : les voitures pourraient être beaucoup plus sûres si des ordinateurs de bord faisaient de la « sécurité active », en intervenant entre le conducteur et les parties électromécaniques, comme cela se fait dans les avions. Des projets ambitieux ont été lancés dans ce sens, mais ont ils ont été abandonnés en raison du coût. L'objectif principal reste donc de fabriquer des systèmes critiques à moindre coût.

Quelles sont les limites du *Model Checking* ?

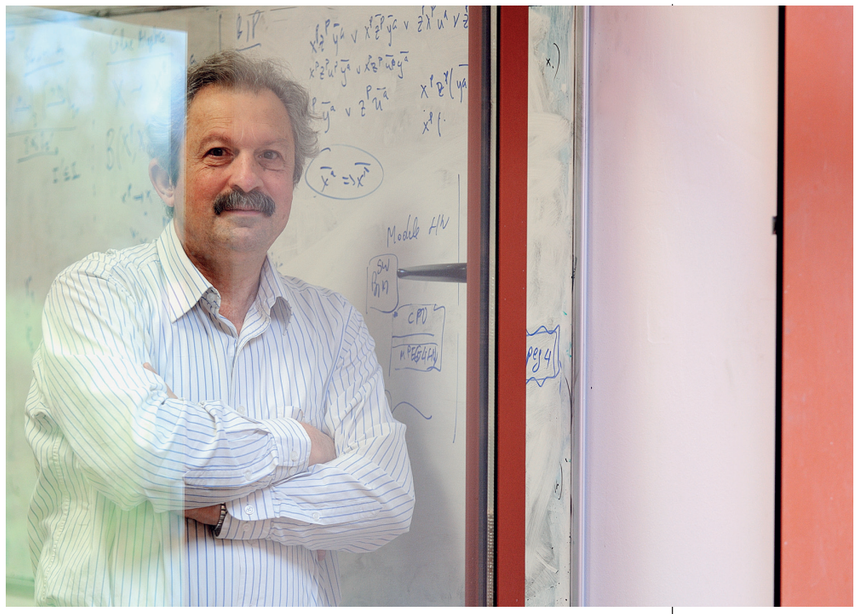
JOSEPH SIFAKIS : Tout d'abord, l'ensemble matériel-logiciel reste impossible à vérifier rigoureusement. Sinon, la principale limite est la complexité des systèmes. Le *Model Checking* permet d'explorer les systèmes de complexité moyenne, par exemple un processeur, ou un protocole de communication. Mais on ne sait pas extraire de modèle des systèmes d'exploitation, par exemple. On peut analyser partiellement les systèmes très complexes, tout dépend du type de propriétés que l'on souhaite vérifier. Il est également hors de portée de vérifier un ensemble d'ordinateurs travaillant de concert, formant un système de systèmes.

Ces limites sont-elles provisoires, ou inhérentes à la complexité des systèmes informatiques ?

JOSEPH SIFAKIS : À mon avis, certaines vérifications resteront impossibles. Les systèmes de systèmes sont trop complexes. Ainsi les États-Unis souhaitaient mettre au point un contrôle du trafic aérien de nouvelle génération. Actuellement, les routes et horaires des avions sont réservés à l'avance. C'est très inefficace : lorsqu'un avion prend du retard, l'ensemble du trafic aérien est retardé. L'idée des Américains était de faire un contrôle « dynamique » : chaque avion se voit allouer un trajet, qui est modifié en temps réel en fonction du trafic. Plus de 5 milliards de dollars ont été dépensés

L'ENTRETIEN DU MOIS
→ Joseph Sifakis

En informatique, on se heurte aujourd'hui à un « mur de la complexité »



pour l'étude de ce système, et tout a été arrêté au bout de dix ans, car les études ont montré que le projet était à la fois dangereux et irréalisable. De même, un projet d'autoroutes automatiques en Californie, où des systèmes automatiques prendraient le contrôle des voitures, s'est révélé infaisable, pour les mêmes raisons de complexité. Un autre exemple est le « Web des objets », dans lequel des appareils seraient accessibles par Internet. Il faut pouvoir s'assurer que les protocoles de communication sont suffisamment fiables pour garantir la sûreté et la sécurité des interactions.

Néanmoins, est-il possible d'améliorer le *Model Checking* ?

JOSEPH SIFAKIS : Les opinions divergent. On se heurte aujourd'hui à un « mur de la complexité ». Certains pensent qu'on peut vaincre cette complexité en construisant de meilleurs algorithmes. Pour ma part, j'ai quitté ce domaine il y a une dizaine d'années, afin de travailler sur une voie nouvelle, celle des « techniques compositionnelles ». Au lieu de vérifier un système globalement, on vérifie ses composants à l'aide du *Model Checking*, puis on construit une théorie permettant de vérifier que les assemblages de composants sont corrects. C'est une voie très active. On rejoint les méthodes de l'ingénierie plus traditionnelle, par exemple la construction, où l'on assemble des « briques ». Tout mon travail consiste à faire une théorie sur les propriétés de la « colle » unissant ces briques. Par exemple, l'une des questions clé est le partage des ressources du système. Lorsque plusieurs programmes tournent en même temps sur un ordinateur, chacun a besoin de mémoire. Si tous lancent un calcul en même temps, il peut y avoir un blocage dû à l'épuisement de la mémoire avant la fin de l'exécution d'un calcul. Mais si l'on ne fait tourner qu'un programme à la fois, on sous-utilise les ressources. Enfin, certains programmes doivent rester prioritaires : un programme lié à la sécurité passera avant les autres. Les techniques compositionnelles visent notamment à définir les priorités et les règles d'ordonnancement dans l'exécution des programmes pour une meilleure utilisation des ressources.

Comment travaillez-vous avec les industriels ?

JOSEPH SIFAKIS : En informatique, la recherche doit aussi s'inspirer des problèmes industriels. L'informatique évolue avec les technologies du matériel. Nous étudions aujourd'hui des questions auxquelles nous n'aurions jamais songé il y a cinq ans. Par exemple, comment programmer efficacement des puces ayant une centaine de processeurs ? Il faut développer des méthodes de programmation spécifiques. Mais en France, la relation avec les industriels n'est pas à la hauteur de ce qu'on pourrait attendre. Il existe beaucoup de projets collaboratifs, subventionnés par l'État et l'Europe, mais peu de relations directes entre un laboratoire public et un industriel, car alors, c'est l'industriel seul qui doit financer cette recherche. La recherche publique est très évaluée, mais ce n'est pas le cas de la recherche et développement de l'industrie, qui reçoit pourtant beaucoup de moyens de la part des pouvoirs publics : crédit d'impôt recherche, pôles de compétitivité… Je ne suis pas certain qu'elle en fasse le meilleur usage.

Quelles sont les implications du prix Türing pour vous ?

JOSEPH SIFAKIS : Pour la communauté scientifique qui connaissait déjà mes travaux, c'est une confirmation. Le prix n'apporte pas de soutien public direct et peu de collaborations scientifiques supplémentaires. En revanche, je suis davantage sollicité par les médias et les pouvoirs publics. ■■ **Propos recueillis par Cécile Michaut**

Reportage
Photos :
Guillaume Atger/
fedephoto.com