COMITE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Conseil scientifique de l'Institut des sciences de l'information et de leurs interactions (INS2I)

Recommandations relatives au séminaire thématique « Sureté, Vérification & Confiance »

Pouvant mener potentiellement à la catastrophe, les défaillances des systèmes informatiques coûtent considérablement à la société. Au plan humain tout d'abord on pourra rappeler en exemple la tragédie du Thérac 25 où en six incidents au moins trois personnes succombèrent à des doses massives de radiations conséquences en particulier d'une situation de concurrence critique dans le logiciel de contrôle et de façon plus générale d'une mauvaise conception rendant difficile sa vérification. En ce qui concerne le plan économique, certaines études estiment à plusieurs milliers de milliards de dollars le coût de la mauvaise qualité du logiciel aux États-unis en 2022.

On peut différencier les catégories de défaillances en fonction de leur origine.

Une première famille concerne les problèmes dus à des erreurs d'implantation ou de conception. La chasse aux bugs peut s'appuyer sur, d'une part les propriétés de sûreté des langages de programmation (typage fort, etc.) et d'autre part sur tous les apports des méthodes formelles, du test jusqu'à la preuve formelle assistée pour les applications critiques.

Une deuxième famille rassemble les fautes provoquées par un attaquant, c'est le registre de la sécurité. Là aussi les techniques de vérification formelle vont apporter des garanties souhaitables, tant au niveau de l'implantation des primitives cryptographiques qu'à celui des protocoles ou de leurs liens avec les interfaces utilisateurs.

Une troisième famille, enfin s'inscrit dans le cadre global de l'IA de confiance et résulte de divers facteurs comme les comportements inattendus des programmes reposant sur l'apprentissage automatique. Qu'ils soient soumis à un entraînement supervisé ou non ces programmes doivent proposer des réponses correspondant à ce qu'on en attend et dans certains cas pour les raisons qu'on attend.

Pour faire le point sur ces problématiques le CSI a invité:

- David Pichardie, professeur à l'ENS Rennes, actuellement chez META : « Analyse sémantique des logiciels par assistants de preuve et analyse statique »
- Karthikeyan Bhargavan directeur de recherche Inria Paris : « High-Assurance Cryptographic Programs »
- Jean-Michel LOUBES, professeur à l'université Toulouse Paul Sabatier : « Panorama sur l'IA de confiance : enjeux et acteurs en France ».

J.-M. Loubes intervient au sujet de l'IA de confiance dans le cadre de l'apprentissage automatique, c'est-à-dire de la minimisation d'une fonction de distance entre une prédiction et une observation

pour toutes les données. Les applications de l'apprentissage sont particulièrement sensibles : on peut les voir en particulier dans le contexte de la conduite autonome et des activités policières.

Obtenir une IA de confiance est garantir que les décisions prises par le système le sont pour de bonnes raisons et des raisons explicables or même si on prétend garantir que le monde algorithmique des prédictions correspond aux données, il reste une séparation nette entre les données et le monde réel et entre le monde réel et le monde idéal modélisé. Ces différences induisent des biais dans le processus d'apprentissage, dans la définition des variables qui sont pertinentes ou non (voire qui ne devraient pas être considérées), dans les performances réelles de l'algorithme en action, enfin dans la confusion entre corrélation et causalité. Il est fondamental de savoir détecter ces biais.

Une approche souvent prônée dans la littérature repose sur le choix de modèles intrinsèquement interprétables pour les applications critiques, même si ceux-ci ne présentent pas à l'heure actuelle les même performances que les modèles d'apprentissage statistique conventionnels. L'explicabilité de tels modèles repose sur l'utilisation d'approximations localement interprétables, une meilleure compréhension de l'importance des différentes variables et l'analyse contrefactuelle du comportement de ces modèles.

Le concept de bornes est également crucial car s'il est en effet illusoire d'atteindre une confiance totale dans un modèle complexe il est en revanche envisageable de borner son comportement. Interprétabilité n'est pas enfin pas correction : la confiance passe aussi par le défi de l'implantation, c'est-à-dire de la conformité de ce qui est effectivement exécuté par rapport à la définition de l'algorithme. Une réflexion sur les langages à utiliser et leurs liens avec les outils du calcul (optimisations absconses, non-déterminisme, variété des architectures matérielles...) est engagée par la communauté.

David Pichardie décrit à cette fin différents aspects de la vérification et l'articulation de ses approches, illustrés par des travaux à l'intersection des langages, de la vérification et de la sécurité. La compilation certifiée est le premier domaine présenté. Il s'agit de certifier correcte, à partir de la définition des sémantiques formelles du langage source et du langage cible, et à l'aide d'un assistant à la preuve interactif (dans les travaux présentés, l'assistant Coq), la conservation de la sémantique des codes par le compilateur, voire à en extraire le code correct par construction. Cette approche supporte en particulier des extensions à différentes formes d'optimisation.

Si la preuve formelle (avec assistant) assure que le comportement sera correct dans les cas d'application nominaux l'analyse statique s'intéresse quant à elle à la recherche de familles de bugs spécifiques, c'est-à-dire à la détection de cas où l'on n'est plus dans le contexte nominal. Dans ce cadre contraint, l'expressivité limitée autorise une automatisation des tâches : les approches par analyse statique sont d'ailleurs considérées comme "presse bouton". Se pose alors la question de la correction de l'outil automatique lui-même. On peut aborder la certification de l'analyse statique en établissant formellement que l'analyseur est correct ou bien en garantissant que ses résultats sont corrects par une validation de leur trace. Enfin des propriétés de sécurité, par exemple, la résistance aux attaques de type "cache timing", peuvent être établies par la mise en évidence de la propriété sur le code source puis sa conservation par l'étape de compilation. Ces techniques permettent de dépasser une garantie qui n'était généralement assurée que par une forme de discipline de programmation qu'il est illusoire de pouvoir maintenir sur des développements complexes et en interaction.

Karthikeyan Bhargavan poursuit sur la problématique d'environnements complexes à fort enjeu, en particulier les applications à base cryptographique.

La période est faste pour la cryptographie, nouvelles technologies, nouvelles approches et omniprésence des applications (internet bien sûr avec https, messageries sécurisées, essor des cryptomonnaies et blockchains). Cet engouement s'appuie toutefois sur des techniques entièrement nouvelles et fondées sur des résultats académiques dont peu, hormis les quelques experts du domaine, ont une compréhension profonde. Dans ces conditions, quelle confiance accorder aux systèmes cryptographiques ? Le besoin cryptographie à hautes garanties est bien là.

Vérifier la correction fonctionnelle d'une application de sécurité commence bien sûr par un travail de spécification ; l'effort de preuve est, lui, très dépendant de la façon dont le code a été développé. Un des freins à une généralisation de la vérification aux applications existantes est en effet le volume de l'existant, à savoir la quantité de code qui n'a pas été écrit dans une perspective de certification et qu'on n'est pas prêt à récrire à cette fin. Et encore ne s'agit-il ici que des cœurs critiques (primitives cryptographiques, protocoles et gestion des sessions et clefs) représentant par exemple pour la messagerie Signal environ 10% du développement total. Certains problèmes peuvent être éliminés par analyse statique (typiquement la sortie des cas nominaux) mais la correction fonctionnelle reste une affaire de spécialiste sur un code adapté. Les organismes de standardisation demandant aujourd'hui des preuves (IETF/TLS), il est de plus en plus fait usage de (méthodologies adaptées et de) langages spécialisés dans certains développements actuels tant un code qui n'a pas été écrit pour la vérification est (à peu près) impossible à vérifier.

L'approche proposée ici consiste à 1) écrire du code générique, 2) vérifier la correction du code générique, 3) spécialiser le code et compiler avec des techniques comparables à compcert vers la cible matérielle considérée.

Aux écueils déjà mentionnés s'ajoutent en effet la diversité des architectures matérielles cibles et les subtiles optimisations pour maintenir des performances élevées (ralentir l'exécution des protocoles chiffrés dans les communications internet par exemple aurait un coût insupportable). Il convient de mentionner également la quasi impossibilité d'obtenir les spécifications exactes des matériels, gardées par le secret industriel.

Conclusions et recommandations

Un des problèmes soulignés par les exposés est la dépendance à un corpus de code toujours plus imposant et dont établir la correction fonctionnelle est hors de portée. Les techniques d'analyse statique permettent de garantir qu'on reste dans les plages de conditions d'utilisations nominales, condition préliminaire à toute tentative de preuve de correction fonctionnelle), encore faut-il que celles-ci soient clairement définies et annoncées.

Le séminaire met, ainsi, en avant le besoin d'une nécessaire transparence pour assurer la confiance (au sens sûreté) : quelles sont les usages acceptables ? quelles sont les conditions nominales ? et par là quelle notion de vérification adopter ? Il insiste sur une discipline de développement qui demanderait que les nouveaux outils logiciels fondamentaux soient développés avec leur vérification formelle en perspective ; il regrette la difficulté d'accès aux réelles spécifications des plateformes matérielles, souvent couvertes par le secret industriel.

L'opacité des développements est aggravée par une complexité croissante et de moins en moins maîtrisée de l'écosystèmes des langages et des matériels. Favoriser et définir des langages avec de bonnes propriétés et en particulier des systèmes de types offrant de solides garanties font partie des moyens relevés au cours du séminaire pour aider à en maintenir le contrôle.

Les conditions de transparence remplies, c'est finalement, comme le montrent les exposés, la composition des approches de vérification qui permet des avancées remarquables dans l'obtention de garanties fortes en s'adressant aux différents niveaux des développements : spécifications, élimination des situations anormales (analyse statique), correction fonctionnelle (preuve formelle), conservation de la sémantique (compilation certifiée…). On ne peut que constater l'importance cruciale de ces spécialités dont l'impact dépasse largement le contexte des applications critiques ainsi qu'une démocratisation de leur usage.

Enfin, l'omniprésence de l'IA et en particulier l'apprentissage automatique à tous les niveaux de la société et de l'économie créée un défi de premier plan qui appelle clairement l'émergence de nouvelles techniques pour assurer et mesurer le niveau de confiance. Ceci implique des travaux sur le front des méthodes explicables. Ces derniers ne doivent pas être limités au cadre des applications critiques : ils sont également indispensables pour différents usages infiltrants comme actuellement les modèles de langages. Largement affectés par le phénomène d'hallucination, ces modèles interpellent dans leurs usages envisagés tels que le conseil (technique, scientifique, juridique, etc.) mais aussi l'enseignement.

Le conseil scientifique de l'INS2I recommande donc qu'un effort significatif soit consenti dans les politiques scientifiques des divers établissements pour que le front de la science avance sur ces sujets dont l'attente économique et sociétale est majeure.

Il souligne l'importance cruciale d'une sensibilisation à la vérification : à travers les disciplines de développement enseignées, en encourageant le développement et l'utilisation de langages appropriés offrant certaines garanties de sûreté.

Il rappelle enfin le besoin d'une véritable transparence tant dans les cadres d'utilisation que dans les spécifications réelles des outils, sur les plans logiciel et matériel. À titre d'exemple, le conseil attire l'attention sur les initiatives de type open-source hardware qui constituent un pas intéressant dans cette direction.

Gilles SASSATELLI
Président du Conseil scientifique de l'INS2I

Recommandation adoptée le 8 décembre 2023
Vote : 20 oui / 20 votants

Destinataires :

- M. Antoine PETIT, président-directeur général du CNRS
- M. Alain SCHUHL, directeur général délégué à la science du CNRS

- Mme Adeline NAZARENKO, directrice de l'Institut des sciences de l'information et de leurs interactions (INS2I)
- M. Olivier COUTARD, président du Conseil scientifique du CNRS
- M. Fabien JOBARD, président de la Conférence des présidents du Comité national (CPCN)
- Mme Christine ASSAIANTE, porte-parole de la Coordination des responsables des instances du CoNRS (C3N)
- Mme Claudine GILBERT, présidente du CSI de l'Institut de physique (INP), M. Olivier DRAPIER, président du CSI de l'Institut national de physique nucléaire et de physique des particules (IN2P3), M. Serge SIMOENS, président du CSI de l'Institut des sciences de l'ingénierie et des systèmes (INSIS), Mme Beatrice MARTICORENA, présidente du CSI de l'Institut national des sciences de l'Univers (INSU), M. Olivier SANDRE, président du CSI de l'Institut de chimie (INC), Mme Nathalie VIENNE-GUERRIN, présidente du CSI de l'Institut des sciences humaines et sociales (INSHS), M. Yaël GROSJEAN, président du CSI de l'Institut des sciences biologiques (INSB), M. Remi CARLES, président du CSI de l'Institut national des sciences mathématiques et de leurs interactions (INSMI), Mme Patricia GIBERT, présidente du CSI de l'Institut écologie et environnement (INEE)