A N D S I

**Association Nationale des Dirigeants en Sciences de l'Information**
**www.andsi.fr**

# Les méthodes formelles

Compte rendu rédigé par ANDSI & Pierre Delort

## En bref…

Olivier HERMANT est enseignant-chercheur en informatique à l'école des Mines de Paris. Invité par Pierre DELORT, président de l'ANDSI, il présentera un exposé sur l'application des méthodes formelles, dans lesquelles la France possède une expertise particulière, aux domaines de la sûreté (transports de masse, nucléaire civil), de la sécurité (processeurs, passeports biométriques) et des mathématiques.

*L'Association Nationale des Directeurs des Systèmes d'Information organise des débats et en diffuse des comptes rendus, les idées restant de la seule responsabilité de leurs auteurs. Elle peut également diffuser les commentaires que suscitent ces documents.*

### Introduction

On se demande si pour une propriété P et un programme p, on a P(p). Le théorème d'incomplétude de Kurt Gödel a démontré que toute chose avait des limites. Rice a quant à lui démontré que toute propriété sur les programmes qui est sémantique et non triviale est indécidable.

Le cahier des charges ne peut pas être rempli intégralement. Un compromis peut être fait sur l'exhaustivité, l'automatisation, l'incomplétude ou la précision, c'est-à-dire qu'on accepte d'obtenir de faux positifs ou de faux négatifs.

L'explosion de la fusée Ariane en 1997 met en évidence qu'il n'est pas toujours possible de procéder à un test avant la mise en œuvre.

### De la preuve

La conjecture de Kepler propose une preuve calculatoire. Les ordinateurs utilisés dans un cadre logique se révèlent plus endurants que les mathématiciens pour vérifier une preuve.

L'outil Coq a fait preuve de son efficacité démonstrative depuis 1995. Il permet de vérifier la preuve de manière mécanique et automatisable à l'infini. Si son degré de certitude est extrêmement élevé, il n'est pas fiable à 100 %. Bien que puissant, son cadre logique peut se révéler incohérent. De surcroît, il n'est pas possible de démontrer que le formalisme Coq est cohérent. Des bugs peuvent apparaître à cause de l'implémentation, du compilateur, du système d'exploitation, du microprocesseur ou d'un rayon cosmique. Toutefois, les risques peuvent être minimisés en vérifiant l'implémentation au moyen d'autres systèmes.

La logique de Hoare suppose que si la propriété P (précondition) est vraie, alors la propriété Q (postcondition) sera vraie après exécution de l'instruction s. Le calcul peut être effectué de manière presque automatique, sauf dans le cas des boucles.

## Des programmes corrects par construction

Pour ce qui est des applications pratiques, par exemple, le bug de la pédale d'accélérateur de la Toyota Camry aux États-Unis a coûté 1,2 milliard de dollars faute d'utilisation de la méthode formelle. Plus on attend pour résoudre un bug, et plus son coût augmente. Pour cette raison, il est préférable de l'identifier au moment de la conception.

En France, l'Agence nationale de la sécurité des systèmes d'information (ANSSI) délivre des certificats fondés sur des critères communs. Le niveau de sûreté le plus élevé requiert une certification formelle du code, par exemple, dans les domaines du transport, du nucléaire ou des cartes à puce.

Le principe consiste à écrire la spécification et à générer le code automatiquement ou par raffinements successifs. La méthode B est utilisée en France depuis les années 1990. Fondée sur la théorie des ensembles, elle consiste à générer des milliers d'obligations de preuve qui sont démontrées de façon automatique.

**PDT :** Qu'est-ce qu'une obligation de preuve ?
**OH :** Il s'agit, par exemple, de démontrer une métrique qui décroît.

L'approche sceptique invite à douter de la fiabilité des prouveurs automatiques en leur demandant des certificats. Les preuves sont alors vérifiées dans un système séparé.

Pour augmenter la confiance dans la base de confiance elle-même, il est possible de faire appel à la compilation certifiée. L'outil CompCert permet de vérifier que l'assembleur généré respecte la sémantique du code source. Sa fiabilité lui vaut d'être notamment utilisé par Airbus.

**Int :** Combien de personnes ont-elles été nécessaires pour développer CompCert ?
**OH :** Vingt personnes durant cinq à sept ans ont œuvré à sa conception.

Les tests sont impossibles pour la conception matérielle. Les vérifications doivent donc être menées le plus en amont possible.

Les langages synchrones sont aussi utilisés. Leur hypothèse de départ est que le système informatique réagit instantanément aux événements extérieurs. Ils sont beaucoup utilisés pour inventer des circuits.

## De la modélisation

Dans un système réactif, un programme ne part jamais seul, et doit donc être placé dans un environnement. Les équations différentielles du système lui-même peuvent être modélisées pour vérifier que la consigne donnée, par exemple à un avion, sera bien suivie, mais d'autres programmes concurrents peuvent être utilisés. L'aspect temporel est primordial. Les prédicats sont vérifiés avec le *model checking* qui est automatique.

## De l'analyse

L'objectif de l'interprétation abstraite est l'analyse des valeurs dans un programme. L'exemple typique est la vérification que celui-ci n'admet pas de division par zéro.

La guerre du Golfe témoigne de l'importance de la vigilance à l'égard des bugs informatiques. Des missiles Scud sont tombés sur un camp militaire américain, car les soldats avaient omis de redémarrer la machine (batterie de missiles Patriot) tous les jours pour la remettre à zéro, et éviter ainsi la dérive dans le temps des nombres flottants. Par la suite, une remise à zéro automatique de leur programme toutes les vingt-quatre heures a été mise en place. Mais lorsqu'il des avions militaires US se sont rendus d'une base à Hawaï à Okinawa

(Japon), la ligne de changement d'heure a été franchie et leur système de géolocalisation a dysfonctionné. Il eût été plus pertinent (mais amenant d'autres risques) de prévoir une synchronisation avec l'horloge atomique.

La sûreté de fonctionnement constitue une application primaire, étudiée et comprise depuis les années 1960-1980. Les spécifications d'un programme sont vérifiées particulièrement pour ses propriétés fonctionnelles. Les modèles pour la sécurité informatique possèdent trois composantes : la confidentialité, l'intégrité et l'authenticité.

Pour la cryptographie, il est possible de modéliser et de mécaniser des types d'attaques en étendant la logique de Hoare.

Les travaux sur la confidentialité sont moins nombreux. S'il est possible de formaliser la *differential privacy*, il ne sera pas possible de statuer a posteriori sur la présence dans le *data set*.

**PDT :** La *differential privacy* demande l'introduction de bruit dans les données. C'est la seule méthode (relativement à k-anonimity ou l-diversity) qui est sûre relativement à une réidentification.
**OH :** Certains s'attachent à tracer les données pour démontrer que le système ne peut pas être corrompu.

La modélisation de protocoles est bien maîtrisée, et utilisée pour la communication, la sécurité ou les cartes à puces. Il a été prouvé qu'elle était non interférente, c'est-à-dire qu'une application ne peut pas interférer avec une autre, ce qui constitue une spécificité des cartes à puces françaises.

L'aspect semi-formel consiste à formaliser un texte. Cette méthode est utilisée pour protéger les passeports, sur lesquels les clefs de sécurité privée sont dérivables du chiffre du bas.

L'analyse de code binaire avec le code Binsec permet de mettre en évidence l'absence d'escalade de privilèges et de détecter les vulnérabilités. Il possède aussi la capacité de modéliser les attaques physiques et de démontrer que le code y résiste.

## Conclusion

Il est donc possible de démontrer qu'un programme fonctionne correctement sur le fondement des mathématiques et de la logique. Au XXI$^{e}$ siècle, l'écriture de programme ne saurait être laissée à la main de l'Homme. Les pays de l'Union européenne disposent d'une véritable expertise à ce sujet. Il n'existe pas de bug informatique, mais seulement des erreurs humaines. Or l'ordinateur les amplifie toujours. Pour les éviter, il est nécessaire de hausser le niveau d'exigence et de formaliser.

### Débat

**Int :** Cette approche commence-t-elle à être prise en compte dans les logiciels de gestion vendus sur le marché ?
**OH :** Microsoft a effectué un grand pas en avant en changeant son langage de développement (scripts d'Excel). Un langage doté d'assurances statiques constitue également un avantage. De manière générale, les outils qui incorporent des éléments formels éliminent la plupart des bugs.

**Int :** À la SNCF, tous les codes étaient intégrés nativement dans les années 1980 et 1990. Les coûts de développement ont été réduits dans les années 2000 en diminuant le niveau d'expertise des informaticiens. La logique qui prévaut désormais est l'acceptation des bugs et leur résolution a posteriori pour les téléphones et les ordinateurs. J'ai l'impression qu'aujourd'hui l'informatique se divise en plusieurs catégories : le code

critique, le code d'utilité commune, et le code généré par l'IA. Or on ne peut pas prouver la validité d'un code généré par une IA.
**OH :** On peut envisager de demander à l'IA de générer des preuves.
**Int :** Pour les systèmes électroniques, l'IA a généré une règle qui échappe à la pensée humaine à partir d'une façon de penser émergente. La preuve est difficile à rapporter dans ce cadre.
**OH :** Il est possible de demander à un système formel de vérifier que la preuve est correcte.

**Présentation des orateurs**

Olivier Hermant, Ingénieur ENSTA & PhD. Université de Paris Cité est Professeur au CRI de Mines Paris-PSL et chercheur associé INRIA. Il enseigne et conduit des recherches sur les méthodes formelles.