Vérification automatique de programmes

David Delahaye

Faculté des Sciences
David.Delahaye@lirmm.fr

Master Informatique M2 2021-2022

Pourquoi les méthodes formelles ?

Développement de systèmes critiques

Systèmes critiques : système dont la panne peut avoir des conséquences dramatiques (morts, dégâts matériels importants, conséquences graves pour l'environnement).

Domaines d'application critiques :

- Transports : avions, trains, automobiles ;
- Production d'énergie : contrôle des centrales nucléaires ;
- Santé : chaînes de production de médicaments, appareil médicaux ;
- Système financier : paiement électronique ;
- Domaine militaire.

Est-ce rare d'avoir des bugs dans les systèmes critiques ?

Bug ? Vous avez dit bug ?

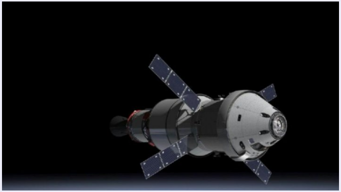
« catastrophes » dues à des erreurs informatiques

- 2003 : arrêt en cascade et simultanées de 256 centrales électriques en Amérique du Nord.
  50 millions de foyers privés d'électricité, 11 morts,
  6 milliards de dollars de dégâts.
- 1985-1987 : dysfonctionnement du logiciel de la machine de radiothérapie Therac 25.
  Surdosage de radiations (jusqu'à 100 fois la dose de radiations).
  Mort directe de 6 patients.
- 2012 : problème dans le logiciel algorithmique de passage d'ordre de l'entreprise Knight capital group.
  Fortes fluctuations de l'action, perte de 440 millions de dollars.
  L'entreprise a frôlé la faillite, recapitalisation en urgence.

Vous avez besoin d'un exemple plus récent ?

Bug ? Vous avez dit bug ?

Vaisseau américain Orion



- Premier vol habité d'Orion retardé à 2023 (17/09/2015) ;
- « Trop de paramètres et d'incertitudes sont en jeu pour permettre de déterminer si le vaisseau peut être prêt en 2021, a renchéri l'administrateur associé de la Nasa, Robert Lightfoot, lors d'une conférence de presse téléphonique.
  "C'est très loin d'être certain car des choses peuvent arriver, comme l'expérience nous l'a montré", a-t-il ajouté, citant notamment des "inconnus" comme des problèmes potentiels dans la mise au point des logiciels. »

Méthodes formelles

Différents méthodes

1. Le typage des langages de programmation est historiquement une des premières méthodes formelles.
2. La vérification déductive consiste à donner une représentation purement logique et sémantique à un programme.
3. Le « model-checking » analyse exhaustivement l'évolution du système lors de ses exécutions possibles.
4. L'analyse statique par interprétation abstraite calcule symboliquement un sur-ensemble des états accessibles du système.

Dans ce cours, nous verrons (1) et (2).

Preuves formelles

Plusieurs pré-requis

- Avoir une bonne connaissance de la sémantique de son langage ;
- Être capable d'exprimer cette sémantique formellement ;
- Savoir spécifier précisément le comportement de son programme ;
- Faire en sorte que la spécification soit totale.

Plusieurs langages en jeu

- Le langage de programmation ;
- Le langage de spécification ;
- Le langage de preuve.

Si ces trois langages sont réunis au sein du même environnement, c'est beaucoup plus pratique pour le développeur !

Spécification

Qu'est-ce que c'est ?

- C'est le « quoi » du programme, ce qu'il doit faire ;
- Peut-être exprimé dans le langage naturel (mais ambigu) :
  - Exemple : « ce programme calcule la racine carrée ».
- Plus formellement : spécification = type d'un programme.

Plusieurs degrés de spécifications

- Spécifications partielles :
  - Exemple : `sqrt : float` $\rightarrow$ `float` ;
  - Donne de l'information mais pas assez ;
  - Beaucoup de fonctions ont ce type (pas seulement racine carrée).
- Spécifications totales :
  - Exemple : $\forall x \in \mathbb{R}^+ . f(x) \geq 0 \wedge f(x) \times f(x) = x$;
  - Seule racine carrée vérifie cette proposition ;
  - Nécessite un langage basé sur la logique.

Preuves

Objectifs

- Mettre en adéquation un programme et sa spécification ;
- Apporter une garantie sur l'exécution du programme.

Remarques

- Plus simple de faire des preuves sur des programmes fonctionnels ;
- Fonctionnel utilisé aussi pour encoder les preuves dans certains outils ;
- Outils basés sur du fonctionnel : Coq, HOL, PVS, etc. ;
- Outils basés sur de l'impératif : Atelier B.

Une preuve triviale

Spécification

- On cherche à écrire une fonction f telle que :
  $\forall x \in \mathbb{N}. f(x) = x \times x$

Programme

- On considère le programme (fonction) suivant :
  $g(x) = x \times x$

Preuve d'adéquation

- On doit prouver que le programme g vérifie la spécification :
  $\forall x \in \mathbb{N}. g(x) = x \times x$
- On « déplie » la définition de g :
  $\forall x \in \mathbb{N}. x \times x = x \times x$
- Ce qui est trivial.

Une preuve plus difficile

Spécification

- La même que précédemment, c'est-à-dire :
  $\forall x \in \mathbb{N}. f(x) = x \times x$

Programme

- On considère le programme (fonction) suivant :
  $$h(x, i) = \begin{cases} x, & \text{si } i = 0, 1 \\ x + h(x, i-1), & \text{sinon} \end{cases}$$
  $$g(x) = h(x, x)$$

Preuve ?

- Par récurrence (exercice pour la semaine prochaine).

Mécanisation des preuves

Outils d'aide à la preuve

- Beaucoup d'outils existants ;
- Développés par des équipes de recherche ;
- Mais pas que : Atelier B (Alstom, ClearSy) ;
- Mécanisation ne signifie pas automatisation :
  - Outils de preuve interactive (Coq, HOL, etc.) ;
  - Outils de preuve automatique (Vampire, Zenon, etc.).

Transfert industriel ?

- Difficile au début ;
- Investit les milieux R&D progressivement ;
- Plus facile si l'outil vient d'une initiative industrielle (Atelier B) ;
- Plusieurs succès académiques récents changent la donne.

Quelques succès marquants

Ligne de métro 14 (Meteor)

- Ouverte en 1998 ;
- Développée par Siemens (Matra à l'époque) ;
- Utilisation de la méthode B ;
- Théorie des ensembles en première page des journaux ;
- Siemens continue à développer des métros sans conducteur.

Quelques succès marquants

JavaCard (Gemalto)

- Formalisation de l'architecture JavaCard ;
- Utilisation de Coq.

Le compilateur certifié CompCert (Inria, Gallium)

- Produit du code assembleur pour PowerPC, ARM, et x86 ;
- Développé en Coq et certifié correct.

Projet L4.verified (NICTA)

- Formalisation du micro-noyau seL4 ;
- Utilisation d'Isabelle/HOL.

Formalisation des Mathématiques

Théorème des 4 couleurs

- Énoncé :
  - Coloriage de n'importe quelle carte découpée en régions connexes ;
  - Deux régions adjacentes (ou limitrophes), deux couleurs distinctes ;
  - Utilisation de seulement 4 couleurs.
- Prouvé en Coq par Gonthier et Werner (2005).

Classification des groupes finis

- Travaux entre 1955 et 1983 ;
- Théorème « énorme » : 500 articles par plus de 100 auteurs ;
- Preuve « patchée » à plusieurs reprises ;
- Preuve du théorème de Feit-Thompson en Coq par Gonthier (2012) ;
- Preuve complète encore en cours.

Outil d'aide à la preuve Coq (vu en HAI821I)

Caractéristiques

- Développement par l'équipe Inria πr^2 ;
- Preuve de programmes fonctionnels ;
- Théorie des types (calcul des constructions inductives) ;
- Isomorphisme de Curry-Howard (objets preuves).

Implantation

- Premières versions milieu des années 80 ;
- Implantation actuelle en OCaml ;
- Preuve interactive (peu d'automatisation) ;
- En ligne de commande ou avec l'interface graphique CoqIDE ;
- Installer Coq : `https://coq.inria.fr/`.

Plusieurs familles d'outils d'aide à la preuve

Selon la théorie utilisée

- Théorie naïve des ensembles de Cantor (1872), Frege (1879) :
  - Théories incohérentes ;
  - Paradoxe de Russell (1903) : $\{x \mid x \notin x\}$.
- Deux théories alternatives :
  - Théorie des ensembles de Zermelo-Fraenkel (1908) ;
  - Théories des types de Whitehead-Russell (1910).
- Exemples d'outils :
  - Théorie des ensembles : Z, B, Alloy ;
  - Théorie des types : Coq, Isabelle, PVS.

Plusieurs familles d'outils d'aide à la preuve

Selon la logique utilisée

- Logique classique : tiers exclu accepté, $A \vee \neg A$;
- Logique intuitionniste :
  - Rejet du tiers exclu ;
  - Initiée par Brouwer (1930), puis ses étudiants Glivenko, Heyting, Gödel et Kolmogorov ;
  - Initiative vivement critiquée par Hilbert : « Priver le mathématicien du tertium non datur [pas de troisième possibilité] serait enlever son télescope à l'astronome, son poing au boxeur » ;
  - Interprétation de Brouwer-Heyting-Kolmogorov.
- Exemples d'outils :
  - Logique classique : B, PVS, HOL.
  - Logique intuitionniste : Coq, Nuprl, Alfa.

Peut-on automatiser les preuves ?

Question à se poser

- Le problème est-il décidable ?
- Problème posé par Hilbert et Ackermann en 1928 ;
- « Entscheidungsproblem » (problème de décision).

Réponse

- La réponse est négative ;
- Preuves par Church et Turing (indépendamment) en 1936 ;
- Utilisent les résultats de Gödel (théorèmes d'incomplétude) de 1931.

Mais tout n'est pas perdu...

Fragments décidables

- Logique propositionnelle (classique et intuitionniste) ;
- Arithmétique linéaire ;
- Réels (corps réels clos) ;
- Géométrie ;
- Tableaux (informatique).

Pour le reste

- Méthodes de recherche de preuve correctes et complètes ;
- Mais non terminantes.

Preuve automatique

Logique propositionnelle

- Problème décidable (même en intuitionniste), mais forte complexité ;
- Solveurs SAT ;
- DPLL, CDCL et variantes.

Logique du premier ordre (calcul des prédicats)

- Problème indécidable (semi-décidable) ;
- Tableaux, résolution (superposition).

Raisonner modulo théories

Qu'est-ce qu'une théorie ?

- Ensemble d'axiomes (axiomatisation) ;
- Exemples : arithmétique, théorie des ensembles, etc.

Deux grandes familles

- Solveurs SMT (solveur SAT + théories) ;
- Déduction modulo théorie, superdéduction.

Automatiser le raisonnement en Coq ?

Difficultés

- Logique d'ordre supérieur :
  - Indécidable (validité) ;
  - Même l'unification est indécidable !
- Système de types très élaboré :
  - Polymorphisme ;
  - Types dépendants ;
  - Types inductifs.
- Le manque d'automatisation est le prix à payer pour l'expressivité.

Organisation du cours

Plan du cours

1. Rappels en logique propositionnelle et en logique du premier ordre.
2. Déduction automatique en logique propositionnelle.
3. Déduction automatique en logique du premier ordre.
4. Raisonnement modulo égalité.
5. Raisonnement modulo arithmétique linéaire.
6. Raisonnement modulo théories (SMT).
7. Raisonnement modulo théories (déduction modulo théories, etc.).
8. Outils de déduction automatique.

Informations pratiques concernant le cours

Pré-requis et MCC

- Avoir eu un cours de logique (du premier ordre).
- UE dans la parfaite continuité de l'UE de M1 HAI821I (Sécurité logicielle), anciennement HMIN229 (LMD4).
- MCC : 1/3 CC + 2/3 CT.

Supports de cours

- Disponibles sous Moodle (clé : « hai934i ;2021 ») :

  https://moodle.umontpellier.fr/course/view.php?id=23037