Introduction à la vérification formelle

- MOTIVATIONS.
- VÉRIFICATION FORMELLE.
- MODEL-CHECKING.

Systèmes embarqués : intérêt

- Dynamisent l'innovation des systèmes :
  - Télécommunication (téléphonie mobile),
  - Transport (contrôle d'équipements, pilotage) : Audi A8 passe de 3 MB de mémoire à 90 MB en une génération,
  - Militaire (commande des trajectoires de missiles),
  - Aéronautique (contrôle de satellite, simulation de vol, pilotage automatique),
  - Energétique : contrôle de centrales nucléaires.
  - Médical (supervision, contrôle de dosage) :
  - Industrie (contrôle-commande de procédés),
  - Vie quotidienne (domotique, contrôle-commande d'appareils, jeux).

Systèmes embarqués : contraintes

- Un système autonome destiné à être intégré (enfoui) dans un environnement à fortes contraintes (espace mémoire, ressources, consommation, temps réel, mobilité, coût, sécurité).
- Exemples : téléphone mobile, sonde spatiale.
- La plupart des systèmes embarqués sont qualifiés de critiques, lesquels ?

Systèmes critiques

- Des systèmes dont le bon fonctionnement dépend la vie d'êtres humains ou d'énormes sommes d'argent.

- En chiffres :
  - Therac-25 (juillet 85- avril 90) : disfonctionnement d'une machine de radiothérapie (overdose de radiation) $\Rightarrow$ 6 morts.

Problèmes connus

- Crash d'AT&T (15 janvier 1990) : panne du centre d'appel dû à une erreur de programmation 75 millions d'appels jamais rendus et American Airlines perd 200000 réservations.
- Ariane 5 (juin 1996) : lanceur crée pour placer des satellites sur orbite, il a subit un crash après 36 secondes de son lancement à cause d'une erreur dans la conversion d'un flottant sur 64 bits en un entier signé sur 16 bits $\Rightarrow$ 700 millions de $ de perte.
- Ecrasement de l'avion Korear Air (aout 1997) : panne informatique à l'aéroport de Guam $\Rightarrow$ 225 morts.

Problèmes connus

- Bogue de l'an 2000 a couté plus de 800 milliards de $.
- Portable Dell (2006) : problème de batterie fabriquée par Sony qui a dû remplacer 8 millions de batteries.
- Honda (Octobre 2011) a dû rappeler 2.5 millions de véhicules, un problème électronique et informatique.
- Bogue de l'année 2038 pour les logiciels utilisant la représentation POSIX du temps : ordinateur atteint sa limite le 19 janvier 2038 à 3 h 14 min 7 s.

Et si on se passait de méthodes formelles

- Les méthodes semi-formelles sont ambigües.
- Tests coûteux (de 50 à 70% du coût total) et insuffisants :
  - Efficace pour découvrir des erreurs.
  - Non exhaustif : détecte la présence d'erreurs mais pas leur absence !
- Non applicables pour les systèmes parallèles, temps réels et embarqués où aucun risque ne doit être pris.

Besoin en méthodes formelles pour Vérifier

- Quand le zéro défaut est visé, il faut adopter les méthodes formelles.
- Elles sont basées sur des fondements mathématiques, donc exemptes d'erreurs.
- Vérifient qu'un système informatique respecte des propriétés exigées.
- Un propriétaire d'une machine à café publique par exemple, ne voudrait pas que sa machine commence à distribuer du café sans recevoir de l'argent .

Vérification formelle : preuve à l'appui

- Des enquêtes ont montré le rôle de la vérification formelle pour relever des défauts dans :
  - le lanceur Ariane-5,
  - la sonde Mars Pathfinder (vaisseau pour l'exploration de la planète mars),
  - le processeur Intel Pentium 2,
  - la machine de radiothérapie Therac-25.

Historique et évolution

- La vérification formelle a commencé dans les années 60 (logique de Hoare, 1969) qui a mis l'emphase sur les preuves formelles : vérification d'assertions dans les programmes, pré- et post-conditions, invariants….
- Bekic (1971) introduit l'algèbre de processus pour modéliser les systèmes parallèles.
- Scott et Stanchy (1971) modélisent les programmes par des fonctions (entré→sortie) : sémantique dénotationnelle.

Historique et évolution

- Park et al. (1973) introduisent les opérateurs de point fixe permettant de modéliser des propriétés modales.
- Pnueli (1977) propose l'utilisation des logiques temporelles pour vérifier des programmes (logique LTL).
- Clarke et al. (1980) définissent la logique temporelle CTL.
- Kozen et Pratt (1982) introduisent le μ-calcul qui s'avère la logique la plus expressive pour la spécification des propriétés pour les programmes.

Historique et évolution

- Emerson et Lei (1985) définissent la logique CTL*.
- Alur et Dill (1989) définissent les automates temporisés.
- L'approche est restée cependant académique et partiellement adoptée par le génie logiciel (surtout le concept d'assertion est utilisé).
- La difficulté résidait dans la construction d'un modèle pour les logiciels.

Historique et évolution

- L'Intérêt des quelques géants tels que Microsoft, IBM et la NASA pour l'automatisation du processus de génération de ces modèle a changé la vision en génie logiciel.
- Depuis les années 90 à aujourd'hui un énorme pas a été franchi en recherche et développement sur les méthodes formelles.

Deux approches principales pour la vérification formelle

- Preuve automatique (vérification directe sur le code):
  - Exhaustive pour démonter l'absence d'erreurs.
  - Difficile de mettre en œuvre, couteuse en tempe.
  - Semi-automatisable
- Model-checking (vérification sur une abstraction du programme):
  - Méthode exhaustive et automatique.
  - Efficace si l'abstraction est bonne.
  - Attention à l'explosion d'états.
- Approches complémentaires et non concurrentes, sans oublier qu'elles ne remplacent pas les tests.

Model checking

- Le model checking a vu le jour principalement grâce aux travaux de Clarke, Emerson et Sifakis dans les années 80.
- C'est une technique de vérification qui explore tous les états possibles d'un modèle (abstraction) du système.
- Il vérifie que l'abstraction satisfait certaines propriétés.
- Ils sont capables de gérer des espaces d'états d'environ 10^9.
- Aujourd'hui, grâce à l'investissement dans la recherche sur les modèles, il est possible de vérifier des modèles atteignant jusqu'à 10^{476} états.

Model checking : preuve à l'appui

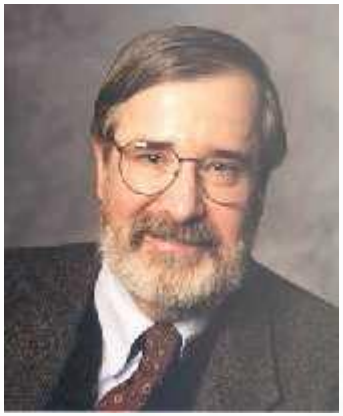
- Bang & Olufsen (outil UPPAAL) : détection d'une erreur dans un protocole de contrôle audio/vidéo.
- IBM : vérification de circuits, 24% de défauts trouvés par model checking dans la conception d'un adaptateur de bus mémoire dont 40% non trouvées par simulation.
- Deep Space 1 (sonde spatiale) : 5 erreurs non découvertes identifiées par model checking.

Model checking : preuve à l'appui

- Microsft (outil SIAM) : vérification de la sûreté et la sécurité de logiciels.
- Philips : audio-contrôle de protocoles de communication temporisés.
- Renault : supervision de fabrication
- Protocole Needham et Schroeder publié en 1978 :
  - Prouvé correct en 1989 par Burrows, Abadi, et Needham.
  - Prouvé erroné en 1995-1996 par Lowe : une preuve automatique, a été donnée en le modélisant en CSP et en utilisant le logiciel de model-checking FDR.

Model checking : preuve à l'appui

- Les chercheurs considérés comme les inventeurs du model-checking (E.M. Clarke, E.A. Emerson, et J. Sifakis) ont reçu en 2007 le prix Turing, équivalent d'un prix Nobel pour l'Informatique : http://www.journaldunet.com/solutions/ssii/interview/joseph-sifakis-prix-turing-2007-microsoft-et-google-utilisent-le-model-checking/



Exemples de propriétés vérifiées

- Sûreté : absence de deadlock, deux processus ne peuvent pas être en même temps en section critique.
- Vivacité : le programme ne bouclera pas à l'infini, la ressource finira par se libérer, une réponse arrivera sûrement dans moins de 5 minutes, exclusion mutuelle (le processus finira par entrer en section critique).
- Equité : l'ordre des entrées dans la section critique respecte l'ordre des demandes.

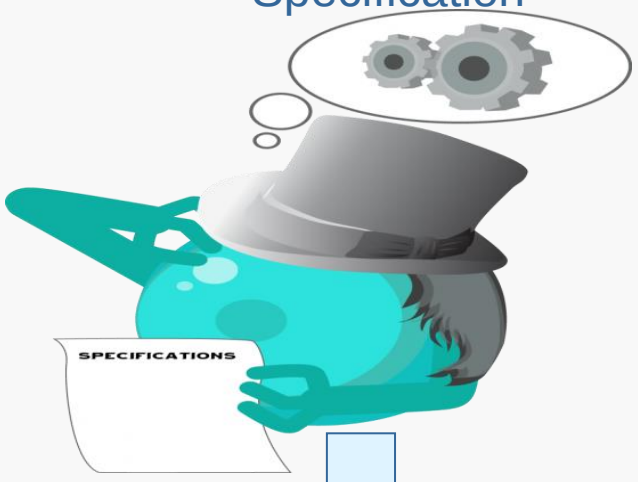
Mais …

- Qu’est ce que c’est que le model checking ?

Schéma du model checking

Système

Spécification



?

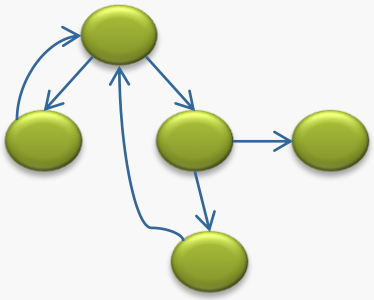
Satisfait

$\models$

ALGORITHME DE MODEL CHECKING

φ

Formule



Modèle

oui

Non + contre exemple → simulation → Localisation de l'erreur

Alors êtes-vous prêts ?