METHODES FORMELLES

M1

OBJECTIFS DU COURS

- Introduire la spécification et la vérification formelle dans la démarche de développement d'un logiciel.
- Connaître les principes généraux des méthodes formelles

PREREQUIS

- Génie Logiciel
- Conception de SI
- Programmation structurée

PROGRAMME

1. Introduction
2. **Spécification et vérification formelle de logiciels**
3. Langage de spécification : JML
4. Méthodes déductives : logique de Hoare
5. Vérification de modèles (ou modelchecking)
6. Preuves formelles
7. Méthodes B

I - INTRODUCTION

DÉFINITION

- Les **méthodes formelles** sont des techniques permettant de raisonner rigoureusement, à l'aide de logiques mathématiques, sur des programmes informatiques ou du matériel électronique, afin de démontrer leur validité par rapport à une certaine spécification.

- Elles sont basées sur les sémantiques des programmes, c'est-à-dire sur des descriptions mathématiques formelles du sens d'un programme donné par son code source.

- Ces méthodes permettent d'obtenir une très forte assurance de l'absence de bug dans les logiciels (Evaluation Assurance Level, Safety Integrity Level).

DÉFINITION

- Elles sont utilisées dans le développement des logiciels les plus critiques. Leur amélioration et l'élargissement de leurs champs d'application pratique sont la motivation de nombreuses recherches scientifiques en informatique.

- Elles sont coûteuses en ressources (humaines et matérielles) et actuellement réservées aux logiciels les plus critiques

EXEMPLE

En vérification formelle, on utilise des valeurs symboliques et on applique les règles qui régissent les opérations + et *.

Les règles pourraient être:

$$\forall x,\ x^2 = x * x \quad (R1)$$

$$\forall x, y, z,\ x * (y + z) = x * y + x * z \quad (R2)$$

$$\forall x, y,\ x * y = y * x \quad (R3)$$

$$\forall x,\ x + x = 2x \quad (R4)$$

$$\forall x, y,\ x + y = y + x \quad (R5)$$

- En se servant de ces règles, on arrive à montrer que :

$$(a + b)^2 = a^2 + b^2 + 2(a * b)$$

DÉMONSTRATION

$$
\begin{aligned}
(a+b)^2 &= (a+b)*(a+b) && (R1)\\
&= (a+b)*a + (a+b)*b && (R2)\\
&= a*(a+b) + b*(a+b) && (R3)\\
&= a*a + a*b + b*a + b*b && (R2)\\
&= a^2 + a*b + b*a + b^2 && (R1)\\
&= a^2 + a*b + a*b + b^2 && (R3)\\
&= a^2 + 2(a*b) + b^2 && (R4)\\
&= a^2 + b^2 + 2(a*b) && (R5)
\end{aligned}
$$

AVANTAGES DES MÉTHODES FORMELLES

- L'avantage principal des méthodes formelles est l'utilisation de concepts de la logique et de la technique mathématique.
- Ces concepts fournissent des outils effectif qui organisent les pensées des concepteurs et qui facilitent la communication entre toutes les personnes concernées par le développement. De plus, ils nous permettent de décrire de manière précise, non ambiguë, les demandes énoncées par l'utilisateur du système logiciel à réaliser.
- Les notions d'ensemble, de relation, de fonction et leurs différentes propriétés et opérations, avec les quantifications universelles et existentielles, nous permettent d'établir une spécification d'une manière simple et claire et de démontrer mathématiquement les propriétés de la spécification.

AVANTAGES DES MÉTHODES FORMELLES

- Les avantages techniques d'une spécification formelle par rapport à une spécification informelle, sont la précision et la clarté. Des imprécisions et des ambiguïtés peuvent facilement se glisser dans les spécifications informelles. Ceci peut ouvrir la voie à plusieurs interprétations. Par contre, les termes de spécifications formelles n'ont qu'une seule interprétation.
- Un autre avantage des spécifications formelles est que les questions sont posées et répondues avec précision et d'une manière scientifique.
- De plus, les méthodes formelles fournissent des spécifications qui peuvent être rigoureusement vérifiées, analysées et testées dès les premières étapes du cycle de développement, ce qui n'est pas le cas dans les méthodes informelles. Cela signifie qu'il possible de détecter et de corriger des fautes dès les premières étapes, ce qui réduit le coût et la durée du développement et améliore la qualité du logiciel.

AVANTAGES DES MÉTHODES FORMELLES

- Les spécifications rigoureuses jouent un triple rôle dans le développement du logiciel.
  - D'abord, les spécifications documentent avec précision les décisions de conception, indépendamment de l'implantation, et servent de base pour la revue de cette conception.
  - Durant l'implantation, les mêmes spécifications supportent le développement en parallèle. D'une part, elles renseignent les utilisateurs de ce qu'ils peuvent s'attendre et, d'autre part, elles renseignent les programmeurs de ce qu'ils doivent faire, et servent de base pour la phase de test.
  - Finalement, durant la maintenance, ces mêmes spécifications supportent l'analyse de changements et aident à la formation de nouveaux personnels.

INCONVÉNIENTS

- **Difficulté** : les praticiens n'ont pas toujours la connaissance nécessaire. Les formalismes ne sont pas toujours très lisibles ou accessibles. Malgré les progrès significatifs accomplis ces dernières années pour rendre les notations formelles plus compréhensibles et utilisables, la rédaction et l'emploi de spécifications formelles exigent toujours un certain niveau d'aptitude mathématique, ainsi qu'un effort important."
- **Pauvreté des processus et des outils support** : peu de méthodes présentent un processus clair et réaliste, un environnement complet et convivial fait défaut même si des prototypes existent. Les preuves sont souvent données de façon anecdotique dans les ouvrages. Le formalisme doit permettre autant que possible d'automatiser la preuve.

INCONVÉNIENTS

- **Diversité** : plusieurs standards existent (Z, VDM, modèles algébriques, OSDL, etc.).
- **Difficultés de modélisation** : manque de structuration, erreurs, exceptions, concurrence. On a besoin d'outils et de méthodes pour construire les spécifications complexes. La modélisation passe aussi par la capitalisation du savoir (modèles, méthodes utilisées).
- **Adéquation** : la spécification formelle aide à cerner les oublis et les contradictions, mais on ne peut prouver que ce qui a été spécifié, pas ce qui était (implicitement) souhaité par le client.

INCONVÉNIENTS

- **Preuve** : des outils de preuve existent (calcul de schémas en Z, réécriture dans les algèbres), mais ils sont insuffisants. Des études sont en cours pour les enrichir et donner les algorithmes de test et de preuve.
- **Domaine d'application** : actuellement, les spécifications formelles ne sont appliquées qu'à un nombre réduit d'applications. Certains éléments des langages de programmation (le parallélisme, l'arithmétique en virgule flottante, les structures de données complexes) sont encore difficiles à modéliser de manière satisfaisante."

INCONVÉNIENTS

- **raffinage** : la théorie du raffinage ou réification est relativement simple à comprendre : on transforme petit à petit les structures de données et de contrôle jusqu'à atteindre les langages de programmation. Il faut noter que le nombre d'étapes de raffinage peut être important et que la preuve du bien-fondé de la transformation est souvent très lourde.
- **structuration** : on a besoin d'outils de structuration des spécifications pour en simplifier l'écriture, l'accès, la preuve et la réutilisation. Certains outils existent (dérivation de schéma en Z, sous-typage et relation d'importation dans les spécifications algébriques) mais ne sont pas toujours suffisants.

USAGE INDUSTRIEL DES MF

- Les méthodes formelles sont utilisées dans l'industrie, notamment dans le domaine de l'énergie nucléaire. La station nucléaire Darlington utilise un système logiciel vérifié par la méthode formelle SCR (Software Cost Reduction), pour arrêter les réacteurs en cas d'urgence.
- Beaucoup d'applications ont été développées avec succès, en utilisant la notation Z. Parmi ces applications, on note la modélisation des oscilloscopes de Tektronix, la nouvelle version du moniteur de télétraitement CICS .
- Un autre exemple qui englobe l'utilisation de méthodes formelles est le développement du TRANSPUTER par le manufacturier de semi-conducteur Inmos. Il a été estimé que l'utilisation de méthodes formelles a permis de réduire le temps de développement de moitié par rapport aux méthodes traditionnelles.

USAGE INDUSTRIEL DES MF

- Les méthodes formelles ont été utilisées avec succès dans le domaine du transport, où la sûreté des automatismes est primordiale. Les logiciels tiennent une part importante dans le fonctionnement des systèmes de contrôle/commande du mouvement. L'utilisation de méthodes formelles, pour développer ce type de logiciels, est une solution au double impératif: la qualité et la sûreté de fonctionnement.
- Une des applications, parmi les applications les plus connues dans le domaine du transport, est celle du Métro de Paris, utilisant la méthode B.

DIFFÉRENTES APPROCHES ET CATÉGORIES

Différents corpus mathématiques ont été utilisés pour élaborer des raisonnements formels sur les logiciels. Cette diversité d'approche a engendré des « familles » de méthodes formelles.

- **Les méthodes basées sur le typage** des langages de programmation: considérée comme une des premières méthodes formelles, permet de restreindre des langages de programmes à des classes de programmes dont le comportement statique est assuré.
- **Les méthodes basées sur le *model checking*** *qui* analyse exhaustivement l'évolution du système lors de ses exécutions possibles. Par exemple, pour démontrer l'absence d'erreurs à l'exécution, on pourra tester l'absence d'états d'erreur dans l'ensemble des états accessibles du système. Il s'agit alors d'une forme de test exhaustif, mais mené à l'aide d'algorithmes astucieux permettant d'énumérer les états du système sous une forme symbolique économique.

DIFFÉRENTES APPROCHES ET CATÉGORIES

- **Les méthodes basées sur l'*analyse statique*** *par interprétation abstraite*, qui schématiquement, calcule symboliquement un sur-ensemble des états accessibles du système ;

- **Les méthodes basées sur la vérification déductive** (calcul de plus faible précondition, logique de séparation) consistant à donner une représentation purement logique et sémantique à un programme. Il s'agit alors de réduire le problème de la vérification des programmes à celui de démonstration de théorèmes de correction et de complétude. Ces théorèmes permettent de formaliser le comportement attendu (spécification) d'un programme et leur démonstration permet d'en affirmer leur vérité.

NIVEAUX D'UTILISATION DE MF

Les méthodes formelles peuvent être utilisées à plusieurs niveaux:

- **Niveau 0: spécification formelle** peut être entreprise et un programme développé à partir de cette manière informelle. Cela a été baptisée *Lite méthodes formelles* . Cela peut être l'option la plus rentable dans de nombreux cas.
- **Niveau 1: le développement formel** et la **vérification formelle** peuvent être utilisées pour produire un programme d'une manière plus formelle. Par exemple, des preuves de propriétés ou le **raffinement** de la **spécification d'un** programme peut être entrepris. Cela peut être le plus approprié dans les systèmes de haute intégrité impliquant la sécurité.
- **Niveau 2: Théorème expérimentateurs** peuvent être utilisés pour effectuer des preuves vérifiées de machine entièrement formelle. Cela peut être très coûteux et n'est pratiquement valable si le coût des erreurs est extrêmement élevé (par exemple, dans les parties critiques de la conception du microprocesseur).

FIN