Nom : Sujet : Jury : Page : /

1.- Exemples de méthodes et outils pour la correction de programmes

Prérequis : · Bases de sémantique (II)
· Bases de logique (II)
· Bases de programmation (I)

Niveau : L1 (I) / L3 (II)

Motivation : Aujourd'hui, les systèmes informatiques sont omniprésents. Il est alors nécessaire de s'assurer de leur bon fonctionnement, que ce soit d'un point de vue utilisateur (service), système critique (aviation), ...

I] Discipline et contrats

1.) Spécifications

Definition 1 : Une spécification est un ensemble explicite d'exigences à satisfaire par un programme.

Remarque 2 : Une spécification peut être formelle ou informelle selon les besoins.

Exemples 3 : · Un code embarqué en aéronautique aura une spécification formelle pour s'assurer de son bon fonctionnement.
· Un kernel n'a de manière générale de spécification formelle compte tenue du caractère changeant dû au matériel.

2.) Documentation

Définition 4 : La documentation est un texte associé à une fonction, un module, une librairie, ... explicitant ses prérequis, ses effets et ses garanties associées.

Remarque 5: Il faut bien différencier la documentation des commentaires servant à expliquer des parties du programme (les choix effectués, ...).

Activité 6: Documenter et commenter une librairie simple de manipulation de listes en Python, C et OCaml.

3.) Vérifications expérimentales

Proposition 7: Dans le cas d'une spécification informelle, il est nécessaire d'utiliser des outils de vérification exécutant le programme.

a.- Tests

Definition 8: Un test est une partie d'un programme, généralement une assertion ou une fonction, dédiée à la vérification expérimentale de la bonne exécution d'un programme.

Proposition 9: Les tests permettent de couvrir plusieurs niveaux de correction. On trouve de manière générale les tests unitaires pour des fonctions simples et les tests d'intégration pour un programme, un module, ...

Remarque 10: Les tests sont particulièrement utiles pour éviter les regressions, introduction d'un bug faisant qu'une fonctionnalité qui fonctionnait auparavant ne l'est plus.

b.- Débogage

Définition 11: Un outil de débogage est un outil d'instrumentation d'un code source permettant des exécutions contrôlées d'un programme pour observer précisément son comportement.

Exemples 12 : . gdb est un outil de débogage du langage C.
. valgrind est un outil de débogage de la mémoire.

Activité 13 : Utilisation du débugger Python.

II) Méthodes formelles de correction

1.) Logique de Hoare

a.- Notions théoriques

Definition 14 : Un triplet de Hoare $\{\varphi\}\ pg\ \{\psi\}$ est la donnée de deux formules logiques φ et ψ, appelées respectivement précondition et postcondition, et d'un programme pg.

Exemple 15 : Algorithme de tri :

$$\{in = arg\}\ tri(arg)\ \left\{\begin{array}{l}\exists\ \sigma \text{ permutation telle que } \sigma(in) = arg \\ \wedge\ \forall\ i \in [\![1 ; |arg|-1]\!],\ in[i-1] \leq in[i]\end{array}\right\}$$

Figure 16 : Partie d'un système de preuve sur les triplets de Hoare pour un langage impératif simple

$$\frac{}{\{\varphi\}\ x := e\ \{\varphi[x \leftarrow e]\}}\ \text{assign} \qquad \frac{\{\varphi\}\ pg^1\ \{\theta\} \quad \{\theta\}\ pg^2\ \{\psi\}}{\{\varphi\}\ pg^1 ; pg^2\ \{\psi\}}\ \text{seq}$$

$$\frac{\{I \wedge e\}\ c\ \{I\}}{\{I\}\ \text{while } e \text{ do } c\ \{I \wedge \neg e\}}\ \text{while}$$

Définition 17: On dit qu'un triplet de Hoare $\{\varphi\}\ pg\ \{\psi\}$ est prouvable et l'on note $\vdash \{\varphi\}\ pg\ \{\psi\}$ s'il existe un arbre de dérivation dont la conclusion est $\{\varphi\}\ pg\ \{\psi\}$.

Définition 18: On dit qu'un triplet de Hoare $\{\varphi\}\ pg\ \{\psi\}$ est valide et l'on note $\models \{\varphi\}\ pg\ \{\psi\}$ si
$\forall \sigma,\ \sigma \models \varphi \Rightarrow [\![pg]\!]\sigma \models \psi$.

Théorème 19: Soient φ et ψ des formules logiques et pg un programme. Alors, $\vdash \{\varphi\}\ pg\ \{\psi\} \iff \models \{\varphi\}\ pg\ \{\psi\}$.

Remarque 20: La logique de Hoare simple permet de :
- donner une spécification formelle aux programmes
- prouver ces spécifications, souvent de manière fastidieuse.
De plus, elle dépend fortement de la logique dans laquelle elle se place (logiques modales, ...).

b. – Utilisation en pratique

Proposition 21: La logique de Hoare utilisée seule peut rapidement donner des preuves complexes à réaliser à la main. Elle est généralement utilisée en base d'un système plus complexe (système de type, ...) ou avec un prouveur automatique.

Activité 22: Présentation du prouveur automatique Why3.

2/ Système de types

a. – Théorie

Définition 23: Un type est un ensemble de valeurs partageant des propriétés communes.

Exemples 24: int (nombres entiers), string (chaînes de caractères)

Remarque 25: Les types peuvent décrire des systèmes plus ou moins riches selon leurs définitions (types dépendants, polymorphisme, ...).

Définition 26: Un système de type est un ensemble de règles décrivant les opérations précises sur les types, permettant d'associer un type à un programme.

Exemples 27: Règles de typage

$$\frac{0 \leqslant n \leqslant 2^8}{n : \text{int}} \qquad \frac{\vdash u : A \to B \quad \vdash v : A}{\vdash u\ v : B}$$

```
Cons: 'a -> 'a list -> 'a list
```

Remarque 28: « Well-typed programs cannot go wrong. » - Milner
Cela signifie que dans un système de type suffisamment puissant, il est possible de garantir des propriétés sur un programme bien typé : pas d'erreur de segmentation, pas d'adressage non aligné, ...

Activité 29: Comparaison de programmes bien/mal typés en C et en Python.

b.- Pratique

Proposition 30: On différencie les langages de programmation faiblement typés des langages fortement typé. Ces derniers utilisent la théorie associée aux types pour garantir des propriétés.

Exemples 31 :
- C et JavaScript sont faiblement typés.
- OCaml, Rust et Python sont fortement typés.

Remarque 32 : Un typage fort demande un surcoût à la compilation ou l'interprétation du programme.

Proposition 33 : Il est possible, pour des systèmes de type simples, d'inférer le type d'une expression. Cela s'effectue par le principe sous-jacent d'unification.

DEV : Algorithme de Robinson

Remarque 34 : Le typage est également lié aux spécifications. En effet, les types des arguments et de sortie d'une fonction permettent de connaître plus précisément les prérequis et garanties associés à la fonction.