Vérification de Programmes

Spécification

Données	désignation	type	pré-condition
Résultats	désignation	type	post-condition

Vérification

Vérification dynamique	Aléatoire	Fonctionnelle	Structurelle
Vérification statique	Informelle	Formelle	

Vérification dynamique

Test	Couple (D,R_D)	Données	Résultats attendus pour D
Algorithme A	Test réussi : $A(D) = R_D$	$A(D) \neq R_D \rightarrow$ A contient erreur	

objectifs du test

Programme code et/ou spécifications — s**pécification + écriture**

résultats attendus

programme+ environnement — **Exécution**

Comparaison

\+ -

Archivage : test ; données ; résultats

Analyse : inductive ; déductive

Modification : programme ; test

Tests Typologie

Unitaire/ Module	*Simuler autres modules*
Intégration	*Intégration progressive des modules ; Test alpha (simulation de l'utilisateur)*
Recettes	*Fournis par le client*
Réception	*Test à la réception avec fournisseur ;* Test bêta (ouvert/fermé)
Non régression	Lors de la modification d'une partie assurer la cohérence avec les autres parties.

Tests Stratégies

Aléatoire	
Fonctionnelle	Couverture du domaine ; Un représentant ; Valeurs frontières
Structurelle	Couverture des instructions ; Couverture des arcs du graphe de contrôle
	Couverture des chemins du graphe de contrôle (Métrique de McCabe : |E|-|X|+2)
	Couverture des conditions élémentaires

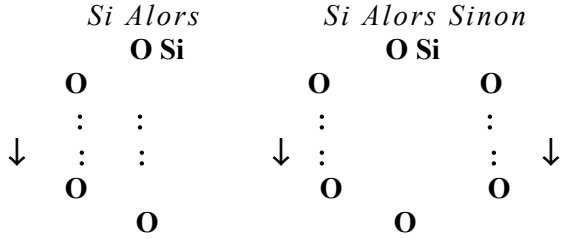
Exemple : Analyse fonctionnelle

Calcul des puissances positives (au sens large) de 2. Pour les petites valeurs, inférieurs (au sens large) à 100, on dispose d'une fonction donnée. Pour les grandes valeurs, inférieurs (au sens large) à 500, on dispose d'une autre fonction.

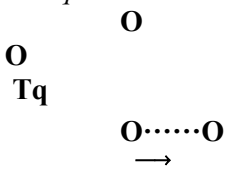
- Domaine Découpes explicites ?? Découpes implicites ??
- Jeux de Test

Exemple : Conditions élémentaires
Si (a > 0) et (b ≠ 0) **Alors** a ← a/b
Sinon
Si (x = 3) et (b = 0) **Alors** a ← 3*b **Fsi**
Fsi
Conditions élémentaires :: Interprétations

Graphe de contrôle
Affectation ; E/S ; Appel S-Prg



Tantque



Exemple : Analyse structurelle
Lire(a); Lire(b);
Tantque a≠b **Faire**
Si a > b **Alors** a ← a – b **Sinon** b ← b – a **Fsi**
Ftanque
Écrire(a) ;

Lire(a);
Si 0 > a **Alors** a ← - a **Fsi**
b ← Sqrt(a)

Lire(a); Lire(b);
Si a > 0 **Alors** c ← - a **Sinon** b ← b - a **Fsi**
Si b > 0 **Alors** c ← b **Sinon** a ← b **Fsi**

Vérifications statiques
Stratégies
Informelle (Equipes / durée / tâches)
- Simuler machine
- Check-list défauts usuels

Variables non initialisées ; allocation et libération de mémoire ; paramètres formels et effectifs ; récursivité et variables non protégées ; boucles infinies ; test d'égalité avec nombres flottants ; …

Formelle
Exécution symbolique : Logique de Hoare ; Weakest Précondition (p^3f) de Dijkstra

Exécution symbolique : La valeur symbolique d'une variable ou d'un ensemble de variables est un ensemble de valeurs définit par une expression logique

Exécution de A sur une valeur symbolique définie par {p} conduisant à une valeur symbolique définie par {q} : **{p} A {q}**

Exécution symbolique ≡ transformation d'expressions logiques

Exemple
m ← 0 ;
Tantque n ≥ (m+1)² **Faire** m ← m+1 **Ftanque**

Langage algorithmique
Syntaxe
Soient S_1, …, S_n des instructions : **« S_1 ; … ; S_n ; »** est une séquence d'instructions
Instructions <Var> ; <Expression> ; <Cond>
Affectation <Var> **:=** <Expression>
Conditionnelle : Si
Si <Cond> **Alors** <Séquence d'instructions> **Fsi**
Conditionnelle : Sisin
Si <Cond> **Alors** <Séquence d'instructions> **Sinon**<Séquence d'instructions> **Fsi**

Répétitive
Tantque <Cond> **Faire**<Séquence d'instructions> **Ftq**
Pour *i : = a* ***à*** *b [****pas*** *c]* ***Faire*** *<Séquence d'instructions>* ***Fpour***

Sémantique Usuelle
Restrictions
Variables synonymes (Pointeurs ; Tableaux (sauf avec adressage direct) ;Fonctions & Procédures

Logique des prédicats
1er ordre

Termes : Formules élémentaires booléennes : $x \in [1, ..n]$; $y \neq 3$; $x \leq 2y + 3$; vrai ; …
Variables : x, y, …
Opérateur logiques : $\neg$; $\wedge$; $\vee$; $\Rightarrow$; $\Leftrightarrow$
Quantificateurs : $\forall$; $\exists$
Symboles de liaison : « , » ; « (» ; «) » ; …

Règles d'inférence : $(\Im, F)$
- Utilisation : $(F \in \Im) \vdash F$
- Augmentation : si $(\Im \vdash F)$ et $(G \notin \Im)$ alors $\{G\} \cup \Im \vdash F$
- Synthèse : si $((\Im, G) \vdash F)$ alors $\Im \vdash (G \Rightarrow F)$
- Modus ponens : si $(\Im \vdash (F \Rightarrow G))$ et $(\Im \vdash F)$ alors $\Im \vdash G$
- Double négation : $(\Im \vdash F)$ si et seulement si $(\Im \vdash \neg\neg F)$
-Tiers exclu : si $((\Im, G) \vdash F)$ et $((\Im, G) \vdash \neg F)$ alors $\Im \vdash \neg G$
- Instantiation : si $(\Im \vdash \forall x\, F)$ alors $\Im \vdash F[x \leftarrow t]$
- Généralisation : si $(\Im \vdash F)$ alors $\Im \vdash \forall x\, F$ x libre dans $\Im$
- Existence : $(\Im \vdash \exists x F)$ si et seulement si $\Im \vdash \neg \forall x\, \neg F$

Syntaxe

Avec x variable , t terme , p et q prédicats, sont alors prédicats :

t ; $\neg p$; $p \wedge q$; $p \vee q$; $p \Rightarrow q$; $p \Leftrightarrow q$; $\forall x\ p$; $\exists x\ p$; (p)

Variable libre / Variable liée

p : $\forall i, (i \in [1 \ldots j]) \Rightarrow (T[i] \leq T[i+1])$ q : $(i \neq j)\ (\forall i, (i \in [1 \ldots j]) \Rightarrow (T[i] \leq T[i+1]))$

Sémantique

I : interprétation

I(p)	I(q)	I(p⇒q)
V	V	V
V	F	F
F	V	V
F	F	V

Avec I(t) définie pour tout terme t, on obtient :

- $I(\neg p) = \neg I(p)$
- $I(p \wedge q) = I(p) \wedge I(q)$; $I(p \vee q) = I(p) \vee I(q)$
- $I(p \Rightarrow q) = I(p) \Rightarrow I(q)$; $I(p \Leftrightarrow q) = I(p) \Leftrightarrow I(q)$
- $I(\exists x\ p) = V$ s'il existe une valeur de x telle que $I(p) = V$, $I(\exists x\ p) = F$ sinon
- $I(\forall x\ p) = V$ si pour toute valeur de x on a $I(p) = V$, $I(\forall x\ p) = F$ sinon

Formule ouverte / fermée (close)

Un prédicat est **valide** s'il est à vrai quelque soit l'environnement.

L'environnement est défini par les variables libres du prédicat au point de contrôle du programme où ce dernier apparaît.

p : $\forall i \forall j \forall k((i \in [1 \ldots j]) \wedge (k \in [1 \ldots j]) \wedge (i \leq k)) \Rightarrow (T[i] \leq T[k])$

6	**8**	**11**	**12**	**20**	**36**	**40**	**58**

6	**8**	**20**	**36**	**11**	**12**	**40**	**58**

q : $\forall i \forall j \forall k((i \in [j \ldots j]) \wedge (k \in [j \ldots j]) \wedge (i \leq k)) \Rightarrow (T[i] \leq T[k])$

Logique de Hoare

Système formel

Alphabet

Procédé de construction de mots/formules

Axiomes

Règles de déductions

$$\frac{m^1, \ldots, m^k}{m}$$

$m^1, \ldots, m^k$: prémisses

m : conclusion

Preuve de f : $(f_1, \ldots, f_n)$

$f_n = f$

f_i : axiome

f_i : $f_i = R_j(f_1, \ldots, f_{i-1})$;

On établit la preuve en partant du but

Logique de Hoare

Formules : {p} A {q}

- A algorithme écrit dans le langage présenté
- p et q prédicats de la logique du 1er ordre

Axiomes : $\{p[x := y]\}\ x := y\ \{p\}$ $\{p\}$ skip $\{p\}$

Règles de déduction :

[Seq]: $\dfrac{\{p\}\ A_1\ \{t\}\ ,\ \{t\}\ A_2\ \{q\}}{\{p\}\ A_1 ; A_2\ \{q\}}$

[Si]: $\dfrac{\{p \wedge c\}\ A\ \{q\}\ ,\ (p \wedge \neg c) \Rightarrow q}{\{p\}\ \textbf{Si}\ c\ \textbf{Alors}\ A\ \textbf{Fsi}\ \{q\}}$

[Sisin]: $\dfrac{\{p \wedge c\}\ A_1\ \{q\}\ ,\ \{p \wedge \neg c\}\ A_2\ \{q\}}{\{p\}\ \textbf{Si}\ c\ \textbf{Alors}\ A_1\ \textbf{Sinon}\ A_2\ \textbf{Fsi}\ \{q\}}$

[Tq]: $\dfrac{\{p \wedge c\}\ A\ \{p\}}{\{p\}\ \textbf{Tantque}\ c\ \textbf{Faire}\ A\ \textbf{Ftantque}\ \{p \wedge \neg c\}}$

Règles de conséquences :

[Pré]: $\dfrac{\{t\}\ A\ \{q\}\ ,\ p \Rightarrow t}{\{p\}\ A\ \{q\}}$ renforcement

[Post]: $\dfrac{\{p\}\ A\ \{t\}\ ,\ t \Rightarrow q}{\{p\}\ A\ \{q\}}$ affaiblissement

[Et]: $\dfrac{\{p\}\ A\ \{q\}\ ,\ \{p\}\ A\ \{t\}}{\{p\}\ A\ \{q \wedge t\}}$

[Ou]: $\dfrac{\{p\}\ A\ \{q\}\ ,\ \{t\}\ A\ \{q\}}{\{p \vee t\}\ A\ \{q\}}$

Théorème de **Cohérence** (Soudness) Si {p} A {q} est déductible alors il est valide.
Théorème de **Complétude relative** (Relative Completness) Si {p} A {q} est valide alors il est déductible.

Indécidabilité

Correction Partielle Pas de prise en compte de l'arrêt des répétitives

Correction Totale

[Tq+]: $\dfrac{\{p \wedge c \wedge (f(V(c)) = v_0)\}\ A\ \{p \wedge (f(V(c)) < v_0)\}}{\{p\}\ \textbf{Tantque}\ c\ \textbf{Faire}\ A\ \textbf{Ftantque}\ \{p \wedge \neg c\}}$

« Weakest Precondition » de Dijsktra
wp : plus faible précondition $\{p\}\ A\ \{q\}$ wp(A,q)

wp(A,q) : ensemble de tous les états initiaux tels que l'exécution de A débutant dans l'un de ces états termine dans l'état q.

$$\textbf{\{p\} A \{q\} ≡ ((p ∧ wp(A, vrai)) ⇒ wp(A,q))}$$

Règles

[**Aff**]: wp(x := y,q} ≡ q[x :=y]
[**Skip**]: wp(Skip,q} ≡ q
[**Et**] : wp(A, p∧q} ≡ wp(A, p) ∧ wp(A, q)

[**Ou**] : wp(A, p∨q} ≡ wp(A, p) ∨ wp(A, q)

[**Seq**]: wp(A_1;A_2, q} ≡ wp(A_1,wp(A_2, q))

[**Sisin**]: wp(**Si** c **Alors** A_1 **Sinon** A_2 **Fsi**, q} ≡ (c ⇒ wp(A_1, q)) ∧ (¬c ⇒ wp(A_2, q))

[**Tq**]: wp(**Tantque** c **Faire** A **Ftantque**, q) ≡ $p^3f(\mathbf{x} \rightarrow (\neg c \wedge q) \vee (c \wedge wp(A, \mathbf{x}))$

p^3f : plus petit point fixe

Méthodologie

$p_0 \equiv (\neg c \wedge q)$
$p_1 \equiv (c \wedge wp(A,p_0))$
$p_2 \equiv (c \wedge wp(A,p_1))$
:
:
$p_k \equiv (c \wedge wp(A,p_{k-1}))$
:
:
wp(**Tantque** c **Faire** A **Ftantque**, q) ≡ $\exists k \geq 0, p_k$