TESTS STRUCTURELS DE COUVERTURE
LE POINT DE VUE DU VOYEUR

VV031
v200b

2014-09-24

Librement traduit et adapté du chapitre 12 de *Software Testing and Analysis* de Pezzè et Young

Luc LAVOIE
Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Luc.Lavoie@USherbrooke.ca
http://info.usherbrooke.ca/llavoie

OBJECTIFS

- Comprendre les principes des tests structurels de couverture (TSC).
- Comprendre la complémentarité des tests fonctionnels et des TSC.
- Connaitre le vocabulaire associé au concept.
- Connaitre et pouvoir utiliser les principaux critères d'adéquation faible pouvant être mis en oeuvre grâce aux TSC.
- Comprendre la bonne utilisation et les limites des TSC.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

PRÉSENTATION DES TSC

Tests structurels de couverture

- Tests structurels (rappel)
- TSC (définition)
- Mesures
- Finalité
- Limites
- Complémentarité
- Usages
- Application pratique

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TEST STRUCTUREL (RAPPEL)

- Test construit sur la base d'informations obtenues de la structure du composant logiciel à tester.
- Synonymes : test en boîte blanche, transparente, de verre...
- En anglais : "white-box", "glass-box", or "code-based" testing.
- Le test structurel continue de dépendre d'exigences connues au préalable.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC
TEST STRUCTUREL DE COUVERTURE

- Test structurel visant à mesurer l'étendue d'un essai fonctionnel en regard du code effectif du module.
- L'étendue est comprise ici comme celle des « exécutions » réalisées (lors de l'essai) par rapport à toutes celles possibles.
- Une exécution est décrite par la séquence (la trace) des instructions effectuées.
- Pour réduire le nombre de cas des tests et déterminer des représentants parmi des exécutions analogues, on a recours à une classification des exécutions sur la base du graphe de contrôle du module (voir PY-5).

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC
MESURES

- Typiquement l'étendue de l'exécution (la couverture) peut alors être mesurée par rapport
  - aux instructions (blocs de base),
  - aux branchements,
  - aux conditions,
  - aux chemins.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC
FINALITÉ

- Déterminer s'il est souhaitable d'ajouter des tests ou des cas de test.
  - Si un essai n'exerce pas une partie du module, on en déduit qu'il n'est pas complet, que l'essai ne « couvre » pas le module.
- Utiliser différemment des cas de tests existants.
  - Il est plus facile et moins couteux de réutiliser des cas de tests existants que d'en élaborer de nouveaux.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC
LIMITES

- Une couverture de 100 % ne garantit pas de révéler tous les défauts.
  - L'exécution d'une instruction fautive n'induit pas nécessairement une erreur,
    - car elle peut livrer le bon résultat pour un sous-ensemble des données de test,
    - car l'erreur induite localement peut être corrigée globalement.
  - Une bonne couverture par rapport à un critère n'induit pas nécessairement une bonne couverture par rapport aux autres critères.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC
Complémentarité

- Certaines erreurs ne peuvent être détectées à partir des seules exigences.
  - Typiquement, lorsqu'une exigence est mise en oeuvre par plusieurs composants ou de plusieurs façons.
  - Exemple : collision lors d'une insertion dans une table dispersée (invisible par rapport aux exigences).
- Certaines erreurs ne peuvent être détectées même avec une couverture à 100 %.
  - Typiquement, un cas omis.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC
USAGES

- L'évaluation systématique et la maximisation de la couverture demeurent souhaitables
  - pour augmenter la confiance,
  - pour identifier explicitement les aspects non couverts,
  - pour détecter des erreurs ou des défauts non détectés par ailleurs (par application de l'axiome de diversité).

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC
APPLICATION PRATIQUE

- Construire l'essai fonctionnel puis identifier les cas de tests manquants grâce aux TSC.
- Interpréter
  - Les différences acceptables
    - entre la spécification et la mise en oeuvre.
  - Les lacunes devant être corrigées
    - incomplétude des exigences (cas particuliers non couverts),
    - incomplétude des cas de tests par rapport aux exigences,
    - inexactitude de la mise en oeuvre.
- Tester
  - En ayant automatisé la prise de mesure et, si possible, les tests eux-mêmes.
- Mesurer
  - Les taux de couverture sont de bons indicateurs
    - de progression,
    - de complétion (parfois),
    - qualité des tests (mais avec circonspection).

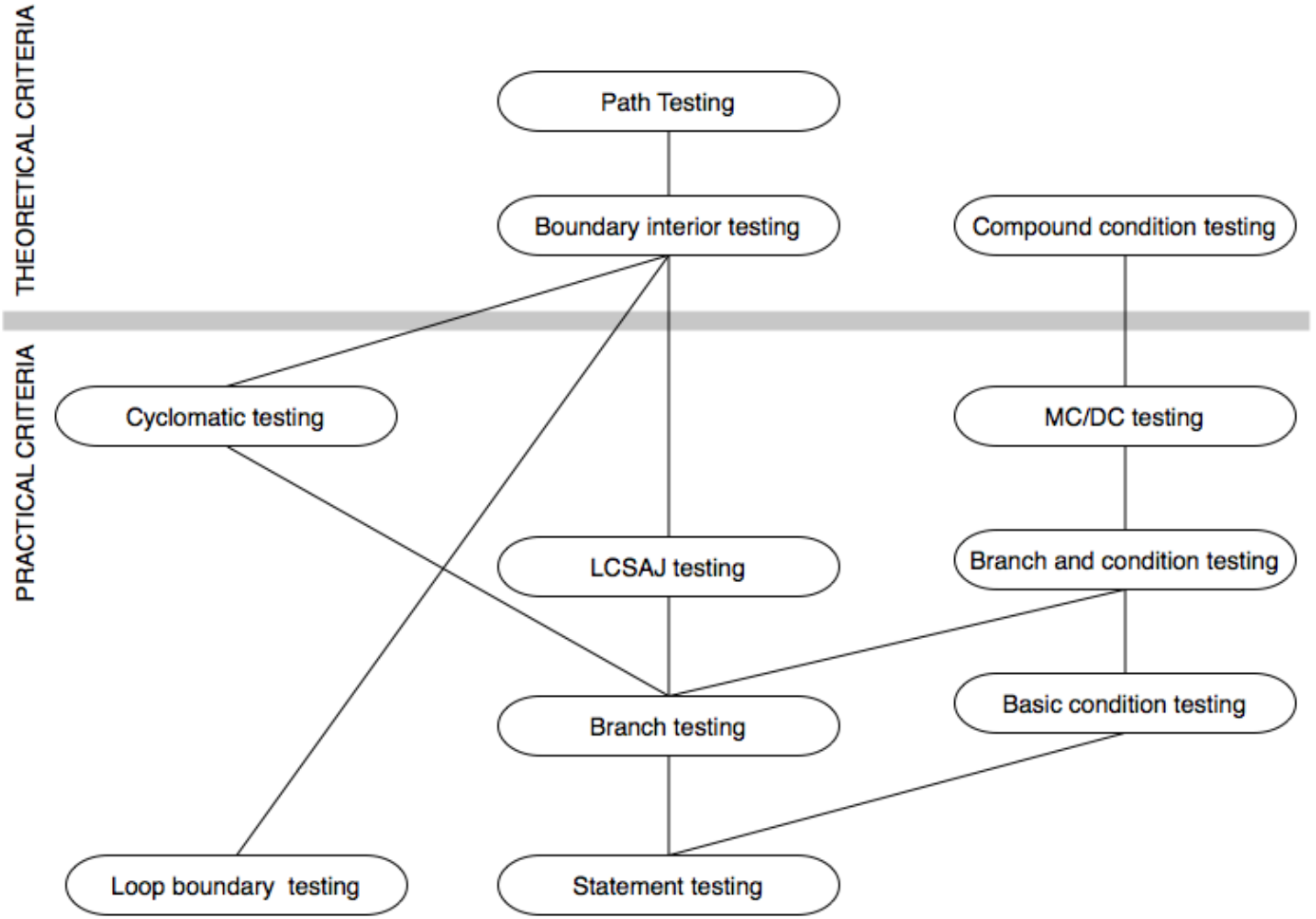
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

LES PRINCIPAUX CRITÈRES

Un survol

- C_{INS}
- C_{BLOC}
- C_{BR}
- C_{COND}
- C_{BC}
- C_{COMP}
- $C_{MC/DC}$
- C_{LCSAJ}

LES PRINCIPAUX CRITÈRES (UNE VUE D'ENSEMBLE)



TSC - C_{INS}
CRITÈRE PAR INSTRUCTIONS

- Principe (critère d'adéquation faible)
  - Toute instruction doit être exécutée au moins une fois.

- Mesure

$$\frac{\#\text{ instructions exécutées}}{\#\text{ instructions}}$$

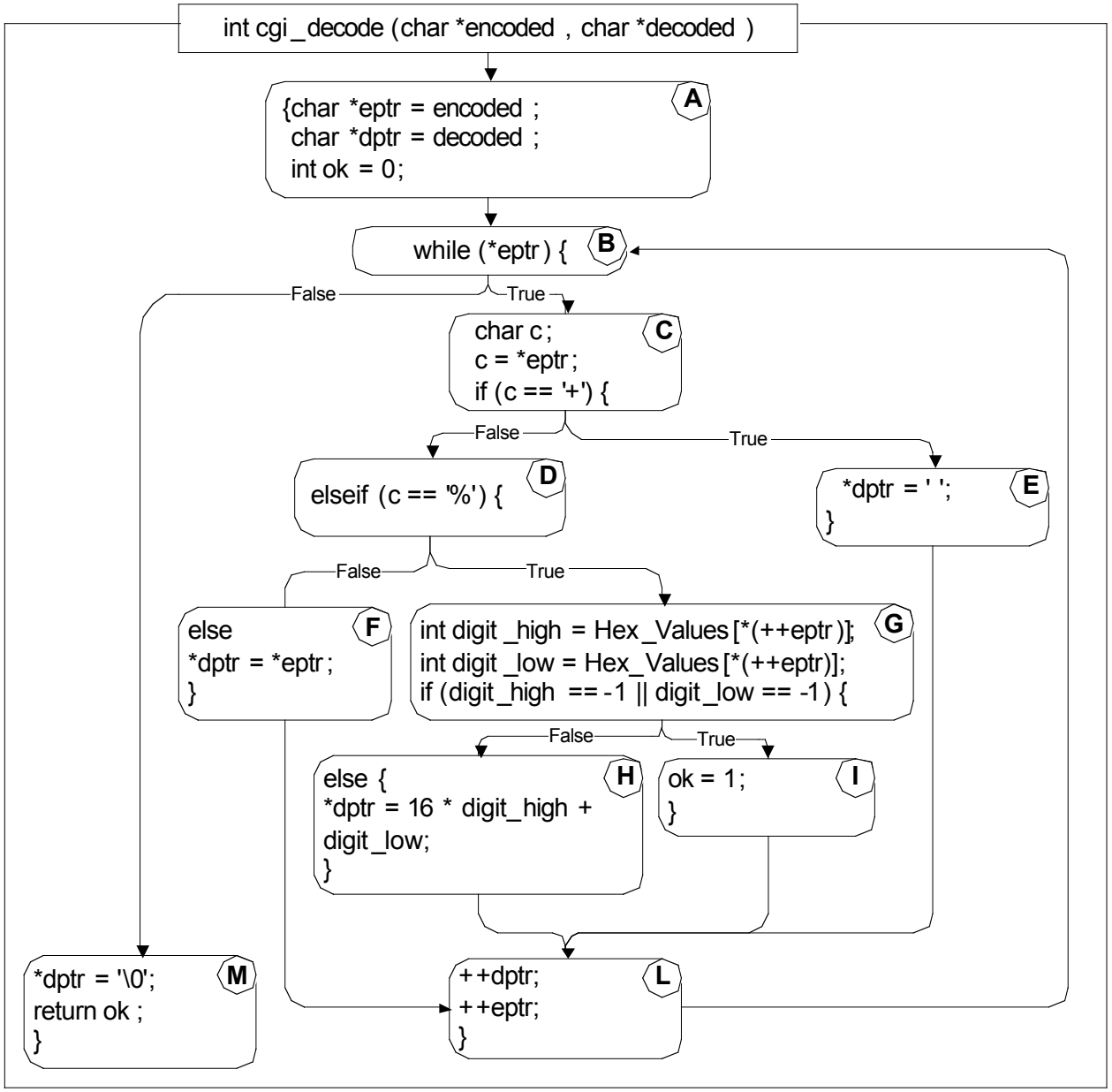
- Motivation
  - Un défaut dans une instruction ne peut être révélée qui si elle est exécutée.

TSC - C_{BLOC}
Critère par blocs d'instructions

- Les noeuds du graphe de contrôle correspondent aux suites ininterruptibles d'instructions.
  - Même concept appliqué à un niveau de granularité différent.
- Aucune différence essentielle
  - $(C_{INS} = 100\ \%) \equiv (C_{BLOC} = 100\ \%)$
  - C_{INS} et C_{BLOC} sont monotones entre elles
    - Si l'une croît, l'autre aussi.
    - Si l'ajout d'un cas de test améliore l'une, il améliore l'autre aussi.
    - La valeur de l'augmentation peut varier.
- Avantage
  - C_{BLOC} coute moins cher en temps et en espace.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

EXEMPLE



T_0 =
{"", "test+case%1Dadequacy"}

C_{INS} = 17/18 = 94 %

C_{BLOC} = 10/11 = 91 %

T_1 =
{"adequate+test%0Dexecution%7U"}

C_{INS} = 18/18 = 100 %

C_{BLOC} = 11/11 = 100 %

T_2 =
{"%3D", "%A", "a+b", "test"}

C_{INS} = 18/18 = 100 %

C_{BLOC} = 11/11 = 100 %

TSC
LA COUVERTURE N'EST PAS LA TAILLE

- Le taux de couverture ne dépend pas du nombre de cas de test
  - T_0, T_1 :
    - $C_{INS}(T_1) > C_{INS}(T_0)$
    - $Card(T_1) < Card(T_0)$
  - T_1, T_2 :
    - $C_{INS}(T_1) = C_{INS}(T_2)$
    - $Card(T_1) > Card(T_2)$
- Minimiser le nombre de cas de test est rarement un objectif
  - Constat
    - Un ensemble de plusieurs petits cas de test bien ciblés facilitent le diagnostic.
  - Exemple
    - La défaillance d'un cas de test de T_2 apporte plus d'information que celle d'un cas de test de T_1.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{BR}
CRITÈRE PAR BRANCHEMENTS

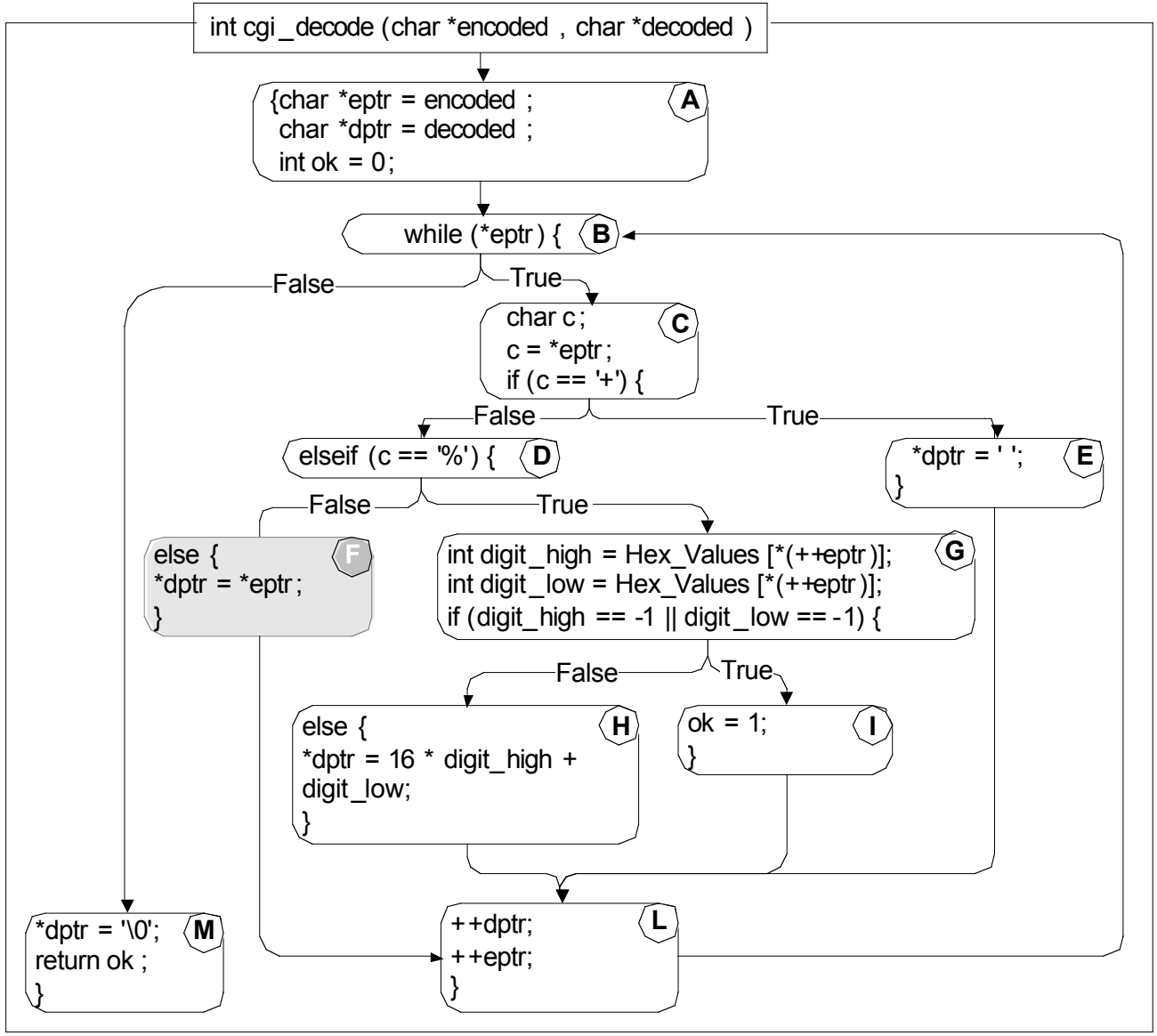
- Principe (critère d'adéquation faible)
  - Tout branchement doit être exécuté au moins une fois.

- Mesure

$$\frac{\#\text{ branchements exécutés}}{\#\text{ branchements}}$$

- Motivation
  - Couverture des blocs « vides »
  - Couverture partielle des « cas manquants »

EXEMPLE



Supposons que le noeud (bloc) F ait été omis par le programmeur

T_3 = {“”, “+%0D+%4J”}

$C_N = 10/10 = 100\ \%$

$C_A = 14/15 = 93\ \%$

$C_{BLOC} = 10/10 = 100\ \%$

$C_{BR} = 7/8 = 88\ \%$

La couverture de tous les noeuds (blocs) n’induit pas celle de tous les arcs (branchements)

TSC – C_{BR}
De la supériorité des arcs sur les noeuds

- Le parcours de tous les arcs induit celui de tous les noeuds (si le graphe est connexe)
- Non l'inverse (voir T_3)
- Le critère d'adéquation faible par rapport aux branchements domine celui par rapport aux instructions.

TSC – C_{BR}
De l'insuffisance des branchements

- La couverture d'un branchement n'induit pas celle de toutes ses conditions élémentaires
- Exemple
  - Supposons que le bloc G contienne l'erreur suivante
    - digit_high == -1 || digit_low == 1
  - Le critère de branchement peut être satisfait uniquement sur la base de la première condition (digit_high == -1)
    - il en découle que l'expression fautive n'est pas testée

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{COND}
CRITÈRE PAR CONDITIONS (ÉLÉMENTAIRES)

- Principe (critère d'adéquation faible)
  - Lors de l'exécution d'un cas de test, chaque condition élémentaire doit prendre les valeurs « vrai » et « faux ».

- Mesure

$$\frac{\text{\# valeurs distinctes prises par les conditions}}{2 * \text{\# conditions}}$$

- Motivation
  - Couverture des catégories issues des exigences telles que transcrites dans le code

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{COND}
De l'insuffisance des conditions

- Il est possible de couvrir toutes les conditions, sans couvrir tous les branchements

- Exemple
  - T_4 = {"first+test%9Ktest%K9"}

- Les deux critères sont incomparables
  - aucun ne domine l'autre

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{BC}
CRITÈRE PAR BRANCHEMENTS ET CONDITIONS

- Principe (critère d'adéquation faible)
  - Lors de l'exécution d'un cas de test,
    - chaque branchement doit être parcouru
    - chaque condition élémentaire doit prendre les valeurs « vrai » et « faux »,

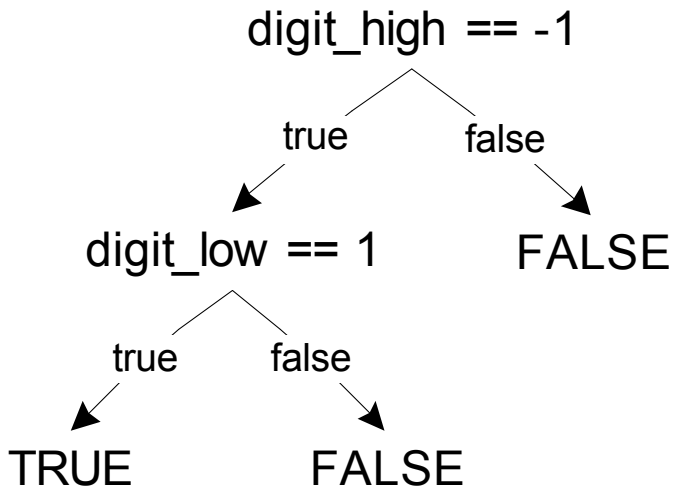
- Mesure
  - bp : nombre de branchements parcourus
  - vc : nombre de valeurs distinctes prises par les conditions
  - nb : nombre de branchements
  - nc : nombre de conditions

$$\frac{bp + vc}{nb+(2*nc)}$$

- Motivation
  - Couverture de tous les branchements
  - Couverture des catégories issues des exigences telles que transcrites dans le code
  - C_{BC} domine C_{COND} et C_{BR}

TSC – C_{BC}
ENCORE INSUFFISANT !

- C_{BC} couvre
  - toutes les conditions élémentaires,
  - tous les branchements,
  - mais PAS toutes les combinaisons de conditions élémentaires
- Exemple



Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{COMP}
CRITÈRE PAR COMPOSITION DE CONDITIONS

- Principe (critère d'adéquation faible)
  - Couvrir toutes les façons distinctes d'obtenir un branchement

- Mesure
  - Soit *p* le nombre d'expressions conditionnelles du module, composées chacune de k_i conditions élémentaires (*i=1..p*)
  - Soit *vc* le nombre de valeurs distinctes prises par les conditions élémentaires

$$\frac{vc}{\sum_{i=1..p} 2^{k_i}}$$

- Motivation
  - La méthode ultime (insurpassable) d'obtenir au moins un exemplaire de cas de test pour chaque façon de faire un branchement au sein du code

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{COMP}
L'EXPLOSION EXPONENTIELLE

- Le court-circuitage des expressions booléennes peut réduire l'explosion exponentielle, mais pas toujours
- Exemple de réduction de 32 à 13 : (((a || b) && c) || d) && e

Cas de test	a	b	c	d	e
(1)	T	—	T	—	T
(2)	F	T	T	—	T
(3)	T	—	F	T	T
(4)	F	T	F	T	T
(5)	F	F	—	T	T
(6)	T	—	T	—	F
(7)	F	T	T	—	F
(8)	T	—	F	T	F
(9)	F	T	F	T	F
(10)	F	F	—	T	F
(11)	T	—	F	F	—
(12)	F	T	F	F	—
(13)	F	F	—	F	—

TSC – $C_{MC/DC}$
CRITÈRE PAR MODIFICATION DE DÉCISIONS ET DE CONDITIONS

- Principe (critère d'adéquation faible)
  - Ne retenir que les décisions de branchement dont au moins une condition élémentaire prise isolément peut modifier la valeur (et donc le branchement)
  - S'assurer que chaque condition intervient dans au moins une telle décision pour chacune des valeurs « vrai » ou « faux »

- Mesure
  - Soit *p* le nombre d'expressions conditionnelles du module, composées chacune de k_i conditions élémentaires (*i=1..p*)
  - Soit *mdc* le nombre de décisions distinctes déterminées par au moins une condition élémentaire

$$\frac{mdc}{\sum_{i=1..p}(k_i + 1)}$$

- Motivation
  - S'approcher le plus possible de C_{COMP} tout en limitant linéairement le nombre de cas de tests

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – $C_{MC/DC}$ Exemple

- Reprenons (((a || b) && c) || d) && e

Cas de test	a	b	c	d	e	résultat
(1)	**true**	--	**true**	--	**true**	true
(2)	false	**true**	true	--	true	true
(3)	true	--	false	**true**	true	true
(6)	true	--	true	--	**false**	false
(11)	true	--	**false**	**false**	--	false
(13)	**false**	**false**	--	false	--	false

- Les valeurs en jaune sont les valeurs déterminantes

TSC – $C_{MC/DC}$ Synthèse

- Domine tous les critères précédents, sauf C_{COMP}.
- Le meilleur compromis entre maximisation de la couverture et faisabilité pratique.
- Un seul désavantage : le temps de précalcul légèrement plus long que celui des critères moins complets.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{CH}
CRITÈRE PAR CHEMINS

- Principe (critère d'adéquation faible)
  - S'assurer que chaque chemin possible est parcouru au moins une fois

- Mesure

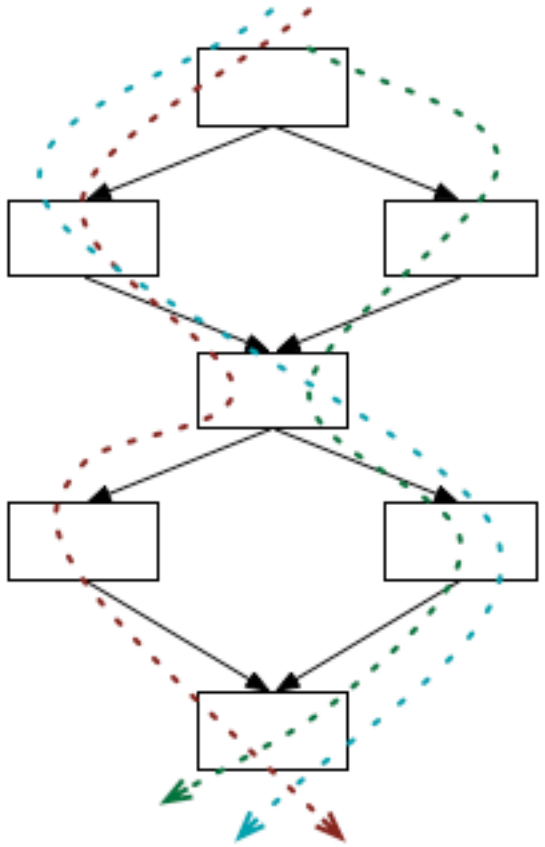
$$\frac{\#\ \textit{chemins distincts parcourus}}{\#\ \text{chemins distincts}}$$

- Motivation
  - Tester la validité des décisions entre elles

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{CH}
Illustration

- Un chemin est une suite ordonnée et d'arcs connexes
- Couverture de nouveaux cas de figure
- Explosion exponentielle, voire non bornée



TSC – C_{CH}
Les limites

- Le nombre de chemins est généralement non borné (boucles)
  - le critère est donc impossible à satisfaire
- Pour rendre opérationnel le critère
  - on aura donc retours à une méthode partitionnelle
- Les principes de partitions principalement utilisés sont
  - les dépendances des chemins
    (par exemple : éviter le retour à un déjà parcouru par le chemin)
  - la longueur des chemins traversés
    (par exemple : inférieur à 4)
  - le nombre d'itérations des boucles
    (par exemple : 0, 1, plusieurs)

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{INT}
CRITÈRE PAR LIMITES INTÉRIEURES

- Principe (critère d'adéquation faible)
  - Ne considérer que les chemins à partir du « départ »
  - Prendre en compte les chemins jusqu'à concurrence :
    - d'un noeud déjà parcouru (par le même chemin)
    - de la « sortie »

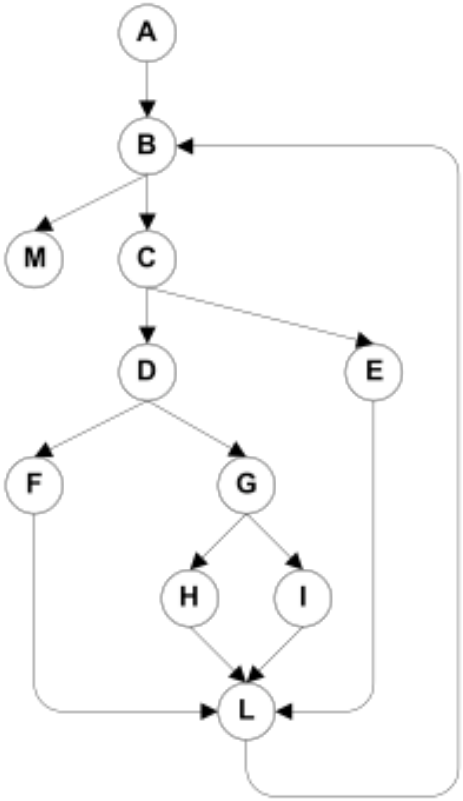
- Mesure

$$\frac{\#\ \textit{chemins limités parcourus}}{\#\ \textit{chemins limités}}$$

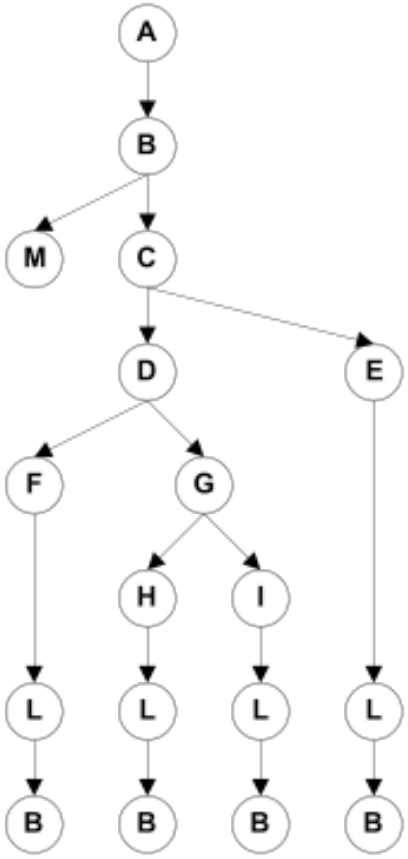
- Motivation
  - Tester les parcours de boucles sans répétition (en négligeant donc l'effet de fermeture, de point fixe)

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{INT}
UN EXEMPLE



(i)



(ii)

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{INT}
DU NON BORNÉ À L'EXPONENTIEL

- Le nombre de chemins est désormais borné, mais exponentiel

```
if (a) {
   S1;
}
if (b) {
   S2;
}
if (c) {
   S3;
}
...
if (x) {
   Sn;
}
```

- Chaque instruction S_i peut faire ou non partie d'un chemin, dès lors le nombre de chemins est 2^n où n est le nombre d'instructions.
- En général, il est long et difficile de produire les données de tests nécessaires à la couverture de tous les cas de test.

TSC – $C_{LIN}(N)$
Critère par séquence linéaire (LCSAJ)

- Principe (critère d'adéquation faible)
  - Considérer tous les chemins constitués de *n* séquences linéaires ou moins
- Mesure

$$\frac{\#\ chemins\ parcourus}{\#\ chemins\ possibles}$$

- Motivation
  - Borner a priori l'effort requis tout en maintenant une bonne représentativité des chemins
- *LCSAJ*
  - *Linear code sequence and jump*

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – $C_{LIN}(N)$ APPLICATION

- Représentation des séquences linéaires
  <début, fin, destination>
  avec association d'un numéro unique à chacune.
- Accumulation des numéros de séquences linéaires franchies dans une file bornée de longueur *k* avec décompte du chemin accumulé chaque fois qu'elle est pleine.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – $C_{LIN}(N)$ GÉNÉRALISATION

- TER1 = C_{BLOC} (blocs d'instructions)
- TER2 = C_{BR} (branchements)
- TER3 = $C_{LIN}(1)$
- …
- TERn+2 = $C_{LIN}(n)$

- Question : donner un exemple de cas appartenant à TER3, mais pas TER2

TSC – $C_{BOUCLE}(k)$

CRITÈRE PAR REPRÉSENTANTS DE BOUCLE

- Principe (critère d'adéquation faible)
  - Couvrir chacune des boucles à l'aide de *k* représentants en fonction du nombre d'itérations (typiquement *k*=3 : 0, 1, plusieurs).
- Mesure

$$\frac{\#\ \textit{nombre de représentants de boucles effectifs}}{k * \#\ \textit{nombre de boucles}}$$

- Motivation
  - Représenter les deux cas initiaux et l'hypothèse d'induction (correspondant à la recherche du point fixe).
- Note
  - k > 1
  - k est interprété de la façon suivante :
    { 0, 1, 2, ..., k-2, « (k-1) ou plus » }

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – $C_{BOUCLE}(K)$
APPLICATION

- Suppose des boucles avec une seule entrée et une seule sortie.
- Adaptation pour les boucles de type « repeat » (on début à 1 plutôt qu'à 0).
- Adaptation possible également pour les boucles à sortie centrale.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{CYC}
CRITÈRE CYCLOMATIQUE

- Principe (critère d'adéquation faible)
  - Considérer les seuls chemins de la base cyclomatique, tous les autres étant composés de ceux-ci

- Mesure

$$\frac{\#\ chemins\ cyclomatiques\ parcourus}{e-n+2}$$

- Motivation
  - Réduire le nombre de chemins considérés au plus petit nombre garantissant une représentativité de tous les chemins

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{CYC}
BASE CYCLOMATIQUE

- Une base cyclomatique est un ensemble (minimal) de chemins permettant de composer tous les autres.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – CCYC

APPLICATION PRATIQUE

- La taille *t* de la base cyclomatique d'un graphe *g* est donnée par

  $t(g) = e - n + c$

  où

  e : le nombre d'arêtes (branchements),
  n : le nombre de noeuds (blocs d'instructions),
  c : le nombre de composantes connexes.

- Par construction, on ajoute une arête de la sortie vers l'entrée afin de s'assurer que le graphe comporte au moins une composante fortement connexe (donc on ajoute aussi un à *e*).

- Dans le cas d'un programme « structuré » (une entrée et une sortie), *c=1* et l'équation devient

  $t(g) = e - n + 2$

  (où *e* est le nombre original d'arêtes).

TSC – C_{CYC}
APPLICATION PRATIQUE

- Par définition, une base est formée de chemins indépendants.
- En général, il peut y avoir plusieurs bases.
- En pratique, lors de l'exécution du programme on compte chacun des chemins indépendants rencontrés.
- Pour permettre la comparaison rapide des chemins et la détermination de leur (in)dépendance, ceux-ci sont représentés par des vecteurs dont chaque dimension est associée à une arête et dont la valeur est le nombre d'exécutions.

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC – C_{CYC}
APPLICATION PRATIQUE

```
if (a) {
   S1;
}
if (b) {
   S2;
}
if (c) {
   S3;
}
...
if (x) {
   Sn;
}
```

- Le nombre total de chemins différents est exponentiel, mais le nombre de chemins de la base cyclomatique est

  n+1.

TSC – C_{APP}

CRITÈRE PAR APPEL DE PROCÉDURE

- Principe (critère d'adéquation faible)
  - S'assurer que toutes les procédures (méthodes, fonctions, routines) aient été utilisées au moins une fois.
- Mesure

$$\frac{\#\ \textit{procédures distinctes appelées}}{\#\ \textit{procédures distinctes}}$$

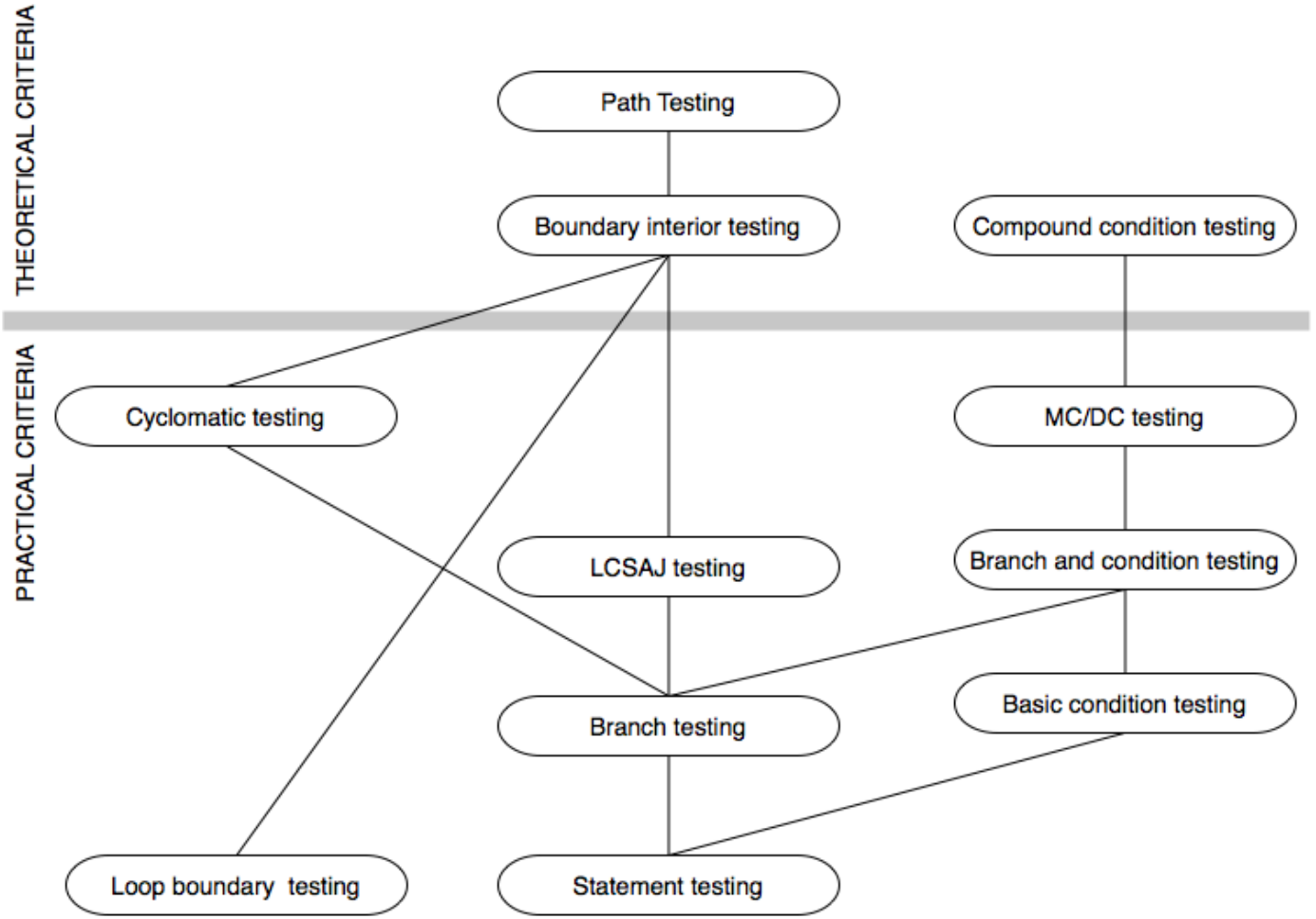
- Motivation
  - Cout très faible (voire nul), permet une démarche progressive dans l'élaboration des cas de tests.
  - Plus approprié pour les tests d'intégration (en supposant que tous les composants ont préalablement été couverts par des tests unitaires).
  - « if unit testing has been effective, then faults that remain to be found in integration testing will be primarily interface faults, and testing effort should focus on interfaces between units rather than their internal details »

TSC – CAPP
DIFFICULTÉS PARTICULIÈRES ET LIMITES

- Points d'entrée et points de sortie multiples
  - augmentent la complexité des tests et rendent l'atteinte de la couverture plus difficile
- Couverture d'appel
  - un complément souvent nécessaire en pratique dans le cadre des essais d'intégrations

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

RELATION DE DOMINANCE (VERSION CORRIGÉE)



TSC
SYNTHÈSE (…)

- Les tests de couverture sont nécessaires
  - Pour guider l'amélioration des essais
  - Pour qualifier objectivement le niveau d'adéquation faible d'un essai
  - En particulier MC/DC
    - Pour respecter plusieurs normes de qualité, dont RTCA/DO-178B et EUROCAE ED-12B

Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

TSC
SYNTHÈSE (2/3)

- Sometimes criteria may not be satisfiable
  - The criterion requires execution of
    - statements that cannot be executed as a result of
      - defensive programming
      - code reuse (reusing code that is more general than strictly required for the application)
    - conditions that cannot be satisfied as a result of
      - interdependent conditions
    - paths that cannot be executed as a result of
      - interdependent decisions

TSC
SYNTHÈSE (3/3)

- Large amounts of fossil code may indicate serious maintainability problems
  - But some unreachable code is common even in well-designed, well-maintained systems
- Solutions:
  - make allowances by setting a coverage goal less than 100%
  - require justification of elements left uncovered
    - RTCA-DO-178B and EUROCAE ED-12B for modified MC/DC

SUMMARY

- We defined a number of adequacy criteria
  - NOT test design techniques!
- Different criteria address different classes of errors
- Full coverage is usually unattainable
  - Remember that attainability is an undecidable problem!
- …and when attainable, "inversion" is usually hard
  - How do I find program inputs allowing to cover something buried deeply in the CFG?
  - Automated support (e.g., symbolic execution) may be necessary
- Therefore, rather than requiring full adequacy, the "degree of adequacy" of a test suite is estimated by coverage measures
  - May drive test improvement