Fiabilité Logicielle

Tests boite noire

Arnaud Labourel

2025-26

amU Faculté des sciences Aix Marseille Université

Le test boite noire

Le test boite noire :

- L'implémentation n'est pas connue.
- Les cas de test ne sont produits qu'à partir de la spécification
- Test fonctionnel : on vérifie la totalité du comportement du système (contrairement aux tests unitaires).
- Permet de tester l'implémentation, mais sert aussi à révéler des erreurs ou des omissions dans la spécification.

Difficultés du test en boite noire

- Difficultés liées à la spécification
  - exploitation de la spécification pour l'extraction de cas de test et la définition d'oracles
  - complexe car combinaison possible de plusieurs outils pour une même spécification
- Pas de méthode rigoureuse, ni systématique : de très nombreuses données possibles notamment en entrée.
- Pas de métriques de couverture du code (contrairement aux tests boite blanche).

$\Rightarrow$ besoin de méthodes pour concevoir des suites de tests efficaces (couverture spécification).

Conception d'une suite de tests en boite noire

Concepts utiles :

- Classe d'équivalence
- Arbre de décision
- Graphe de causalité

Section 1

Classes d'équivalence

Classes d'équivalence : principes

Objectifs : avoir un jeu de test de taille raisonnable en évitant les redondances

- Partitionner l'espace des données en classe d'équivalence
- Ne produire et exécuter qu'un seul cas de test par classe d'équivalence

Exemple : La fonction `abs(x)` retourne la valeur absolue de l'entier `x`.

Classe	`x<0`	`x >= 0`
Donnée de Test	-2	3

Classes d'équivalence : principes

Difficulté : identifier les classes d'équivalence

Nécessite une analyse de la spécification pour produire le **partitionnement des données** :

- Relations entrée/sorties.
- Conformité des entrées :
  - Classe valide : données acceptables par le programme.
  - Classe invalide : données refusées par le programme.

Suite de tests pour une variable :

- couvrir toute classe valide,
- couvrir toute classe invalide.

Une donnée invalide fait **dysfonctionner** le programme

Trouver les classes d'équivalence

- Condition nécessaire booléenne (tout nom commence par une lettre) :
  - 1 classe valide
  - 1 classe invalide
- Condition spécifiant un intervalle $x \in [a, b]$ avec $a \leq b$ réels :
  - 1 classe valide $[a, b]$
  - 2 classes invalides $]-\infty, a[$, $]b, +\infty[$
- Condition spécifiant un nombre d'items (exemple : choisir des articles parmi 3)
  - 1 classe valide (1,2, ou 3 valeurs)
  - 2 classes invalides : 0 article, $+3$ articles
- Ensemble de valeurs possibles (exemple : étudiant, tuteur, professeur) :
  - 1 classe valide par élément
  - une classe invalide (les éléments hors de la classe)

Trouver les classes d'équivalence (II)

Classes invalides usuelles

Exemple : le fichier contient exactement 3 flottants

- Nombre d'arguments ou de données
  - < 3 arguments
  - > 3 arguments
- Types des arguments / données
  - la première, la deuxième ou la troisième donnée n'est pas un flottant (3 cas de test).

Classes d'équivalence - choix des données de test

En supposant plusieurs variables d'entrée, les Données de Tests (DT) doivent couvrir toutes les classes pour toutes les variables.

- Une DT peut couvrir plusieurs classes valides.
- Une DT ne doit couvrir qu'une seule classe invalide

Exemple :

Spécification : `x` doit être entre 0 et 100 et `y` entre 0 et 5.

Donnée	valide	invalide
x	$[0..100]$(C1)	< 0 (C2), > 100 (C3)
y	$[0..5]$(C4)	< 0 (C5), > 5 (C6)

Suite de tests - exemple

Spécification : x doit être entre 0 et 100 et y entre 0 et 5.

Donnée	valide	invalide
x	$[0..100]$(C1)	< 0 (C2), > 100 (C3)
y	$[0..5]$(C4)	< 0 (C5), > 5 (C6)

Suite de test :

- $x = 5, y = 2$ couvre (C1) et (C4)
- $x = -1, y = 2$ couvre (C2)
- $x = 101, y = 2$ couvre (C3)
- $x = 5, y = -1$ couvre (C5)
- $x = 5, y = 9$ couvre (C6)

Exemple de spécification

Une application Web calcule le taux d'imposition et le taux de réduction en fonction du revenu du foyer et du nombre d'enfants selon les règles suivantes :

- Le revenu doit être compris entre 0€ et 100K€ ;
- Les revenus de moins de 10K€ ne sont pas imposables ;
- Les revenus à partir de 10k€ sont imposés au taux de 20% ;
- Les foyers avec 2 enfants scolarisés ou plus ont une réduction de 15% ;
- Le montant des revenus et le nombre d'enfants (de 0 à 10) sont fournis via un formulaire avec des champs de saisie libres. Seuls des entiers positifs sont acceptés par le système pour les deux champs.

Classes d'équivalence sur l'exemple

Entrée revenu r

- Entrée non numérique : invalide
- Entrée non entière : invalide
- $[0; 10k[$: valide (non imposable)
- $[10k; 100k]$: valide (imposable)
- $]-INF; 0[$: invalide
- $]100k; +INF[$: invalide

Entrée nombre d'enfants n

- Entrée non numérique : invalide
- Entrée non entière : invalide
- $[0; 2[$: valide
- $[2; 10]$: valide
- $]-INF; 0[$: invalide
- $]10; +INF[$: invalide

Couverture des classes d'équivalence :

- DT1 : $r = -1000$, $n = -10$: erreur
- DT2 : $r = 5$, $n = 1$: non imposable
- DT3 : $r = 50000$, $n = 3$: imposable à 5%
- DT4 : $r = 150000$, $n = 11$: erreur
- DT5 : $r = 2,5$, $n = 1,1$: erreur
- DT5 : $r =$cinq, $n =$six : erreur

Tests aux limites

Idée : erreurs souvent liées aux cas limites donc utiliser ces cas limites comme des DT (**en plus** des représentants de classe)

```
if( x > 0 ) {...} else {...}
```

au lieu de

```
if( x >= 0 ) {...} else {...}
```

Retourne la valeur absolue de x : $]-\infty, 0[$, $[0, +\infty[$.

Cas limites : -1 pour la première classe et 0 pour la seconde à ajouter aux "cas typiques", par exemple -3, 3

Déterminer les cas limites : règles standards et bon sens !

Tests aux limites : autres exemples

- `N` entier maximal : `N-1`, `N`, `N+1`
- Intervalle $[a, b]$: `a`, `b` (éventuellement `a-1` et $b+1$ via autres classes).
- Fichier : vide, taille maximale, trop grand.
- Indice d'un tableau : 0, indice maximal (indices valides).
- Chaine d'au plus `n` caractères : chaine vide, de taille `n-1`, `n`, $n+1$.

Gestion de la combinatoire

Application avec plusieurs entrées

- Tests unidimensionnels : on considère les classes des variables séparément
- Tests multidimensionnels : corrélation des valeurs de variables dans les tests

$\mathtt{x} \in X, \mathtt{y} \in Y$, chacune ayant leur propres classes d'équivalence valides $(C_x^1, C_x^2, \ldots, C_y^1, C_y^2, \ldots)$.

Comment calculer les classes pour `foo(x,y)` (et donc le nombre de cas de tests nécessaires pour les couvrir) ?

Gestion de la combinatoire

Différentes méthodes pour calculer les classes pour `foo(x,y)` (et donc le nombre de cas de tests nécessaires pour les couvrir) :

- considérer le produit cartésien des classes $C_x^1 \times C_y^1, C_x^1 \times C_y^2, ...$ (nombre : $|X| * |Y|$)
- considérer $C_x^1 \times Y, C_x^2 \times Y, ..., X \times C_y^1, X \times C_y^2, ...$ (nombre : $|X| + |Y|$ au maximum, sans doute moins)
- partitionner directement $X \times Y$ (plutôt que de considérer les classes d'équivalence des variables individuelles).
- couvrir les paires de classes d'équivalence (combinatoire moindre s'il y a plus de 2 variables en jeu).

Gestion de la combinatoire : variables booléennes

Cas des variables booléennes : 2 classes par variable (V, F)

On suppose n variables

- Nb de combinaisons possibles : 2^n
- Nb de paires : $\frac{n(n-1)}{2}$ avec 4 possibilités par paire.
- Une DT peut couvrir plusieurs paires

Le raisonnement s'étend aux tuples de variables opérant sur des domaines finis.

Combinatoire : variables booléennes - exemple

On suppose 3 variables booléennes : X, Y et Z

Détermination d'une suite de tests minimale couvrant toutes les paires possibles :

	$DT1$	$DT2$	$DT3$	$DT4$
X	1	1	0	0
Y	1	0	1	0
Z	1	0	0	1

Les $(3 * 2)/2 \ * \ 4 = 12$ possibilités sont couvertes.

- DT1 : 3 paires
- DT2 : 3 paires
- DT3 : 3 paires
- DT4 : 3 paires

Section 2

Arbres de décision

Arbres de décision : principes

Certaines spécifications peuvent s'analyser avec un arbre de décision qui donne les classes de tests.

- chaque nœud de l'arbre définit une classe qu'on va raffiner selon un critère usuellement binaire (Oui/Non).
- les fils d'un nœud correspondent aux différents choix possibles
- les feuilles correspondent à une classe,
- le chemin de la racine à une feuille définit un cas de test.
- les données de tests sont souvent définies par les critères rencontrés sur le chemin.

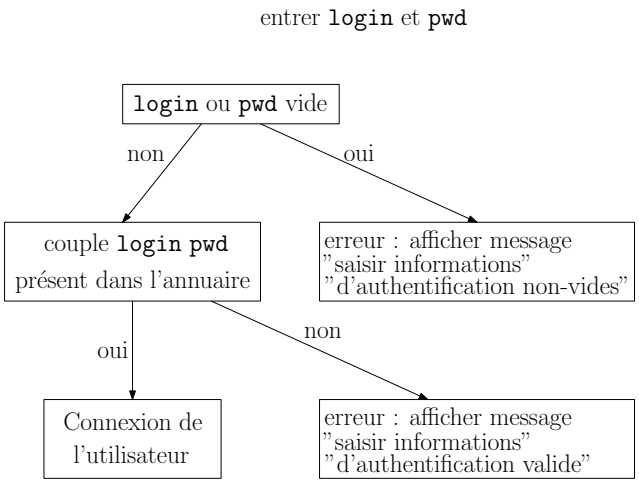
Arbres de décision : méthodes

1. Identifier les concepts permettant de définir des classes : un argument, nom de fichier, fichier, lire, format csv, une ligne unique, valeurs réelles.
2. Déduire une analyse top-down. La satisfaction de chaque critère donne un branchement (décision binaire ou d'arité supérieure).

Arbres de décision : exemple

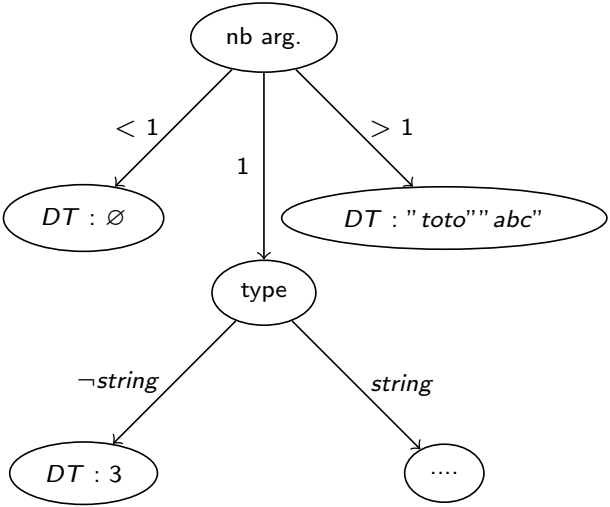
Un utilisateur tente de se connecter à une application Web via `login` et `pwd`.

- `login` et `pwd` vides : erreur
- `login` ou `pwd` incorrects : erreur
- `login` et `pwd` corrects : accès autorisé



Arbres de décision : exemple

fonction pour laquelle un seul argument de type `String` est attendu.



Arbres de décision : bilan

- Méthode simple et efficace.
- Analyse top-down.
- Peut se combiner avec d'autres techniques (arbres de défaillances).
- Ne détecte pas certaines relations complexes.
- Taille : n décisions binaires $\Rightarrow 2^n$ feuilles (pire des cas).

Section 3

Graphe Causes/Effets

Graphe de causalité

Graphe de causalité :

Formaliser la spécification avec un graphe donnant les relations de causalité entre entités du problème.

- **Cause :** entrées (variable d'entrée, valeur système, action utilisateur, . . .)
- **Effet :** sorties (variable de sortie, modification de variable d'état,. . .)

Construction du graphe :

- Noeuds : cause, effets, noeuds intermédiaires (et $\vee$, ou $\wedge$) éventuellement
- Arcs : normaux ou négation : traduisant la causalité
- Ecriture de contraintes logiques entre causes, effets (et, ou, . . .).

Graphe de causalité : contraintes

Contraintes entre les causes :

- Partition totale : exactement 1 dans $C_1, \ldots, C_n$
- Partition partielle : aucun ou exactement 1 dans $C_1, \ldots, C_n$
- Nécessité : au moins 1 dans $C_1, \ldots, C_n$
- Causalité : $C_1 \Rightarrow C_2$
- Exclusion : $C_1 \Leftrightarrow \neg C_2$

Graphe de causalité : exemple

Renouvellement annuel de contrat d'une compagnie d'assurance.

- 0 accident et age $\leq$ 25 ans : augmenter de 50 euros
- 0 accident et age $>$ 25 ans : augmenter de 20 euros
- 1 accident et age $\leq$ 25 ans : augmenter de 100 euros $+$ envoyer une mise en garde
- 1 accident et age $>$ 25 ans : augmenter de 50 euros $+$ envoyer une mise en garde
- plus d'un accident : résiliation.

Graphe de causalité : causes/effets

Causes :

- avoir 0 accident (C_0), 1 accident (C_1), plus d'un accident $(C_{>1})$.
- avoir moins de 25 ans $(C_{\leq 25})$, avoir plus de 25 ans $(C_{>25})$

Effets :

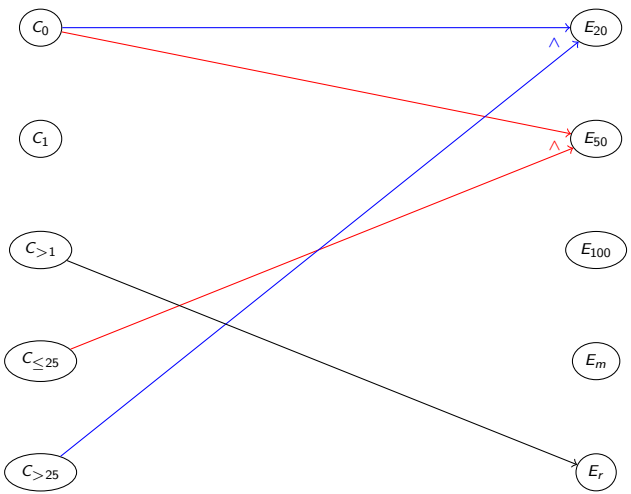
- augmenter de 20 euros (E_{20}), 50 euros (E_{50}), 100 euros (E_{100}),
- envoyer une mise en garde (E_m),
- résilier le contrat (E_r)

Graphe de causalité : construire le graphe

1. Identifier les effets $E_1, E_2, \dots, E_p$,
2. Identifier les causes $C_1, C_2, \dots, C_n$
3. Tracer les arcs causes/effets + opérations et, ou, non entre arcs
4. Rajouter les contraintes entre causes

Graphe de causalité : le graphe de l'exemple

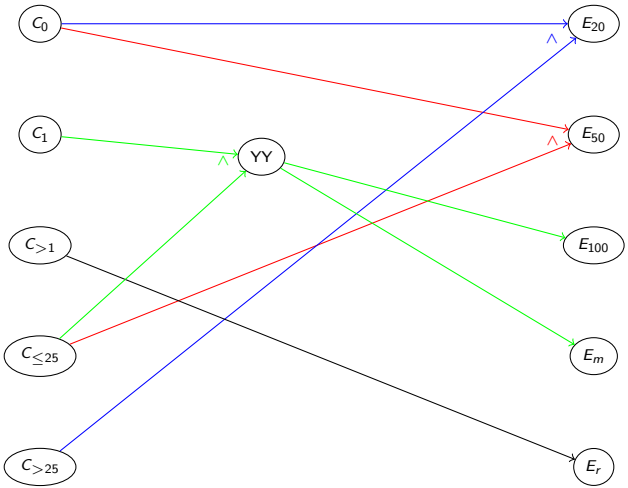
$C_0 \wedge C_{\leq 25} \Rightarrow E_{50}$
$C_0 \wedge C_{>25} \Rightarrow E_{20}$
$C_{>1} \Rightarrow E_r$



Graphe de causalité : le graphe de l'exemple

$C_0 \wedge C_{\leq 25} \Rightarrow E_{50}$
$C_0 \wedge C_{>25} \Rightarrow E_{20}$
$C_{>1} \Rightarrow E_r$

$C_1 \wedge C_{\leq 25} \Rightarrow E_{100} \wedge E_m$

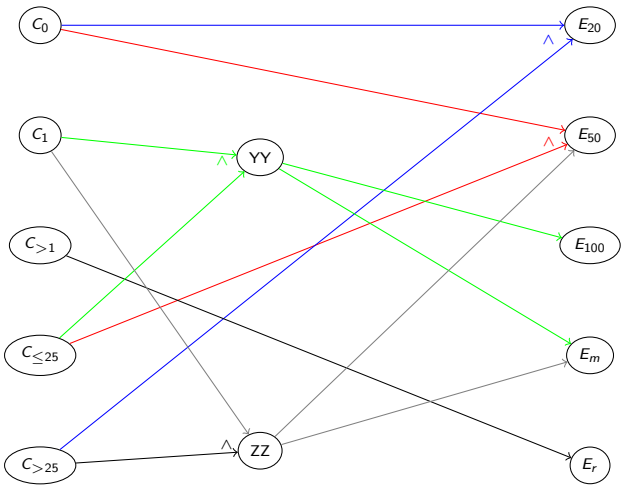


Graphe de causalité : le graphe de l'exemple

$$C_0 \wedge C_{\leq 25} \Rightarrow E_{50}$$
$$C_0 \wedge C_{>25} \Rightarrow E_{20}$$
$$C_{>1} \Rightarrow E_r$$

$$C_1 \wedge C_{\leq 25} \Rightarrow E_{100} \wedge E_m$$

$$C_1 \wedge C_{>25} \Rightarrow E_{50} \wedge E_m$$

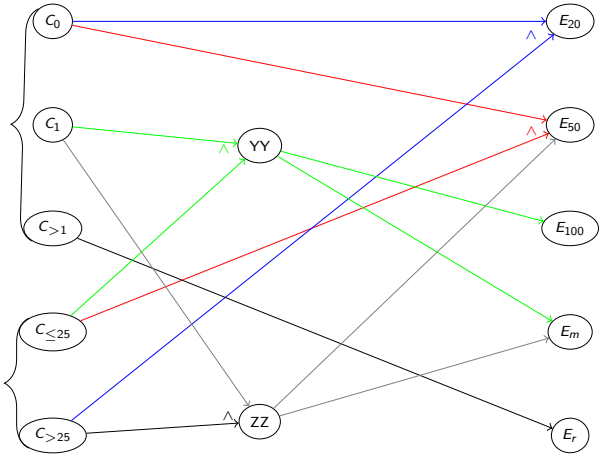


Graphe de causalité : le graphe de l'exemple

$$C_0 \wedge C_{\leq 25} \Rightarrow E_{50}$$
$$C_0 \wedge C_{>25} \Rightarrow E_{20}$$
$$C_{>1} \Rightarrow E_r$$

$$C_1 \wedge C_{\leq 25} \Rightarrow E_{100} \wedge E_m$$

$$C_1 \wedge C_{>25} \Rightarrow E_{50} \wedge E_m$$



{ exactement 1 vrai dans nombre d'accidents et dans age

Autre exemple

Connexion à un système bancaire mobile.

Spécification

Un utilisateur saisit son identifiant et son mot de passe ou son numéro de compte bancaire et son mot de passe pour vérifier son identité. Ainsi, pour se connecter au système bancaire mobile, un mot de passe est nécessaire, mais un identifiant ou un numéro de compte bancaire doit être saisi avec celui-ci.

Causes et effets

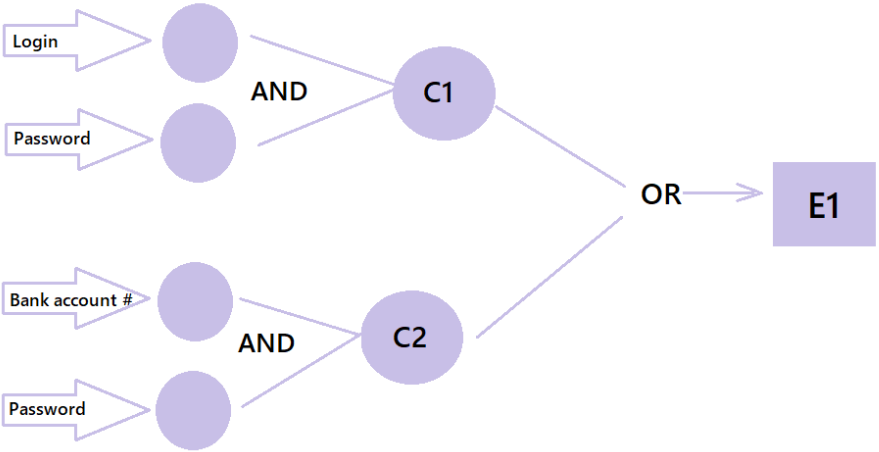
Causes :

- C1 : identifiant et mot de passe correctement saisis.
- C2 : numéro de téléphone portable et mot de passe correctement saisis.
- C3 : identifiant mal saisi, mot de passe correct.
- C4 : numéro de compte bancaire mal saisi, mot de passe correct.
- C5 : mot de passe mal saisi, identifiant correct.
- C6 : mot de passe mal saisi, numéro de compte bancaire correct.
- C7 : identifiant et mot de passe insérés de manière incorrecte.
- C8 : numéro de compte bancaire et mot de passe insérés de manière incorrecte.

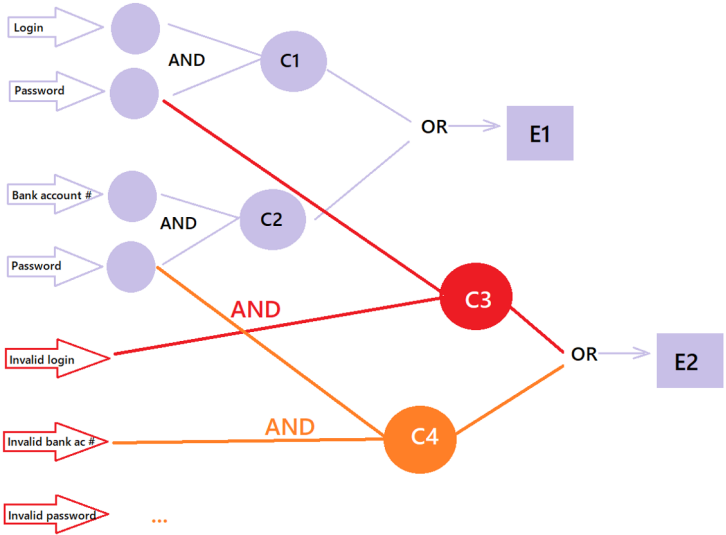
Effets :

- E1 : Un client peut passer la vérification et se connecter au système.
- E2 : Pas de connexion au système. Un message d'erreur s'affiche.

Graphes de causalité : le graphe de l'exemple



Graphes de causalité : le graphe de l'exemple



Graphe de causalité : exploitation du graphe

On utilise le graphe pour générer des cas de test.

- Causes et effets sont vus comme des variables booléennes (les premières d'entrée et les secondes de sortie)
- Génération automatique d'une matrice de décision (matrice booléenne, Vrai : 1 Faux : 0) à partir du graphe cause-effet
- Élimination des combinaisons invalides : *comportements inobservables* (représentées par une contrainte sur les causes)

Graphe de causalité : matrice de décision

(C_0)	(C_1)	$(C_{>1})$	$(C_{\leq 25})$	$(C_{>25})$	E_{20}	E_{50}	E_{100}	E_m	E_r
1	0	0	1	0	0	1	0	0	0
1	0	0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	1	1	0
0	1	0	0	1	0	1	0	1	0
0	0	1	1	0	0	0	0	0	1
0	0	1	0	1	0	0	0	0	1

Remarque : il n'existe pas de ligne avec à la fois $C_{\leq 25}$ et $C_{>25}$ à Vrai (1).

Stratégie de construction de la matrice

Pour un effet E

- on identifie les valeurs des causes C_i qui produisent cet effet
- on rajoute tous les effets induits par ces causes

(C_0)	(C_1)	$(C_{>1})$	$(C_{\leq 25})$	$(C_{>25})$	E_{20}	E_{50}	E_{100}	E_m	E_r
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
							1		

Stratégie de construction de la matrice

Pour un effet E

- on identifie les valeurs des causes C_i qui produisent cet effet
- on rajoute tous les effets induits par ces causes

(C_0)	(C_1)	$(C_{>1})$	$(C_{\leq 25})$	$(C_{>25})$	E_{20}	E_{50}	E_{100}	E_m	E_r
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	1	0	1	0			1		

Stratégie de construction de la matrice

Pour un effet E

- on identifie les valeurs des causes C_i qui produisent cet effet
- on rajoute tous les effets induits par ces causes

(C_0)	(C_1)	$(C_{>1})$	$(C_{\leq 25})$	$(C_{>25})$	E_{20}	E_{50}	E_{100}	E_m	E_r
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	1	0	1	0	0	0	1	1	0

Extraction de cas de test

Chaque ligne de la matrice définit un ou plusieurs cas de test

- les données de test correspondent aux causes C_i à Vrai
- le résultat attendu du test est un effet E à Vrai (ou l'ensemble des effets à Vrai)

(C_0)	(C_1)	$(C_{>1})$	$(C_{\leq 25})$	$(C_{>25})$	E_{20}	E_{50}	E_{100}	E_m	E_r
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	1	0	1	0	0	0	1	1	0

- CT1 = DT : C_1, $C_{\leq 25}$ et EV : E_{100}
- CT2 = DT : C_1, $C_{\leq 25}$ et EV : E_m

Exploitation du graphe via SAT

- Transposer en un système d'équations booléennes.
- Résolution avec un solveur propositionnel (Z3, CVC4,. . .) :
  1. Déterminer les valeurs des C_i donnant E donne les DT,
  2. Savoir si un effet dépend d'une condition,

Possibilité de modifier le système pour accélérer la résolution.

exclusion mutuelle : $\neg(C_0 \wedge C_1) \wedge \neg(C_1 \wedge C_{>1}) \wedge \neg(C_{>1} \wedge C_0) \wedge \neg(C_{\leq 25} \wedge C_{>25})$

couverture : $(C_0 \vee C_1 \vee C_{>1})$

implications :
$$\begin{aligned} C_0 \wedge C_{\leq 25} &\Rightarrow E_{50} \\ C_0 \wedge C_{>25} &\Rightarrow E_{20} \\ C_1 \wedge C_{\leq 25} &\Rightarrow E_{100} \wedge E_m \\ C_1 \wedge C_{>25} &\Rightarrow E_{50} \wedge E_m \\ C_{>1} &\Rightarrow E_r \end{aligned}$$

Exploitation du graphe via SAT (II)

Sortie de Z3 : un modèle satisfaisant les contraintes

```
[C0 = False, C1 = False, C2 = True,
Cgt25 = True, Cle25 = False,
E50 = False, E20 = False, E100 = False,
Em = False, Er = True]
```

- chercher des modèles minimaux pour les effets (si les autres effets ne sont pas forcés à Vrai, les mettre à Faux)
- un modèle = un cas de test.
- possibilité d'énumérer tous les modèles (mininaux)

Graphe Causes/Effet : réduction de la combinatoire

Si n causes alors (sans contraintes / au pire) 2^n lignes dans la table (2^n cas de test)

- ne pas tester les causes redondantes : Si $\{C_1, C_2\}$ et $\{C_1, C_2, C_3\}$ produisent les mêmes effets, ne tester que le premier
- éliminer les CT redondants en identifiant ceux ayant des comportements similaires (mêmes effets) bien qu'ayant des causes différentes, c'est-à-dire ne tester qu'un seul cas de test pour un ensemble d'effets donné

Graphe Causes/Effet : bilan

Avantages

- Méthode rigoureuse pour formaliser la spécification
- Permet d'identifier des cas de tests pertinents
- Permet de réduire la combinatoire des cas de tests
- Peut être automatisée via des solveurs propositionnels

Inconvénients

- Nécessite une analyse fine de la spécification
- Peut être difficile à appliquer sur des systèmes complexes
- Ne garantit pas la couverture totale de la spécification