UNIVERSITÉ PARIS SUD

L3 informatique et MIAGE

Année 2016-2017

Génie logiciel avancé

Test structurel à partir du graphe de flot de données

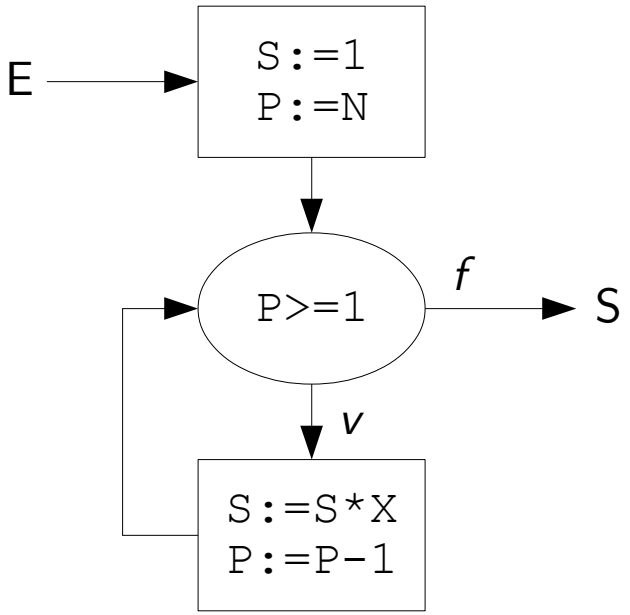
Delphine Longuet

delphine.longuet@lri.fr

http://www.lri.fr/~longuet/Enseignements/16-17/L3-GLA

Graphe de flot de contrôle

Graphe orienté faisant apparaître la structure du code en termes de chaînes d'instructions, de branchements conditionnels et de boucles

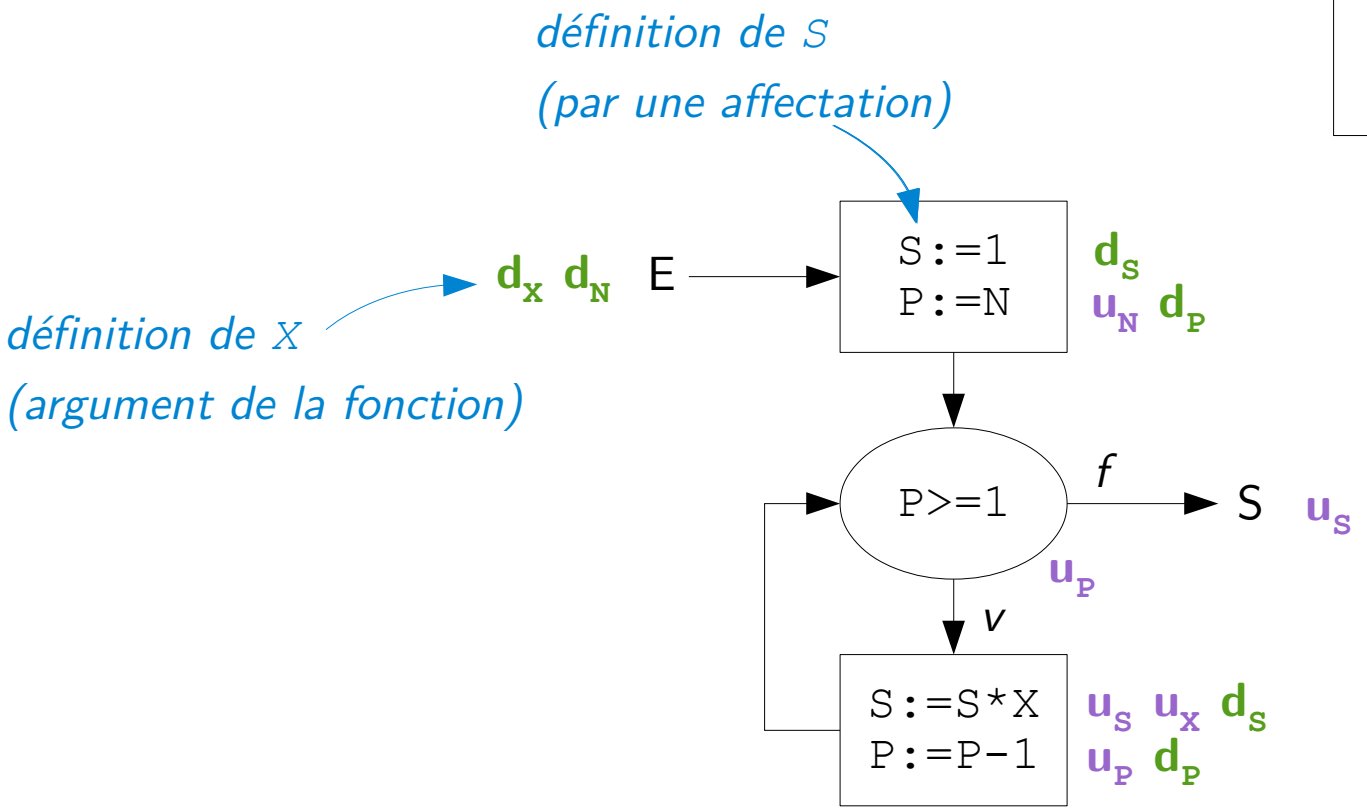


Graphe de flot de données

Annotation du graphe de flot de contrôle par les définitions et les utilisations de variables

Notations :

- Définition de X : d_X
- Utilisation de X : u_X

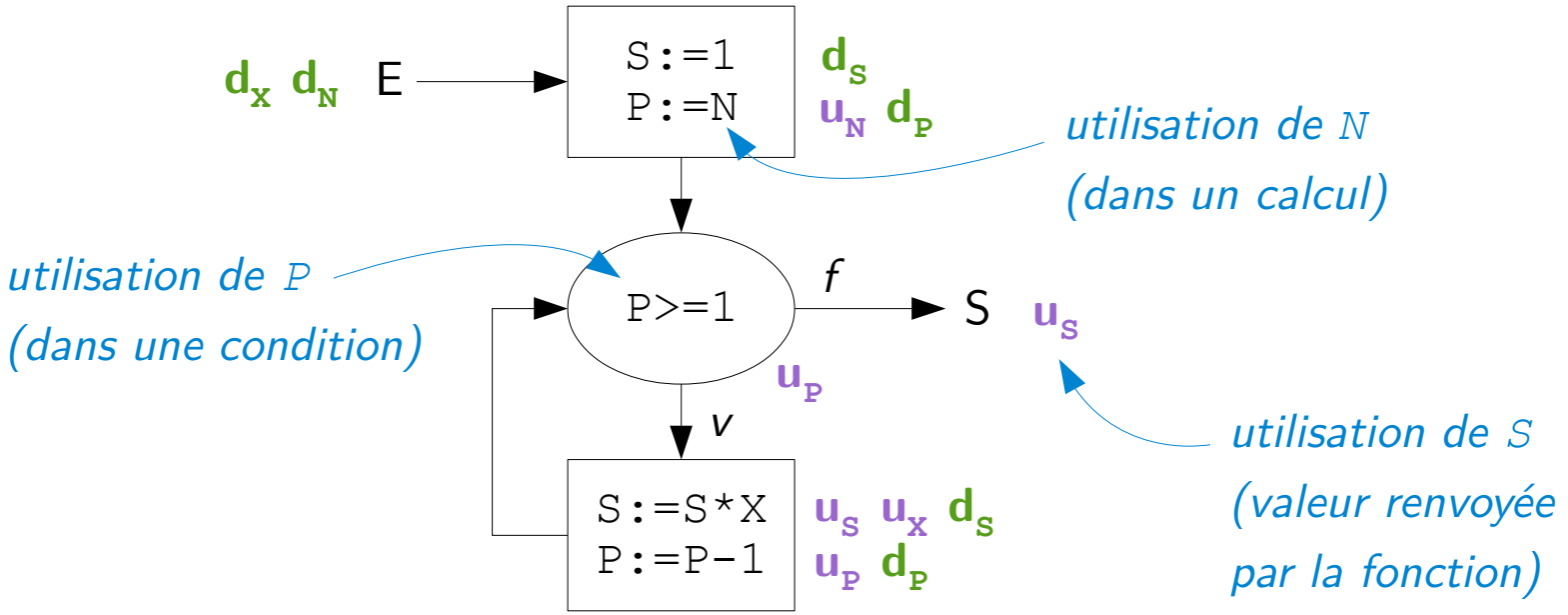


Graphe de flot de données

Annotation du graphe de flot de contrôle par les définitions et les utilisations de variables

Notations :

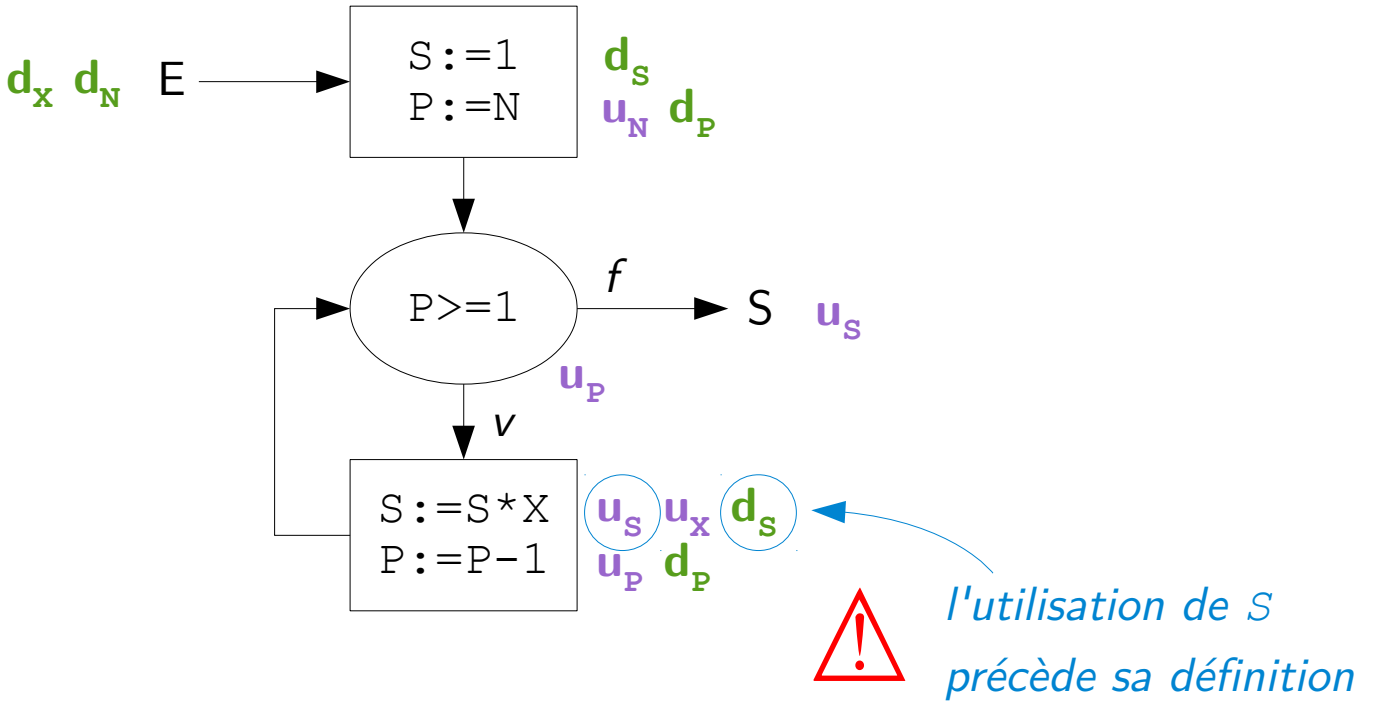
- Définition de X : d_X
- Utilisation de X : u_X



Graphe de flot de données

Annotation du graphe de flot de contrôle par les définitions et les utilisations de variables

Base pour l'analyse des dépendances entre les définitions et utilisations d'une même variable : ordre des annotations important



Analyse du flot de données

Analyse des dépendances entre les définitions et utilisations d'une variable

Statiquement : Une utilisation peut correspondre à plusieurs définitions

Dynamiquement : Chaque utilisation correspond à une seule définition

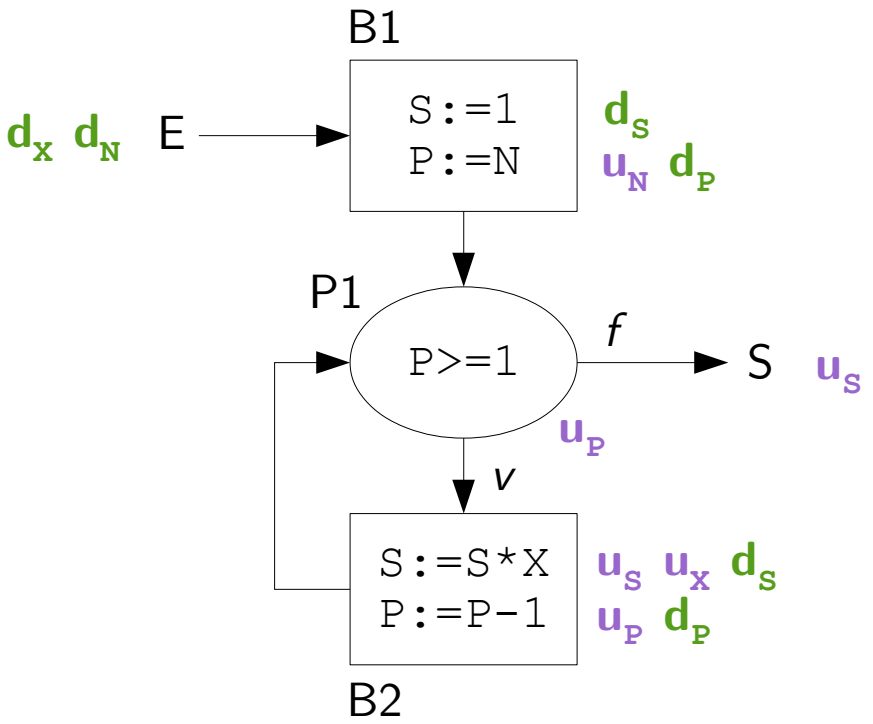
Flot de données pour une variable `X` :

- Pour une exécution du programme : suite des définitions et utilisations (mot sur l'alphabet $\{\mathbf{d_x}, \mathbf{u_x}\}$)
- Pour toutes les exécutions du programme : ensemble des suites de définitions et utilisations (langage sur l'alphabet $\{\mathbf{d_x}, \mathbf{u_x}\}$)

Analyse du flot de données

Flot de données pour `P` :

- Pour le chemin E B1 P1 B2 P1 S : $d_P\ u_P\ u_P\ d_P\ u_P$
- Pour le programme : $d_P\ u_P\ (u_P\ d_P\ u_P)^*$



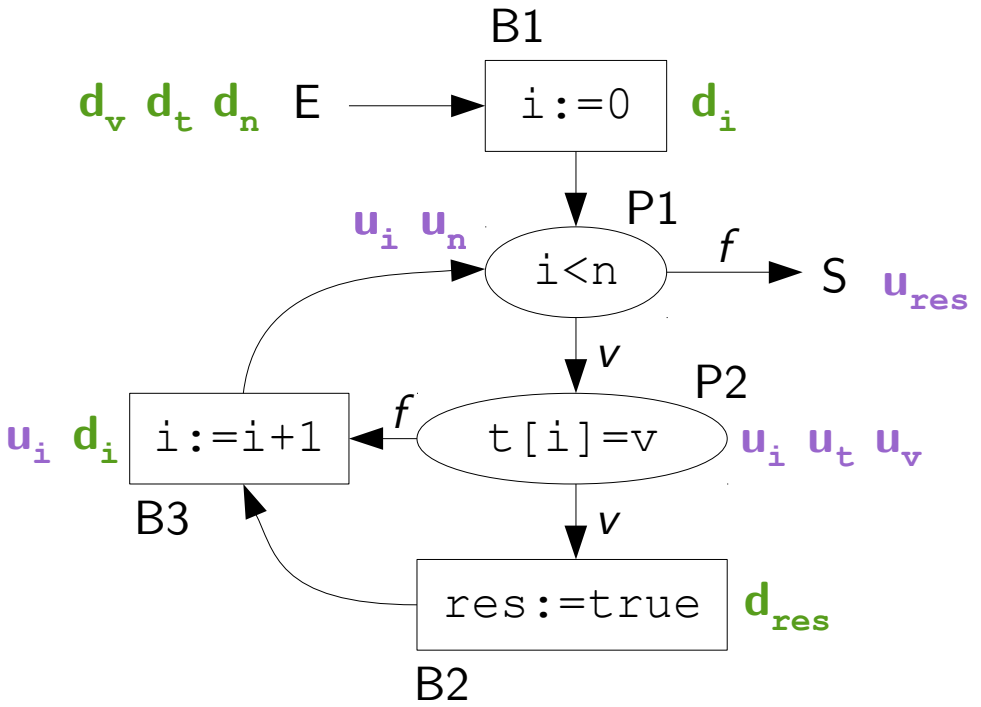
Analyse du flot de données

Flot de données pour `res` :

- Pour le chemin E B1 P1 P2 B2 B3 P1 S: $d_{res}\ u_{res}$
- Pour le programme : $(d_{res})^*\ u_{res}$

Anomalie : `res` peut être utilisée sans avoir été initialisée (tableau vide ou `v` absent)

```
bool find(int t[],int n,int v)
int i := 0;
while (i < n) do
   if (t[i] = v)
   then bool res := true;
   i := i+1;
return res;
```



Analyse du flot de données

Anomalies du flot de données : Présence d'une anomalie dans un chemin si le flot de données associé pour $\mathtt{X}$ est de la forme :

- $u_{\mathtt{X}}$... (commence par une utilisation) : variable non initialisée
- ...$d_{\mathtt{X}}$ (finit par une définition) : mise à jour jamais utilisée
- ...$d_{\mathtt{X}}d_{\mathtt{X}}$... (redéfinition sans utilisation) : mise à jour non utilisée

Anomalie $\neq$ erreur

Mais présence d'anomalies peut rendre certains critères de couverture impossibles à satisfaire (comment tester une définition si elle n'est jamais utilisée ?)

Couverture du flot de données

Notations (bloc = bloc d'instructions ou condition) :

- $d_B(x)$: le bloc B contient une définition de x
- $u_B(x)$: le bloc B contient une utilisation de x

Une définition $d_B(x)$ atteint une utilisation $u_{B'}(x)$ si et seulement si x n'est pas redéfini entre les blocs B et B'

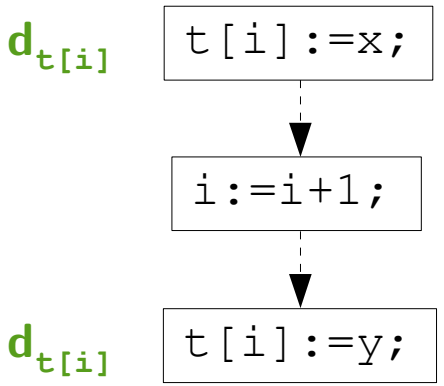
Redéfinition d'une variable

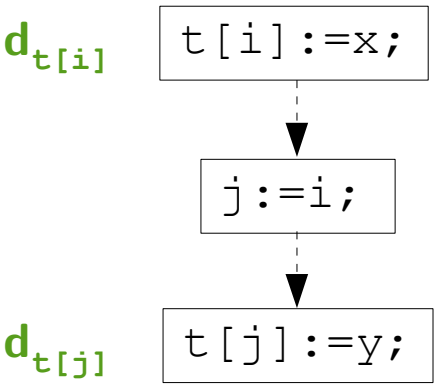
Une définition de x est tuée sur un chemin C s'il existe dans C une redéfinition non ambiguë de x

Non ambiguë ?

- Appel de procédure `p(x,y)` : `x` peut-elle être modifiée par `p` ?
- Définition d'un élément de tableau `t[i] := exp` : tue-t-elle `t[j]` ?
- Définition de la valeur référencée par un pointeur `*a := exp` : tue-t-elle la valeur référencée par un autre pointeur ?

Anomalies du flot de données ?





Critère « toutes les définitions »

Satisfait par un ensemble de chemins T si :

pour toute variable x,

pour toute définition $\mathsf{d}_B(x)$,

il existe au moins une utilisation $\mathsf{u}_{B'}(x)$ atteinte par $\mathsf{d}_B(x)$

telle qu'il existe un chemin de T qui contient BCB'

où C est un chemin sans redéfinition de x

Intuitivement : Toutes les définitions sont utilisées au moins une fois

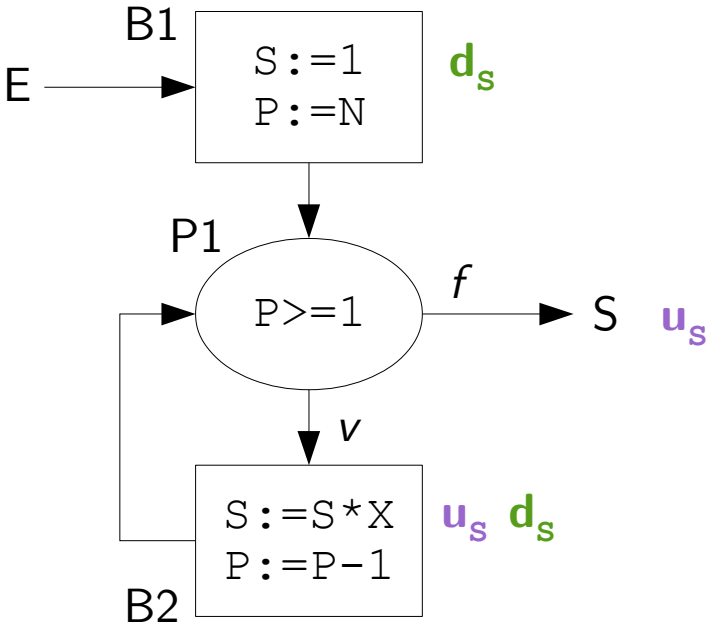
Critère « toutes les définitions »

Pour la variable `S` : définitions en B1 et en B2

Il suffit de couvrir une utilisation de chacune de ces définitions

Par exemple, avec le chemin E B1 P1 B2 P1 S

- Définition en B1 utilisée en B2
- Définition en B2 utilisée en S



Limite du critère : certains couples définition-utilisation non couverts (utilisation en S de la définition en B1 et utilisation en B2 de la définition en B2)

Critère « toutes les utilisations »

Satisfait par un ensemble de chemins T si

pour toute variable x,

pour toute définition $\mathsf{d}_B(x)$,

pour toute utilisation $\mathsf{u}_{B'}(x)$ atteinte par $\mathsf{d}_B(x)$,

il existe un chemin de T qui contient BCB'

où C est un chemin sans redéfinition de x

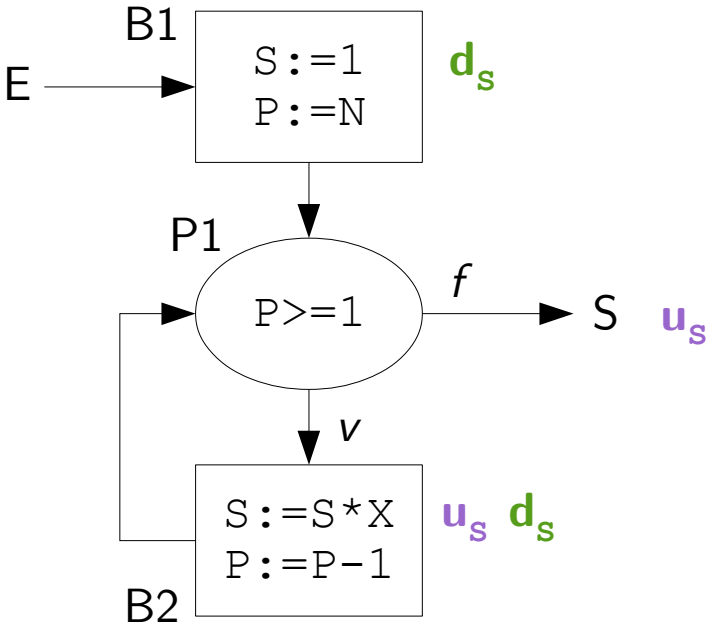
Intuitivement : Toutes les utilisations accessibles par chaque définition

Critère « toutes les utilisations »

Pour la variable S :

- Définitions en B1 et en B2
- Utilisations en B2 et en S

Il suffit de couvrir chaque utilisation de chaque définition



	B	B'	chemin
(1)	B1	B2	E B1 P1 B2 P1 S
(2)	B1	S	E B1 P1 S
(3)	B2	B2	E B1 P1 B2 P1 B2 P1 S
(4)	B2	S	déjà couvert par (1)

⚠ Couple (B1,S) non couvert par (1) car redéfinition en B2

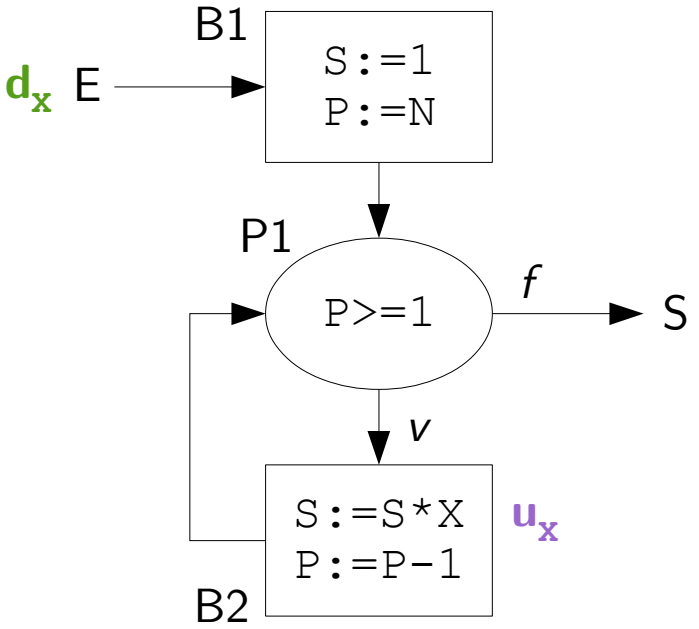
Critère « toutes les utilisations »

Limite du critère « toutes les utilisations » : un seul chemin entre une définition et son utilisation

Définition de X en E utilisée en B2

Chemins entre E et B2 sans redéfinition de X :

E B1 P1 (B2 P1)* B2



Pour limiter le nombre de chemins : seulement les chemins simples entre d_x et u_x

Chemins simples

Chemin simple : chemin sans circuit ou contenant un circuit élémentaire

Autrement dit : au plus un nœud apparaît deux fois dans le chemin

Chemins simples :

- B2 B4 B5 P2 S (chemin sans circuit)
- P1 B2 B4 P1 B3 (chemin contenant un circuit élémentaire)

B1 P1 P2 B3 P1 P2 S : pas un chemin simple

Critère « tous les DU-chemins »

Satisfait par un ensemble de chemins T si

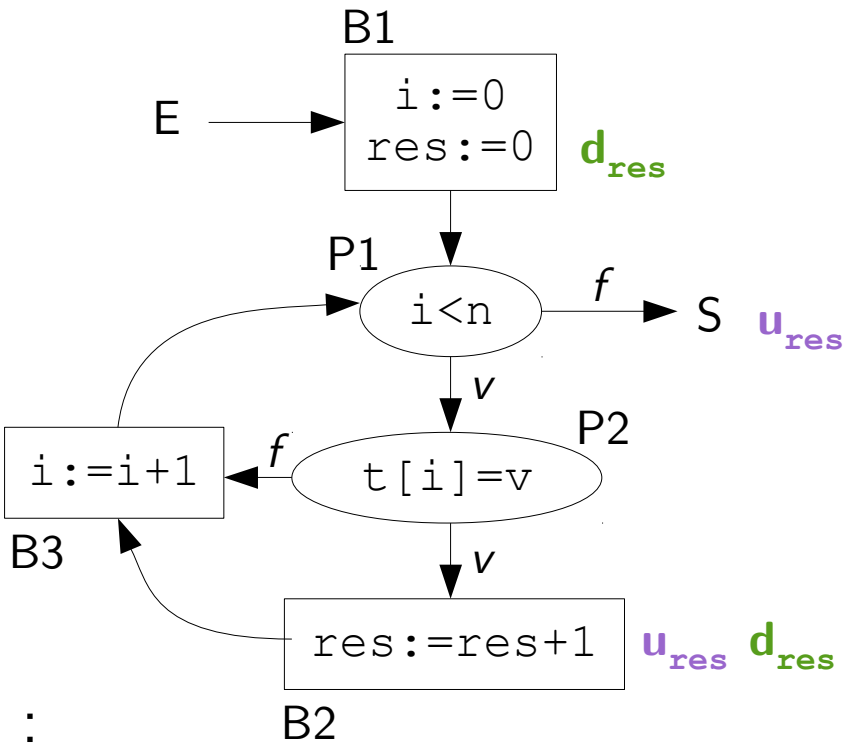
pour toute variable x,
pour toute définition $\mathsf{d}_B(x)$,
pour toute utilisation $\mathsf{u}_{B'}(x)$ atteinte par $\mathsf{d}_B(x)$,
pour tout sous-chemin BCB'
où C est un chemin sans redéfinition de x
et BCB' est simple,
il existe un chemin de T qui contient BCB'

Intuitivement : Tous les chemins (simples) pour chaque couple définition-utilisation (DU)

Critère « tous les DU-chemins »

Pour la variable `res` :

- Définitions en B1 et en B2
- Utilisations en B2 et en S



Couverture du couple (B1,S) :

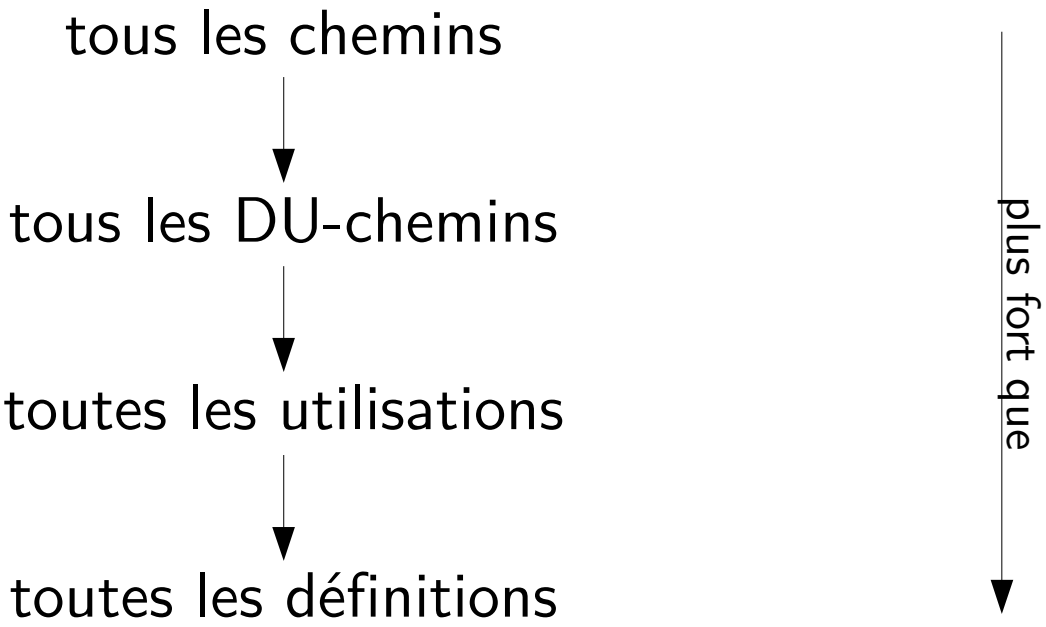
- Chemin sans circuit :

  E B1 P1 S
- Chemin contenant un circuit élémentaire sans redéfinition de `res` :

  E B1 P1 P2 B3 P1 S

Couverture de tous les chemins simples menant de la définition à son utilisation

Hiérarchie des critères de couverture pour le flot de données



Conclusion

Méthodes de sélection de tests structurels :

- ✔ Couverture des exécutions orientée contrôle ou données
- ✔ Mesure de la couverture des tests
- ✔ Méthodes outillées (génération automatique ou mesure de couverture)
- ✔ Complémentaire au test fonctionnel
- ✔ Efficace sur des petites portions de code (test unitaire)

- ✗ Tests dédiés à une implantation : chaque changement de code nécessite de générer de nouveaux tests
- ✗ Passe mal à l'échelle

Conclusion

Utilisations des méthodes de test structurel

- Seules, pour générer un jeu de tests pour un programme selon un ou plusieurs critères de couverture
  - ↳ Outils de génération automatique de tests : Pathcrawler (langage C, CEA), Pex (langage C#, Microsoft)...
- Pour évaluer la qualité (en termes de couverture) d'un jeu de tests existant, et le compléter pour satisfaire complètement les critères visés
  - ↳ Outils de mesure de couverture : CodeCover, Emma, Clover (Java), gcov (C)...

En pratique : test fonctionnel puis couverture de toutes les décisions et de toutes les utilisations

Exemple : norme DO-178C en avionique

Niveau	Définition	Critère structurel requis
A	Un défaut du système ou sous-système étudié peut provoquer un problème catastrophique - Sécurité du vol ou atterrissage compromis - Crash de l'avion	MC/DC
B	Un défaut du système ou sous-système étudié peut provoquer un problème majeur entraînant des dégâts sérieux voire la mort de quelques occupants	toutes les décisions
C	Un défaut du système ou sous-système étudié peut provoquer un problème sérieux entraînant un dysfonctionnement des équipements vitaux de l'appareil	toutes les instructions

Critère de couverture requis en fonction du niveau de criticité du logiciel