Modèles de calcul et Validation d'algorithmes
L3 MIAGE, UGA
2019/2020
raphael.charrondiere@inria.fr

DM : Testons, exploitons

Correction

Exercice 1 Sommer et multiplier les entiers

On considère le programme **P** suivant :
$x := 0$; $z := 1$; **while** z $\leqslant$ y **do** ($x := x + z$; $z := z + 1$)

1. Donner l'invariant et le variant de boucle du programme **P**, justifiez la réponse. *En Bonus Utiliser la logique de Hoare pour montrer* $\{y = n \wedge n \geqslant 0\}\mathbf{P}\{x = \frac{n\times(n+1)}{2}\}$. *(On annotera le programme entre les commandes comme dans l'exemple en cours.)*
   Correction : On définit le variant par $y - x$, cette quantitée est strictement décroissante, et l'on sort de la boucle quand elle devient strictement négative.
   On définit l'invariant par $x = \frac{z(z-1)}{2}$. En effet juste avant la boucle x vaut 0 et z vaut 1. Si c'est vrai en début de boucle $x = \frac{z(z-1)}{2}$, on note avec un prime les variables en fin d'itération, on a $x' = x + z$ et $z' = z + 1$, ainsi $x' = \frac{z(z-1)}{2} + z = \frac{z^2-z+2z}{2} = \frac{z^2+z}{2} = \frac{z(z+1)}{2} = \frac{z'(z'-1)}{2}$
   En utilisant la logique de Hoare (l'implication $(*)$ utilise ce que l'on vient d'écrire)

$\{y = n \wedge n \geqslant 0\} \implies \{0 = 0 \wedge \wedge 1 \leqslant y + 1\}$
$x := 0$
$\{x = \frac{1(1-1)}{2} \wedge 1 \leqslant y + 1\}$
$z := 1$
$\{x = \frac{z(z-1)}{2} \wedge z \leqslant y + 1\}$
while$(z \leqslant y)$
  $\{x = \frac{z(z-1)}{2} \wedge z \leqslant y + 1 \wedge z \leqslant y\} \implies \{x + z = \frac{z(z+1)}{2} \wedge z + 1 \leqslant y + 1\}(*)$
  $x := x + z$
  $\{x = \frac{z(z+1)}{2} \wedge z + 1 \leqslant y + 1\}$
  $z := z + 1$
  $\{x = \frac{z(z-1)}{2} \wedge z \leqslant y + 1\}$
$\{x = \frac{z(z-1)}{2} \wedge z > y \wedge z \leqslant y + 1\} \implies \{x = \frac{n(n+1)}{2}\}$

Exercice 2 Hésiter entre deux instructions

On se propose d'introduire du non-déterminisme dans IMP. On introduit ainsi la commande **maybe do** <commande> **otherwise do** <commande>.

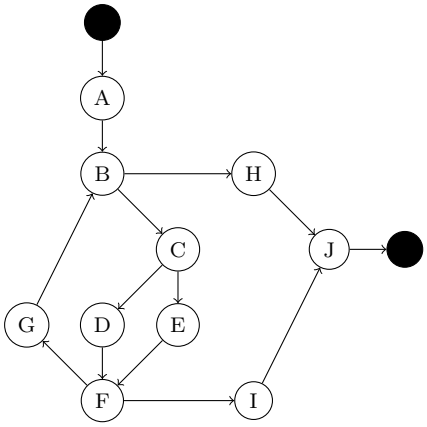
1. Donner les deux règles d'inférence pour la sémantique à grands pas.
   Correction :
   $$\frac{\sigma, c_1 \Downarrow \sigma_F}{\sigma, \textbf{maybe do } c_1 \textbf{ otherwise do } c_2 \Downarrow \sigma_F} \qquad \frac{\sigma, c_2 \Downarrow \sigma_F}{\sigma, \textbf{maybe do } c_1 \textbf{ otherwise do } c_2 \Downarrow \sigma_F}$$
2. Donner les deux règles d'inférence pour la sémantique à petits pas.
   Correction :
   $$\frac{}{\sigma, \textbf{maybe do } c_1 \textbf{ otherwise do } c_2 \to \sigma, c_1} \qquad \frac{}{\sigma, \textbf{maybe do } c_1 \textbf{ otherwise do } c_2 \to \sigma, c_2}$$

3. Comment traduire cette commande dans un graphe de flot de contrôle ? (Faut-il créer un nouveau type de nœud, ou au cas contraire, comment intégrer cette commande dans le graphe ?)
Correction : Structurellement le « maybe » est similaire au if, c'est comme un nœud de condition, sauf qu'il n'y a pas de condition et l'éxecution peut passer par l'un des deux arcs sortants. On peut alors remplacer la condition par le symbole **?**. Note : Outre dessiner le noeud, il était attendu d'expliquer la sémantique/le comportement du noeud.

Exercice 3 Parti indépendantiste

Cette partie porte sur les chemins indépendants.

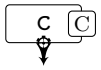
1. Calcul des chemins indépendants



Donnez une base de chemin indépendants. Justifiez votre réponse.
Correction : On peut calculer la complexité cyclomatique pour savoir combien une base a de chemins indépendants. Ici il y a 12 nœuds et 14 arcs, donc elle est de 4. Proposons les chemins ① =ABHJ, ② = ABCDFIJ, ③ = ABCEFIJ et ④ = ABCDFGBHJ. Il y a plein de manière de montrer qu'ils sont indépendants, en particulier en notant v_i le vecteur correspondant au chemin ⓘ, une combinaison linéaire des vecteurs correspondants $av_1 + bv_2 + cv_3 + dv_4$ ne doit s'annuler que si a, b, c, d sont tous nuls.
On résoud donc $av_1 + bv_2 + cv_3 + dv_4 = 0$, chaque ligne des vecteurs correspond à un arc. La ligne de l'arc (GF) nous donne l'équation $d = 0$, celle de (CE) $c = 0$, celle de (CD) $b + d = 0$ celle de (BH) $a + d = 0$. La seule solution est alors $a = b = c = d = 0$

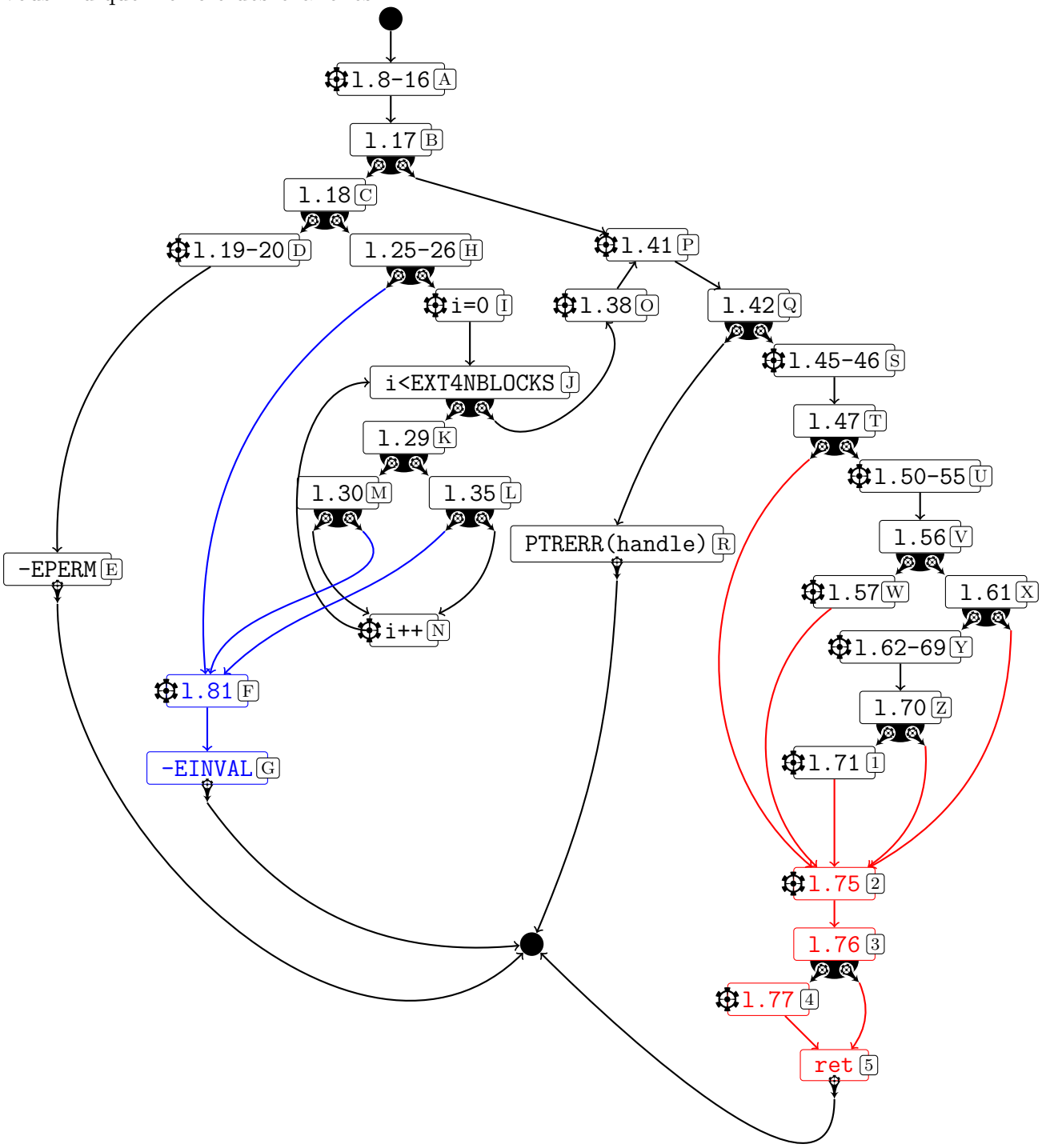
2. Soit un graphe $\mathcal{G} = (N, A)$ où N est l'ensemble de ses $n = \#N$ nœuds et A l'ensemble de ses $a = \#A$ arcs. On rappelle que la complexité cyclomatique se définit par $v(\mathcal{G}) = 2 + a - n$.
Montrez qu'en notant ϕ le nombre de nœuds de décision, alors $v(\mathcal{G}) = \phi + 1$.
Correction : Les arcs sont orientés, pour les compter il suffit de compter les arcs sortants pour chaque nœud. Pour un nœud d'entrée il y a un arc sortant, pour un nœud de sortie il n'y a pas d'arc sortant, pour un nœud de commande il y a un arc sortant, pour un nœud de décision il y a deux arcs sortants. Si l'on note ψ le nombre de nœuds de commandes, étant donné qu'il y a un unique nœud d'entrée et un unique nœud de sortie on a $n = \phi + \psi + 1 + 1$, le nombre d'arc est $a = \phi + 2\psi + 1$. On peut alors calculer $v(\mathcal{G}) = 2 + a - n = 2 + \psi + 2\phi + 1 - \phi - \psi - 1 - 1 = \phi + 1$

Exercice 4 Graphe de flot de contrôle

1. Nous avons vu en cours les nœuds de commande et les nœuds de condition, il peut être utile de définir un autre type de nœud pour traduire l'instruction `return c`.
Proposez une définition pour ce type de nœud.
Indice : Ai-je toujours le droit de rajouter un tel nœud s'il respecte la définition donnée en réponse ? Si non, alors la définition donnée est fausse/incomplète.
Correction : L'instruction return est la dernière éxécutée, on définit alor le nœud return de la manière suivante : ce type de nœud possède 1 seul arc sortant connecté à l'unique nœud de sortie. (On pourrait rajouté qu'il y a au moins un arc entrant, mais par définition de graphe de flot de contrôle, seul le nœud d'entrée n'a pas d'arc entrant). Dans la question suivante, on représentera le nœud `return c` par 

2. Dessiner le graphe de flot de contrôle de la fonction `ext4_convert_meta_bg` en annexe.

Remarque : On considère que les appels de fonctions sont des instructions élementaires.
Remarque : Pour simplifier la mise en page, dans les nœuds qui ne sont pas des nœuds de décision, on inscrira uniquement les numéros de lignes exécutées.
Remarque : Il s'agit d'une fonction prise dans le noyau Linux, cette fonction participe au redimensionnement de partitions ext4.
Correction : Note : les couleurs n'ont qu'une fonction de lisibilité, elles n'ont pas de sens, même si on pourrat leur en donner un. Note : en suivant les conventions du cours, le branche vrai est toujours à droite, mais rien ne vous empêchait de ne pas suivre cette convention si vous indiquez le rôle des branches.



Exercice 5 Enchaîner les ifs

Un magasin veut inciter les clients à acheter les produits alimentaire en fin de vie en premier

pour éviter les pertes. Il baisse alors les prix suivants plusieurs critères, en considérant la date de péremption et le stock disponible. Si plusieurs critères sont applicables, le dernier critère de la liste est appliqué

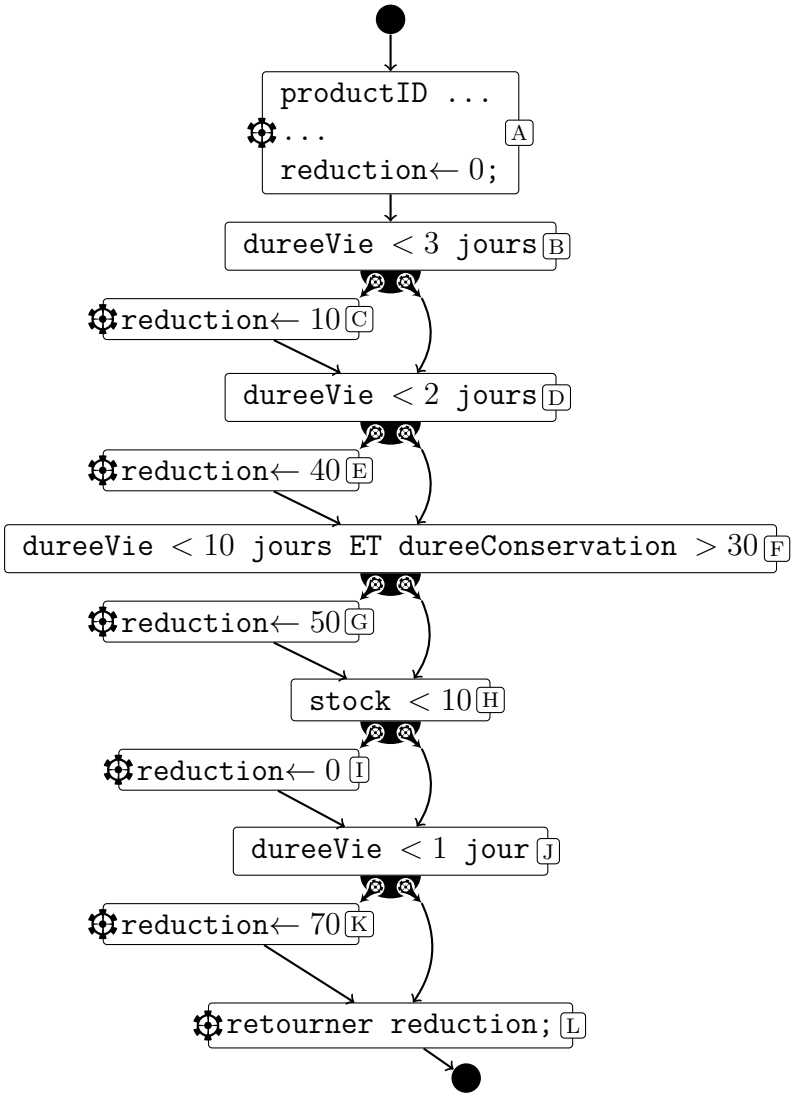
C1 : Si la date de péremption est dans les trois jours, le produit reçoit une réduction de 10%
C2 : Si la date de péremption est dans les deux jours, le produit reçoit une réduction de 40%
C3 : Si la date de péremption est dans les dix jours et qu'il s'agisse d'un article à longue conservation, le produit reçoit une réduction de 50%
C4 : Si le stock est faible le produit n'est pas diminué
C5 : Si la date de péremption est le jour même, le produit reçoit une réduction de 70%

Le magasin veut coder un programme pour afficher la réduction à appliquer. Prenons la solution la plus simple :

```
productID ← readInput();
dureeVie ← getExpiration(productID)-clock();
dureeConservation ← getDureeConservation(productID);
stock ← getStock(productID);
reduction← 0;
si dureeVie<3 jours alors reduction ← 10;
si dureeVie<2 jours alors reduction ← 40;
si dureeVie<10 jours ET dureeConservation>30 jours alors reduction ← 50;
si stock<10 alors reduction ← 0;
si dureeVie<1 jour alors reduction ← 70;
retourner reduction;
```

Algorithme 1 : Réduction

1. Donner le graphe de flot de contrôle.
   Correction : Note : en suivant les conventions du cours, le branche vrai est toujours à droite, mais rien ne vous empêchait de ne pas suivre cette convention si vous indiquez le rôle des branches.



2. Donner des données de test pour couvrir le critère "tous les arcs"
   Correction :
   En utilisant part exemple les données de test
   — [dureeVie=0, dureeConsevation =31, stock =3]
   — [dureeVie=5, dureeConsevation = 10, stock = 30]
   on couvre tout les arc (ici, toutes les conditions sont vraies pour la première DT, et fausses pour la seconde). Cette réponse n'est qu'un exemple, et il n'y a pas de réponse unique. On pourrait aussi proposer
   — [dureeVie=5, dureeConsevation =31, stock =3]
   — [dureeVie=0, dureeConsevation =5, stock =7]
   — [dureeVie=2, dureeConsevation = 10, stock = 24]
3. Écrire l'expression algébrique des chemins.
   En réutilisant la notation du cours A ? qui signifie zéro ou une fois A, on obtient ABC ?DE ?FG ?HI ?JK ?
   (Note : plusieurs syntaxes sont possibles, par exemple A(B+BC)(D+DE)...)
4. Pour chaque chemin donner une donnée de test (on supposera que `clock()` renvoie le 1/1/2020).
   Indice : Regardez la question suivante.
   Correction : (Note : Personne n'a voulu utiliser la date d'expiration, pourtant c'était ce qu'il fallait utiliser et non dureeConservation. Je vais faire comme tout le monde du coup). Certains chemins n'ont pas de données de tests, (par exemple si dureeVie< 2 alors dureeVie< 3)
   — `ABCDEFGHIJKL` [dureeVie=0, dureeConsevation =31, stock =3]
   — `AB DEFGHIJKL` Pas de DT
   — `ABCD FGHIJKL` Pas de DT
   — `ABCDEF HIJKL` [dureeVie=0, dureeConsevation = 10, stock = 3]

— `ABCDEFGH JKL` [dureeVie=0, dureeConsevation = 100, stock = 30]
— `ABCDEFGHIJ L` [dureeVie=1, dureeConsevation = 100, stock = 3]
— `AB D FGHIJKL` Pas de DT
— `AB DEF HIJKL` Pas de DT
— `AB DEFGH JKL` Pas de DT
— `AB DEFGHIJ L` Pas de DT
— `ABCD F HIJKL` Pas de DT
— `ABCD FGH JKL` Pas de DT
— `ABCD FGHIJ L` [dureeVie=2, dureeConsevation = 100, stock = 3]
— `ABCDEF H JKL` [dureeVie=0, dureeConsevation = 10, stock = 30]
— `ABCDEF HIJ L` [dureeVie=1, dureeConsevation = 10, stock = 3]
— `ABCDEFGH J L` [dureeVie=1, dureeConsevation = 100, stock = 30]
— `AB D F HIJKL` Pas de DT
— `AB D FGH JKL` Pas de DT
— `AB D FGHIJ L` [dureeVie=5, dureeConsevation = 100, stock = 3]
— `AB DEF H JKL` Pas de DT
— `AB DEF HI JL` Pas de DT
— `AB DEFGH J L` Pas de DT
— `ABCD F H JKL` Pas de DT
— `ABCD F HI JL` [dureeVie=2, dureeConsevation = 10, stock = 3]
— `ABCD FGH J L` [dureeVie=2, dureeConsevation = 100, stock = 30]
— `ABCDEF H J L` [dureeVie=1, dureeConsevation = 10, stock = 30]
— `AB D F H JKL` Pas de DT
— `AB D F HIJ L` [dureeVie=20, dureeConsevation = 100, stock = 5]
— `AB D FGH J L` Pas de DT
— `AB DEF H J L` Pas de DT
— `ABCD F H J L` [dureeVie=2, dureeConsevation = 10, stock = 30]
— `AB D F H J L` [dureeVie=5, dureeConsevation = 10, stock = 30]

5. Quel problème y a-t-il à décorréler la structure du code ?
   Correction : Le problème est que certains chemins sont impraticables, c'est à dire qu'il n'y a pas forcément de données de test pour un chemin donné comme le montre la question précédente.

6. Pour pouvoir exploiter un critère « tous les chemins », il faut réécrire le code. Proposez une nouvelle version du code. On demande à ce que l'instruction `retourner` soit utilisée une seule fois.
   Correction :

```
productID ← readInput();
dureeVie ← getExpiration(productID)-clock();
dureeConservation ← getDureeConservation(productID);
stock ← getStock(productID);
reduction← 0;
si dureeVie<1 jour alors reduction ← 70;
sinon si stock<10 alors ;
sinon si dureeVie<10 jours ET dureeConservation>30 jours alors reduction ← 50;
sinon si dureeVie<2 jours alors reduction ← 40;
sinon si dureeVie<3 jours alors reduction ← 10;
retourner reduction;
```

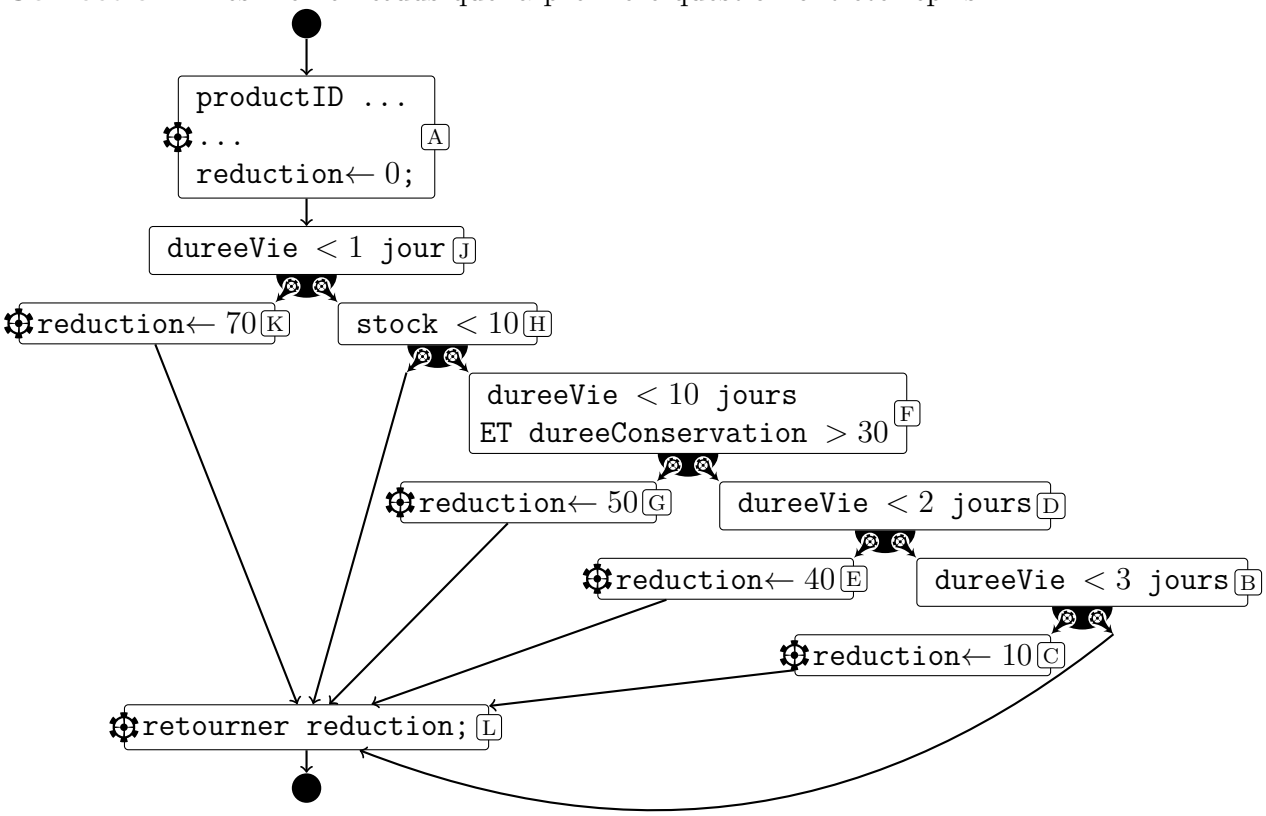
Algorithme 2 : Réduction

   (Note, la réponse n'est pas unique, on peut trouver d'autres variantes de cet algorithme qui répondent à l'énoncé.)

7. Donner le graphe de flot de contrôle du nouveau code, puis les données de test pour couvrir

le critère « tous les chemins ».

Correction : Les même nœuds que la première question ont été repris.



Il y a maintenant 6 chemins, donnons en la liste avec une donnée de test pour chacun, l'ensemble des données de test couvrira alors le critère « tous les chemins ».

— `AJKL` [dureeVie=0, dureeConsevation = 10, stock = 30]
— `AJHL` [dureeVie=2, dureeConsevation = 10, stock = 3]
— `AJFGL` [dureeVie=2, dureeConsevation = 100, stock = 30]
— `AJFDEL` [dureeVie=1, dureeConsevation = 10, stock = 30]
— `AJFDBCL` [dureeVie=2, dureeConsevation = 10, stock = 30]
— `AJFDBL` [dureeVie=5, dureeConsevation = 10, stock = 30]