

# DM Informatique Théorique

Marion Medeville  
Elias Rhouzlane

November 28, 2014

## Abstract

Nous avons listé dans ce document toutes les signatures et axiomes des fonctions de notre programme.

## Lexique

**CMDBoisson** : *CommandeBoisson*

**S, Stocks** : *Machine.Stocks*

**I, Ingredients** : *Machine.Ingredients*

## Signatures et Axiomes

$\text{Machine} : \text{Monnaie} \times \text{CMDBoisson} \longrightarrow (\text{Monnaie} \times \text{Boisson}) \cup \text{Erreur}$

## Fonction tarifs

### Signature

- $\emptyset \longrightarrow E = (\text{NomIngredient} \times \text{Prix})^{\# \text{Ingredient}}$

### Axiomes

- $\forall \text{ nom, prix} \in E, \text{get\_prix}(\text{nom}) = \text{prix}$
- $\text{dim}E = \# \text{ Ingrédients}$

## Fonction `get__max__stock`

### Signature

- $Texte \longrightarrow Taille$

### Axiomes

- $\forall nom \in Texte \Rightarrow get\_stock(nom) \in Stocks$
- $\forall taille\_max \in Taille \Rightarrow taille\_max \in \mathbb{N}$
- $\forall stock \in C, \exists get\_max\_stock(nom)$  avec  $nom = get\_nom(stock)$

## Fonction `get__stock__size`

### Signature

- $Texte \longrightarrow Taille$

### Axiomes

- $nom \in Texte \Rightarrow get\_stock(nom) \in Stocks$
- $taille \in Taille \Rightarrow taille \in \mathbb{N}$  et  $taille = longueur(get\_stock(nom))$
- $\forall stock \in C, \exists get\_stock\_size(nom)$  avec  $nom = get\_nom(stock)$

## Fonction `get__stock`

### Signature

- $Text \longrightarrow Stock$

### Axiomes

- $\forall nom \in Text, \exists! stock \in Stocks$  tel que  $get\_nom(stock) = nom$

## MODE FONCTIONNEMENT

### Fonction commander

#### Signature

- $Monnaie \times Cmd_{Boisson} \rightarrow (Monnaie \times Monnaie \times Boisson) \cup Monnaie$

#### Axiomes

- Soient,  
 $(a, b, c, d, e, f), (g, h, i, j, k, l), (m, n, o, p, q, r), (s, t, u, v, w, x) \in (\{0, 1\})^6$   
quatre tuples binaires de longueur 6.  
 $\exists(a, b, c, d, e, f)$  tel que  $mo = (a, b, c, d, e, f)$  avec  $mo \in Monnaie$ ,  
 $\exists(g, h, i, j, k, l)$  tel que  $cmdb = (g, h, i, j, k, l)$  avec  $cmdb \in CMD_{Boisson}$ ,  
Si commande impossible retourner  $mo$ ,  
Sinon retourner  $mo_1, mo_2, boisson$  tel que  
 $\exists(m, n, o, p, q, r)$  tel que  $mo_1 = (m, n, o, p, q, r)$  avec  $mo_1 \in Monnaie$ ,  
 $\exists(s, t, u, v, w, x)$  tel que  $mo_2 = (s, t, u, v, w, x)$  avec  $mo_2 \in Monnaie$ ,  
 $mo_1 = mo - 2\text{€}$   
 $mo_1 + mo_2 = mo - \text{Prix}(cmdb)$

### Fonction preparer\_\_commande

#### Signature

- $Stocks \times (Ingrédient \times Quantité)^{\#Ingrédient} \longrightarrow Boisson$

#### Axiomes

- Soit  $(a, b, c, d, e, f) \in N^6$ ,  $(g, h, i, j, k, l) \in N^6$  et  $(m, n, o, p, q, r) \in N^6$  trois tuples de binaire de 6 entiers.  
 $Ingrédients = (a, b, c, d, e, f)$ ,  $Commande = (g, h, i, j, k, l)$ ,  $Boisson = (m, n, o, p, q, r)$   
 $BoissonCommandée = Boisson$

## Fonction verifier\_\_commande

### Signature

- $CMD_{boisson} \longrightarrow \{V, F\}$

### Axiomes

- Soit  $(a,b,c,d,e,f) \in N^6$  un tuple de binaire de 6 entiers. Commande =  $(a,b,c,d,e,f)$   
 $a,b,c,d,e,f \in \{0,1\}$   
 $\text{longueur}(\text{Commande}) = \text{nombre d'ingrédients} + 1$

## Fonction verifier\_\_stock\_suffisant

### Signature

- $Stocks \times (Ingrédient \times Quantité)^{\#Ingrédient} \longrightarrow \{V, F\}$

### Axiomes

- Soit  $(a,b,c,d,e,f) \in N^6$  un tuple de binaire de 6 entiers.  
Commande =  $(a,b,c,d,e,f)$   
Soit Stock un dictionnaire contenant en clé le nom de l'ingrédient et en valeur le stock restant correspondant.  
 $\text{Commande}[i] \in \text{Stock}[i]$

## Fonction verifier\_\_monnaie

### Signature

- $Monnaie \longrightarrow \{V, F\}$

### Axiomes

- Soit  $(a,b,c,d,e,f) \in N^6$  un tuple de binaire de 6 entiers.  
Monnaie =  $(a,b,c,d,e,f)$   
 $\text{Longueur}(\text{Monnaie}) = \text{nombre de pièces différentes}$

## Fonction ramener\_deux\_euros

### Signature

- $Monnaie \longrightarrow Monnaie \times Monnaie$

### Axiomes

- Soit  $(a,b,c,d,e,f) \in N^6$ ,  $(g,h,i,j,k,l) \in N^6$  et  $(m,n,o,p,q,r) \in N^6$  trois tuples de binaire de 6 entiers.  
Monnaie =  $(a,b,c,d,e,f)$ , Monnaie1 =  $(g,h,i,j,k,l)$ , Monnaie2 =  $(m,n,o,p,q,r)$   
 $g \in a, h \in b, I \in c, j \in d, k \in e, l \in f$ ,  
somme(m,n,o,p,q,r)=2  
somme(a,b,c,d,e,f)=somme(g,h,I,j,k,l)+2  
Monnaie[i]=Monnaie1[i]+Monnaie2[i]

## Fonction vérifier\_rendu\_monnaie\_possible

### Signature

- $Monnaie \times Prix \longrightarrow \{V, F\}$

### Axiomes

- Soit  $(a,b,c,d,e,f) \in N^6$  un tuple de binaire de 6 entiers.  
Monnaie =  $(a,b,c,d,e,f)$   
Si  $\exists (g,h,i,j,k,l) \in N^6$  un tuple de binaire de 6 entiers tel que  
 $(g,h,i,j,k,l) = Monnaie - PrixBoisson$ , alors retourner Vrai  
Sinon retourner Faux

## Fonction rendre\_monnaie

### Signature

- $Monnaie \times Stocks \longrightarrow Monnaie$

### Axiomes

- Soit  $(a,b,c,d,e,f) \in N^6$  et  $(g,h,i,j,k,l) \in N^6$  deux tuples de binaire de 6 entiers.  
Monnaie = (a,b,c,d,e,f), Monnaie2 = (g,h,i,j,k,l)  
Monnaie[i] ∈ Sotck[i]  
somme(a,b,c,d,e,f) = somme(g,h,I,j,k,l)-prix(commande)

## Fonction formater\_commande

### Signature

- $CMD_{Boisson} \longrightarrow (Ingredient \times Quantité)^{\#Ingredients}$

### Axiomes

- Soit  $(a,b,c,d,e,f) \in N^6$  un tuple de binaire de 6 entiers. Soit  $(g,h,i,j,k) \in N^5$  un tuple de 5 entiers appartenant à  $[0, 3]$ .  
Commande = (a,b,c,d,e,f) et CommandeAjustée = (g,h,i,j,k)  
g = transformation du tuple (a,b) binaire en entier, h=c, i=d, j=e, k=f.

## Fonction get\_prix\_boisson

### Signature

- $(Ingredient \times Quantité)^{\#Ingredients} \longrightarrow Prix$

### Axiomes

- Soit  $(a,b,c,d,e) \in N^5$  un tuple de binaire de 6 entiers.  
Commande = (a,b,c,d,e)  
Somme = prix[ingrédient] × Commande[ingrédient]

## Fonction match

### Signature

- $(Ingredient \times Quantité)^{\#Ingredients} \longrightarrow TypeBoisson \times (TypeSupplement)^n$

### Axiomes

- Pour  $i$  dans  $[0 ; 5]$  : Si Ingrédients[i] ∈ composition d'une boisson  $\alpha$  et  $\alpha[Ingrédients] \in Ingrédients$  alors retourner Vrai  
Retourner Faux

MODE MAINTENANCE :

## Fonction changer\_\_prix\_\_unitaire

### Signature

- $Text \times (\mathbb{N} \cup (\mathbb{N} \times \mathbb{N})^n) \longrightarrow \emptyset$

### Axiomes

- Prix\_Unitaire = Nouveau\_Prix

## Fonction prix\_\_unitaire

### Signature

- $Text \longrightarrow Prix$

### Axiomes

- Prix = Prix[Ingrédient]

## Fonction set\_\_max\_\_stock

### Signature

- $Text \times \mathbb{N} \longrightarrow \emptyset$

### Axiomes

- $\text{Max\_Stock}[\text{Ingrédient}] = \text{Nouveau\_Max\_Stock}$

### Fonction reset

#### Signature

- $\emptyset \longrightarrow \emptyset$

### Axiomes

- ancien historique = nouvel historique

### Fonction vider\_\_caisse

#### Signature

- $\emptyset \longrightarrow \emptyset$

### Axiomes

- Soit  $(a,b,c,d,e,f) \in N^6$  un tuple de binaire de 6 entiers.  
Caisse =  $(a,b,c,d,e,f)$   
Caisse\_vide =  $(0,0,0,0,0,0)$

### Fonction get\_\_all\_\_stock

#### Signature

- $\emptyset \longrightarrow \text{Stocks}$

### Axiomes

- Soit Stock2 un dictionnaire ayant pour clé le nom de chaque ingrédients et pour valeur le stock correspondant.  
nombre de clés du dictionnaire = nombre d'ingrédients



## Fonction remplir\_stock

### Signature

- $Text \longrightarrow \emptyset$

### Axiomes

- $Stock(Ingredient) = Max\_stock$

## Fonction remplir\_tout\_stock

### Signature

- $\emptyset \longrightarrow \emptyset$

### Axiomes

- $Stocks[i] = Max\_Stock[i]$

## Fonction ajouter\_stock

### Signature

- $Text \times \mathbb{N} \longrightarrow \emptyset$

### Axiomes

- Si  $Stock + quantite \in Max\_Stock$ , alors  $Nouveau\_Stock = Stock + Quantite$ , Sinon  $Nouveau\_Stock = Stock$

## Fonction historique

### Signature

- $\emptyset \longrightarrow Historique$

### Axiomes

- Axiome :

## COMPLEXITES/ORDRE DE GRANDEUR

Def tarifs :

- Complexité : 12
- Ordre de grandeur :  $O(1)$

Def stocks :

- Complexité : 12
- Ordre de grandeur :  $O(1)$

Def changer\_prix\_unitaire :

- Complexité :  $3n$
- Ordre de grandeur :  $O(n)$

Def prix\_unitaire :

- Complexité : 2
- Ordre de grandeur :  $O(1)$

Def set\_max\_stock :

- Complexité :  $n + 3$
- Ordre de grandeur :  $O(n)$

Def reset :

- Complexité :  $2n + 25$
- Ordre de grandeur :  $O(n)$

Def vider\_caisse :

- Complexité :  $5n$
- Ordre de grandeur :  $O(n)$

Def get\_stock :

- Complexité : 3
- Ordre de grandeur :  $O(1)$

Def get\_stock\_size :

- Complexité : 3
- Ordre de grandeur :  $O(1)$

Def get\_stock\_max :

- Complexité : 5
- Ordre de grandeur :  $O(1)$

Def get\_all\_stock :

- Complexité :  $5n$
- Ordre de grandeur :  $O(n)$

Def remplir\_stock :

- Complexité :  $3n + 10$
- Ordre de grandeur :  $O(n)$

Def remplir\_tout\_stock :

- Complexité :  $3n + 17$
- Ordre de grandeur :  $O(n)$

Def ajouter\_stock :

- Complexité : 19
- Ordre de grandeur :  $O(1)$

Def get\_historique :

- Complexité : 3
- Ordre de grandeur :  $O(1)$

Def display\_stats :

- Complexité :  $5 + 6n$
- Ordre de grandeur :  $O(n)$

Def verifier\_commande :

- Complexité :  $11 + 4n$
- Ordre de grandeur :  $O(n)$

Def trad :

- Complexité : 23
- Ordre de grandeur :  $O(1)$

Def \_\_get\_boites :

- Complexité :  $4n + 5$
- Ordre de grandeur :  $O(n)$

Def \_\_verifier\_monnaie :

- Complexité :  $15 + 20n + 12n^2$
- Ordre de grandeur :  $O(n^2)$

Def \_\_verifier\_rendu\_monnaie\_possible :

- Complexité :  $6 + 20n + 13n^2$
- Ordre de grandeur :  $O(n^2)$

Def match :

- Complexité :  $8n^2 + 5n + 2$
- Ordre de grandeur :  $O(n^2)$

Def `_calculer_prix_boisson` :

- Complexité :  $3n^3 - 3n^2$
- Ordre de grandeur :  $O(n^3)$

Def `calculer_prix_boisson` :

- Complexité :  $24 + 3n^3 - n^2$
- Ordre de grandeur :  $O(n^3)$

Def `__verifier_stock_suffisant` :

- Complexité :  $5n + 2$
- Ordre de grandeur :  $O(n)$

Def `__preparer_commande` :

- Complexité :  $22 + 17n + 8n^2$
- Ordre de grandeur :  $O(n^2)$

Def `commander` :

- Complexité :  $91 + 80n + 41n^2 + n^3$
- Ordre de grandeur :  $O(n^3)$