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

Machine : Monnaie × CMDBoisson $\longrightarrow$ (Monnaie × Boisson) ∪ Erreur

Fonction tarifs

Signature

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

Axiomes

- ∀ nom, prix ∈ E, get_prix(nom) = prix
- $dimE$ = # 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,
  $(a, b, c, d, e, f), (g, h, i, j, k, l), (m, n, o, p, q, r), (s, t, u, v, w, x) \in (\{0, 1\}^6)^4$
  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€

  $mo_1 + mo_2 = mo$ – Prix(cmdb)

Fonction preparer_commande

Signature

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

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}

  longueur(Commande) = nombre d'ingrédients + 1

Fonction verifier_stock_suffisant

Signature

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

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.

  Commande[i] $\in$ 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)

  Longueur(Monnaie) = 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] $\in$ 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]$\times$Commande[ingrédient]

Fonction match

Signature

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

Axiomes

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

  Retourner Faux

MODE MAINTENANCE :

Def changer_prix_unitaire :

- Signature : Ingrédient $\times$ Nouveau_Prix Prix_Unitaire
- Axiome : Prix_Unitaire = Nouveau_Prix

Def prix_unitaire :

- Signature : Ingrédient Prix
- Axiome : Prix = Prix[Ingrédient]

Def set_max_stock :

- Signature : Ingrédient $\times$ Nouveau_Max_Stock Max_Stock
- Axiome : Max_Stock[Ingrédient] = Nouveau_Max_Stock

Def reset :

- Signature : Distributeur Distributeur
- Axiome : ancien historique = nouvel historique

Def vider_caisse :

- Signature : Caisse Caisse_vide
- Axiome : 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)

Def get_stock :

- Signature : Ingrédient Stock
- Axiome : Ingrédient = Sucre, ou Ingrédient = Lait, ou Ingrédient = Thé, ou Ingrédient = Chocolat, ou Ingrédient = Café.

Stock ∈ Max_Stock

Def get_all_stock :

- Signature : Ingrédients × Stock Stock2
- Axiome : 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

Def remplir_stock :

- Signature : Ingrédient × Max_Stock Stock
- Axiome : Stock(Ingrédient) = Max_stock

Def remplir_tout_stock :

- Signature : Ingrédients × Max_Stock Stocks
- Axiome : Stocks[i] = Max_Stock[i]

Def ajouter_stock :

- Signature : Stock ×Max_Stock × Quantité Nouveau_Stock
- Axiome : Si Stock + quantité ∈ Max_Stock, alors Nouveau_Stock = Stock + Quantité,

Sinon Nouveau_Stock = Stock

Def historique :

- Signature :
- 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 : $O(n^3)$