

# Rapport Mega Machine à Caoua

Paul Ecoffet

Mathieu Seurin

Vendredi 28 Novembre 2014

Nous allons ici détailler les fonctions utilisées dans notre programme (signature et axiomes) ainsi que faire l'analyse de leur complexité en notation **O**. Après chaque méthode, nous indiquerons le fichier correspondant aux tests que nous avons effectués, ainsi que les noms des tests.

## Fonctions de Machine, mode fonctionnement

order :

1. Signature :  $(\text{Monnaie}, \text{Commande}) \Rightarrow (\text{Boisson} \cup \emptyset \times \text{Monnaie}) \cup \text{Exception}$
2. Axiomes :
  - $\forall \text{commande} \in \text{Drink}$ , Drink l'ensemble de tous les drinks possibles tel que  $\forall \text{type} \in \text{Commande.stock}$ ,  $\text{commande.stock}[\text{type}] \leq \text{Machine.Stock}[\text{type}]$   
 $\forall \text{monnaie} \in \text{Coins}$  tel que  $\text{monnaie.compute\_surplus}(\text{Machine.MaxCashInput}) \neq \text{Error}$   
et  $\text{monnaie.value} > \text{commande.price}$   $\text{order}(\text{commande}, \text{monnaie}) \Rightarrow \text{Drink}(\text{commande}), \text{change}$
  - $\forall \text{commande} \notin \text{Drink}$ , Drink l'ensemble de tous les drinks possibles  
 $\forall \text{monnaie} \in \text{Coins}$   
 $\text{order}(\text{commande}, \text{monnaie}) \Rightarrow \text{InvalidOrderException}$
  - $\forall \text{Commande} \in \text{Drink}$ , Drink l'ensemble de tous les drinks possibles  
 $\forall \text{Monnaie} \in \text{Coins}$   
tel que  $\text{Monnaie.compute\_surplus}(\text{Machine.MaxCashInput}) = \text{NoChangePossibleException}$   
 $\text{order}(\text{commande}, \text{monnaie}) \Rightarrow \text{None}, \text{monnaie}$
  - $\forall \text{commande} \in \text{Drink}$ , Drink l'ensemble de tous les drinks possibles tel que  $\forall \text{type} \in \text{Commande.stock}$ ,  $\text{commande.stock}[\text{type}] > \text{Machine.Stock}[\text{type}]$   $\text{order}(\text{commande}, \text{monnaie}) \Rightarrow \text{NotEnoughStockException}$

3. Complexité :  $\text{Max}(\forall \text{ functions} \in \text{order: Complexité}(\text{functions})) = O(2^n)$   
complexité de `Coins.compute_surplus`, avec  $n$  le nombre de pièces dans  
coins.
4. Test : `test_machine.py`
  - `test_order_simple()`
  - `test_order_complex()`
  - `test_order_fail_not_enough_cash()`
  - `test_order_fail_not_drink()`
  - `test_order_fail_no_stock()`
  - `test_order_cant_get_maxcash()`

## Fonctions de Machine, mode Maintenance

**reset :**

1. Signature :  $\emptyset \Rightarrow \emptyset$
2. Axiomes :

$$\text{machine.reset()} \Rightarrow \begin{cases} \text{machine.__stocks}[\text{type}] = 0 & \forall \text{type} \in \text{machine.StocksType} \\ \text{machine.__coins}[\text{type}] = 0 & \forall \text{type} \in \text{machine.CoinsType} \\ \text{machine.__cash} = 0 \end{cases}$$

**edit\_prices :**

1. Signature :  $(\text{dictionnaire\_prix}) \Rightarrow \emptyset \cup \text{TypeError}$
2. Axiomes :
  - $\forall \text{ stock} \in \{\text{'thé'}, \text{'café'}, \text{'lait'}, \text{'chocolat'}\},$   
 $\forall \text{ prix} \geq 0$   
 $\text{edit\_prices}(\text{stock}=\text{prix}) \Rightarrow \text{machine.__stock\_prices}[\text{stock}] = \text{prix}$
  - Si  $\text{stock} = \text{'sucre'} \forall 0 \leq \text{prix}_i \leq \text{prix}_{i+1}, i \in [0, 3]$   $\text{edit\_prices}(\text{sucre}=[\text{prix}_i])$   
 $\text{machine.__stock\_prices}[\text{stock}] = [(\text{prix}_i)] \forall i \in [0, 3]$
3. Complexité :  $O(n)$  avec  $n$  le nombre de produits payant
4. Test : `test_machine.py`
  - `test_edit_prices`

### **edit\_stocks :**

1. Signature : (dictionnaire\_stocks)  $\Rightarrow \emptyset$
2. Axiomes :
  - $\forall \text{ stock} \in \{\text{'thé'}, \text{'café'}, \text{'lait'}, \text{'chocolat'}, \text{'sucre'}\},$   
 $\forall \text{ machine.quantite}[\text{stock}] < \text{quantite} \leq \text{machine.quantite\_max}[\text{stock}]$   
 $\text{machine.edit\_stock}(\text{stock}=\text{quantite}) \Rightarrow \text{machine.quantite}[\text{stock}] = \text{quantite}$
  - $\forall \text{ stock} \in \{\text{'thé'}, \text{'café'}, \text{'lait'}, \text{'chocolat'}, \text{'sucre'}\},$   
 $\forall \text{ quantite} \leq \text{machine.quantite}[\text{stock}] \text{ ou } \text{quantite} > \text{machine.quantite\_max}[\text{stock}]$   
 $\text{machine.edit\_stock}(\text{stock}, \text{quantite}) \Rightarrow \text{machine.quantite}[\text{stock}] = \text{machine.quantite}[\text{stock}]$
3. Complexité :  $O(n)$  avec  $n$  le nombre de stocks différents
4. Test : *test\_machine.py*
  - test\_edit\_stocks

### **refill\_stocks :**

1. Signature :  $\emptyset \Rightarrow \emptyset$
2. Axiomes :
  - $\forall \text{ stock} \in \text{Machine.StocksType},$   
 $\text{machine.refill\_stock}() \Rightarrow \text{machine.quantite}[\text{stock}] = \text{machine.quantite\_max}[\text{stock}]$
3. Complexité :  $O(n)$  avec  $n$  le nombre de stocks différents
4. Test : *test\_machine.py*
  - test\_edit\_prices

### **edit\_coins :**

1. Signature : coins  $\Rightarrow \emptyset$
2. Axiomes :
  - $\forall \text{ pieces} \in \text{machine.CoinsType} \text{ et } \text{pieces} \in \text{coins},$   
 $\forall 0 \leq \text{coins}[\text{pieces}] \leq \text{machine.\_max\_coins}[\text{pieces}]$   
 $\text{edit\_coins}[\text{coins}] \Rightarrow \text{machine.\_coins}[\text{pieces}] = \text{coins}[\text{pieces}], \forall \text{pieces}$
3. Complexité :  $O(n)$  avec  $n$  le nombre de types de pièces différentes gérées par la machine
4. Test : *test\_machine.py*
  - test\_edit\_prices

### refill\_coins :

1. Signature :  $\emptyset \Rightarrow \emptyset$
2. Axiomes :
  - $\text{machine.refill\_coins}() \Rightarrow \forall \text{valeur} \in \text{Machine.CoinsType}, \text{machine.coins}[\text{valeur}] = \text{machine.max\_coins}[\text{valeur}]$
3. Complexité :  $O(n)$  avec  $n$  le nombre de types de pièces différents
4. Test : *test\_machine.py*
  - test\_edit\_prices

### remove\_stocks :

1. Signature :  $\text{stock\_dict} \Rightarrow \emptyset$
2. Axiomes :
  - $\forall A = (\text{stock\_type}, \text{value})_i, i \in \mathbb{N}, \text{stock\_type}_i \in \text{Machine.StocksType}$   
 $\text{machine.remove}(A) \Rightarrow \forall \text{stock\_type}, \text{value} \in A, \text{machine.\_stocks}[\text{stock\_type}] = \text{machine.\_stocks}[\text{stock\_type}] - \text{value}$
3. Complexité :  $O(n)$  avec  $n$  le nombre de types de stocks gérés par la machine
4. Test : *test\_machine.py*
  - test\_remove\_stocks

### add\_to\_cash :

1. Signature:  $\text{Coins} \Rightarrow \emptyset$
2. Axiomes :
  - $\forall \text{coins} \in \text{Coins},$   
 $\text{machine.add\_to\_cash}(\text{coins}) \Rightarrow \forall \text{type}, \text{quantite} \in \text{coins},$   
 $\text{machine.cash}[\text{type}] = \text{machine.cash}[\text{type}] + \text{quantite}$
3. Complexité :  $O(1)$

## Fonctions de Coins

Coins hérite de `collections.Counter`.

## compute\_surplus

1. Signature:  $\text{value} \Rightarrow \text{change} \in \text{Coins} \cup \text{NoChangePossibleException}$
2. Axiomes:

- $\forall \text{ coins in Coins, coins.value} \geq \text{value},$

$$\text{coins.compute\_surplus}(x) \Rightarrow \begin{cases} (\text{coins} - \text{change}).\text{value} = \text{value} & \text{si possible} \\ \text{NoChangePossibleException} & \text{si impossible} \end{cases}$$

- $\forall \text{ coins in Coins, coins.value} \leq \text{value},$   
 $\text{coins.compute\_surplus}(x) \Rightarrow \text{NoChangePossibleException}$

3. Complexité:  $O(2^n)$ ,  $n$  le nombre de pièces dans coins.
4. Tests: *test\_coins.py*
  - test\_compute\_surplus

## compute\_change

1. Signature :  $\text{change\_value} \Rightarrow \text{Coins}$
2. Axiomes :

- $\forall \text{ coins in Coins, coins.value} \geq \text{value},$

$$\text{coins.compute\_change}(x) \Rightarrow \begin{cases} (\text{change}).\text{value} = \text{change\_value} & \text{si possible} \\ \text{NoChangePossibleException} & \text{si impossible par division} \end{cases}$$

- $\forall \text{ change, change} > \text{coins.value},$   
 $\text{coins.self\_compute}(\text{change}) \Rightarrow \text{NoChangePossibleException}$

3. Complexité:  $O(n)$  avec  $n$  le nombre de types de pièces dans coins.
4. Tests: *test\_coins.py*
  - test\_compute\_change
  - test\_compute\_change\_not\_enough\_cash
  - test\_compute\_change\_impossible

## value

1. Signature :  $\emptyset \Rightarrow \mathbb{N}$
2. Axiomes :

- $\forall \text{ coins in Coins, coins} = (\text{valeur}, \text{quantite})_{n \in \mathbb{N}},$

$$\text{coins.value} = \sum_{i=1}^n \text{valeur}_i \times \text{quantite}_i$$

3. Complexité:  $O(n)$  avec  $n$  le nombre de types de pièces dans coins.
4. Tests: *test\_coins.py*
  - test\_coins\_value

## Méthodes de Drink

### price

1. Signature :  $\emptyset \Rightarrow \mathbb{N}$
2. Axiomes :
  - $\forall \text{stock}_i \in \text{drink.stocks}, \forall \text{price}_i \in \text{drink.stock\_prices}, i \in \{1, \dots, n\}$

$$\text{drink.price} = \sum_{i=1}^n \text{stock}_i \times \text{price}_i$$

3. Complexité :  $O(n)$  avec  $n$  le nombre de types de stock différents dans drink.
4. Tests: *test\_drink.py*
  - test\_drink\_price

### has\_beverage

1. Signature:  $\emptyset \Rightarrow T, F$
2. Axiomes :
  - $\forall \text{drink avec } \exists x \in \{\text{'chocolate'}, \text{'tea'}, \text{'coffee'}\}, x \in \text{drink.stock}, \text{drink.has\_beverage} \Rightarrow T$
  - $\forall \text{drink avec } \forall x \in \{\text{'chocolate'}, \text{'tea'}, \text{'coffee'}\}, x \notin \text{drink.stock}, \text{drink.has\_beverage} \Rightarrow F$
3. Complexité :  $O(1)$
4. Tests: *test\_drink.py*
  - test\_drink\_has\_beverage

## Méthodes des Logs

### Log.message

1. Signature:  $\emptyset \Rightarrow \emptyset$

2. Axiomes:  $\forall \text{ log\_message}$   
 $\text{log.message} = \text{log\_message}$
3. Complexité :  $O(1)$
4. Tests: *test\_logs.py*
  - test\_log

## StockLog.message

1. Signature:  $\emptyset \Rightarrow \emptyset$
2. Axiomes:
  - $\forall (\text{stock}, \text{quantity})_n \in \text{prev\_stock},$   
 $\forall (\text{n\_stocks}, \text{n\_quantite})_n \in \text{log.cur\_stock} \text{ log.message} \Rightarrow$   

$$\sum_{i=1}^n \text{"}\{p\_stock_i\} : \{p\_quantite_i\} \rightarrow \{n\_quantité_i\} \{n\_quantité-p\_quantite\}\text{"}$$
3. Complexité :  $O(1)$
4. Tests: *test\_logs.py*
  - test\_stock\_log\_no\_changes
  - test\_stock\_log\_message
  - test\_stock\_log\_str

## OrderLog.message

1. Signature:  $\emptyset \Rightarrow \emptyset$
2. Axiomes:
  - $\forall \text{ commande} \in \text{Drink}, \text{Drink l'ensemble des Drinks possible}$   
 $\forall \text{ monnaie} \in \text{Coins}$   
 $\text{Orderlog.message} \Rightarrow \text{"}\{\text{commande}\} \text{ which cost } \{\text{commande.price}\}$   
 $\text{the customer gave } \{\text{monnaie.value}\}\text{"}$
3. Complexité :  $O(1)$
4. Tests: *test\_logs.py*
  - test\_cash\_log\_message

## EndOrderLog.message

1. Signature:  $\emptyset \Rightarrow \emptyset$
2. Axiomes:
  - $\text{EndOrderLog.message}() \Rightarrow \text{"That's all folks"}$

3. Complexité :  $O(1)$
4. Tests: *test\_logs.py*
  - test\_end\_order\_log\_message

## CoinsLog

CoinsLog est un alias de StockLog avec un intitulé différent.

1. Test: *test\_logs.py*
  - test\_coins\_log\_message

## CashLog.message

1. Signature:  $\emptyset \Rightarrow \emptyset$
2. Axiomes:
  - $\forall \text{monnaie}_i \in \text{Coins } \text{CashLog.message} \Rightarrow$   
 $\text{“monnaie}_i \rightarrow \text{monnaie}_{i-1}(\text{monnaie}_i - \text{monnaie}_{i-1})\text{”}$
3. Complexité :  $O(1)$
4. Tests: *test\_logs.py*
  - test\_cash\_log\_message