Guide : Création d'un fichier json pour le chargement d'arbre

1 * Créer un arbre vide :

```
{
    "Parameters": [],
    "Compute rules": {
      "AND": [],
      "OR": [],
      "NOT": [],
      "XOR": [],
      "SOR": []
    },
    "Optimization rules": [],
    "Nodes": []
}
```

(Ici rien n'est rempli)

2 * Dans un premier temps : spécifier les noms des paramètres entre les crochets à droite de « parameters » en les séparant par une virgule. (Il n'y a pas de limite sur le nombre de paramètres)

Exemple :

```
"Parameters": [
  "Time",
  "Money",
  "Success probability"
],
```

Ici, on entre les paramètres notés « Time », « Money », et « sucess probability »

3 * Ensuite, en dessous de « COMPUTE RULES », pour chaque porte logique on entre les valeurs des règles de calculs suivante (une pour chaque paramètre séparé par une virgule) :

+ pour la somme
* pour le produit
m pour le minimum
M pour le maximum

L'ordre des règles de calculs suivent l'ordre dans lequel ont été entrés les noms des paramètres ci-dessus.

Exemple :

```
"Compute rules": {
  "AND": [
    "+",
    "+",
    "*"
  ],
  "OR": [
    "M",
    "+",
    "+"
  ],
```

Concernant la porte « AND » :
Ici on entre la règle + pour le paramètre 'time'
+ pour le paramètre 'Money'
Et * pour le paramètre 'sucess probability'

On répète l'opération pour l'ensemble des portes logiques restantes

4 * La spécification des paramètres (la partie « optimization rules ») :

On entre m (resp. M) pour spécifier si on cherche le minimum (resp Maximum) séparé par des virgules et toujours suivant l'ordre des paramètres entrés en 1ère étape

Exemple :

```
"Optimization rules": [
  "m",
  "m",
  "M"
],
```

Ici, on spécifie la minimisation du paramètre ‘time’ et ‘money’
Et la maximisation du paramètre ‘success probability’

5 * spécification de la liste des nœuds qui compose l’arbre :

La partie ‘Node’ ouvre une liste de nœud ou chacun d’eux est composé d’information entre accolades :

```
{
  "ID": 1,
  "Name": "node1",
  "Parents": [],
  "Children": [],
  "Gate": "",
  "Values": [1,1,1]
}
```

- un id (ici 1)
- un nom (ici ‘node1’)
- une liste d’Id de parents (ici la liste est vide)
- une liste d’Id de fils (ici la liste est vide également)
- une porte logique suivant les valeurs suivantes : « » (comme ici, pour les feuilles) ou bien ‘AND’ ‘OR’ ‘XOR’ etc.
- enfin une liste de valeur où chaque valeur est associée au paramètre déclaré plus haut. L’ordre dans lequel placer les valeurs est le même que l’ordre de déclaration des paramètres.

<u>On peut en déclarer plusieurs comme suit :</u>

```
"Nodes": [
  {
    "ID": 1,
    "Name": "node1",
    "Parents": [],
    "Children": [2],
    "Gate": "AND",
    "Values": []
  },
  {
    "ID": 2,
    "Name": "node2",
    "Parents": [1],
    "Children": [],
    "Gate": "",
    "Values": [2,2,"x"]
  },
```

Chaque nœud doit avoir un id différent et une cohérence parent-enfant.
De plus, il est possible de mettre un caractère (tel que ‘x’ par exemple) dans les valeurs sans que cela ne pose problème.

D’autres exemples d’arbres plus complets sont disponibles dans le jeu de test fourni

Mode d'emploi de l'application :

- **1er contact avec le programme :**

*** Lors du démarrage :**

Au lancement, s'affiche l'interface suivante :

```
Create a new tree: type N or n
Load a JSON file: type L or l

Choice: n
```

S'offre alors deux options :

- charger un arbre provenant d'un fichier json : entrer L ou l

```
Create a new tree: type N or n
Load a JSON file: type L or l

Choice: l

Enter the JSON file name: <name_file>.json
```

On entre alors le nom du fichier (sans oublier l'extension .json)

- créer un nouvel arbre via l'interface terminale que l'on verra ultérieurement : entrer N ou n

```
PARAMETERS
Type Y or y to choose a parameter

********************************

Time: y
Money: y
Damages: n
Success probability: n
Custom parameters: n
```

Est proposé une liste de paramètres à intégrer à l'arbre
Entrer 'y' pour garder le paramètre et 'n'pour ne pas les retenir

(Time, Money, Damages, Success Probability sont des paramètres proposés par défaut, il est possible de créer un paramètre personnalisé en entrant 'y' dans « custom parameters »)

*** Pour chacun des paramètres :**

Face à cette interface :

```
RULES

Choose the compute rule for each logical gate and parameter
Choices among the sum (type +), the product (type *), the maximum (type M) and the minimum (type m)

********************************************************************************************************

Gate AND
Valid rules: sum and product

Parameter: Time
Choice: +

Parameter: Money
Choice: +
```

Configuration des règles de calcul paramètres : on procède Par porte, PUIS par paramètres.

Pour chacune des portes (ici AND, OR, K/N), sont affichées sur la ligne d'en dessous les règles « valid rules », elles sont communes à chaque paramètre. Ainsi pour chaque porte, entrer successivement les symboles associés aux règles de calcul (* pour le produit, m pour le minimum…)

```
*****************************************************************************************************

Gate AND
Valid rules: sum and product

Parameter: Time
Choice: +

Parameter: Money
Choice: +

-------

Gate OR
Valid rules: maximum and minimum

Parameter: Time
Choice: m

Parameter: Money
Choice: m

-------

Gate k/n
Valid rules: sum and product

Parameter: Time
Choice: +

Parameter: Money
Choice: +
```

Exemple : Pour la porte AND, les uniques règles valides sont la somme (+) et le produit (*), on sélectionne + pour le paramètre ‘time’ et + pour le paramètre ‘money’

- **Création d’un arbre :**

```
CREATION OF THE NODES AND THE VALUES OF THE TREE
================================================

Root node

Name: node1
Number of children: 2
Logical gate: AND

*

Node 2 (parent: 1 (node1))

Name: node2
Number of children: 0

Parameter: Time
Enter a value: 7

Parameter: Money
Enter a value: 2*m

*

Node 3 (parent: 1 (node1))

Name: node3
Number of children: 0

Parameter: Time
Enter a value: t

Parameter: Money
Enter a value: 3
```

- On commence par entrer les informations nécessaires du nœud racine :
Name : le nom
Number of children: Le nombre d’enfants
(Le nombre de parent est mis à 0 par défaut)
Logical gate : la porte logique (‘AND’, ‘OR’…)

- On réitère pour les autres nœuds :
Name : le nom
Number of children: Le nombre d’enfants
(Le nombre de parent est mis à 0 par défaut)
Et uniquement si le nombre de fils est différent de 0 :
On renseigne ‘Logical gate’ : la porte logique (‘AND’, ‘OR’…)

- Le programme de création d’arbre ne s’arrête que lorsque tous les nœuds (renseigné lors de l’enregistrement des fils) sont remplis.

- Affichage d'un arbre dans l'interface :

```
PARAMETERS

Time; Money

*

NODES

Node 1 (root)
  Name: node1
  Children's id.: 2 3
  Logical gate: AND
    Time: sum
    Money: sum

Node 2 (leaf)
  Name: node2
  Parent's id.: 1
  Values: 7; 2*m

Node 3 (leaf)
  Name: node3
  Parent's id.: 1
  Values: t; 3

*

SCENARIO

Time: minimum
Money: minimum
```

Affichage des paramètres (les noms)

Affichage des nœuds (NODES) :
'root' signifie que le nœud est la racine de l'arbre : n'a pas de parent
Pour chaque nœud, on affiche le Nom (ici 'node1')
La porte logique (ici 'AND')
Les règles de calculs pour chacun des paramètres
(Ici sum pour le paramètre 'Time' et sum pour 'Money')

'leaf' signifie que le nœud est une feuille de l'arbre : n'a pas de fils
Pour chaque feuille, sont listées les valeurs pour chaque paramètre du nœud

(Exemple : pour le nœud 2, la valeur du paramètre 'time' vaut 7 et pour le paramètre 'Money' la valeur est de 2*m (m est une variable))

Spécification des scénario (en dessous de l'inscription « SCENARIO »).
Ici, les deux paramètres seront minimisés (« minimum »)

- Actions principales du menu :

```
Quit: type Q or q

~

Action

Parameter: enter P or p
Node: type N or n
Data: type D or d

~

Choice:
```

Pour quitter : taper q ou Q

Pour les actions propres aux paramètres : taper P ou p
Pour les actions propres aux nœuds : taper N ou n
Pour les actions propres aux paramètres : taper D ou d

- Paramètres :

3 actions sont disponibles, on peut :
Ajouter (entrer a ou A), Supprimer (entrer D ou d), Modifier (entrer M ou m) un paramètre.

```
Parameter
  Adding: type A or a
  Deleting: type D or d
  Modifying: type M or m

Action:
```

Il suffit ensuite de lire les instructions pas à pas (pour l'ajout d'un paramètre, les règles de calculs ainsi que les valeurs pour chaque feuille sont demandées, pour la suppression d'un paramètre les valeurs aux feuilles de l'arbre se suppriment automatiquement).

- **Node**

2 actions sont disponibles : on peut ajouter (taper A ou a) ou supprimer un nœud (taper D ou d)

```
Node
  Adding: type A or a
  Deleting: type D or d

Action :
```

Le reste se fait en lisant les instructions sur le terminal (deux modes de suppression sont disponibles, soit on supprime tout le sous arbre pour lequel le nœud est racine, soit on ne supprime uniquement que le nœud (et le sous arbre est recollé))

- **Data :**

Le sous-menu data affiche les chemin et scénarios possibles de l'arbre :
Affiche les chemins sous forme de liste imbriqué
Affiche les expressions symboliques associée

```
Dataframe

                Chemin Time Money
0  [1, [[2], [3], [4]]]    4    26
1  [1, [[2], [3], [5]]]   11    14
2  [1, [[2], [4], [5]]]    8    30
3  [1, [[3], [4], [5]]]   11    20
```

L'interface demande de sélectionner un type de tri (croissant, décroissant ou min_max).
Il suffit d'écrire le type de test qu'on veut effectuer. Si l'utilisateur sélectionne un tri croissant ou décroissant, il devra ensuite choisir sur quelle variable il veut effectuer ce tri puis dire s'il veut la maximiser ou la minimiser.
Si l'utilisateur sélectionne un tri min_max, il va commencer par dire s'il veut pondérer les variables qu'il va choisir par la suite.
Ensuite il va sélectionner une première variable, puis dire s'il veut la minimiser ou la maximiser et pour finir ajouter un poids (uniquement s'il a répondu "oui", à la question pour avoir une pondération).
Il va ensuite pouvoir sélectionner une autre variable et reproduire le même schéma pour rentrer les données. Une fois qu'il aura ajouté toutes les variables qu'il veut utiliser pour son tri, il répondra "stop" à la question où le programme demande quelle variable il veut ajouter.

Il est également possible à la suite de toutes ces procédures, d'évaluer des variables passées en valeur de paramètres pour certain nœud.
L'affichage du dataframe se présente comme suit :
D'abord le chemin est affiché

```
1: [1, [[2], [3], [4]]]
```

Ensuite L'expression symbolique pour chaque paramètre est affichée (ici l'exemple avec le paramètre 'time')

```
Time: (1 + t + 2) = t + 3
```

Par la suite il est possible d'évaluer la variable de l'expression symbolique et d'avoir un résultat sans inconnue.

Exemple plus complet :

```
4 chemins

Paramètres
Time; Money

~

1: [1, [[2], [3], [4]]]
  Time: (1 + t + 2) = t + 3
    t := 1
    => t + 3 = 4
  Money: (8 + 3 + 15) = 26

2: [1, [[2], [3], [5]]]
  Time: (1 + t + 5) = t + 6
    t := 5
    => t + 6 = 11
  Money: (8 + 3 + m) = m + 11
    m := 3
    => m + 11 = 14

3: [1, [[2], [4], [5]]]
  Time: (1 + 2 + 5) = 8
  Money: (8 + 15 + m) = m + 23
    m := 7
    => m + 23 = 30

4: [1, [[3], [4], [5]]]
  Time: (t + 2 + 5) = t + 7
    t := 4
    => t + 7 = 11
  Money: (3 + 15 + m) = m + 18
    m := 2
    => m + 18 = 20
```