Interpréteur d'expressions arithmétiques

Consignes – Ce devoir doit être réalisé individuellement. Chaque question vous demande de construire une machine de Turing. On utilisera le simulateur en ligne disponible sur `https://turingmachinesimulator.com/`. Vos réponses devront être interprétable par ce simulateur : en cas d'erreur de compilation votre réponse ne sera pas corrigée. Toutes les machines à construire opèreront donc sur un ou plusieurs rubans bi-infinis, seront déterministes, et commenceront avec leur entrée sur le premier ruban, et la tête de lecture sur le symbole le plus à gauche de l'entrée.

Structure du rendu – Vos réponses devront être présentées sous forme de fichers texte, interprétable par le simulateur. Vous rassemblerez les fichiers correspondant à vos réponses à chaque question dans une archive (`.zip` ou `.tar.gz`), nommée `nom.prenom.dm-cc`, qui devra me parvenir par email à l'adresse suivante : `paul.brunet@u-pec.fr`. Par exemple, voici l'arborescence d'un rendu possible :

```
brunet.paul.dm-cc.zip
├── ex1-q1.txt
├── ex1-q2.txt
├── ex2-q1.txt
├── ex2-q2.txt
└── ex3-q1.txt
```

Conseils – Il est recommandé d'utiliser des machines à plusieurs rubans pour certaines questions. Comme on va combiner les différentes machines que l'on construit, il est judicieux de choisir des noms d'états différents pour chaque machine. Pensez bien à *tester* vos réponses sur *plusieurs* mots d'entrée !

On veut construire une machine de Turing qui interprète des expressions arithmétiques en unaire vers des entiers binaires. Plus précisément, on veut une machine qui, sur une chaîne de caractères en entrée $(1111 + (11 * 111))$, qui encode le calcul $(4 + (2 \times 3))$, produit la chaîne de caractères 1010, c'est à dire le code binaire de l'entier 10. On va pour cela procéder en plusieurs étapes :

1. Implémenter l'addition et la multiplication d'entiers en unaire.
2. Implémenter la conversion d'entiers de unaire à binaire.
3. Construire un parseur, qui met une expression arithmétique infixe sous forme suffixe, aussi appelée « notation polonaise inverse ».
4. Assembler ces éléments pour obtenir l'interpréteur souhaité.

Il est recommandé de se documenter sur la notation polonaise inverse, par exemple en consultant la page suivante :

`https://fr.wikipedia.org/wiki/Notation_polonaise_inverse`.

Exercice 1. Opérations arithmétiques . . . . . 3 points

Question 1. (1 point) Construire une machine « add » telle que :

— L'alphabet d'entrée d'« add » est $\{0, 1\}$.

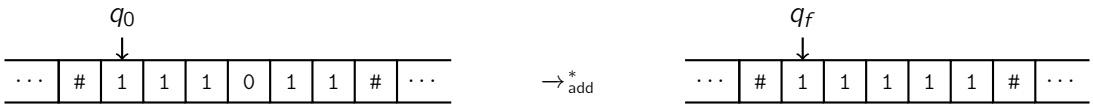
— Sur une entrée $w = 1^n01^m$, la machine accepte, et s'arrête avec :

  — le ruban contenant la sortie 1^{n+m} ;

  — si le ruban est non-vide ($n + m \neq 0$), la tête de lecture sur la première case non-vide à partir de la gauche.

— La machine rejette les mots mal formattés.

Par exemple, si on note q_0 et q_f respectivement les états initiaux et finaux, on devrait avoir :



Question 2. (2 points) En utilisant la machine précédente, construire « mult » telle que :

— L'alphabet d'entrée de « mult » est $\{0, 1\}$.

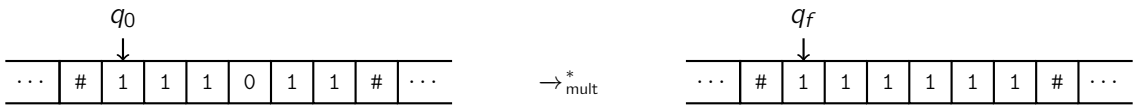
— Sur une entrée $w = 1^n01^m$, la machine accepte, et s'arrête avec :

  — le ruban contenant la sortie $1^{n\times m}$;

  — la tête de lecture sur la première case non-vide (si le ruban est non-vide).

— La machine rejette les mots mal formattés.

Par exemple, si on note q_0 et q_f respectivement les états initiaux et finaux, on devrait avoir :



Exercice 2. Conversion unaire vers binaire . . . . . 3 points

Pour un entier $n \in \mathbb{N}$, on note $[n]_{code}$ sa représentation binaire, avec le bit de poids fort à gauche. Par exemple :

$$[0]_{code} = 0 \qquad [3]_{code} = 11$$
$$[1]_{code} = 1 \qquad [4]_{code} = 100$$
$$[2]_{code} = 10 \qquad [5]_{code} = 101.$$

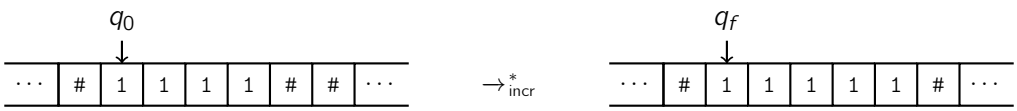
Question 1. (1 point) Construire une machine « incr » telle que :

— L'alphabet d'entrée d'« incr » est $\{0, 1\}$.

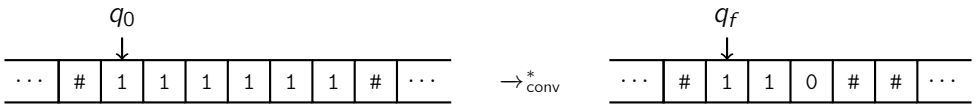
— Sur une entrée $w = [n]_{code}$, la machine accepte, et s'arrête avec : le ruban contenant la sortie $[n + 1]_{code}$ et la tête de lecture sur la première case non-vide.

— La machine rejette les mots mal formattés, c'est à dire qu'elle s'arrête sur un état non-final pour tous les mots $\left\{w \in \{0, 1\}^\star \;\middle|\; \forall n \in \mathbb{N},\ [n]_{code} \neq w\right\}$.

Par exemple, si on note q_0 et q_f respectivement les états initiaux et finaux, on devrait avoir :



Question 2. (2 points) En utilisant la question précédente, implémenter une machine réalisant la conversion unaire binaire, c'est à dire une machine « conv » telle que :



Exercice 3. Parseur ..... 7 points

Une expression arithmétique peut se représenter de manière arborescente. Par exemple, l'expression $4 + (2 \times 3)$ peut se représenter comme le résultat de l'opération $+$ appliquée à 4 et au résultat d'une seconde opération $\times$ appliquée à 2 et à 3. 4 et $\times(2, 3)$ sont dits « opérandes » de l'expression qui a $+$ comme opérateur. Selon ce principe, une autre notation pour cette expression arithmétique est $+(4, \times(2, 3))$, aussi appelée notation « polonaise » et notamment utilisée sur les calculatrice HP. Il est à noter qu'il devient ainsi possible de se passer tout à fait de parenthèses : il n'y a aucune ambiguïté lorsqu'on utilise la notation $+4 \times 23$. La notation qui nous intéresse ici est la notation polonaise *inverse* : pour chaque opération $n + m$, on écrit $n\,m+$. L'expression $4 + (2 \times 3)$ devient donc $4\,2\,3 \times +$.

Dans notre cas, on va vouloir prendre en entrée des chaînes de caractères sur l'alphabet $\{1, +, *, (,)\}$ où :

— 1 sert à représenter les entiers en unaire ;

— $+$ et $*$ sont utilisé pour représenter de manière infixe les opérations arithmétiques d'addition et de multiplication ;

— les parenthèses doivent être utilisées à chaque application d'opération.

Plus précisément, une entrée bien formattée doit être conforme à la grammaire ci-dessous :

$$S \to 1^n \mid (S + S) \mid (S * S)$$

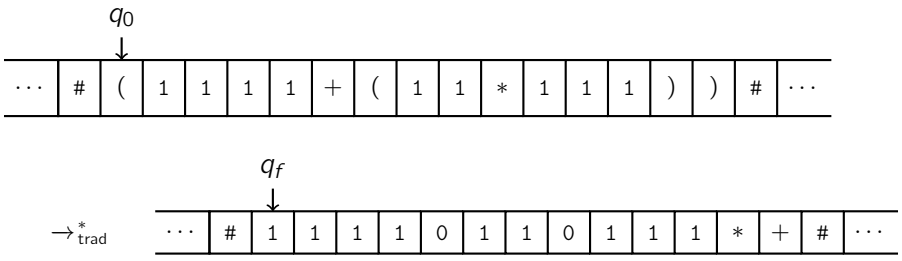
Par exemple $(1111 + (11 * 111))$ est bien formattée, tout comme 1111. En revanche 1(11) ne l'est pas, pas plus que $(1 + 11 * 1111)$, $11 + 1$, ou $(1 + 11 + 111)$.

Question 1. (3 points) Construire une machine qui décide le langage des expressions bien formattées.

On veut traduire ces expressions en notation polonaise inverse (NPI). Ceci implique d'effacer les parenthèses, de déplacer les symboles $+$ et $*$, et d'insérer des séparateurs (on va utiliser 0) entre les arguments. Ainsi, l'expression $(1111 + (11 * 111))$ deviendra $11110110111 * +$ On pourra utiliser le pseudo-algorithme suivant, qui utilise une pile (last in first out) d'opérateurs :

1. On lit l'entrée, en effaçant les parenthèses ouvrantes, et en recopiant les entiers.
2. Lorsequ'on lit un opérateur $+$ où $*$, on le remplace par 0 et on l'ajoute à la pile.
3. Lorsequ'on lit une parenthèse fermante, on retire un opérateur du sommet de la pile, et on l'écrit à la place de cette parenthèse.

Question 2. (4 points) En utilisant l'algorithme ci-dessus, construire une machine qui rejette les expressions mal formattées, et convertit les autres en NPI.



Exercice 4. Interpréteur ..... 7 points

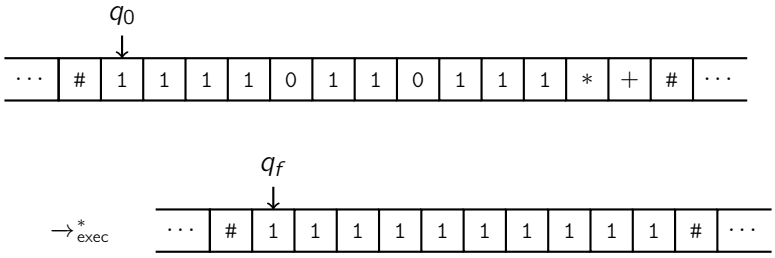
Une fois l'expression en NPI, l'interpréter (i.e. calculer le résultat) est aisé [1]. L'algorithme maintient une pile de nombres :

— Lorsequ'on lit un nombre, on l'ajoute à la pile.

— Lorsequ'on lit un opérateur, on retire les deux premiers nombres de la pile, on leur applique l'opération, et on place le résultat au sommet de la pile.

Après avoir fini de lire l'entrée, on doit avoir un seul nombre dans la pile : le résultat.

Question 1. (5 points) En utilisant les machines de l'exercice et l'algorithme ci-dessus, constuire une machine qui interprète les expressions en NPI. Par exemple, sur l'entrée

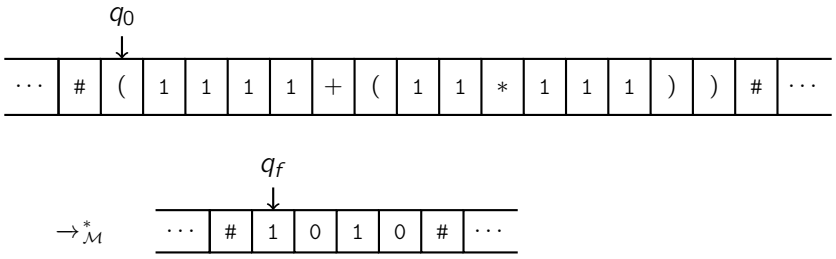


Question 2. (2 points) En combinant les différentes machines réalisées jusque là, construire une machine qui prend en entrée des expressions arithmétiques avec :

— les entiers en unaire ;

— les opérateurs en notation infixe.

Cette machine doit évaluer l'expression, et convertir le résultat en binaire. Autrement dit elle doit avoir sur notre exemple le comportement suivant :



1. Un exemple d'exécution de l'algorithme d'evaluation est présenté sur `https://fr.wikipedia.org/wiki/Notation_polonaise_inverse`.