# Compilation - N3, S5
# TP 5 : Génération de code MVaP

1. L'objectif de ce TP c'est la génération de code dans le langage MVaP d'expressions arithmétiques utilisant les fractions et d'expressions booléennes.
2. Votre ami pour la documentation de ANTLR :
   `https://github.com/antlr/antlr4/blob/master/doc/index.md`
3. Pour la liste des instructions de la machine MVaP et leur description, voir le fichier MVaP_cheat_sheet.

**Installation du simulateur de la machine MVaP.** Suivre les instructions du fichier installMVaP.txt pour installer le simulateur de la machine MVaP. Il y a également un rappel des lignes de commande pour utiliser MVaPAssembler pour assembler un programme MVaP (c'est la partie édition de lien et création exécutable) et pour utiliser le simulateur de la machine MVaP.

**Question 5.1** Un entier relatif est une suite de chiffres et il peut être signé (négatif ou positif), et un rationnel [1] est un nombre qui est le quotient de deux entiers relatifs, que l'on notera a/b où a est appelé numérateur et b est appelé dénominateur.

1. Ecrire un programme MVaP qui :
   (a) lit deux nombres rationnels r1 et r2 depuis l'entrée, *i.e.*, lit quatre entiers, interprété comme numérateur(r1), dénominateur(r1), numérateur(r2), dénominateur(r2) ;
   (b) affiche un rationnel r, *i.e.*, affiche numérateur(r) puis dénominateur(r) tel que r=r1+r2 ;
   (c) termine avec la pile vide.
2. Ecrire un programme MVaP qui :
   (a) lit un nombre rationnel r et un entier n, *i.e.*, lit trois entiers, interprété comme numérateur(r), dénominateur(r), et n ;
   (b) affiche un rationnel r', *i.e.*, affiche numérateur(r') puis dénominateur(r') tel que $r' = r^n$, *i.e.*, la puissance n de r ;
   (c) termine avec la pile vide.

**Question 5.2** On veut maintenant écrire un langage de programmation, appelé langage de Rationnel, et qui n'a qu'une seule instruction `Afficher(expr)`, où `expr` est une expression. Le langage de Rationnel manipulera 3 types d'expression : entières, rationnelles et booléennes. Chaque instruction doit être terminée par un point-virgule. Le compilateur prend en entrée une suite d'instructions et génère un autre programme équivalent dans le langage de MVaP. Le code généré par votre compilateur devra toujours terminer par la commande `HALT`. Le résultat de l'instruction `Afficher(expr)` est l'affichage de la valeur de `expr` qui est un

— entier si `expr` est une expression de type entier,
— un booléen si `expr` est une expression booléenne. Afficher un booléen revient à afficher 1 si le booléen vaut vrai, et 0 sinon,
— un rationnel si `expr` est une expression rationnelle. Dans ce cas, il faut afficher le numérateur et ensuite le dénominateur à la ligne suivante.

Proposez à l'aide d'ANTLR un compilateur pour le langage de Rationnel dont la machine cible est la machine MVaP. Ce qui suit est une description des expressions du langage de Rationnel.

1. Wikipedia : `https://fr.wikipedia.org/wiki/Nombre_rationnel`.

**Opérations sur les rationnels et les entiers.** Voici les opérations possibles sur les rationnels :
— Addition +, multiplication ∗, division : et soustraction − de nombres rationnels,
— L'obtention du numérateur num ou du dénominateur denum,
— Puissance ∗∗ (*e.g.*, 2/3∗∗4 donne 16/81) ; dans a/b∗∗n, n est toujours un entier.

Dans le langage de Rationnel, il n'y a aucune opération dans les entiers. Les entiers sont utilisés avec :
— / constructeur des rationnels (*e.g.*, 3/4),
— ou pour calculer la puissance d'un rationnel avec l'opérateur ∗∗.
Ainsi, si on écrit par exemple 3+4, 3 et 4 doivent être interprétés, respectivement, comme les rationnels 3/1 et 4/1. De même pour les opérations ∗, : et −.

**Expressions.** Le langage des expressions devra permettre :

1. les expressions bien parenthésées sur les rationnels avec les opérateurs standards :
   — addition : +,
   — multiplication ∗,
   — soustraction − (à la fois binaire et unaire) et
   — division :.
   Les priorités habituelles devront être respectées.
2. les expressions booléennes bien parenthésées avec les opérateurs standards : et, ou, non. Les priorités suivantes devront s'appliquer : non>et>ou.

   Le code MVaP généré pour les expressions booléennes doit être une évaluation paresseuse, *i.e.*, pour une expression <∗exprC1∗> et <∗exprC2∗> la machine ne doit pas évaluer le code généré pour <∗exprC2∗> si <∗exprC1∗> est évaluée à faux, et de même pour <∗exprC1∗> ou <∗exprC2∗> la machine ne doit pas évaluer le code généré pour <∗exprC2∗> si <∗exprC1∗> est évaluée à vrai.
3. les comparaisons strictes et larges binaires (*i.e.*, entre deux éléments) d'expressions rationnelles, avec les opérateurs standards :<, <=, >,>=,== et <> (différent).
4. l'opération d'entrée lire () : lecture d'un entier ou d'un rationnel ou d'un booléen, selon le contexte, *e.g.*, 3∗∗ lire () et 4/ lire () devront lire un entier, alors que 3 + lire () devra lire un nombre rationnel, c'est à dire lire deux entiers successivement. Pour lire un booléen, on va considérer tout entier strictement positif comme vrai, et 0 comme faux.

**Remarque.** Vous avez pu remarquer dans la question 5.1 que pour générer, par exemple, le code MVaP de l'addition de deux rationnels, vous aviez besoin d'espace mémoire intermédiaire. Comme dans le langage de Rationnel on manipule des expressions rationnelles et non pas seulement des valeurs rationnelles, une astuce pour avoir d'espace mémoire intermédiaire est que le compilateur en réserve un certain nombre - le strict nécessaire - comme premières instructions de tout programme.