INF7641 - Devoir 1

Jean Privat

Hiver 2025

Devoir à remettre le mardi 4 février avant 9h par courriel à privat.jean@uqam.ca (format PDF).

Il est suggéré de dessiner les automates et les arbres avec graphviz.

Le travail est individuel.

Expressions régulières et automates

Exercice 1

Soient l'alphabet {`a`, `b`} et les deux expressions régulières suivantes (écrites en syntaxe SableCC4) :

```
r1 = ('a'+ 'b'+)* 'a'+ ;
r2 = ('a' 'b')* ('b' 'a')* ;
```

Indiquez si les mots suivants appartiennent à `r1`, à `r2`, aux deux ou à aucun des deux :

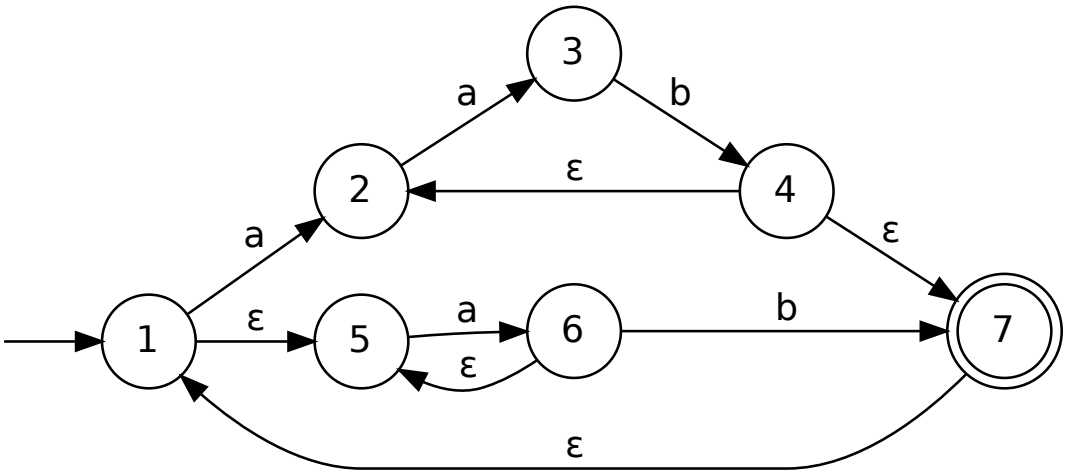
- `aaaa`
- `aabb`
- `abab`
- `bbbb`

Exercice 2

En utilisant la technique vue en cours, dessinez le NFA de l'expression régulière `'b' ('a' 'a'*|'b'*) 'a'`.

Exercice 3

En utilisant l'algorithme vu en cours, transformez en DFA le NFA suivant :



C'est beau la Pologne

La notation dite polonaise (ou préfixe) est une forme d'écriture d'expressions algébriques où les opérateurs sont placés avant les opérandes.

Si l'arité (le nombre d'opérandes) de chaque opérateur est fixe, alors cette représentation permet de se passer des parenthèses et des règles de priorité à apprendre par cœur.

Par exemple « `* + 2 3 4` » en notation polonaise correspond à « `(2 + 3) * 4` » en notation infixe classique.

C'est une notation dont un analyseur syntaxique se code très facilement à la main, même si la lisibilité des expressions est très discutable.

Exercice 4

Soit l'analyseur lexical (la classe `language_polonaise.Lexer`) généré par le fichier SableCC4 suivant :

```
Language polonaise;
Lexer
  int = ('0'..'9')+;
  add = '+';
  sub = '-';
  mul = '*';
  pow = '**';
  neg = 'neg';
  Token int, add, sub, mul, pow, neg;
  Ignored #9, #10, #13, #32; // tab, line break, space
Parser
  exp = ;
```

Développez une classe Java `Polonaise` qui calcule l'expression polonaise passée en premier argument et affiche le résultat.

Note : les opérateurs `+`, `-`, `*` et `**` sont d'arité 2 et `neg` (l'opposé d'un nombre) est d'arité 1.

```
$ java Polonaise "* + 2 3 4"
20
```

```
$ java Polonaise "* neg - 2 + 3 ** 4 5 6"
6150
```

Quelques directives et informations supplémentaires

- Important: n'utilisez pas la classe `Parser`, mais seulement `Lexer` (et `Node` et `Token`).
- Utilisez `long` pour représenter les nombres et ignorez les débordements (pour simplifier).
- Vous pouvez utiliser `switch/case`, par rapport à l'enum `Node.Type`.
- La récursivité, c'est la vie.
- `Long.parseLong()`, `StringReader` et `Math.pow()` peuvent aider.
- Affichez un message d'erreur si l'expression est terminée trop tôt ou s'il reste des jetons superflus.
- Note: Un seul fichier `Polonaise.java` de moins de 50 lignes suffit largement.

Exercice 5 (à faire après le 28 janvier)

Complétez la partie `Parser` du fichier SableCC4 pour écrire la grammaire des expressions polonaises avec le même langage que celui reconnu par la classe `Polonaise` (mêmes opérandes et opérateurs).

Exercice 6 (à faire après le 28 janvier)

Dessinez l'AST de « `* neg - 2 + 3 ** 4 5 6` ».

Exercice 7 (bonus)

Le langage des expression polonaises est-il régulier ? Donnez une preuve.