

Travail Ecrit

juin 2017

Nom et Prénom

Classe :

Barème

(1)	(2)	(3)	(4)	Total
12 pts	12 pts	12 pts	14 pts	50
9	11	3	10	33

4.5

b a b a a b a

1. Définir par un *automate fini* déterministe le langage suivant sur l'alphabet {a,b} : l'ensemble des mots qui contiennent la suite "ababaab". Exemple : "baababaabbb".

2. Pour chaque expression régulière, donner un mot de longueur minimale sur l'alphabet {a,b} qui **n'appartient pas** au langage (ou indiquer si c'est impossible) :

- a) $b^* (ab)^* (ba)^* a^*$: abb ✓
- b) $(a \cup b)^* a (a \cup b)^* b (a \cup b)^*$: ba a ou bien b ✓ 1 lettre : a ou b 2 lettres : ba - 1/epsilon
- c) $b^* (a \cup ba)^* b^*$: abba ✓

3. Lesquelles de ces formules sont satisfaisables :

1 3 4 5 6 7 8 9 (-4)

Lesquelles sont des tautologies :

2 3 5 7 8 9 (-5)


1. $\Diamond a \vee \Box a$ 2. $\Box (a \rightarrow \Diamond a)$ 3. $(\Box a \rightarrow \Diamond \neg a)$ 4. $(a \rightarrow b) \rightarrow b$ 5. $a \vee \Diamond (a \rightarrow b)$ 6. $(a \rightarrow b) \vee b$ 7. $a \rightarrow (a \vee a)$ 8. $\neg (a \vee (a \rightarrow a))$ 9. $(a \rightarrow b) \vee (b \rightarrow a)$

4. Voici un analyseur de langage par descente récursive.

```
public class Lexer {
    public Lexer(String w);
    public void goToNextSymbol();
    public static final char
        END_OF_STRING_SYMBOL='\0';
    public char crtSymbol();
}
```

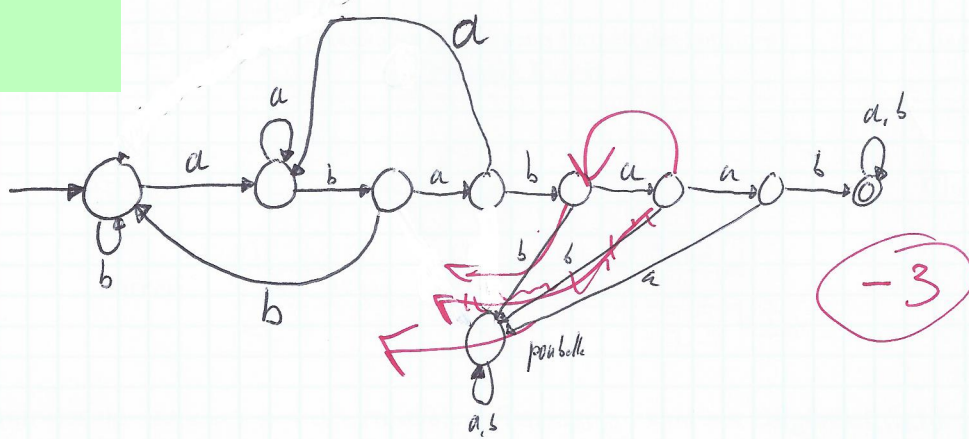
```
public class SyntaxAnalyzer {
    public static boolean isAccepted(String s) {
        ?????
    }
}
```

```
static void parseS(Lexer lex) throws Exception {
    if (lex.crtSymbol()=='b') {
        lex.goToNextSymbol();
        return;
    }
    if (lex.crtSymbol()!='v') throw new Exception();
    lex.goToNextSymbol();
    parseS(lex);
    while (lex.crtSymbol()=='g') {
        lex.goToNextSymbol();
        parseS(lex);
    }
    if (lex.crtSymbol()!='n') throw new Exception();
    lex.goToNextSymbol();
    if (lex.crtSymbol()=='f')
        lex.goToNextSymbol();
}
```

- a) Ecrire la méthode principale `isAccepted()`.
- b) Exprimer le langage correspondant sous forme de *diagrammes syntaxiques*. 
- c) Donner tous les mots de 6 lettres acceptés par cet analyseur.
- d) En plus des mots acceptés, on aimerait accepter toutes les variantes où on ajoute, n'importe où, des caractères 'z' (si "ata" est accepté, on acceptera aussi "zzatza" ou "ataz").

Expliquer (pas nécessaire de programmer) où/comment adapter le code

1



4

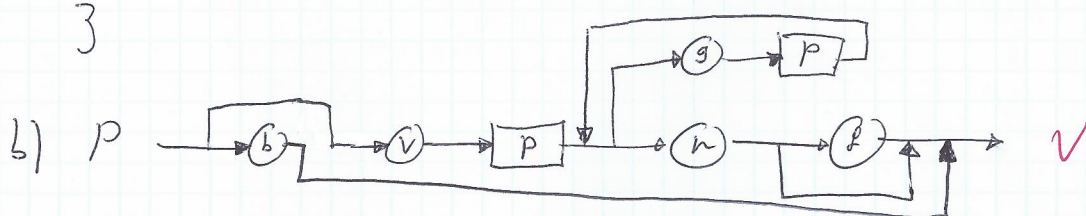
```

public static boolean isAccepted (String s) {
    Lexer lex = new Lexer (s);
    Parser p = new Parser (lex);
    try { parsS(lex); }
    catch (Exception e) { return false; }
    return true;
}

```

-2/end of string

3



c) ~~des variables~~
~~variables~~
~~right~~

v b g b n t ✓

~~v v b b n t~~
~~v v b b n~~

-2

d) avant et après chaque 0, mettre un ϵ dans
 il faut mettre des while (lex.currSymbol() != '2') { lex.goToNextSymbol(); }
 ✓