



# ALF

## Expressions régulières et Lexer

**Keith Cooper, Linda Torczon, *Engineering a Compiler***

- Chapitre 2
  - 2.4
  - 2.5
  - 2.6

**Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools (2nd Edition)***

- Chapitre 3
  - 3.1
  - 3.3

**Terence Parr, *The Definitive ANTLR 4 Reference*, (2nd Edition)**

- Expressions régulières
- Lexer
- ANTLR



# Donald Knuth

---



- Américain
- TeX
- The Art of Computer Programming
- Stanford

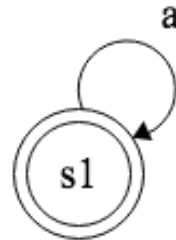
- Une façon de décrire la forme d'une string
- Les string sont reconnaissables par un automate fini
- Une string avec des caractères spéciaux

# Syntaxe d'expression régulière

Character	Description	Exemple	String
*	Zéro ou plusieurs fois	a*, (ab)*	aaaaaaaaa, ababababab
+	Une ou plusieurs fois	a+, (ab)+	aaaaaaaa, ababababab
?	Zéro ou une fois	a?, (ab)?	a, ab
^	début de string	^ab*	abbbbbbbb
\$	fin de string	b*a\$	bbbba
.	tout symbole	.	a, b, c
[ ]	Ensemble	[abc]	a, b
\s	Espace blanc	a\s b	a b
[^ ]	ensemble complémentaire	[^abc]	e, d
( )	groupe	(abc)+	Abcabcababc
	Ou	a b, (ab) (ba)	a, ab

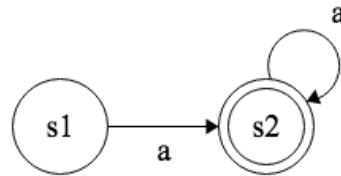
# Automate fini pur $a^*$

---



# Automate fini pur a+

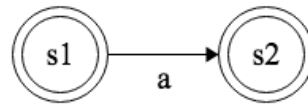
---





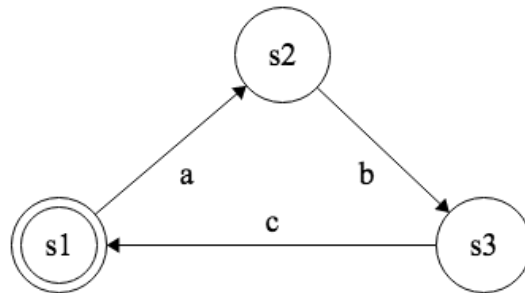
# Automate fini pur a?

---

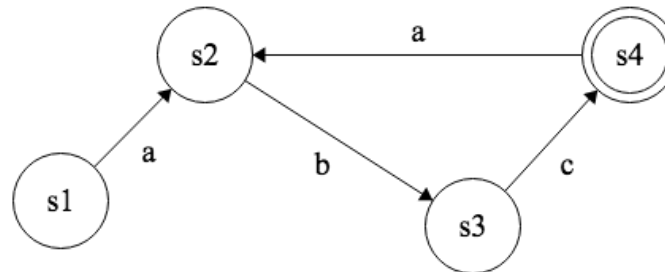


# Automate fini pour $(abc)^*$

---

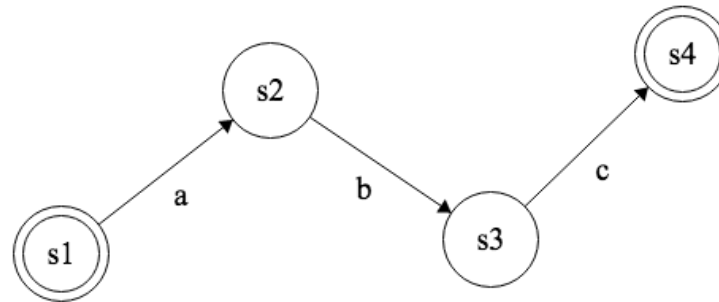


# Automate fini pour $(abc)^+$



# Automate fini pour (abc)?

---



# Exemples

---

- Groupe
- Téléphone
- Email
- Nom de variable
- Numéro entier
- Numéro avec virgule

# JavaScript RegEx

---

```
var regex = new RegEx ('...');
```

```
var regex = /.../;
```

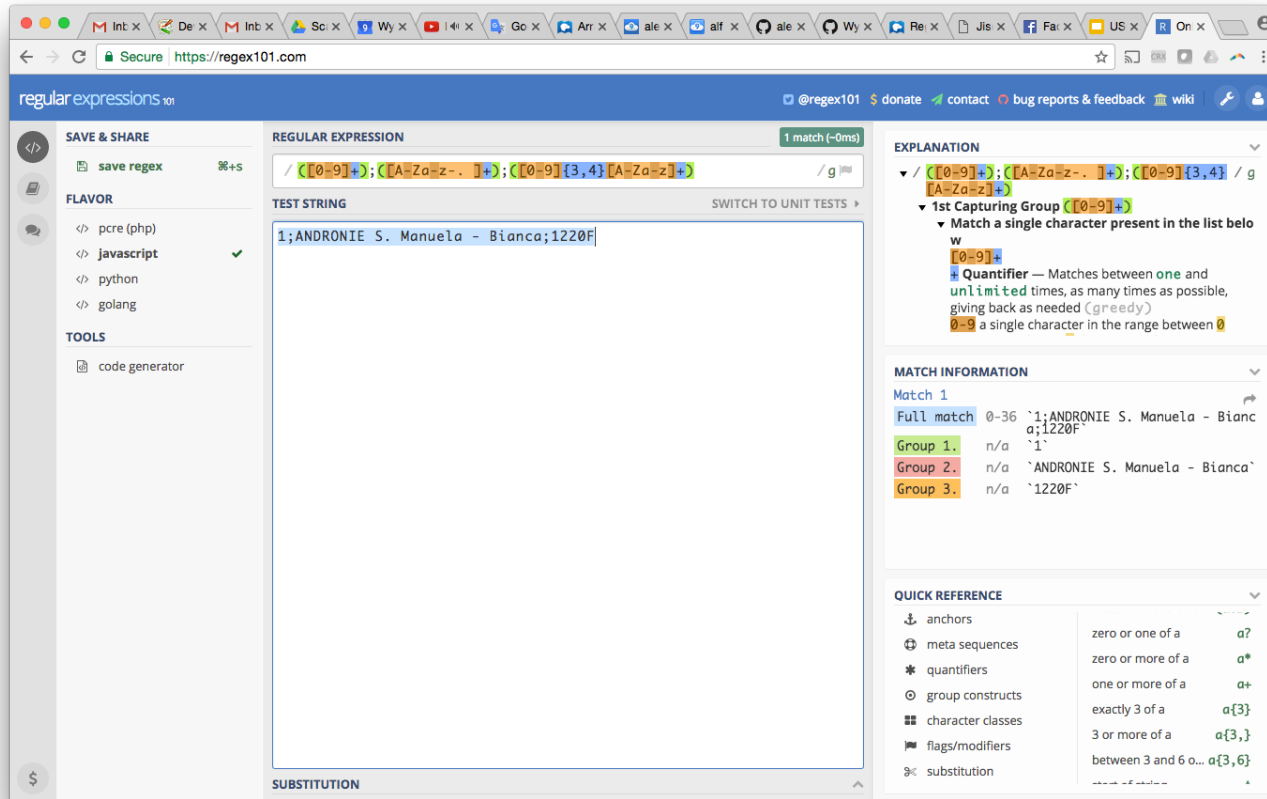
Documentation

[https://developer.mozilla.org/en/docs/Web/JavaScript/Guide/Regular Expressions](https://developer.mozilla.org/en/docs/Web/JavaScript/Guide/Regular_Expressions)

- RegEx
  - regex.exec (string)
    - Array
  - regex.test (string)
    - Boolean
- String
  - str.match (regex)
    - Array
  - str.search (regex)
    - Index of match

# RegEx101

- <https://regex101.com> (vérificateur de regex)



The screenshot shows the regex101.com website interface. The browser address bar displays "Secure https://regex101.com". The page title is "regular expressions 101". The main content area is divided into several sections:

- SAVE & SHARE**: Includes a "save regex" button and a "FLAVOR" dropdown menu with options for pcre (php), javascript (checked), python, and golang. There is also a "TOOLS" section with a "code generator" button.
- REGULAR EXPRESSION**: The input field contains the regex `/[0-9]+;[A-Za-z-. ]+;[0-9]{3,4}[A-Za-z]+/g`. A status bar indicates "1 match (-0ms)".
- TEST STRING**: The input field contains the string `1;ANDRONIE S. Manuela - Bianca;1220F`.
- EXPLANATION**: Provides a detailed breakdown of the regex components:
  - `[0-9]+`: Quantifier — Matches between one and unlimited times, as many times as possible, giving back as needed (greedy). `0-9` a single character in the range between 0 and 9.
  - `[A-Za-z-. ]+`: Match a single character present in the list below
    - `A`: Matches the character `A`.
    - `a`: Matches the character `a`.
    - `z`: Matches the character `z`.
    - `-`: Matches the character `-`.
    - : Matches the character .
  - `[0-9]{3,4}`: Match a single character present in the list below
    - `0`: Matches the character `0`.
    - `1`: Matches the character `1`.
    - `2`: Matches the character `2`.
    - `3`: Matches the character `3`.
    - `4`: Matches the character `4`.
  - `[A-Za-z]+`: Match a single character present in the list below
    - `A`: Matches the character `A`.
    - `a`: Matches the character `a`.
    - `z`: Matches the character `z`.
- MATCH INFORMATION**: Shows the results of the match:
  - Match 1**: Full match: `0-36` ``1;ANDRONIE S. Manuela - Bianca;1220F``
  - Group 1.**: `n/a` ``1``
  - Group 2.**: `n/a` ``ANDRONIE S. Manuela - Bianca``
  - Group 3.**: `n/a` ``1220F``
- QUICK REFERENCE**: A table of common regex symbols and their meanings:

Symbol	Description	Example
<code>^</code>	anchors	
<code>.</code>	meta sequences	zero or one of a <code>a?</code>
<code>*</code>	quantifiers	zero or more of a <code>a*</code>
<code>+</code>	quantifiers	one or more of a <code>a+</code>
<code>{n}</code>	group constructs	exactly 3 of a <code>a{3}</code>
<code>{n,m}</code>	character classes	3 or more of a <code>a{3,}</code>
<code> </code>	flags/modifiers	between 3 and 6 o... <code>a{3,6}</code>
<code>%</code>	substitution	



# Syntaxe d'expression régulière (JS)

Character	Description	Exemple	String
{n}	n fois	a{3}	aaa
{n,m}	au moins n, au plus m	a{3,7}	aaa, aaaa
\w	alphanumérique et _	\w	a, ab
\t	TAB	^a\t*	a »TAB »a
\n	fin de ligne	a\nb	a b
\r	retour chariot	a\rb	a\rb
a(?!b)	a seulement si non suivi par b	a(?!b)	a, aa
a(?=b)	a seulement si suivi par b	a(?=b)	ab
( )	group	a(ab)a	aaba

# JavaScript Example

---

```
var regex = new RegEx ('([0-9]+);([A-Za-z-\.  
]+);([0-9]{3,4}[A-Za-z]+)');
```

```
var regex = /([0-9]+);([A-Za-z-\.  
]+);([0-9]{3,4}[A-  
Za-z]+)/;
```

```
var match = regex.exec ('1;ANDRONIE S.  
Manuela - Bianca;1220F');
```

`([0-9]+);([A-Za-z-\.\ ]+);([0-9]{3,4}[A-Za-z]+)`

match:

`[ '1;ANDRONIE S. Manuela - Bianca;1220F',`

`'1',`

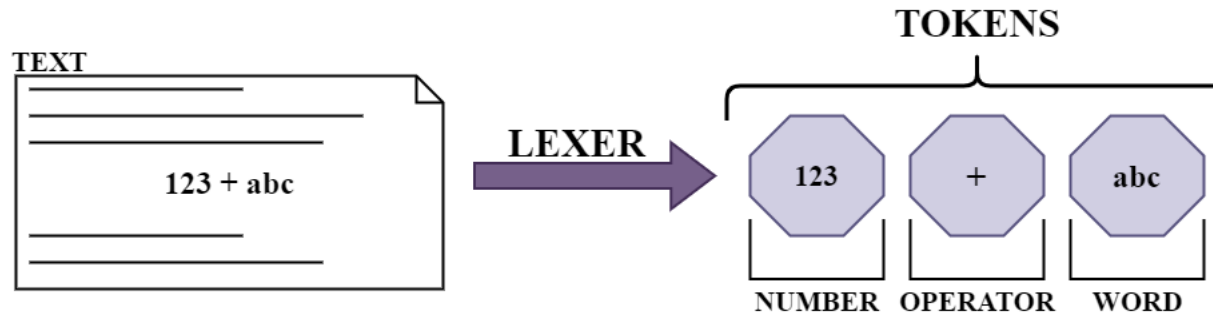
`'ANDRONIE S. Manuela - Bianca',`

`'1220F',`

`index: 0,`

`input: '1;ANDRONIE S. Manuela - Bianca;1220F fdsfs' ]`

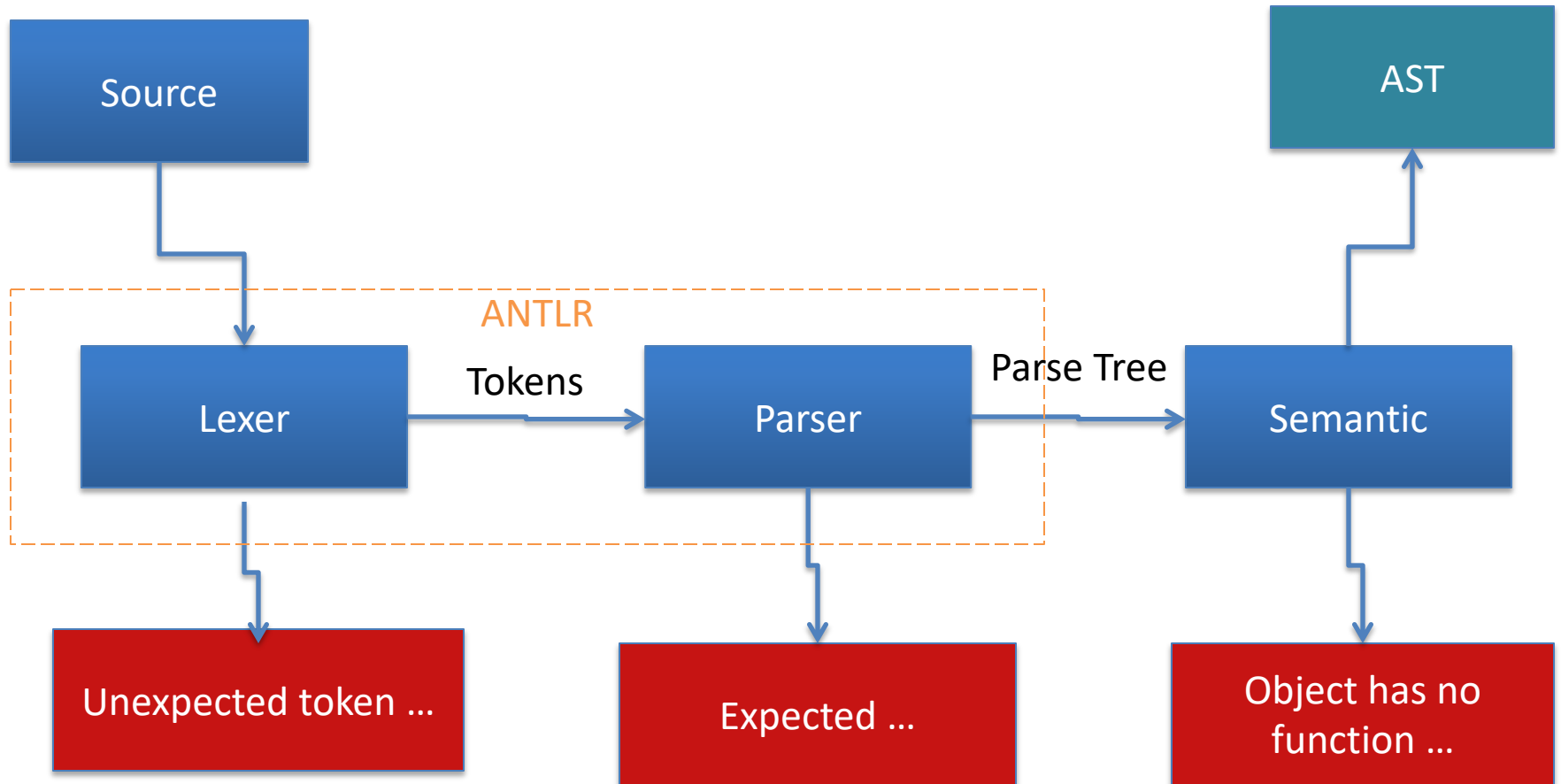
# Lexer



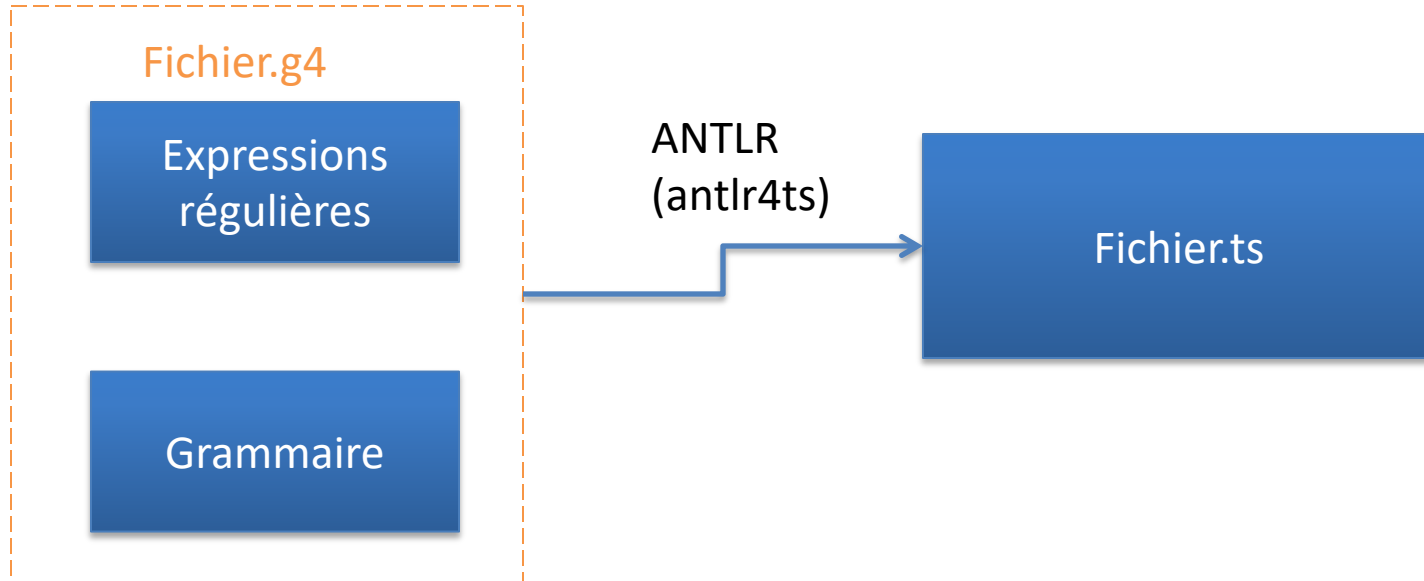


**ANTLR**

# Frontent



# Générateur de lexer/parseur



- Lexer
  - Expressions réguliers
- Parser
  - Grammaire



# Fichier.l (lex ou flex)

```
grammar Expr;
```

```
NEWLINE : [\r\n]+ ;
```

```
INT : [0-9]+ ;
```

```
prog: (expr NEWLINE)* ;
```

```
expr: expr ( '*' | '/' ) expr  
    | expr ( '+' | '-' ) expr  
    | INT  
    | '(' expr ')'  
    ;
```

# Fichier.g4

grammar Expr;

Nom

NEWLINE : [\r\n]+ ;

INT : [0-9]+ ;

Expressions Régulières

prog: (expr NEWLINE)\* ;

expr: expr ('\*' | '/') expr

| expr ('+' | '-') expr

| INT

| '(' expr ')'

;

Règles de Grammaire

# Nom

---

grammar Expr;

Nom

Expression Régulières

Règles de Grammaire

# Expressions Régulières

Nom

```
NEWLINE : [\r\n]+ ;  
INT : [0-9]+ ;
```

Expressions Régulières

Règles de Grammaire

# Règles de Grammaire

Nom

Expressions Régulières

```
prog: (expr NEWLINE)* ;  
expr: expr ('*' | '/') expr  
      | expr ('+' | '-') expr  
      | INT  
      | '(' expr ')'  
      ;
```

Règles de Grammaire

# Expr.g4 (vide)

---

grammar Expr;

prog: ;

- Expressions régulières
  - Mathématique
  - JavaScript
- Lexer
- ANTLR

# Questions

---

