Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Wh

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features Coqlex: Generating formally verified lexers

W. Ouedraogo G. Scherer L. Straßburger

Siemens Mobility - INRIA Saclay - LIX ouedraogo@lix.polytechnique.fr

Scalp Working Group Days - 16/02/2023

Presentation Overview



Lexers: What? Why?

W. edraogo, icherer, L. 3 Coglex: What? Why?

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

hy? Coqlex overview Coqlex Library Coqlex Generator

4 Evaluation

Execution time Usability and features

5 Conclusion

Compilation

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why?

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

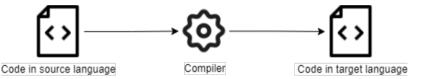
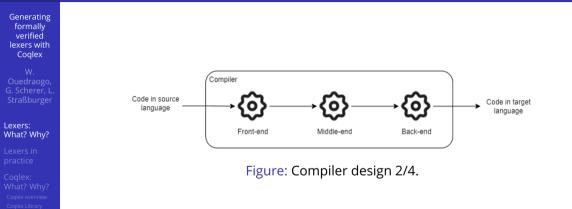


Figure: Compiler design 1/4.

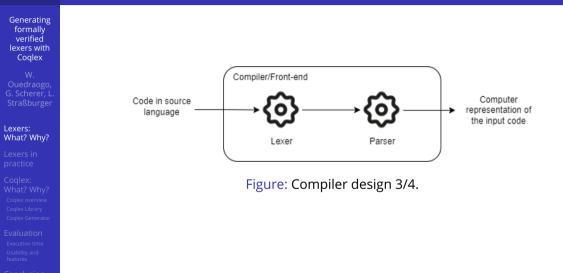
◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ● □ ● ● ● ●

Compilation



Conclusion

Compilation



◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ● □ ● ● ● ●

Front-end overview

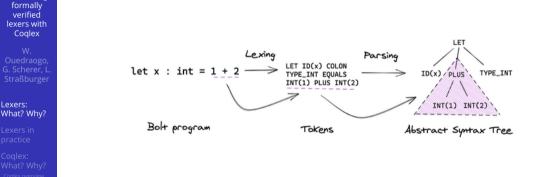


Figure: Compiler design 4/4.

Conclusion

Generating

◆□ > ◆□ > ◆臣 > ◆臣 > ○ 国 - の Q @

Lexers in practice

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features From a given string, a lexer generates a stream/list of tokens: part of the input string (lexeme) associated with meaning.

Common additional features

- 1 Ignore white spaces and comments
- 2 Detect/reject keywords
- **3** Track line/column numbers

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Lexers and parser are usually generated using generators

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Important concepts

- Lexical buffer
- 2 Lexical rules
- 3 Selection system

Lexical buffer data structure:

- tracks positions
- used as lexer input
- is updated by lexers

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion

Important concepts

1 Lexical buffer

Lexical rules

3 Selection system

```
1 (*OCamllex syntax*)
2 rule my_lexer = parse
3 'b' 'a'* 'b' { 0 }
4 | 'a'* { my_lexer lexbuf }
5 | 'c' { 20 }
6 | 'c'+ { 21 }
```

◆□ > ◆□ > ◆臣 > ◆臣 > ○ 国 - の Q @

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion

Important concepts	1
1 Lexical buffer	2
2 Lexical rules	3
•	4
3 Selection system	5

The priority and longuest match rule: the semantic action of the first lexical rule whose regex matches the longest prefix of the input string.

rule my_lexer = parse
 'b' 'a'* 'b' { 0 }

| 'c' { 20 }
| 'c' + { 21 }

/ 'a' * { my_lexer lexbuf }

▲□▶▲□▶▲目▶▲目▶ 目 のへの

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features Important concepts

Lexical buffer
 Lexical rules

3 Selection system

1	rule my_l	exe	er =	= parse	
2	'b' 'a'	*	'b'	{ 0 }	
3	'a'*	{	my_	_lexer lexbuf	}
4	'c'	{	20	}	
5	'c'+	{	21	}	

Tokens for 'c'?

Generating formally verified lexers with Coglex

Levers in practice

Important concepts **1** Lexical buffer **2** Lexical rules 3 Selection system

3

4

5 6

7

8

```
rule my_lexer = parse
1
   'b' 'a'* 'b' { 0 } (*No match*)
2
   / 'a'* { my lexer lexbuf } (*Matches ``*)
     'c'
           { 20 } (*Matches 'c' *)
     'c'+ { 21 } (*Matches `c` *)
    (* result tokens: [20]*)
```

Tokens for 'c'?

◆□▶ ◆□▶ ◆三▶ ◆三▶ ○ ○ ○ ○

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features Important concepts

Lexical buffer
 Lexical rules

3 Selection system

1	rule my_l	exe	er =	= parse	
2	'b' 'a'	* '	b'	{ O }	
3	'a'*	{	my_	_lexer lexbuf	}
4	'c'	{	20	}	
5	'c'+	{	21	}	

Tokens for 'cc' ?

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features Important concepts
 Lexical buffer
 Lexical rules
 Selection system

1

2

3

4

5 6

7

8

```
rule my_lexer = parse
  'b' 'a'* 'b' { 0 } (*No match*)
  | 'a'* { my_lexer lexbuf } (*Matches ``*)
  | 'c' { 20 } (*Matches `c` *)
  | 'c'+ { 21 } (*Matches `cc` *)
  (* result tokens: [21]*)
```

◆□▶ ◆□▶ ◆三▶ ◆三▶ ○ ○ ○ ○

Tokens for 'cc' ?

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features Important concepts

Lexical buffer
 Lexical rules

3 Selection system

1	rule my_l	ex	er =	= par	se	
2	'b' 'a'	*	'b'	{ 0	}	
3	'a'*	{	my_]	exer	lexbuf	}
4	'c'	{	20	}		
5	'c'+	{	21	}		

Tokens for 'aabbc' ?

Generating formally verified lexers with Coglex

```
Levers in
practice
```

Important concepts **1** Lexical buffer **2** Lexical rules 3 Selection system

3

4

5 6

7

8

```
rule my_lexer = parse
1
    'b' 'a' * 'b' { 0 } (*No match*)
2
   / 'a' * { my_lexer lexbuf } (*Matches 'aa'*)
     'c' { 20 } (*No match*)
     'c'+ { 21 } (*No match*)
    (* result tokens: ...*)
```

◆□▶ ◆□▶ ◆三▶ ◆三▶ ○ ○ ○ ○

Tokens for 'aabbc'?

Important concepts

Generating formally verified lexers with Coglex

Levers in practice

1 Lexical buffer **2** Lexical rules 3 Selection system

```
rule my_lexer = parse
1
    'b' 'a'* 'b' { 0 } (*Matches `bb`*)
2
   / 'a' * { my lexer lexbuf } (*No match*)
3
     'c' { 20 } (*No match*)
4
     'c'+ { 21 } (*No match*)
5
6
7
    (* result tokens: [0; ...]*)
```

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Tokens for 'aabbc'?

8

Generating formally verified lexers with Coglex

Levers in practice

Important concepts **1** Lexical buffer **2** Lexical rules 3 Selection system

```
rule my_lexer = parse
1
    'b' 'a'* 'b' { 0 } (*No match*)
2
   / 'a' * { my lexer lexbuf } (*No match*)
3
     'c' { 20 } (*Matches `c` *)
4
     'c'+ { 21 } (*Matches `c` *)
5
7
    (* result tokens: [0; 20]*)
```

◆□▶ ◆□▶ ◆三▶ ◆三▶ ○ ○ ○ ○

Tokens for 'aabbc'?

6

8

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features Important concepts

Lexical buffer
 Lexical rules

3 Selection system

1	rule my_l	exe	er =	= parse	
2	'b' 'a'	*	'b'	{ 0 }	
3	'a'*	{	my_	_lexer lexbuf	}
4	'c'	{	20	}	
5	′c′+	{	21	}	

Tokens for 'd'?

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Why

```
Lexers in 
practice
```

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion

Important concepts

Lexical buffer
 Lexical rules

3 Selection system

```
rule my_lexer = parse
1
    'b' 'a'* 'b'
                  { 0 } (*No match*)
2
    / 'a'* { my lexer lexbuf } (*Matches ``*)
3
     'c'
            { 20 } (*No match*)
4
           { 21 } (*No match*)
     'c'+
5
6
7
    (* result tokens: Infinite loop*)
8
```

Tokens for 'd' ?

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

Generating formally verified lexers with Coglex

Levers in practice

Important concepts

1 Lexical buffer **2** Lexical rules

3 Selection system

4

6

7

```
1
 rule my_lexer = parse
   'b' 'a'* 'b' { 0 } (*No match*)
2
     'c'
           { 20 } (*No match*)
3
     'c'+
           { 21 } (*No match*)
    (* result tokens: Error*)
```

Tokens for 'd'?

◆□▶ ◆□▶ ◆三▶ ◆三▶ ○ ○ ○ ○

Coqlex: What? Why?

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Contribution

A Coq library

2 A generator

Goals

- 1 Simplify lexer implementation
- 2 Allow to write proofs on implemented lexers
- 3 Usable
- 4 Easy to

understand/review/improve

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, L Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Implementation details

- Typing a lexer
- 2 Brzozowski derivatives
- 3 Selection system
- Optimization

lexer(T) :=
nat ->lexbuf ->Result(T) x lexbuf

action(T) :=
lexbuf ->Result(T) x lexbuf

▲□▶▲□▶▲□▶▲□▶ □ ● ● ●

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion

Implementation details

- 1 Typing a lexer
- 2 Brzozowski derivatives (1/4)
- 3 Selection systems
- **4** Optimization

Regex definition

regex ::=

$$\begin{array}{ll} \varnothing_r & L(\varnothing_r) = \varnothing \\ \epsilon_r & L(\epsilon_r) = \{\epsilon\} \\ \llbracket a \rrbracket & L(\llbracket a \rrbracket) = \{a\} \\ e_1 + e_2 & L(e_1 + e_2) = L(e_1) \cup L(e_1) \\ e_1 \cdot e_2 & L(e_1 \cdot e_2) = \\ & \{s_1 + + s_2 | s_1 \in L(e_1) \land s_2 \in L(e_2)\} \\ e^* & L(e^*) = \{s^n | s \in L(e) \land n \in \mathbb{N}\} \end{array}$$

・ロト・西ト・ヨト ・ヨー うらぐ

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion

Implementation details

1 Typing a lexer

 2 Brzozowski derivatives (2/4)

3 Selection system

4 Optimization

the nullable funtion

nullable $\emptyset_r = \text{false}$ nullable $\epsilon_r = \text{true}$ nullable $\llbracket a \rrbracket = \text{false}$ nullable $(e_1 + e_2) =$ nullable $e_1 \lor \text{nullable} e_2$ nullable $(e_1 \land e_2) =$ nullable $e_1 \land \text{nullable} e_2$ nullable $e_1 \land \text{nullable} e_2$

・ロト・西ト・ヨト ・ヨー うらぐ

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Wh

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Implementation details

- 1 Typing a lexer
- 2 Brzozowski derivatives (3/4)
- 3 Selection system
- 4 Optimization

the derive funtion

$$\begin{split} & \varnothing_r/c &= \varnothing_r \\ & \epsilon_r/c &= \varnothing_r \\ & [a]] \ / \ c &= \begin{cases} \epsilon \text{ if } a = c \\ & \varnothing_r \text{ otherwise} \end{cases} \\ & (e_1 + e_2)/c = (e_1/c) + (e_2/c) \\ & (e_1 \cdot e_2)/c &= \begin{cases} (e_1/c \cdot e_2) + e_2/c \text{ if nullable } e_1 \\ & (e_1/c \cdot e_2) \text{ otherwise} \end{cases} \\ & e^*/c &= (e/c) \cdot e^* \end{split}$$

・ロト・日本・日本・日本・日本

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Implementation details

- 1 Typing a lexer
- 2 Brzozowski derivatives (4/4)
- 3 Selection system
- Optimization

Matching a string

$$r//\epsilon = r \quad r//az = (r/a)//z$$

matches $r \ z =$ nullable (r//z)

・ロト・西ト・ヨト ・ヨー うらぐ

1 Typing a lexer

derivatives

4 Optimization

3 Selection system • Score: Sr

Selection

2 Brzozowski

Generating formally verified lexers with Coglex

Coglex Library

Implementation details Score computation nullable r = true nullable r = false $\mathbb{S}_l(r,\epsilon) = 0$ $\mathbb{S}_l(\mathbf{r},\epsilon) = -\infty$ $\mathbb{S}_l(r/a, z) = n$ $\mathbb{S}_l(r, az) = n+1$ $\mathbb{S}_{l}(r/a, z) = -\infty$ nullable r = true $\mathbb{S}_{l}(r,az)=0$ $\mathbb{S}_{l}(r/a, z) = -\infty$ nullable r = false $\mathbb{S}_{l}(r,az) = -\infty$

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, L Straßburgei

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Implementation details

1 Typing a lexer

2 Brzozowski derivatives

3 Selection systems

- Scores
- Selection
- Optimization

Semantic rule selection Choosing the first rule with the highest score (first argmax).

Problem: Lexing in quadratic time.

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, L Straßburgei

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion

Implementation details

- 1 Typing a lexer
- Brzozowski derivatives
- 3 Selection systems
- Optimization

Idea: Stop \mathbb{S}_l as soon as possible

Details

- 1 Adding faster constructions ex: ['0'- '9'] vs '0' | '1' | ... | '9'
- 2 Regexp simplification ex: $(r^*)^* \equiv r^*, r \cdot \varnothing_r \equiv \varnothing_r$
- **3** Stopping for trivial cases ex: $\mathbb{S}_l(\epsilon_r, s) = 0$, $\mathbb{S}_l(\emptyset_r, s) = -\infty$

◆□▶ ◆□▶ ◆三▶ ◆三▶ ○○○

Coqlex generator

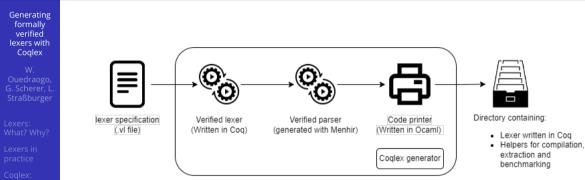


Figure: Coqlex generator architecture.

▲□▶ ▲□▶ ▲ □▶ ▲ □▶ ▲ □ ● ● ● ●

Evaluation Execution time Usability and features

Coglex Generator

Conclusion

State of the art

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Wh

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion

Related work and tools

1 Lexers written by hand (ex: CakeML)

▲□▶▲□▶▲□▶▲□▶ □ のQ@

- 2 Nipkow
- OCamllex (Lex, Flex)
- 4 Verbatim/Verbatim++

Coqlex vs OCamllex vs Verbatim++



W. Ouedraogo G. Scherer, I Straßburge

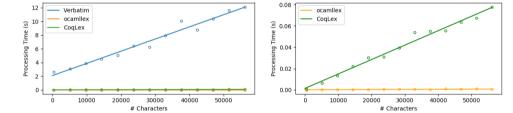
Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion



	Verbatim++	Coqlex	OCamllex
Tokens per sec.	$1.7 imes10^3$	$2.23 imes10^5$	$3.9 imes10^7$
Time to process 56ko.	12.11 s	$7.7 imes 10^{-2} ext{ s}$	$4.4 imes 10^{-4} ext{ s}$

Coqlex vs OCamllex vs Verbatim++

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, L Straßburgei

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

	Coqlex	OCamllex	Verbatim++	
Lexer language	Coq	OCaml	Coq	
	lexbuf ->			
Semantic action	Result(token)	OCaml code	token	
Semantic action	х			
	lexbuf			
Error handling	yes	yes	no	
mechanism	yes	yes	110	
Formally verified	yes	no	yes	
lexers	yes		yes	
Execution speed	100x slower	fastest (reference)	10000x slower	
	roox slower		10000X Slower	
Generator	yes	yes	no	

Conclusion

Generator language

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Why Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

.vl syntax vs .mll syntax

Listing 1: looping.vl

```
1 rule my_lexer = parse
2 'b' 'a' * 'b' { ret 0 }
3 | 'a' * { my_lexer }
```

4 | EOF { ret 1 }

Listing 2: looping.mll

1 rule my_lexer = parse
2 'b' 'a'* 'b' { 0 }
3 | 'a'* { my_lexer lexbuf }
4 | EOF { 1 }

Remark: We proved that this lexer loops if the input string starts by a character *x* such that $x \neq a'$, and $x \neq b'$

Coqlex industrial use-case

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Wh

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Coqlex in industry

- 1 In a Ada-to-Ada formally verified compiler
- 2 Biggest program: 2380 files (25MB of code)
- 3 Formally verified front-end
- **4** Compilation timee: x4 compared to the unverified front-end version.

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Coqlex in a nutshell

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, I Straßburge

Lexers: What? Wh

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

Conclusion

Usable

- 2 Simple
- 3 Formally verified
- 4 Common lexer features are implemented

- Coq proof of S₁, correctness and completeness
- Coq proof of Lexical rule selection is correctness and completeness
- Coq proof of Optimizations correctness

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Future work

Generating formally verified lexers with Coqlex

W. Ouedraogo, G. Scherer, L Straßburger

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features Conclusion

Improvements for Coqlex

- 1 Speed up
- 2 Termination proof
- Ocamllex <-> Coqlex converter

▲□▶▲□▶▲□▶▲□▶ □ のQ@

4 CompCert

The End

Generating formally verified lexers with Coqlex

W. Ouedraogo G. Scherer, L Straßburgei

Lexers: What? Why

Lexers in practice

Coqlex: What? Why? Coqlex overview Coqlex Library Coqlex Generator

Evaluation Execution time Usability and features

References

- Brzozowski Janusz A. (1964) Derivatives of regular expressions JACM 1964
- Joshua B Smith (2007) Ocamllex and Ocamlyacc
- Derek Egolf and al. (2022)

Verbatim++: verified, optimized, and semantically rich lexing with derivatives CPP 2022 27 – 39.

W. OUEDRAOGO and al. (2022) Git link for Coqlex: https://gitlab. inria.fr/wouedrao/coqlex

Thank you. Questions? Comments?

▲□▶▲□▶▲□▶▲□▶ □ ● ● ●