WIKIPEDIA

Comparison of parser generators

This is a list of notable lexer generators and parser generators for various language classes.

Contents

Regular languages

Regular languages are a category of languages (sometimes known as Chomsky Type 3) which can be matched by a state machine (more specifically, by a deterministic finite automaton or a nondeterministic finite automaton) constructed from a regular expression. In particular, a regular language can match constructs like "A follows B", "Either A or B", "A, followed by zero or more instances of B", but cannot match constructs which require consistency between non-adjacent elements, such as "some instances of A followed by the same number of instances of B", and also cannot express the concept of recursive "nesting" ("every A is eventually followed by a matching B"). A classic example of a problem which a regular grammar cannot handle is the question of whether a given string contains correctly-nested parentheses. (This is typically handled by a Chomsky Type 2 grammar, also known as a context-free grammar.)

Name	Lexer algorithm	Output languages	Grammar, code	Development platform	License
Alex	DFA	Haskell	mixed	all	BSD
AnnoFlex	DFA	Java	mixed	Java Virtual Machine	BSD
AustenX	DFA	Java	separate	all	BSD
Booze–tools (https://github.com/kjosib/booze–tools)	DFA	state machine is runtime–generated or saved as JSON	mixed	Python	Public Domain
C# Flex	DFA	C#	mixed	.NET CLR	GNU GPL
C# Lex	DFA	C#	mixed	.NET CLR	?
CookCC	DFA	Java	mixed	Java Virtual Machine	Apache License 2.0
DFAlex	DFA	no code generation required	Java	Java	Apache License 2.0
Dolphin	DFA	C++	separate	all	Proprietary
flex	DFA table driven	C, C++	mixed	all	BSD
gelex	DFA	Eiffel	mixed	Eiffel	MIT
golex	DFA	Go	mixed	Go	BSD–style
gplex	DFA	C#	mixed	.NET CLR	BSD–like
JFlex	DFA	Java	mixed	Java Virtual Machine	BSD
JLex	DFA	Java	mixed	Java Virtual Machine	BSD–like
lex	DFA	C	mixed	POSIX	Proprietary, CDDL
lexertl	DFA	C++		all	GNU LGPL
LRSTAR	DFA	C++	separate	Windows	BSD
Quex	DFA direct code	C, C++	mixed	all	GNU LGPL
Ragel	DFA	C, C++, Assembly, Objective C, D, Go, Ruby, Java, C#, OCaml, Crack, Rust, Julia	mixed	all	GNU GPL, MIT[1]
RE/flex	DFA direct code, DFA table driven, and NFA regex libraries	C++	mixed	all	BSD
re2c	DFA direct code	C	mixed	all	Public domain

Deterministic context-free languages

Context-free languages are a category of languages (sometimes known as Chomsky Type 2) which can be matched by a sequence of replacement rules, each of which essentially maps each non-terminal element to a sequence of terminal elements and/or other nonterminal elements. Grammars of this type can match anything that can be matched by a regular grammar, and furthermore, can handle the concept of recursive "nesting" ("every A is eventually followed by a matching B"), such as the question of whether a given string contains correctly-nested parentheses. The rules of Context-free grammars are purely local, however, and therefore cannot handle questions that require non-local analysis such as "Does a declaration exist for every variable that is used in a function?". To do so technically would require a more sophisticated grammar, like a Chomsky Type 1 grammar, also known as a Context-sensitive grammar. However, parser generators for context-free grammars often support the ability for user-written code to introduce limited amounts of context-sensitivity. (For instance, upon encountering a variable declaration, user-written code could save the name and type of the variable into an external data structure, so that these could be checked against later variable references detected by the parser.)

The deterministic context-free languages are a proper subset of the Context-Free languages which can be efficiently parsed by Deterministic pushdown automata.

Name	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License
ANTLR4	ALL(*)[2]	EBNF	C#, Java, Python, JavaScript, C++, Swift, Go	mixed	generated	Java Virtual Machine	Yes	BSD
ANTLR3	LL(*)	EBNF	ActionScript, Ada95, C, C++, C#, Java, JavaScript, Objective-C, Perl, Python, Ruby	mixed	generated	Java Virtual Machine	Yes	BSD
APG	Recursive descent, Backtracking	ABNF	C, C++, JavaScript, Java	separate	none	all	No	GNU GPL
AXE	Recursive descent	AXE/C++	C++17, C++11	mixed	none	any platform with standard C++17/C++11 compiler	No	Boost
Beaver	LALR(1)	EBNF	Java	mixed	external	Java Virtual Machine	No	BSD
Bison	LALR(1), LR(1), IELR(1), GLR	YACC	C, C++, Java	mixed	external	all	No	GNU GPL with Exception
Bison++[note 1]	LALR(1)	?	C++	mixed	external	POSIX	No	GNU GPL
Bisonc++	LALR(1)	?	C++	mixed	external	POSIX	No	GNU GPL
Booze-tools (https://github.com/kjosib/booze-tools)	LALR(1) or LR(1) (canonical or minimal)	BNF with macros in place of EBNF	state machine can be runtime-generated or saved as JSON	mixed, separable	included	Python	No	Public domain
BtYacc	Backtracking Bottom-up	?	C++	mixed	external	all	No	Public domain
byacc	LALR(1)	YACC	C	mixed	external	all	No	Public domain
BYACC/J	LALR(1)	YACC	C, Java	mixed	external	all	No	Public domain
CL-Yacc	LALR(1)	Lisp	Common Lisp	mixed	external	all	No	MIT
Coco/R	LL(1)	EBNF	C, C++, C#, F#, Java, Ada, Object Pascal, Delphi, Modula-2, Oberon, Ruby, Swift, Unicon, Visual Basic .NET	mixed	generated	Java Virtual Machine, .NET Framework, Microsoft Windows, POSIX (depends on output language)	No	GNU GPL

Product	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License

Name	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License
CookCC	LALR(1)	Java annotations	Java	mixed	generated	Java Virtual Machine	No	Apache License 2.0
CppCC	LL(k)	?	C++	mixed	generated	POSIX	No	GNU GPL
CSP	LR(1)	?	C++	separate	generated	POSIX	No	Apache License 2.0
CUP	LALR(1)	?	Java	mixed	external	Java Virtual Machine	No	BSD-like
Dragon	LR(1), LALR(1)	?	C++, Java	separate	generated	all	No	GNU GPL
eli	LALR(1)	?	C	mixed	generated	POSIX	No	GNU GPL, GNU LGPL
Epsilon Grammar Studio	Recursive descent, Backtracking	ABNF	C++	separate	generated	Microsoft Windows	Yes	proprietary
Essence	LR(???)	?	Scheme 48	mixed	external	all	No	BSD
Eto.Parse	LL(k)	BNF, EBNF or C#	N/A (state machine is runtime generated)	separate	internal	.NET Framework	No	MIT
eyapp	LALR(1)	?	Perl	mixed	external or generated	all	No	Perl
Frown	LALR(k)	?	Haskell 98	mixed	external	all	No	GNU GPL
geyacc	LALR(1)	?	Eiffel	mixed	external	all	No	MIT
GOLD	LALR(1)	BNF	x86 assembly language, ANSI C, C#, D, Java, Pascal, Object Pascal, Python, Visual Basic 6, Visual Basic .NET, Visual C++	separate	generated	Microsoft Windows	Yes	Modified Zlib
GPPG	LALR(1)	YACC	C#	separate	external	Microsoft Windows	Yes	BSD
Grammatica	LL(k)	BNF dialect	C#, Java	separate	generated	Java Virtual Machine	No	BSD
HiLexed	LL(*)	EBNF or Java	Java	separate	internal	Java Virtual Machine	No	GNU LGPL
Hime Parser Generator	LALR(1), GLR	BNF dialect	C#, Java, Rust	separate	generated	.NET Framework, Java Virtual Machine	No	GNU LGPL
Hyacc	LR(1), LALR(1), LR(0)	YACC	C	mixed	external	all	No	GNU GPL

Product	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License

Name	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License
Irony	LALR(1)	C#	N/A (state machine is runtime generated)	separate	internal	.NET Framework	Yes	MIT
iyacc	LALR(1)	YACC	Icon	mixed	external	all	No	GNU GPL
jacc	LALR(1)	?	Java	mixed	external	Java Virtual Machine	No	BSD
JavaCC	LL(k)	EBNF	Java, C++, JavaScript (via GWT compiler)[3]	mixed	generated	Java Virtual Machine	Yes	BSD
jay	LALR(1)	YACC	C#, Java	mixed	none	Java Virtual Machine	No	BSD
JFLAP	LL(1), LALR(1)	?	Java	?	?	Java Virtual Machine	Yes	?
JetPAG	LL(k)	?	C++	mixed	generated	all	No	GNU GPL
JS/CC	LALR(1)	EBNF	JavaScript, JScript, ECMAScript	mixed	internal	all	Yes	BSD
KDevelop-PG-Qt	LL(1), Backtracking, Shunting yard	?	C++	mixed	generated or external	all, KDE	No	GNU LGPL
Kelbt	Backtracking LALR(1)	?	C++	mixed	generated	POSIX	No	GNU GPL
kmyacc	LALR(1)	?	C, Java, Perl, JavaScript	mixed	external	all	No	GNU GPL
Lapg	LALR(1)	?	C, C++, C#, Java, JavaScript	mixed	generated	Java Virtual Machine	No	GNU GPL
Lemon	LALR(1)	?	C	mixed	external	all	No	Public domain
LEPL	Recursive descent	Python	Python (no generation, library)	separate	none	all	No	MPL/GNU LGPL
Lime	LALR(1)	?	PHP	mixed	external	all	No	GNU GPL
LISA	LR(?), LL(?), LALR(?), SLR(?)	?	Java	mixed	generated	Java Virtual Machine	Yes	Public domain
LLgen	LL(1)	?	C	mixed	external	POSIX	No	BSD
LLnextgen	LL(1)	?	C	mixed	external	all	No	GNU GPL
LLLPG	LL(k) + syntactic and semantic predicates	ANTLR-like	C#	mixed	generated (?)	.NET Framework, Mono	Visual Studio	GNU LGPL
LPG	Backtracking LALR(k)	?	Java	mixed	generated	Java Virtual Machine	No	EPL
LRSTAR	LALR(1), LR(1), LR(*)	EBNF, Yacc-like	C++	separate	generated	Windows	Visual Studio	BSD
Menhir	LR(1)	?	OCaml	mixed	generated	all	No	QPL
ML-Yacc	LALR(1)	?	ML	mixed	external	all	No	?

Product	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License

Name	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License
Monkey	LR(1)	?	Java	separate	generated	Java Virtual Machine	No	GNU GPL
Msta	LALR(k), LR(k)	YACC, EBNF	C, C++	mixed	external or generated	POSIX, Cygwin	No	GNU GPL
MTP (More Than Parsing)	LL(1)	?	Java	separate	generated	Java Virtual Machine	No	GNU GPL
MyParser	LL(*)	Markdown	C++11	separate	internal	any platform with standard C++11 compiler	No	MIT License
NLT	GLR	C#/BNF-like	C#	mixed	mixed	.NET Framework	No	MIT
ocamlyacc	LALR(1)	?	OCaml	mixed	external	all	No	QPL
olex	LL(1)	?	C++	mixed	generated	all	No	GNU GPL
parglare (https://github.com/igordejanovic/parglare)	Scannerless LALR(1)/SLR(1)/GLR	BNF-like, Python	N/A (state machine is runtime generated)	mixed	none	all	No	MIT
Parsec	LL, Backtracking	Haskell	Haskell	mixed	none	all	No	BSD
Parse::Yapp	LALR(1)	?	Perl	mixed	external	all	No	GNU GPL
Parser Objects	LL(k)	?	Java	mixed	?	Java Virtual Machine	No	zlib
PCCTS	LL	?	C, C++	?	?	all	No	?
PLY	LALR(1)	BNF	Python	mixed	generated	all	No	MIT License
PlyPlus	LALR(1)	EBNF	Python	separate	generated	all	No	MIT License
PRECC	LL(k)	?	C	separate	generated	DOS, POSIX	No	GNU GPL
QLALR	LALR(1)	?	C++	mixed	external	all	No	GNU GPL
RPATK	Recursive descent, Backtracking	BNF	C (no generation, library)	separate	none	all	No	GNU GPL
SableCC	LALR(1)	?	C, C++, C#, Java, OCaml, Python	separate	generated	all	No	GNU LGPL
SLK	LL(k) LR(k) LALR(k)	EBNF	C, C++, C#, Java, JavaScript	separate	external	all	No	SLK[4]
SP (Simple Parser)	Recursive descent	Python	Python	separate	generated	all	No	GNU LGPL
Spirit	Recursive descent	?	C++	mixed	internal	all	No	Boost
Sprache	LL, Backtracking	C#	interpreted	mixed	internal	.NET Framework	No	MIT
Styx	LALR(1)	?	C, C++	separate	generated	all	No	GNU LGPL
Sweet Parser	LALR(1)	?	C++	separate	generated	Microsoft Windows	No	zlib
Tap	LL(1)	?	C++	mixed	generated	all	No	GNU GPL

Product	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License

Name	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License
TextTransformer	LL(k)	?	C++	mixed	generated	Microsoft Windows	Yes	Proprietary
TinyPG	LL(1)	?	C#, Visual Basic	?	?	Microsoft Windows	Yes	CPOL 1.0
Toy Parser Generator	Recursive descent	?	Python	mixed	generated	all	No	GNU LGPL
TP Yacc	LALR(1)	?	Turbo Pascal	mixed	external	all	Yes	GNU GPL
UltraGram	LALR(1), LR(1), GLR	BNF	C++, Java, C#, Visual Basic .NET	separate	external	Microsoft Windows	Yes	Public domain
UniCC (https://github.com/phorward/unicc)	LALR(1)	EBNF	C, C++, Python, JavaScript, JSON, XML	mixed	generated	POSIX	No	BSD
UrchinCC	LL(1)	?	Java	?	generated	Java Virtual Machine	No	?
Whale	LR(?), some conjunctive stuff, see Whale Calf	?	C++	mixed	external	all	No	Proprietary
wisent	LALR(1)	?	C++, Java	mixed	external	all	No	GNU GPL
Yacc AT&T/Sun	LALR(1)	YACC	C	mixed	external	POSIX	No	CPL & CDDL
Yacc++	LR(1), LALR(1)	YACC	C++, C#	mixed	generated or external	all	No	Proprietary
Yapps	LL(1)	?	Python	mixed	generated	all	No	MIT
yecc	LALR(1)	?	Erlang	separate	generated	all	No	Erlang
Visual BNF	LR(1), LALR(1)	?	C#	separate	generated	.NET Framework	Yes	Proprietary
YooParse	LR(1), LALR(1)	?	C++	mixed	external	all	No	MIT
Parse (https://github.com/MathiasVP/Parse/)	LR(1)	BNF in C++ types	?	?	none	C++11 compliant compiler	No	MIT
GGLL (https://github.com/tassotirap/GGLL.UI)	LL(1)	Graph	Java	mixed	generated	Windows	Yes	MIT
Product	**Parsing algorithm**	**Input grammar notation**	**Output languages**	**Grammar, code**	**Lexer**	**Development platform**	**IDE**	**License**

Parsing expression grammars, deterministic boolean grammars

Name	Parsing algorithm	Output languages	Grammar, code	Development platform	License
Arpeggio (https://github.com/igordejanovic/Arpeggio)	PEG parser interpreter, Packrat	Python (no generation, interpreted)	mixed	all	MIT
AustenX	Packrat (modified)	Java	separate	all	BSD
Aurochs	Packrat	C, OCaml, Java	mixed	all	GNU GPL
BNFlite (https://github.com/r35382/bnflite)	Recursive descent	C++	mixed	all	MIT
Canopy (http://canopy.jcogla n.com/)	Packrat	Java, JavaScript, Python, Ruby	separate	all	GNU GPL
CL-peg	Packrat	Common Lisp	mixed	all	MIT
Drat!	Packrat	D	mixed	all	GNU GPL
Frisby	Packrat	Haskell	mixed	all	BSD
grammar::peg	Packrat	Tcl	mixed	all	BSD
Grako	Packrat + Cut + Left Recursion	Python / C++ (beta)	separate	all	BSD
IronMeta	Packrat	C#	mixed	Microsoft Windows	BSD
Katahdin	Packrat (modified), mutating interpreter	C#	mixed	all	Public domain
Laja	2-phase scannerless top-down backtracking + runtime support	Java	separate	all	GNU GPL
lars::Parser (https://github.com/TheLartians/Parser)	Packrat (supporting left-recursion and grammar ambiguity)	C++	identical	all	BSD
LPeg	Parsing Machine	Lua	mixed	all	MIT
lug (https://github.com/jwtowner/lug)	Parsing Machine	C++17	mixed	all	MIT
Mouse	Recursive descent	Java	separate	Java Virtual Machine	Apache License 2.0
Narwhal	Packrat	C	mixed	POSIX, Microsoft Windows	BSD
Nearley (http://nearley.js.org/)	Earley	JavaScript	mixed	all	MIT
Nemerle.Peg	Recursive descent + Pratt	Nemerle	separate	all	BSD
neotoma	Packrat	Erlang	separate	all	MIT
NPEG	Recursive descent	C#	mixed	all	MIT
OMeta	Packrat (modified, partial memoization)	JavaScript, Squeak, Python	mixed	all	MIT
PackCC	Packrat (modified)	C	mixed	all	MIT
Packrat	Packrat	Scheme	mixed	all	MIT
Pappy	Packrat	Haskell	mixed	all	BSD
parboiled	Recursive descent	Java, Scala	mixed	Java Virtual Machine	Apache License 2.0
Lambda PEG	Recursive descent	Java	mixed	Java Virtual Machine	Apache License 2.0
parsepp	Recursive descent	C++	mixed	all	Public domain
Parsnip	Packrat	C++	mixed	Microsoft Windows	GNU GPL

Name	Parsing algorithm	Output languages	Grammar, code	Development platform	License
peg	Recursive descent	C	mixed	all	MIT
PEG.js (https://pegjs.org/)	Packrat (partial memoization)	JavaScript	mixed	all	MIT
peg-parser	PEG parser interpreter	Dylan	separate	all	
Pegasus	Recursive descent / Packrat (selectively)	C#	mixed	Microsoft Windows	MIT
pegc	Recursive descent	C	mixed	all	Public domain
pest (https://github.com/dragostis/pest)	Recursive descent	Rust	separate	all	MPL
PetitParser	Packrat	Smalltalk, Java, Dart	mixed	all	MIT
PEGTL (https://github.com/taocpp/PEGTL)	Recursive descent	C++11	mixed	all	MIT
PGE	Hybrid recursive descent / operator precedence[5]	Parrot bytecode	mixed	Parrot virtual machine	Artistic 2.0
PyPy rlib	Packrat	Python	mixed	all	MIT
pyPEG	PEG parser interpreter, Packrat	Python	mixed	all	GNU GPL
Rats! (https://cs.nyu.edu/rgrimm/xtc/rats-intro.html)	Packrat	Java	mixed	Java Virtual Machine	GNU LGPL
Spirit2	Recursive descent	C++	mixed	all	Boost
textX (https://github.com/textX/textX)	PEG parser interpreter, Packrat	Python (no generation, interpreted)	separate	all	MIT
Treetop	Recursive descent	Ruby	mixed	all	MIT
Yard	Recursive descent	C++	mixed	all	MIT or Public domain
Waxeye	Parsing Machine	C, Java, JavaScript, Python, Racket, Ruby	separate	all	MIT
PHP PEG	? (PEG Parser?)	PHP	mixed	all	BSD

General context-free, conjunctive or boolean languages

Name	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License
ACCENT	Earley	YACC variant	C	mixed	external	all	No	GNU GPL
APaGeD	GLR, LALR(1), LL(k)	?	D	mixed	generated	all	No	Artistic
Bison	LALR(1), LR(1), IELR(1), GLR	YACC	C, C++, Java, XML	mixed (except XML)	external	all	No	GNU GPL
DMS Software Reengineering Toolkit	GLR	?	Parlanse	mixed	generated	Microsoft Windows	No	Proprietary
DParser	Scannerless GLR	?	C	mixed	scannerless	POSIX	No	BSD
Dypgen	runtime-extensible GLR	?	OCaml	mixed	generated	all	No	CeCILL-B
E3	Earley	?	OCaml	mixed	external, or scannerless	all	No	?
Elkhound	GLR	?	C++, OCaml	mixed	external	all	No	BSD
eu.h8me.Parsing	GLR	?	N/A (state machine is runtime generated)	separate	external	.NET Framework	No	BSD
GDK	LALR(1), GLR	?	C, Lex, Haskell, HTML, Java, Object Pascal, Yacc	mixed	generated	POSIX	No	MIT
Happy	LALR, GLR	?	Haskell	mixed	external	all	No	BSD
Hime Parser Generator	GLR	?	C#, Java, Rust	separate	generated	.NET Framework, Java Virtual Machine	No	GNU LGPL
IronText Library	LALR(1), GLR	C#	C#	mixed	generated or external	.NET Framework	No	Apache License 2.0
Jison	LALR(1), LR(0), SLR(1)	YACC	JavaScript, C#, PHP	mixed	generated	all	No	MIT
Syntax	LALR(1), LR(0), SLR(1) CLR(1) LL(1)	JSON/YACC	JavaScript, Python, PHP, Ruby, C#, Rust, Java	mixed	generated	all	No	MIT
Laja	Scannerless, two phase	Laja	Java	separate	scannerless	all	No	GNU GPL
ModelCC	Earley	Annotated class model	Java	generated	generated	all	No	BSD
parglare (https://github.com/igordejanovic/parglare)	Scannerless LR/GLR	BNF-like	Python interpreted, automata run-time generated	mixed	scannerless	all	No	MIT
P1 (https://github.com/tomjridge/p1)	Combinators	BNF-like	OCaml	mixed	external, or scannerless	all	No	?
P3 (https://github.com/tomjridge/p3)	Earley/combinators	BNF-like	OCaml	mixed	external, or scannerless	all	No	?

Name	Parsing algorithm	Input grammar notation	Output languages	Grammar, code	Lexer	Development platform	IDE	License
P4 (https://github.com/tomjridge/p4)	Earley/combinators, infinitary CFGs	BNF–like	OCaml	mixed	external, or scannerless	all	No	?
Scannerless Boolean Parser	Scannerless GLR (Boolean grammars)	?	Haskell, Java	separate	scannerless	Java Virtual Machine	No	BSD
SDF/SGLR	Scannerless GLR	SDF	C, Java	separate	scannerless	all	Yes	BSD
SmaCC	GLR(1), LALR(1), LR(1)	?	Smalltalk	mixed	internal	all	Yes	MIT
SPARK	Earley	?	Python	mixed	external	all	No	MIT
Tom (https://www.cs.cmu.edu/Groups/AI/areas/nlp/parsing/tom/0.html)	GLR	?	C	generated	none	all	No	"No licensing or copyright restrictions"
UltraGram (http://ultragram.com)	LALR, LR, GLR	?	C++, C#, Java, Visual Basic .NET	separate	generated	Microsoft Windows	Yes	Proprietary
Wormhole	Pruning, LR, GLR, Scannerless GLR	?	C, Python	mixed	scannerless	Microsoft Windows	No	MIT
Whale Calf	General tabular, SLL(k), Linear normal form (Conjunctive grammars), LR, Binary normal form (Boolean grammars)	?	C++	separate	external	all	No	Proprietary
yaep	Earley	yacc like	C	mixed	external	all	No	LGPL
Zecc	Recursive Pattern Matching	Zecc/Zacc	Linkable Library	mixed	scannerless	macOS	Yes	Proprietary

Context-sensitive grammars

Name	Parsing algorithm	Input grammar notation	Boolean grammar capabilities	Development platform	License
LuZc[6][7]	delta chain	modular	Conjunctive, not complimentary	POSIX	proprietary
bnf2xml (http://sourceforge.net/p/bnf2xml/)	recursive descent (is a text filter output is xml)	simple bnf grammar (input matching), output is xml	?	beta, and not a full–fledged EBNF parser	GNU GPLv2

See also

- Compiler–compiler
- List of lexer generators

References

1. http://www.colm.net/open-source/ragel/
2. "Adaptive LL(*) Parsing: The Power of Dynamic Analysis" (http://www.antlr.org/papers/allstar-techreport.pdf) (PDF). Terence Parr. Retrieved 2016–04–03.
3. "Building parsers for the web with JavaCC & GWT (Part one)" (http://consoliii.blogspot.co.uk/2014/04/creating-gwt-compatible-parser-using.html). Chris Ainsley. Retrieved 2014–05–04.
4. http://www.slkpg2.com/license.txt

5. "Parrot: Grammar Engine" (https://parrot.github.com/html/docs/book/pct/ch04_pge.pod.html). The Parrot Foundation. 2011. "PGE rules provide the full power of recursive descent parsing and operator precedence parsing."
6. "LuZ: A context sensitive parser" (https://web.archive.org/web/20161017051112/http://qyxz.netau.net/). 2016–10–17. Archived from the original (http://qyxz.netau.net/) on 2016–10–17. Retrieved 2018–10–17.
7. "LuZc – A conjunctive context–sensitive parser" (http://luzc.zohosites.com/). *luzc.zohosites.com*. Retrieved 2018–10–17.

Notes

1. Bison 1.19 fork

External links

- The Catalog of Compiler Construction Tools (http://catalog.compilertools.net/lexparse.html)
- Open Source Parser Generators in Java (http://java–source.net/open–source/parser–generators)