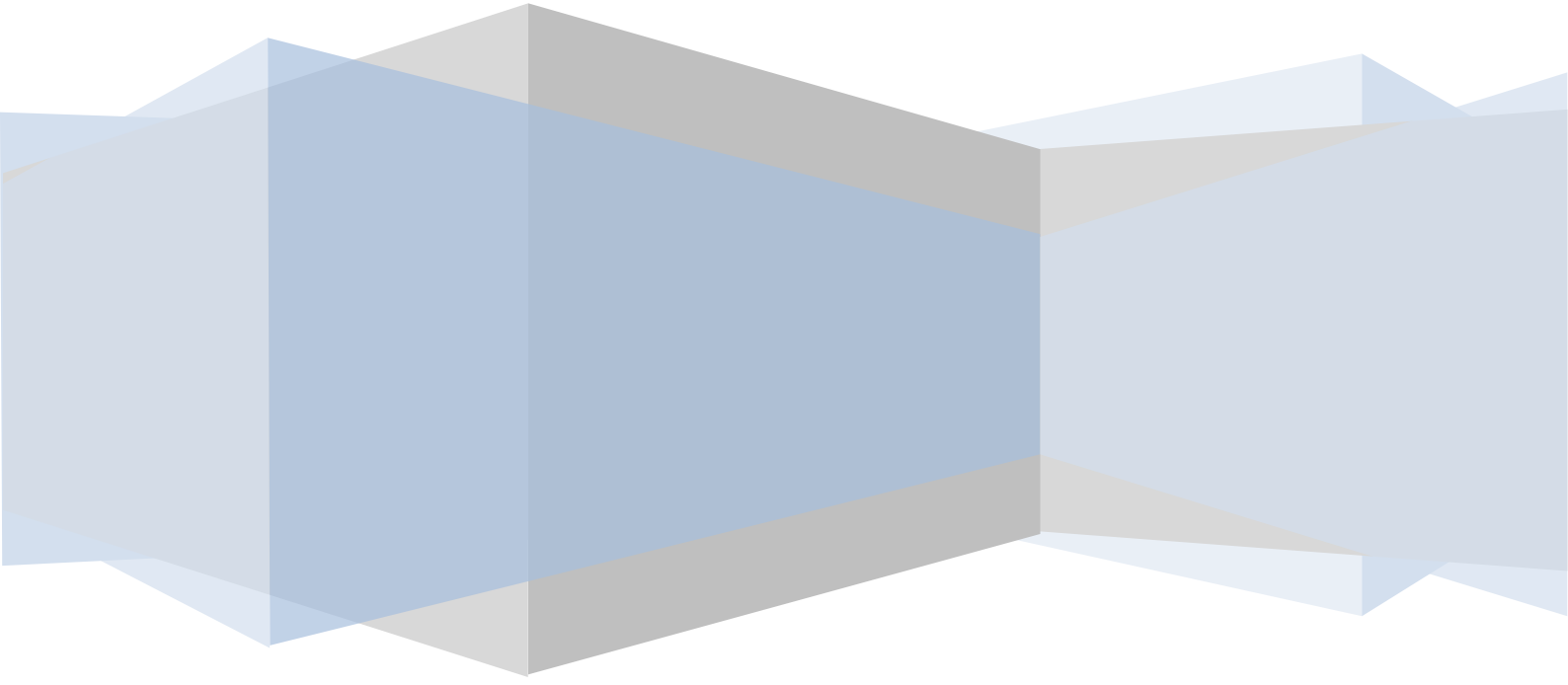
Procesadores del Lenguaje

Herramientas que generan analizadores

Comparativa de herramientas

Grupo 1



Contenido

Componentes del grupo

- ✓ Alina Gheorghita
- ✓ Cristina García
- ✓ Pilar Torralbo
- ✓ Tomás Restrepo
- ✓ Guillermo José Hernández
- ✓ Laura Reyero

Generadores de analizadores léxicos (scanner generators)

Lex

Tipo de analizador generado: léxico.

Código generado: C

Plataforma: Unix

Se puede usar con: Yacc

Licencia: software propietario

Otras versiones: Flex (software libre)

Entrada: una tabla de expresiones regulares y fragmentos correspondientes del programa

Lex ayuda a escribir programas cuyo control de flujo es dirigido por instancias de expresiones regulares en el flujo de entrada. Es muy adecuado para las transformaciones de tipo script y para la segmentación de la entrada en la preparación de una rutina de análisis.

El reconocimiento de las expresiones es realizado por un autómata finito determinista generado por Lex. Los fragmentos de programa escrito por el usuario se ejecutan en el orden en el que las expresiones regulares correspondientes aparecen en el flujo de entrada.

JLex

Tipo de analizador generado: léxico.

Código generado: Java

Plataforma: Multiplataforma.

Se puede usar con: CUP.

Licencia: Open Source y GPL-Compatible

Otras versiones: Jflex

Flex

Tipo de analizador generado: léxico.

Código generado: C, C++ (flex++)

Plataforma: Unix

Se puede usar con: Yacc, Bison.

Licencia: software libre (BSD license).

Complejidad del analizador generado: O(n) sobre la longitud de entrada.

Un analizador léxico Flex normalmente tiene complejidad O(n) sobre la longitud de la entrada. Es decir, se realiza un número constante de operaciones para cada símbolo de entrada. Esta constante es bastante baja (GCC genera 12 Instrucciones para el bucle de reconociemiento del ATN). Nótese que la constante es independiente de la longitud del token, de la longitud de la expresión regular y del tamaño del ATN.

Sin embargo, una característica opcional de Flex puede causar la generación de un analizador con complejidad no lineal: el uso de la macro REJECT en un analizador con el potencial de reconocimiento de tokens muy largos. En este caso, el programador le pide explícitamente a flex "volver atrás e intentarlo de nuevo" después de que haya acertado alguna entrada. Esto causará que el DFA use backtracking para encontrar otros estados finales. En teoría, la complejidad en tiempo es O(n+m^2)\ge O(m^2) donde m es la longitud más larga del token (esto vuelve a ser O(n) si los tokens son "pequeños" con respecto al tamaño de la entrada). La función de REJECT no está habilitada por defecto, y sus implicaciones en el rendimiento están ampliamente documentadas en el manual de Flex .

JFlex

Tipo de analizador generado: léxico.

Código generado: Java.

Plataforma: Multiplataforma (debe ejecutarse en una plataforma que soporte JRE/JDK 1.1 ó superior).

Se puede usar con: CUP, BYacc/J, ANTLR.

Licencia: Software libre. (GPL)

Entrada: Un archivo con extensión ".flex" que incluya las expresiones regulares del lenguaje que se quiere reconocer.

JFlex es un generador de analizador léxico, para Java, escrito en Java. También es una reescritura de la herramienta muy útil JLex.

JFlex está diseñado para trabajar junto con el generador de analizador sintáctico LALR, CUP de Scott Hudson, y la modificación de Java de Berkeley Yacc BYacc/J de Bob Jamison. También se puede utilizar junto con otros generadores de analizadores sintácticos como ANTLR o como una herramienta independiente.

La lista de características se puede encontrar en su página web:

http://jflex.de/features.html

Ragel

Tipo de analizador generado: léxico.

Código generado: Java, C, C++, Objective-C, D.

Plataforma: Multiplataforma.

Se puede usar con:

Licencia: Software libre. (Licencia GNU)

Entrada: Una tabla con expresiones regulares y fragmentos de código

Ragel es un compilador de máquina de estados finitos con soporte de salida para C, C #, Objective-C, D, Java, Go y el código fuente de Ruby. Es compatible con la generación de una tabla o control de flujo impulsado por las máquinas de estado de las expresiones regulares y / o gráficos de estado y también se puede construir analizadores léxicos a través del método “longest-match”.

Una característica única de Ragel es que las acciones del usuario pueden estar asociadas con las transiciones de una máquina de estados arbitraria usando los operadores que están integrados en las expresiones regulares. Ragel también es compatible con la visualización de la máquina generada a través de graphviz.

Comparativa de generadores léxicos

Nombre	Autómata reconocido	Entrada	Lenguaje de salida	Licencia
Flex	AFD(table driven)	Mixto	C	BSD
Lex	AFD(table driven)	Mixto	C	Propietario, CDDL
JLex	AFD	Mixto	Java	BSD-like
JFlex	AFD	Mixto	Java	GNU GPL
Ragel	AFD	Mixto	C,C++,D, Java, Objective-C, Ruby	GNU GPL

Generadores de analizadores léxicos y sintácticos (scanner and parser generators)

Antlr

Analizadores generados: léxico y sintáctico.

Código generado: C, C#, Java, JavaScript, Ada95, Python, Perl

Tipo de analizador sintáctico: Descendente recursivo, LL(k).

Plataforma: Multiplataforma.

Se puede usar con: JFlex.

Licencia: Software libre. (Licencia BSD)

Entrada: Para introducir una gramática en ANTLR, se debe utilizar la notación EBNF y un conjunto de construcciones auxiliares. Permite especificar cada analizador (léxico, sintáctico, semántico) en un fuente independiente o en un único fuente. En este primer ejemplo optaremos por utilizar un solo fuente.

CookCC

Analizadores generados: léxico y sintáctico.

Código generado: Java, Plain, XML, Yacc.

Tipo de analizador sintáctico: LALR(1).

Plataforma: Multiplataforma.

Licencia: Software libre. (Licencia BSD)

Entrada: Se puede especificar el lexer/parser directamente en código Java. También puede ser un fichero XML que contiene una sección para el lexer y otra para el parser (solamente una de ellas es necesaria). También se permiten ficheros de entrada tradicionales de yacc/bison con algunas restricciones.

Grammatica

Analizadores generados: sintáctico.

Código generado: C#, Java

Tipo de analizador sintáctico: Descendente, LL(k).

Plataforma: Multiplataforma (debe ejecutarse en una plataforma que soporte JRE/JDK 1.1 ó superior).

Se puede usar con: JFlex

Licencia: Software libre (licencia LGPL).

Entrada: Gramática basada en una notación similar a BNF.

Mejora herramientas similares (como Yacc y ANTLR) por la creación de código fuente bien comentado y legible, por tener la recuperación automática de errores y mensajes de error detallados, así como soporte para probar y depurar las gramáticas sin generar código fuente

JavaCC

Analizadores generados: léxico y sintáctico.

Código generado: Java.

Tipo de analizador sintáctico: Descendente recursivo, LL(k).

Plataforma: Multiplataforma.

Se puede usar con:

Licencia: Software libre. (Licencia BSD)

Entrada: Para introducir una gramática se debe utilizar la notación EBNF.

YaCC

Analizadores generados: sintáctico.

Código generado: C, C++.

Tipo de analizador sintáctico: Ascendente, LALR(1).

Plataforma: Unix.

Se puede usar con: Requiere un analizador léxico como Lex, Flex.

Licencia: Open Source.

Entrada: Gramática basada en una notación similar a BNF.

Comparativa de generadores léxicos y sintácticos

Nombre	Algoritmo	Notacion Gramatical	Entrada	Salida	Lexico
ANTLR	Descendente	EBNF	Mixto	C, C++, Java, Ada, Perl,Python, Ruby, otros	generado
Yacc	LALR(1)	YACC	Mixto	C	Externo
JavaCC	Descendente (k)	EBNF	Mixto	Java	generado
JFLAP	Descendente(1), LALR(1)	¿?	¿?	Java	¿?
Grammatica	Descendente(k)	dialecto BNF	Por separado	C#; Java	generado
CookCC	LALR(1)	Java, XML, Yacc	Mixto	Java	generado

Conclusiones

Tras el estudio y la comparativa de las diferentes herramientas utilizadas para la generación de analizadores léxicos y sintácticos, nuestra herramienta elegida por el momento como mejor opción es JavaCC.

No nos parecen buenas opciones las herramientas Lex, Flex, Yacc, dado que generan código en C, C++ y dicho lenguaje no está muy dominado por todos los miembros del grupo.

JFlex, JLex y Ragel son descartadas dado que estas herramientas generan solamente analizadores léxicos y por mayor comodidad creemos que es más conveniente utilizar una herramienta que genere también el analizador sintáctico. Lo mismo se aplica a Grammatica ya que genera solamente analizadores sintácticos.

Al no tener suficiente información sobre sus características y funcionamiento, descartamos la aplicación JFlap.

CookCC no incluye tanta documentación como JavaCC y ANTLR por lo cual sería más complicado aprender al usarla.

Se supone que es la herramienta más utilizada en Java. Los propietarios estiman en cientos de miles el número de descargas de la herramienta y los foros de discusión congregan a miles de usuarios interesados en JavaCC.

Se trata de una herramienta desarrollada en Java, lenguaje de programación ampliamente conocido por todos los miembros del grupo. Además, dispone de una amplia documentación, facilitando el uso de la misma.

JavaCC integra en una misma herramienta al analizador lexicográfico y al sintáctico, y el código que genera es independiente de cualquier biblioteca externa, lo que le confiere independencia respecto al entorno. Además, ofrece muchas opciones diferentes para personalizar su comportamiento y el comportamiento de los analizadores generados. Estos aspectos nos interesan ya que no tenemos un objetivo claro, al no conocer la gramática que vamos a utilizar. Dicha característica no se contempla en muchos de los generadores ?

Genera analizadores descendentes, permitiendo el uso de gramáticas de propósito general. Por defecto analiza gramáticas de tipo LL(1), pero permite fijar un tamaño de ventana de tokens mayor (para analizar gramáticas LL(k)) e incluso utilizar un tamaño adaptativo.

Otra de sus ventajas es que las especificaciones léxica (basada en expresiones regulares) y sintáctica (basada en el formato BNF) de la gramática que se va a analizar se incluyen en un mismo fichero. De esta forma la gramática puede ser leída y mantenida más fácilmente. Además, incluye la herramienta JJDoc que convierte los archivos de la gramática en archivos de documentación.

También incluye la herramienta JJTree, un preprocesador para el desarrollo de árboles con características muy potentes.

Es altamente eficiente, lo que lo hace apto para entornos profesionales y lo ha convertido en uno de los metacompiladores más extendidos (quizás el que más, por encima de JFlex/Cup).

De entre los generadores de analizadores sintácticos descendentes, JavaCC es uno de los que poseen mejor gestión de errores. Los analizadores generados por JavaCC son capaces, mediante las excepciones, de localizar exactamente la ubicación de los errores, proporcionando información diagnóstica completa.

Glosario

ATN: Augmented Transaction Network

LALR: Look-Ahead Left to Right Parser

LL: Left to right, Leftmost derivation

EBNF: Extended Backus–Naur Form

BNF: Backus-Naur form

DFA: Deterministic-finite automaton

CDDL = Licencia común de Desarrollo y Distribución

Mixto = Tabla de ER's y fragmentos de código

LGPL = Licencia GNU GPL con una excepción de linkado

Bibliografía

- http://www.antlr.org/
- http://www.escet.urjc.es/~procesal/analizadores.html
- http://es.wikipedia.org/wiki/JavaCC
- http://javacc.java.net/
- http://www.lcc.uma.es/~galvez/ftp/libros/Compiladores.pdf
- http://en.wikipedia.org/wiki/Comparison_of_parser_generators
- http://www.cs.princeton.edu/~appel/modern/java/JLex/
- http://dinosaur.compilertools.net/
- http://www.gnu.org/software/flex/
- http://jflex.de/