UNA HERRAMIENTA DIDÁCTICA PARA LA GENERACIÓN Y UTILIZACIÓN DE ANALIZADORES LR DIRECTOS E INVERSOS

José Fortes Gálvez

Departamento de Informática y Sistemas
Universidad de Las Palmas de Gran Canaria
E-mail: jfortes@dis.ulpgc.es

Resumen: Se presenta una herramienta didáctica desarrollada por el autor, que facilita al alumno trabajar con varios métodos de análisis sintáctico de las familiar LR directo e inverso. Es posible además generar de manera automática sentencias aleatorias no triviales.

1.- INTRODUCCIÓN.

En asignaturas donde se introduce al alumno en lenguajes formales, y en particular en el análisis sintáctico, se suele poner énfasis en el análisis LR, por razones evidentes de potencia de análisis (familia de gramáticas aceptadas) y de utilidad práctica (disponibilidad de herramientas como yacc). Desafortunadamente, los métodos de dicha familia, y en general, la mayoría de los métodos de análisis sintáctico, con la prácticamente única excepción del descenso recursivo, presentan el inconveniente de ser muy laboriosos de desarrollar a mano. Si no se dispone de una buena herramienta, el alumno puede perder mucho tiempo en desarrollar un ejemplo, muchas veces obteniendo un resultado erróneo, por lo minucioso y prolijo de la construcción, lo que induce a frustración y rechazo. En todo caso es impensable construir a mano dichos generadores para gramáticas de un cierto tamaño real. Y, por último, tal dificultad impide al alumno experimentar con sus diseños y analizar las consecuencias del mismo.

El autor ha desarrollado, durante su trabajo de tesis doctoral en el que presenta un enfoque nuevo al análisis de potencia LR, denominado discriminante inverso (siglas DR en inglés), una herramienta que permite:

1. Generar analizadores LR(0), SLR(1), LALR(1), LR(1) y otros de la familia DR. Se pueden visualizar los items, acciones, y tabla de análisis, tanto en formato texto como en formato LaTeX para ser incluidos en la documentación del trabajo.

2. Generar de forma automática sentencias para gramáticas, de forma aleatoria no-trivial.

3. Seguir paso a paso el análisis sintáctico de una entrada (escrita manualmente o generada automáticamente), en cualquiera de los métodos anteriores.

La herramienta ha sido desarrollada en Pascal estándar (con opciones de compilación para varios compiladores de amplio uso) y está disponible libremente (licencia GPL).

En este artículo presentamos brevemente las posibilidades de la herramienta, así como una sesión demostrativa del uso de la misma.

Es de destacar que, además de en nuestra universidad, dicha herramienta se usa desde hace algunos años en los estudios de informática de la Universidad de Niza Sophia Antipolis (Francia), y está siendo objeto de reescritura en Java con el objeto de mejorar la accesibilidad y el interface con el usuario.

Por otro lado, se está reimplementando en C++ de forma eficiente para ser presentada como alternativa para la construcción realista de analizadores de gramáticas de lenguajes de programación, donde además se incorporarán extensiones no canónicas al método discriminante inverso que permitan analizar gramáticas no-LR.

2.- UTILIZACIÓN DE LA HERRAMIENTA

La herramienta se ejecuta desde el prompt del sistema operativo, con el nombre de aut, seguido de una opción que indica el modo de funcionamiento:

- aut gr Genera analizador *discriminante inverso*.
- aut gd Genera analizador de la familia LR directa (Knuth).
- aut s *L* Genera una o más sentencias aleatorias de longitud *L*, si ésta no se especifica, la longitud es asimismo aleatoria.
- aut pr Analiza una secuencia de entrada utilizando el método discriminante inverso.
- aut gr Igualmente para la familia LR directa.

Normalmente se invoca en primer lugar un generador (aut gd), luego se utiliza opcionalmente el generador de sentencias, y por último, se invoca el analizador correspondiente.

a) Ficheros Utilizados por la Herramienta.

La herramienta utiliza los tres ficheros siguientes. Se muestran los nombres por defecto en el directorio actual, si bien se pueden indicar otros nombres o directorios al invocarla, mediante el uso de los especificadores **–gf, -vf, -pf** y **–sf.**

g.aut Fichero de texto plano que contiene la especificación de la gramática en notación BNF. Se verifica que la gramática es reducida, y que no contiene reglas repetidas. Constituye la entrada a los generadores.

v.aut, p.aut Ficheros de vocabulario y de tabla de análisis, respectivamente, en formato interno. Son producidas por los generadores y leídas por los analizadores.

s.aut Fichero de texto conteniendo la secuencia de símbolos a ser analizada. Puede ser escrita manualmente, o bien generada de forma automática con **aut.s**.

No es necesario el uso de analizador léxico. El alumno puede escribir directamente los nombres de los símbolos terminales, lo que hace que su uso sea más inmediato e intuitivo.

Algunas opciones de línea de comando.

Aquí presentamos algunas de las opciones más interesantes de la línea de comandos. El resto pueden verse más adelante.

-n Especifica el número de sentencias a generar automáticamente.

+r S Por defecto, cada vez que se invoca el generador de sentencias, éste produce aleatoriamente una nueva sentencia. A veces es importante la repetibilidad, y entonces podemos usar esta opción donde indicamos un valor de semilla S que nos permitirá producir posteriormente la misma sentencia aleatoria. Por defecto, S=0.

+t Utilizamos normalmente una técnica propia que denominamos de longitudes típicas de no-terminales para la generación de sentencias de forma no trivial. Si utilizamos esta opción inhibidora, se toman las distintas reglas al azar; lo que usualmente da unos resultados muy pobres y distantes de las sentencias de uso normal.

+v Inhibe el modo "verbose" para la invocación en scripts.

-d Prefijo de las muy variadas opciones de visualización. Véase el listado completo más adelante.

3.- UNA SESIÓN EJEMPLO

Cuando se ejecuta el programa sin argumento alguno, se obtiene la siguiente ayuda, así como cuando hay algún error en la línea de comando, se le indica al usuario si desea que se visualice esta ayuda.

```
Host$ aut

Direct and reverse LR tool (v10.0.5 28-1-1999)

Usage:

  aut g r|d[#] {-|+c -d{ailbfgsv} +f -g[#] -i +|-l -r[a] +s -t. +v -x file}
  aut s[#] {-# -d{bdfgvem} +e -g[#] +n[#] +v file}
  aut p r|d {-d{psu} -g[#] +v file}


 Functioning modes:
  g       parser Generation (reads gf; writes vf, pf)
  s       random Sentence generation (reads gf; writes sf), with random or
          given (#) length (quite approximately)
  p       Parse (reads vf, pf, sf) with mostly manual error recovery


 Automaton classes:
  r       Reverse discriminating
  d       Direct (classical)
  #       lookahead length: 0 or 1 (default)
```

```
Files:
 -gf path  BNF Grammar File (default g.aut) with rules like:
              proc_list = "proc" heading ";" proc_list | .
 -sf path  Sentence text File (default s.aut) to parse; spaces separate
           terminals.
 -pf path  Parsing table File (default p.aut) in internal format.
 -vf path  Vocabulary File (default v.aut) in internal format.


Options (some do not produce a usable parser):
 -#      Number of sentences
 -c      resolve (reverse) Conflicts (by asking)
 +c      Continue (reverse) state construction beyond conflicts
 -d      Display options:
   a      Automaton states; suboptions:
    i       state situations (also called Items)
    l       List actions for (stack symbol, lookahead) pairs
   b       intput BNF grammar
   d       sentential forms during Derivation
   f       First and follow sets
   g       internal Grammar
   p       (action counts and) stack windows during Parsing
   s       Statistics
   u       strict parsing User time
   v       Vocabulary; suboptions:
    e       estimated typical lEnghts of phrases
    m       Minimum and maximum lenghts of phrases
 +e      disable use of typical lEngths of phrases
 +f      disable Full right-parts computation in situations
 -g[#]   debuGging [option; default 0]
 -i      ask (reverse) Initial state
 +l      build (direct) LALR(1) automaton
 -l      signal (direct) LALR(1) conflicts in LR(1) automaton
 +n[#]   seed [Number; default 0]: disable randomize
 -r[a]   full (reverse) Recognizer [consider Actions]
 +s      build (direct) SLR(1) automaton
 -t.     produce parsing table in plain Text using '.' for empty entries
 +v      disable Verbose output
 -x      produce output and parsing table in LaTeX notation

Author address: <Fortes Gálvez, José> jfortes@dis.ulpgc.es.
```

Veamos en primer lugar la gramática:

```
Host$ cat g.aut
E = E "+" T | T .
T = T "*" F | F .
F = "(" E ")" | "a" .
```

Ejecutemos el generador LR(1):

```
Host$ aut gd -dais

Direct and reverse LR tool (v10.0.5 28-1-1999)

Invoked:  aut gd -dais

Input grammar statistics:
 Number of rules = 6
 Average length of rightparts = 2.00
 Average number of terminal symbols per rightpart = 0.83
 Vocabulary:
  Number of terminal symbols = 5
  Number of nonterminal symbols = 3

Direct (classical) LR(1) automaton


State 1
S' -> ^ . E $ , $
E -> . E + T , $
E -> . T , $
E -> . E + T , +
```

```
E -> . T , +
T -> . T * F , $
T -> . F , $
T -> . T * F , +
T -> . F , +
T -> . T * F , *
T -> . F , *
F -> . ( E ) , $
F -> . a , $
F -> . ( E ) , +
F -> . a , +
F -> . ( E ) , *
F -> . a , *

NS(E) = Go To 2
NS(T) = Go To 3
NS(F) = Go To 4
NS(() = Shift 5
NS(a) = Shift 6


State 2
S' -> ^ E . $ , $
E -> E . + T , $
E -> E . + T , +

NS($) = Shift 7
NS(+) = Shift 8


State 3
E -> T . , $
E -> T . , +
T -> T . * F , $
T -> T . * F , +
T -> T . * F , *

NS($) = Reduce 3
NS(+) = Reduce 3
NS(*) = Shift 9

.
.
(Estados eliminados)
.
.

State 22
T -> T * F . , )
T -> T * F . , +
T -> T * F . , *

NS()) = Reduce 4
NS(+) = Reduce 4
NS(*) = Reduce 4


State 23
F -> ( E ) . , )
F -> ( E ) . , +
F -> ( E ) . , *

NS()) = Reduce 6
NS(+) = Reduce 6
NS(*) = Reduce 6


End of generation.


Automaton statistics:
 Number of states = 23
 Number of transitions & actions = 72
```

Generemos automáticamente una sentencia sobre s.aut.

```
Host$ aut s15 -dd

Direct and reverse LR tool (v10.0.5 28-1-1999)
```

```
Invoked:  aut s15 -dd
<E>:15
<T>:15
<T>:10 * <F>:3
<T>:6 * <F>:6 * a
<T>:2 * <F>:2 * ( <E>:4 ) * a
<F>:2 * a * ( <E>:2 + <T>:2 ) * a
a * a * ( <T>:3 + <F>:4 ) * a
a * a * ( <T>:1 * <F>:1 + ( <E>:2 ) ) * a
a * a * ( <F>:1 * a + ( <T>:1 ) ) * a
a * a * ( a * a + ( <F>:1 ) ) * a
a * a * ( a * a + ( a ) ) * a
a * a * ( a * a + ( a ) ) * a
```

Obsérvese que se señalan los tamaños de las frases que se van a generar, de forma descendente, para cada no-terminal. Por último, se utilizaría el analizador para esta entrada, que nos confirmaría que corresponde a una sentencia y que es analizada correctamente:

```
Host$ aut pd
Direct and reverse LR tool (v10.0.5 28-1-1999)
Invoked:  aut pd
Successful end of parsing.
```

Y, por último, pidamos que genere en forma de fichero LaTeX la tabla de análisis LALR(1) de esta gramática ejemplo.

```
Host$ aut gd +l -x +v
Host$ cat paut.tex
%
% Direct (classical) LR parsing table in LaTeX format
% VersionNumber= 10.0.5 VersionDate= 28-1-1999
%
% GrammarFileName = g.aut
%
%
\begin{figure}\center
{\small\begin{tabular}{|c|cccccc|ccc|}
\hline
  & $\eof $       & $+$   & $*$   & $($  & $)$   & $a$   & $E$   & $T$ & $F$ \\
\hline
 s1     &       &       &       & s5    &       & s6    & 2     & 3     & 4 \\
 s2     & s7    & s8    &       &       &       &       &       &       & \\
 s3     & r3    & r3    & s9    &       & r3    &       &       &       & \\
 s4     & r5    & r5    & r5    &       & r5    &       &       &       & \\
 s5     &       &       &       & s5    &       & s6    & 10    & 3     & 4 \\
 s6     & r7    & r7    & r7    &       & r7    &       &       &       & \\
 s7     & r1    &       &       &       &       &       &       &       & \\
 s8     &       &       &       & s5    &       & s6    &       & 11    & 4 \\
 s9     &       &       &       & s5    &       & s6    &       &       & 12 \\
 s10    &       & s8    &       &       & s13   &       &       &       & \\
 s11    & r2    & r2    & s9    &       & r2    &       &       &       & \\
 s12    & r4    & r4    & r4    &       & r4    &       &       &       & \\
 s13    & r6    & r6    & r6    &       & r6    &       &       &       & \\
\hline \end{tabular}}
```

```
\caption{\label{g}%
%caption goes here
}\end{figure}
%
```