Algoritmo general y LR(0) [1]

Lourdes González Huesca
Compiladores
Facultad de Ciencias UNAM

Input: Cadena de entrada w y la tabla LR con las funciones ACTION y GOTO.

Output: Si la cadena w está en el lenguaje de la gramática, entonces se devuelven las reducciones del parsing bottom-up; sino se devuelve un error.
El parser inicia con el estado inicial s_0 en la pila y $w\#$ como la cadena de entrada (input), v$\in (\Sigma \cup \Gamma)^\star$.

```
let a be the first symbol of w#;
while(1)  /* repeat forever */
{
     let s be the state on top of the stack;
     if ( ACTION[s; a] = shift t )
     {
         push t onto the stack;
         let a be the next input symbol;
     } else if ( ACTION[s; a] = reduce A→v )
            {
              pop v symbols off the stack;
              let state t now be on top of the stack;
              push GOTO[t; A] onto the stack;
              output the production A→v;
            } else if ( ACTION[s; a] = accept ) break;
                   /* parsing is done */
            else call error-recovery routine;
}
```

Existen diferentes tipos de analizadores **LR** en donde el manejo del autómata de pila está descrito en el algoritmo anterior, la diferencia entre ellos es la construcción de la tabla de parsing. La tabla de parsing LR tiene dos partes: las acciones y los saltos. Para construirla se usa una máquina de decisión donde los estados y las transiciones se calculan con las funciones de cerradura (estados) y goto (transiciones).
Un **item** es una producción de la gramática con una bandera o marca ($\bullet$) en alguna posición del cuerpo o parte derecha para indicar la coincidencia en el proceso de análisis.

- item tipo kernel: es el item de la producción inicial $S \to \bullet E$ (donde E es el símbolo inicial de la gramática original) o cualquier item que **no** tenga un punto más a la izquierda
- item tipo no-kernel: es cualquier item con el punto más a la izquierda excepto el item inicial $S \to \bullet E$

Máquina para LR(0)

Los conjuntos de items **canónicos** se calculan con la función auxiliar de cerradura o CLOSURE:

```
let J = I;
repeat{
   for each item A → α • Bβ in J
       for each grammar production B → γ
          if (B → •γ not in J)
          then add B → •γ
}
until no more items are added to J
```

Dado un conjunto de items I y un símbolo cualquiera de la gramática X, la función GOTO del estado I mediante X es la cerradura de todos los items $A \to \alpha X \bullet \beta$ tal que $A \to \alpha \bullet X\beta$ está en I.

El algoritmo que construye los conjuntos canónicos de items C para una gramática extendida con el símbolo S es:

```
C = CLOSURE({S → • E});
repeat{
   for each set of items I in C
       for each grammar symbol X
          if (GOTO(I,X) is not empty and not in C)
          then add GOTO(I,X) to C
}
until no new sets are added to C
```

[1] Resumen tomado del libro "Compilers, Principles, Techniques and Tools", Aho A. et al., Capítulo 4.