Teoría de Lenguajes
Segunda parte
Algoritmos de Parsing de Earley

Verónica Becher

Segundo cuatrimestre 2020

Bibliografía para esta clase:
A. V. Aho, J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. 1 , Parsing. Prentice Hall, 1972.

`https://www-2.dc.uba.ar/staff/becher/Aho-Ullman-Parsing-V1.pdf` Capítulo 4.2

Algoritmo de parsing de Earley

Dada una gramática libre de contexto el algoritmo de parsing de Earley con una entrada de longituf n hace una cantidad de operaciones en el orden de n^3 y usa espacio n^2.

En caso de que la gramática sea no-ambigua, la cantidad de operaciones es del orden n^2.

Para la mayoría de las gramáticas que definen lenguajes de programación el algoritmo de Earley se puede modificar para obtener cantidad de operaciones y espacio lineales respecto a la longitud de la entrada.

Es un método bottom-up que encuentra la derivación más a la derecha.

Algoritmo de Parsing de Earley

Sea $G = (N, T, P, S)$ una GLC y sea $w = a_1 a_2 \dots a_n$ una cadena de input en T^*.

Definición (Item de Earley)**.** *Decimos que* $[A \to X_1 X_2 \dots X_k \cdot X_{k+1} \dots X_m, i]$ *es un item para* w *si* $A \to X_1 X_2 \dots X_m$ *es una producción en* P *y* $0 \leq i \leq n$.

Si la producción es $A \to \lambda$, el item es $[A \to \cdot, i]$.

Algoritmo de Parsing de Earley

Dada $G = (N, T, P, S)$ y una cadena de entrada $a_1 \dots a_n$ en T^* construimos las listas de items $\ell_1, \ell_2, \dots, \ell_n$ tal que

$$[A \to \alpha \cdot \beta, i] \in \ell_j$$

si y solo si para algún γ y δ

$$S \overset{*}{\Rightarrow} \gamma A \delta, \quad \gamma \overset{*}{\Rightarrow} a_1 \dots a_i \quad \text{y} \quad \alpha \overset{*}{\Rightarrow} a_{i+1} \dots a_j,$$

Luego, $w \in L(G)$ si y solo si hay α tal que $[S \to \alpha \cdot, 0] \in \ell_n$.

Algoritmo de Earley

Algorithm 4.5 Aho Ullman vol 1

Input. Gramática GLC $G = (N, T, P, S)$ y cadena $w = a_1 \dots a_n$ en T^*.

Output. Las listas $\ell_0, \ell_1, \dots, \ell_n$.

Paso 1 Si $S \to \alpha$ es una producción en P agregar a ℓ_0, $[S \to \cdot \alpha, 0]$.

Paso 2 Si $[A \to \alpha \cdot B\beta, 0] \in \ell_0$ y $[B \to \gamma \cdot, 0] \in \ell_0$ (en particular γ puede ser λ) entonces agregar a ℓ_0, $[A \to \alpha B \cdot \beta, 0]$.

Paso 3 Si $[A \to \alpha \cdot B\beta, 0] \in \ell_0$, agregar a ℓ_0 para toda $B \to \gamma$ en P, (si es que aún no está), $[B \to \cdot \gamma, 0]$.

Paso 4 Supongamos que ya hemos construido $\ell_0, \ell_1, \dots \ell_{j-1}$. Si $[B \to \alpha \cdot aB\beta, i]$ en ℓ_{j-1} tal que $a = a_j$, agregar a ℓ_j $[B \to \alpha a \cdot \beta, i]$

Paso 5 Si $[B \to \alpha \cdot A\beta, k] \in \ell_i$ y $[A \to \gamma \cdot, i] \in \ell_j$, agregar a ℓ_j $[B \to \alpha A \cdot \beta, k]$.

Paso 6 Si $[A \to \alpha \cdot B\beta, i] \in \ell_j$, agregar a ℓ_j, para todo $B \to \gamma$ en P, $[B \to \cdot \gamma, j]$.

Notemos que la consideración de un item con un terminal a la derecha del $\cdot$ no produce nuevos items en los pasos $2, 3, 5$ y 6.

Ejemplo

Sea G con las producciones
1.$E \Rightarrow T + E$
2.$E \Rightarrow T$
3.$T \Rightarrow F * T$
4.$T \Rightarrow F$
5.$F \Rightarrow (E)$
6.$F \Rightarrow a$

Input $(a + a) * a$, de longitud 7
Damos $\ell_0, \dots \ell_7$.
Dado que $E \to T\cdot, 0] \in \ell_7$,
$(a + a) * a \in L(G)$

I_0
$[E \longrightarrow \cdot\ T + E, 0]$
$[E \longrightarrow \cdot\ T, 0]$
$[T \longrightarrow \cdot\ F * T, 0]$
$[T \longrightarrow \cdot\ F, 0]$
$[F \longrightarrow \cdot\ (E), 0]$
$[F \longrightarrow \cdot\ a, 0]$

I_1
$[F \longrightarrow (\cdot\ E), 0]$
$[E \longrightarrow \cdot\ T + E, 1]$
$[E \longrightarrow \cdot\ T, 1]$
$[T \longrightarrow \cdot\ F * T, 1]$
$[T \longrightarrow \cdot\ F, 1]$
$[F \longrightarrow \cdot\ (E), 1]$
$[F \longrightarrow \cdot\ a, 1]$

I_2
$[F \longrightarrow a\ \cdot, 1]$
$[T \longrightarrow F\ \cdot * T, 1]$
$[T \longrightarrow F\ \cdot, 1]$
$[E \longrightarrow T\ \cdot + E, 1]$
$[E \longrightarrow T\ \cdot, 1]$
$[F \longrightarrow (E\ \cdot), 0]$

I_3
$[E \longrightarrow T + \cdot\ E, 1]$
$[E \longrightarrow \cdot\ T + E, 3]$
$[E \longrightarrow \cdot\ T, 3]$
$[T \longrightarrow \cdot\ F * T, 3]$
$[T \longrightarrow \cdot\ F, 3]$
$[F \longrightarrow \cdot\ (E), 3]$
$[F \longrightarrow \cdot\ a, 3]$

I_4
$[F \longrightarrow a\ \cdot, 3]$
$[T \longrightarrow F\ \cdot * T, 3]$
$[T \longrightarrow F\ \cdot, 3]$
$[E \longrightarrow T\ \cdot + E, 3]$
$[E \longrightarrow T\ \cdot, 3]$
$[E \longrightarrow T + E\ \cdot, 1]$
$[F \longrightarrow (E\ \cdot), 0]$

I_5
$[F \longrightarrow (E)\ \cdot, 0]$
$[T \longrightarrow F\ \cdot * T, 0]$
$[T \longrightarrow F\ \cdot, 0]$
$[E \longrightarrow T\ \cdot + E, 0]$
$[E \longrightarrow T\ \cdot, 0]$

I_6
$[T \longrightarrow F * \cdot\ T, 0]$
$[T \longrightarrow \cdot\ F * T, 6]$
$[T \longrightarrow \cdot\ F, 6]$
$[F \longrightarrow \cdot\ (E), 6]$
$[F \longrightarrow \cdot\ a, 6]$

I_7
$[F \longrightarrow a\ \cdot, 6]$
$[T \longrightarrow F\ \cdot * T, 6]$
$[T \longrightarrow F\ \cdot, 6]$
$[T \longrightarrow F * T\ \cdot, 0]$
$[E \longrightarrow T\ \cdot + E, 0]$
$[E \longrightarrow T\ \cdot, 0]$

Correctitud de algoritmo de Earley

Teorema 1 (Theorem 4.9 Aho Ullman vol 1)**.** *Dada $G = (N, T, P, S)$ libre de contexto, $[S \to \alpha\cdot, 0] \in \ell_n$ si y solo si $S \to \alpha$ está en P y $\alpha \overset{*}{\underset{R}{\Rightarrow}} a_1 \dots a_n$.*

Complejidad de algoritmo de Earley

Teorema 2 (Theorem 4.10 en Aho Ullman vol 1)**.** *Si la gramática es no-ambigua, para una entrada de longitud n el algoritmo de Earley construye las listas $\ell_1, \dots, \ell_n$ en una cantidad de operaciones del orden n^2*

Si la gramática es ambigua esta cantidad es del orden n^3.

Cada lista ℓ_j tiene a lo sumo $(j + 1)k$ items, donde k es una constante. Mantenemos una lista encadenada entre todos ellos y una table indicando si una producción ya está o no.

Mantenemos además otra lista encadenada entre los items que tiene el mismo símbolo a la derecha del $\cdot$. Esto sirve para los pasos donde se requiere buscar los items que tienen un determinado símbolo a derecha del $\cdot$.

Cuando la gramática es no ambigua cada item se considera una sola vez. Y hay una cantidad constante c de operaciones a realizar para cada item. Los puntos 4,5 y 6 del algoritmo aseguran que la cantidad de operaciones es a lo sumo

$$c\sum_{j=0}^{n}(j+1)k = ck(n+1)(n+2)/2 = c'n^2$$

Los pasos 1,2,3 construyen ℓ_0 y la cantidad de operaciones está acotada por la cantidad de producciones Concluimos que algoritmo realiza en el orden de n^2 operaciones. En caso de que la gramática sea ambigua son n^3.

Derivacion a derecha de Earley

Derivación a derecha del Algoritmo de Earley, Algorithm 4.6 Aho Ullman vol 1

Input. Una GLC $G = (N, T, P, S)$ sin ciclos, con las producciones numeradas $1, \ldots, p$, una cadena de input $w = a_1 \ldots a_n$ y las listas $\ell_0, \ldots \ell_n$

Output. Derivación a derecha de w, o "error".

Método. Si $[S \to \alpha\cdot, 0]$ en ℓ_n entonces w está en $L(G)$. Fijar variable global $\pi = \lambda$. Ejecutar $R([S \to \alpha\cdot, 0], n)$ Emitir secuencia de producciones π.

Rutina $R([A \to X_1 \ldots X_m\cdot, i], j)$

$\pi = h\pi$, donde h es el número de producción $A \to X_1 \ldots X_m$ $k = m$ $j' = j$

Mientras $(k > 0)$

Si $X_k \in T$ entonces

$k = k - 1$

$j' = j' - 1$

Sino (es decir, $X_k \in N$)

Encontrar un item $[X_k \to \gamma\cdot, i']$ en $\ell_{j'}$ para algún i' tal que $[A \to X_1X_2 \ldots X_{k-1} \cdot X_k \ldots X_m, i] \in \ell_{i'}$.

Ejecuta $R([X_k \to \gamma\cdot, i'], j')$.

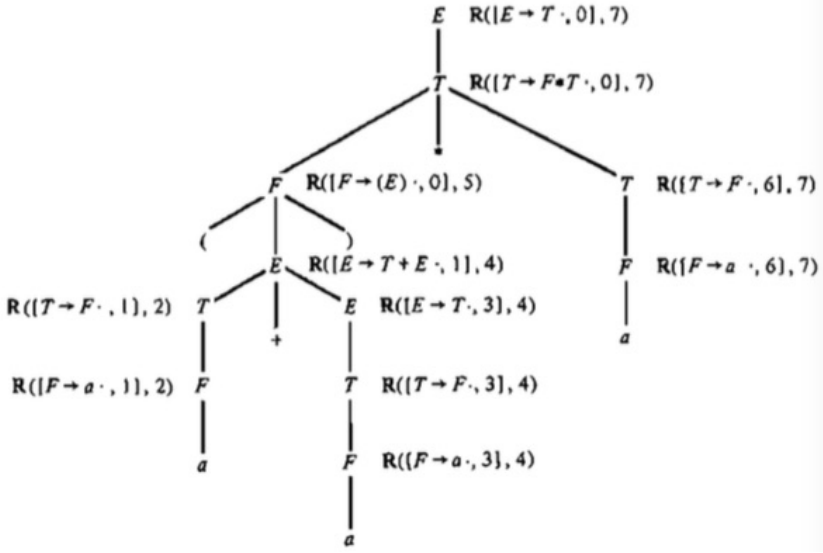
$k = k - 1$

$j' = i'$.

Ejemplo

Sea G con las producciones
1. $E \Rightarrow T + E$
2. $E \Rightarrow T$
3. $T \Rightarrow F * T$
4. $T \Rightarrow F$
5. $F \Rightarrow (E)$
6. $F \Rightarrow a$

La figura muestra la rutina R sobre el árbol de derivación para $(a + a) * a$, de longitud 7, y $\pi = 64642156432$.



Complejidad de algoritmo de Earley

Teorema 3 (Theorem 4.12 en Aho Ullman vol 1)**.** *Si existe una derivación más a la derecha de $a_1 \dots a_n$ el algoritmo de Earley la encuentra en c n^2 operaciones, para una constante c.*

Complejidad de algoritmo de Earley

Demostración. Recordemos que $[A \to \alpha \cdot \beta, i] \in \ell_j$ si y solo si $\alpha \overset{*}{\underset{R}{\Rightarrow}} a_{i+1} \dots a_j$ y existen γ y δ tales que $S \overset{*}{\underset{R}{\Rightarrow}} \gamma A \delta \ \gamma \overset{*}{\underset{R}{\Rightarrow}} a_1 \dots a_i$.

En ℓ_j unimos todos los items con la misma segunda componente.
Acceso y consulta en esta lista son operaciones elementales.
Este preprocesamiento se puede hacer en tiempo constante n^2.

Veamos que la ejecución de $R([A \to \beta\cdot, i], j)$ lleva del orden de $(j - i)^2$ operaciones. El ciclo realiza $|\beta|$ iteraciones. En cada iteración lo costoso es buscar un valor i' examinando $j - i + 1$ listas (cada exámen es elemental) y realizar una llamada recursiva $R([--, i'], j')$, donde $j' - i' < j - i$.

La ejecución de $R([A \to \beta\cdot, i], j)$ invoca a $R([--, i'], j')$ a lo sumo $|P| \times M$ veces (cada vez con un valor distinto de $--$) donde M es la máxima cantidad d esímbolso de una producción.
El total de invocaciones a R (una por cada iteración más las invociaciones recursivas anidadas) es a lo sumo $(j - i) \times |P| \times M$. $\square$