Teoría de Lenguajes
Segunda parte
Teórica 5: Gramáticas de Atributos

Verónica Becher

Segundo cuatrimestre 2020

Bibliografía para esta clase

- (libro del dragón) Aho A.V., Lam M.S., Sethi R. y Ullman J.D. Compilers: Principles, Techniques, and Tools, Second Edition. Addison Wesley, 2007 `http://www.dc.uba.ar/staff/becher/dragon.pdf`
- The Theory of Parsing, Translation and Compiling A. Aho, J. Ullman, Prentice Hall, Volume I and Volume II, 1972 `http://www-2.dc.uba.ar/staff/becher/Aho-Ullman-Parsing-V2.pdf` `https://www-2.dc.uba.ar/staff/becher/Aho-Ullman-Parsing-V1.pdf`
- K. Slonneger, B. Kurtz Formal Syntax and Semantics of Programming Languages: A Laboratory Based Approach Addison-Wesley Longman Publishing,1995 Capitulo 3 Atribute Grammars `http://www-2.dc.uba.ar/profesores/becher/chapter3-slonnegr-iowa`
- Marcelo Daniel Arroyo. Gramáticas de atributos, clasificación y aportes en técnicas de evaluación. Tesis Magister en Ciencias de la Computación Universidad Nacional del Sur. Bahía Blanca Argentina, 2008 `http://repositoriodigital.uns.edu.ar/bitstream/123456789/1999/1/ag-final-1.pdf`

Otras referencias

- D. Knuth. 1968. Semantics of context free languages. Math Systems Theory 2:127-145.
- U. Kastens. 1980. Ordered Attribute Grammars. Acta Informatica 13: 229-256.
- M. Jazayeri W. Ogden. 1975. The Intrinsically Exponential Complexity of the Circularity Problem for Attribute Grammars Communications of the ACM 18(12): 697-706.

Gramáticas de atributos (Knuth 1968)

Las *gramáticas de atributos* son gramáticas libres de contexto extendidas con *atributos, reglas de evaluación y condiciones.*

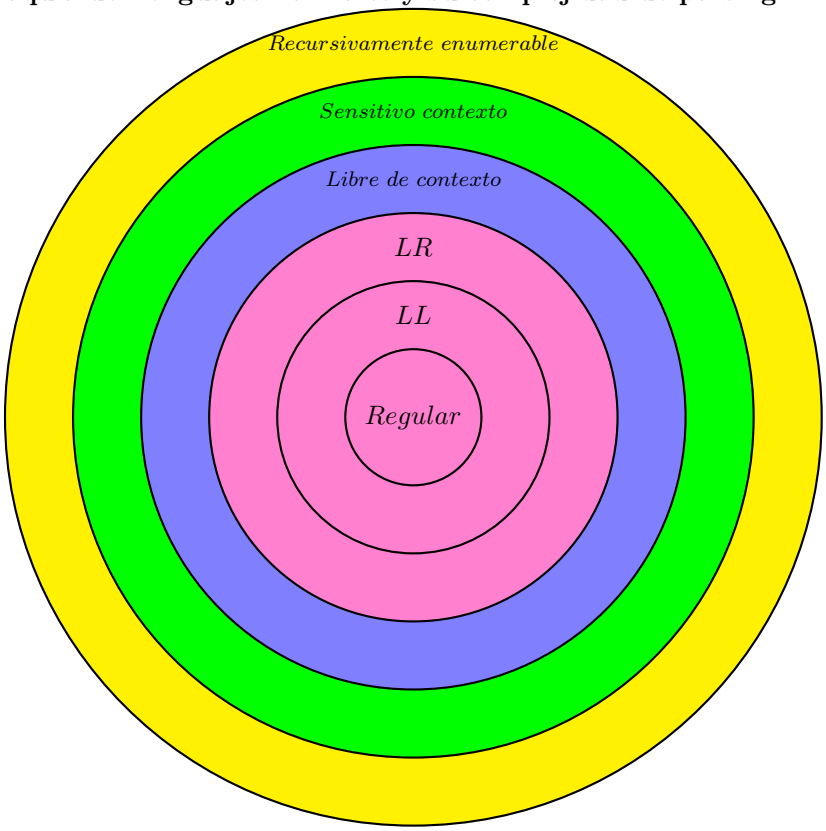
- Proveen sensitividad al contexto
- Dan semántica de lenguajes formales

A cada símbolo de la gramática (terminal o no terminal) le asignamos un conjunto (posiblemente vacío) de atributos. Cada atributo tiene un dominio de valores como números enteros, caracteres y strings, o estructuras más complejas.

Si vemos a la expresión de entrada como un árbol de derivación,las gramáticas de atributos pueden pasar valores de un nodo a su padre usando un atributo *sintetizado*, o pueden pasar valores d eun nodos a sus hijos usando un atributo *heredado*. Además de pasarlos para arriba o para abajo los atributos pueden ser chequeados, asignados o modificados.

Las gramáticas de atributos y acciones semánticas surgieron en los años 1960.

Jeraquía de Lenguajes Formales y su complejidad de parsing



Rosa: complejida dlineal

Azul complejidad cúbica

Verde complejidad PSPACE

Amarillo complejidad NP

Gramática de atributos

Definición (Gramática de Atributos)**.** *Una gramática de atributos es una tupla* $GA = (G, A, V, Dom, F, R)$ *donde,*

1. $G = (N, T, P, S)$ *es una gramática libre de contexto donde todos sus no-terminales son alcanzables y es no-ambigua.*
2. A *es el conjunto de atributos y es la unión de los subconjuntos* $A(X)$ *que son los atributos asociados al símbolo* X *donde* $X \in N \cup T$*.*
3. V *es el conjunto de dominios de los valores de los atributos*
4. $Dom : A \to V$ *asocia a cada atributo un dominio*
5. F *es el conjunto de funciones semánticas*
6. R *es el conjunto de reglas asociadas a cada producción* $p \in P$

   (una producción puede tener muchas reglas)

   $$(X_0.a_0, f, X_1.a_1, \ldots, X_k.a_k)$$

   donde $X.a$ *denota el atributo* a *del símbolo* X*, y la función* $f \subseteq Dom(a_1) \times \ldots \times Dom(a_k) \to Dom(a_0)$*.*

Atributos sintetizados y heredados

El conjunto de atributos de un símbolo está particionado en dos, los sintetizados $S(X)$ y los heredados $I(X)$.

Definición. *Un atributo $X.a$ es* sintetizado *si hay una producción $p : X \to \alpha$ y una regla para p donde ocurre $X.a$ e Un atributo $X.a$ es* heredado *si $X.a$ si hay una producción $p : Y \to \alpha X \beta$ y una regla para p donde ocurre $X.a$.*

Para evitar circularidades obvias se impone que para todo $X \in N$, $S(X) \cap I(X) = \emptyset$. Y el símbolo Start S no tiene atributos heredados.

Sensitividad al contexto

Sabemos que el lenguaje $a^n b^n c^n$, $n \geq 0$ no es libre de contexto.

La siguiente gramática libre de contexto reconoce $a^+ b^+ c^+$

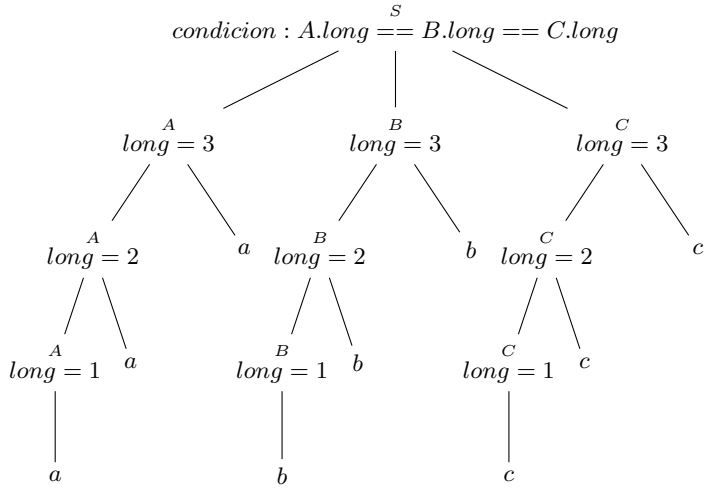
$$
\begin{aligned}
S &\to ABC \\
A &\to a|Aa \\
B &\to b|Bb \\
C &\to c|Cc
\end{aligned}
$$

$a^n b^n c^n$ con atributos sintetizados

Consideremos el atributo *longitud* para cada no terminal, A, B, C y usemolos en la gramática para controlar la cantidad de a b y c.

$S \to$	ABC	$condicion \;\; A.long == B.long == C.long$
$A \to$	a	$A.long = 1$
$A \to$	$A_2 a$	$A.long = A_2.long + 1$
$B \to$	b	$B.long = 1$
$B \to$	$B_2 b$	$B.long = B_2.long + 1$
$C \to$	c	$C.long = 1$
$C \to$	$C_2 c$	$C.long = C_2.long + 1$

$a^n b^n c^n$ con atributos sintetizados

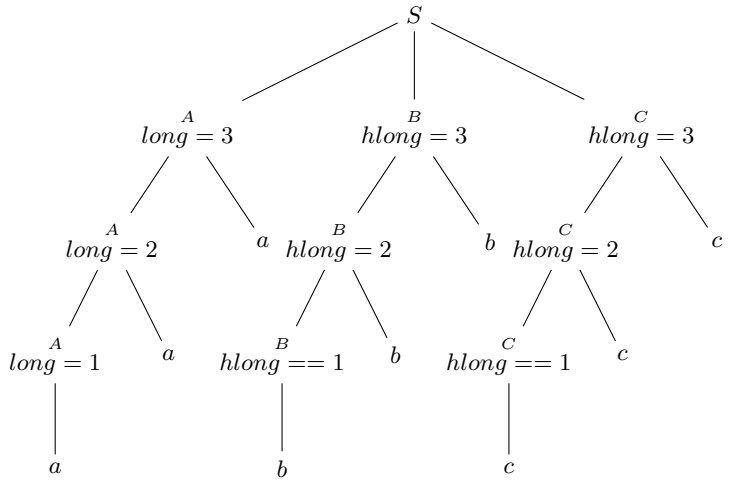


$a^n b^n c^n$ con atributos sintetizados y heredados

Consideremos el atributo *longitud* para cada no terminal, A, B, C
A.long sintetizado B.hlong heredado C.hlong heredado

$S \to$	ABC	$B.hlong = A.long,\ C.hlong = A.long$
$A \to$	a	$A_2.long = 1$
$A \to$	$A_2 a$	$A.long = A_2.long + 1$
$B \to$	b	$condicion\ B.hlong == 1$
$B \to$	$B_2 b$	$B_2.hlong = B.hlong - 1$
$C \to$	c	$condicion\ C.hlong == 1$
$C \to$	$C_2 c$	$C_2.hlong = C.hlong - 1$

$a^n b^n c^n$ con atributos sintetizados y heredados



Semántica

Definición. *Sea G una gramática libre de contexto y GA su gramática de atributos. El significado de una cadena α en $L(G)$, es el conjunto de valores $A(S)$, es decir, el conjunto de los valores de los atributos sintetizados asociados al símbolo de comienzo S.*

Para esto necesitamos un mecanismo para dar los valores de los atributos.

Grafo de dependencias

Sea GA una gramática de atributos.

Definición (Grafo de dependencias)**.** *Definimos el grafo de dependencias de una producción. Cuando una regla define a en función de b (y posiblemente otros atributos), escribimos $X_j.b \to X_i.a$.*

Notar que si un atributo pudiera ser sintetizado y heredado simultáneamente, podría definirse en una producción de la forma $X \to \alpha X \beta$ una regla semántica con la forma $X.a - f(\ldots, X.a, \ldots)$ En este caso la gramática de atributos será circular (ver definición más abajo, ya que un atributo dependerá (tal vez transitivamente) de sí mismo.

Grafo de dependencias y gramática circular

Definición (Arbol atribuido)**.** *Un árbol atribuido $T(GA)$ para la gramática de atributos GA es un árbol de derivación cuyos nodos tienen etiquetas para $(X, X.a_1, \ldots, X.a_k)$ donde $\{X.a_1, \ldots, X.a_k\} = A(X)$.*

Definición (Grafo de dependencias)**.** *Para cada producción $p \in P$ damos un grafo de dependencias $DP(p)$.*

El grafo de dependencias de un árbol atribuido $T(GA)$ está dado por la composición de los grafos de dependencia $DP(p)$, $p \in P$.

Definición (Gramática de atributos circular)**.** *Una gramática de atributos GA es circular si y solo si existe un árbol atribuido $T(GA)$ tal que su grafo de dependencias contiene al menos un ciclo.*

Evaluación de atributos

La evaluación de atributos sobre un árbol atribuido $T(GA)$ es un proceso que computa los valores de las instancias de atributos de $T(GA)$ de acuerdo a las reglas semánticas. Es decir que el valor de una instancia del atributo $X_0.a$ en nodo rotulado X_0, se obtiene computando la regla semántica $X_0.a_0 = f(X_1.a_1, \ldots, X_k.a_k)$ según un orden de evaluación consistente.

Definición (Árbol decorado)**.** *El árbol atribuido, sobre el cual las instancias de los atributos en cada nodo han sido definidas (es decir, cada instancia se ha asociado a un valor de su dominio), se denomina árbol decorado.*

Gramática bien definida

Un orden topológico de un grafo es un orden total de todos sus nodos tal que cada nodo aparece exactamente una vez y si (u, v) está en el grafo entonces u precede a v.

Definición (Gramática bien definida)**.** *Una GA se dice* bien definida *si cualquier orden topológico de su grafo de dependencias de es un orden de evaluación consistente con las dependencias de atributos.*

Definición (Gramática completa)**.** *Una GA es completa si cumple :*

- *Para toda producción p de la forma $X \to \gamma$, todos los atributos sintetizados de X se definen en p.*
- *Para toda producción p de la forma $Y \to \alpha X \beta$ todos los atributos heredados de X se definen en p.*

Observación 1. *Una gramática de atributos está bien definida si es completa y su grafo de dependencia no tiene ciclos.*

Sobre gramáticas bien definidas

Teorema 2. *Sea GA_1 una gramática de atributos bien definida. Es posible obtener una gramática de atributos GA_2, equivalente a GA_1, tal que GA_2 no contiene atributos heredados.*

Demostración.

Hay dos posibles enfoques.

1. Modificación de la gramática libre de contexto: la solución consiste en aplicar repetidamente a las producciones, reglas de factorización y eliminación de recursión a izquierda, para finalmente reemplazar los atributos heredados por sintetizados.

una producción de la forma $X \to \alpha Y \beta Z \gamma$, donde un atributo heredado de Y depende de atributos de Z, y donde $Y \Rightarrow t$, una de estas dos formas:

1. $X \to \alpha t Y$ e $Y \to \beta Z \gamma$.
2. $X \to Y \gamma$ e $Y \to \alpha Z \beta t$.

2. Modificación del dominio de los atributos: Sea G una gramática de atributos con atributos heredados. Sea G' una gramática de atributos obtenida a partir de G en la cual se eliminan todos los atributos heredados y el dominio de los atributos heredados de A se reemplaza por funciones $Ops(A)$ definidas a partir de los atributos sintetizados.edskip

El conjunto $Ops(A)$ contiene funciones de la forma $f(\sigma, x)$, que respetan las dependencias funcionales del atributo σ del símbolo x desde sus atributos heredados con respecto a G (la gramática original).

La evaluación de atributos arroja como resultado la composición de las funciones. Como G' no es circular, hay un valor inicial para la aplicación de la composición de funciones. □

Una generalización del segundo enfoque nos lleva a las gramáticas de alto orden (High Order Attribute Grammars), que permiten que el dominio de los atributos sean árboles sintácticos derivado a partir de otra GA.

Clasificación basada en la estrategia de evaluación

Clasificación de gramáticas de atributos en base a procedimientos de evaluación. En recorridos sucesivos del árbol atribuido se evalúan los atributos que sean posibles. Si el grafo de dependencias es acíclico luego de varios recorridos todos los atributos deberían estar evaluados.

Gramáticas s-atribuidas

La clase de gramáticas s-atribuidas es la clase más restrictiva de GA ya que sólo permite atributos sintetizados

Definición. *Una $GA = (G, A, V, Dom, F, R)$ es s-atribuida si y solo si el conjunto $I(A) = \emptyset$ y el grafo de dependencias directas de cada producción no contiene ciclos.*

Se pueden evaluar en un solo recorrido ascendente del árbol atribuido y es de aplicación directa en parsers ascendentes.

Gramáticas l-atribuidas

Las gramáticas de atributos l-atribuidas ((también denominadas 1 Pass Attribute Grammars)

Esta famila de GA tienen como característica que en cada producción p, los atributos de un símbolo de la parte derecha de p, X_i, dependen sólo de atributos de los símbolos X_j que aparecen a la izquierda de X_i.

Definición. *Una GA es l-atribuida si y solo si es bien definida y para cada producción de la forma $p : X_0 \to X_1 X_2 \dots X_{np}$ se cumple:*

1. *Si $(Xi.a, Xj.b) \in DG(p)$ entonces $i < j$, asumiendo $(1 \leq j \leq np)$,*
2. *$DP(p)$ es acíclico.*

Pueden ser evaluadas en un solo recorrido descendente de izquierda a derecha.

Una pasada no alcanza

Las gramáticas de atributos l-atribuidas han sido ampliamente utilizadas en herramientas de generación de parsers ya que pueden evaluarse durante el parsing. Sin embargo su limitación es bastante notable en muchas construcciones que se encuentran comunmente en lenguajes de programación, como por ejemplo en una declaración tipo Pascal:

`decl: id-list ':' type ';'`

El atributo `type` de cada identificador no puede ser computado hasta que se conozca el tipo de la declaración, por lo que una GA para esta construcción no sería l-atribuida. En la práctica estas limitaciones se resuelven generalmente sintetizando listas de identificadores en un atributo de `id-list` y se asigna el tipo de cada uno (en una tabla de símbolos global) luego de la reducción de la regla `decl`.

GA evaluables en n-recorridos

Obviamente existen gramáticas de atributos que no pueden ser evaluadas en un solo recorrido (ascendente ni descendente), pero que podrían ser evaluadas en varios recorridos (por ejemplo, descendentes, de izquierda a derecha). Cada recorrido evaluará los atributos que pueda (aquellos cuyos atributos de los cuales dependen se hayan evaluado en recorridos anteriores).

Una generalización o extensión del método anterior es alternar recorridos en cada pasada: en la pasada i-ésima se realiza un recorrido descendente de izquierda a derecha (derecha a izquierda) si i es par (impar).

La cantidad máxima de recorridos debe ser una constante para toda cadena de entrada.

Clasificación basada orden de dependencia

Una gramática de atributos se dice *ordenada* si para cada símbolo de $(N \cup T)$ sus atributos pueden ser evaluados en un orden (total) predeterminado, es decir que es posible determinar un orden de evaluación independientemente de su contexto.

Definición. *Una gramática de atributos es (ordenada) si para cada producción existe un orden total tal que el grafo de dependencias aumentado con dichos órdenes es no circular.*

Esta familia es una subclase de las grmáticas ordenadas definidas también por Kastens .para las cuales la decisión de que si una GA pertence a ésta familia no puede ser determinado en tiempo polinomial. Kastens también propuso un algoritmo para computar los ordenes totales en tiempo polinomial. Esto permite evaluadores para esta familia que sean muy eficientes, tanto en tiempo como en espacio y el algoritmo de verificación de circularidad tiene tiempo polinomial.

Ejemplo gramática ordenada

Esta gramática no se puede evaluar en un número fijo de pasadas.

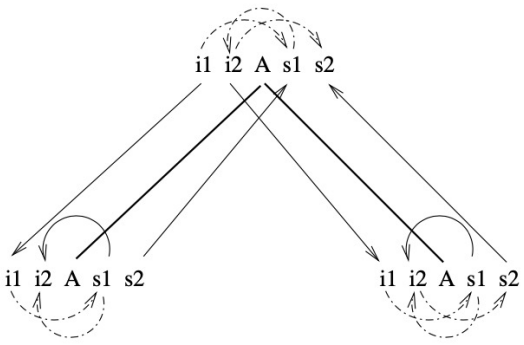


Figura 3.2: Grafo de dependencias de la producción $p1$ aumentado (arcos con línea de puntos) con un orden total sobre las dependencias de los atributos del símbolo A.

$p0:$ $S \to A$
attribution
$A.i1 = S.in$
$A.i2 = A.s1$
$S.out = A.s2$

$p1:$ $A \to AA$
$A1.i1 = A0.i1$
$A1.i2 = A1.s1$
$A0.s1 = A1.s2$
$A2.i1 = A0.i2$
$A2.i2 = A2.s1$
$A0.s2 = A2.s2$

$p2:$ $A \to \lambda$
attribution
$A.s1 = A.i1 + 1$
$A.s2 = A.i2 + 2$

Orden total consistente para A es $\{i1 \to s1 \to i2 \to s2\}$.

Algoritmo de detección de circularidad (Knuth 1968)

La observación que hace posible el algoritmos es la siguiente: Si hay un ciclo en algun grafo de dependencia para la gramática G entonces hay un ciclo simple: uno que pasa por cada nodo

(atribito) a lo sumo una vez. En vez de revisar $2^{m \times m}$ conjuntos de pares, hay que considerar $2^{m \times \log m}$ donde m es la cantidad de atributos de G.

Dada una gramática de atributos G, algoritmo produce una gramática libre de contexto G'. La gramática de atributos G es circular exactamente cuando hay una producción admisible en G' que es un arco (parte de un ciclo simple) y participa en alguna derivación desde el símbolo start de G' llegando a símbolos terminales.

Debemos definir G', la noción de *producción admisible* y la noción de *arco*.

Gramática equivalente G'

Sea G gramática de atributos. T el conjunto de terminales y N el conjunto de sus no-terminales. Definimos G' así:

Conjunto T de símbolos terminales.

Conjunto N' de símbolos no terminales.

Sea p una producción de G y sea X un no terminal en p.

Sea $A(p, X)$ los atributos asociados con esta ocurrencia de X.

Los no terminales de G' son las parejas (X, D) donde $D = \{(a, b) : a, b \in A(p, X)\}$.

Esto se realiza en tiempo polinomial en $|G|$.

Producciones de G'.

Sea $h : (N' \cup T) \to (N \cup T)$ tal que $h(X, D) = X$ y $h(t) = t$, para $t \in T$.

Para cada producción p en G, agregar las producciones $Y_0 \to Y_1 \dots Y_k$ tales que

- $Y_i \in (N' \cup T)$,
- p es $h(Y_0) \to h(Y_1) \dots h(Y_k)$ y si $Y_i = (X_i, D_i)$ entonces todos los atributos mencionados en D_i son miembros de $A(p, X)$.

Este se realiza en tiempo polinomial en $|G|$.

Producciones admisibles

Asumamos que no hay atributos asociados a símbolos terminales.

Sea π una producción en G', $u(\pi)$ la producción de G y $\gamma(u(\pi))$ el grafo de dependencia asociado.

Sean e and f atributos que ocurren en no-terminales (Y, E) y (Z, F) en π (pueden ser iguales y pueden ocurrir en la parte izquierda de π).

Decimos que e y f están π-relacionados si y solo si
hay un camino desde e a f en $\gamma(u(\pi))$, o
$(e, f) \in E$ y el no-terminal (Y, E) ocurre en el lado derecho de π, o
$(e, f) \in F$ y (Z, F) ocurre en el lado derecho de π.
Sea π^+ la clausura transitiva de la relación π.

Definición (Producción admisible)**.** *Supongamos que la parte izquierda de π es (X, D). Decimos que la producción π es* admisible *si , $D = \emptyset$ o para cada $(a, b) \in D$, a y b están π^+ relacionadas.*

Se demuestra por induccion sobre árboles de derivacion parcial que (X, D) deriva una cadena de terminales en G' si y solo si X la deriva en G y para cada $(a, b) \in D$ hay un camino simple de a to b en el grafo de dependencias.

Ejemplos admisibles y no admisibles

a, b, c son atributos de X;
d, e, f, g, h son atributos de Y
i, j son atributos de Z.

Sea el grafo de dependencia dado por
$(a, b), (a, e), (f, b), (f, g), (h, i), (i, j), (j, c)$

- $(X, (a, c)) \to (Y, (e, h))(Z, (i, j))$ es admisible.
- $(X, (a, c)) \to (Y, (e, f), (g, h))(Z, (i, j))$ es admisible.
- $(X, (a, c)) \to (Y, (e, f)(e, h)(g, h))$ no es admisible porque no es simple
- $(X, (a, c)) \to (Y, (e, f))(Z, (i, j))$ no es admisible porque falta la conexión $g \to h$
- $(X, (a, b)) \to (Y, (d, h))(Z, \emptyset)$ es amisible, conexión directa de a y b

Quitar produciones inútiles

Una production π es inútil en una gramática si para cada derivación completa $S \stackrel{*}{\Rightarrow} w$, con $w \in T^*$ la producción π no es usada.

Ciclos

Para terminar el algoritmo debemos identificar las producciones admisibles de G' que forman un arco que cierra un ciclo. Concluiremos que la gramática G tiene un ciclo exactamente cuando hay alguno de estos arcos.

Una producción π es un arco si existe al menos un atributo mencionado en π que está π^+-relacionado consigo mismo.

Ejemplos:

1. si en el grafo de dependencias hay un camino desde c hacia a, la producción $(X, (a, c)) \to (Y, (e, h))(Z(, i, j))$ es un arco.
2. si en el grafo de dependencias hay un camino de j a g y de h a i $(X, \emptyset) \to (Y, (g, h))(Z, (i, j))$ entonces esta produccion es un arco, debido al ciclo $g \to \ldots \to h \to i \ldots \to j \to g$

Algoritmo de deteción de circularidad

Input: G gramática de atributos

Output: Decisión sobre la circularidad del grafo

Generar los símbolos no terminales de G'.

Esto se realiza en tiempo exponencial en $|G|$.

Generar las producciones de G'.

Esto se realiza en tiempo exponencial en $|G|$.

- Testear la admisibilidad de las nuevas producciones, revisando el grafo dependencias de las producciones originales.

  Si (X, D) es la parte izquierda de una produccion y (X_i, D_i) son los no terminales dela parte derecha podemos testear (X, D) si ya sabemos que los no-terminales de la derecha ya tiene la propiedad.

  Esto se realiza para cada prducción en tiempo polinomial en $|G|$.

- Quitar de G' las producciones inútiles, es decir aquelas que no participan en la derivación de un terminal empezando dessde Start.

  Esto se realiza en tiempo polinomial en $|G|$.

- Si en G' hay alguna producción que sea un arco admisible contestar que G es circular. Sino, hay arcos admisibles, contestar que G está bien formada.

  Es lineal en el tamaño del grafo, es decir exponencial en $|G|$.

Complejidad del algoritmo de Circularidad

Teorema 3 (Knuth 1968; Jazayeri, Ogden 1975)**.** *La determinación si una GA es circular tiene complejidadad* 2^{dn} *donde* n *es el tamaño de la gramática*

exponencial respecto del número máximo de no terminales de la parte derecha de las producciones.

El test de circularidad lleva demasiado tiempo

La mayoría de los evaluadores de gramáticas bien definidas no incluyen la prueba de circularidad y utilizan alguno de los siguientes enfoques:

- Para cada árbol atribuído T se construye su grafo de dependencias $GD(T)$ y se verifica que sea acíclico. Luego se produce la secuencia (plan) de evaluación computando un orden topológico de $GD(T)$.

- Computar estáticamente todos los posibles planes de evaluación para la GA dada para que el evaluador seleccione (en tiempo de ejecución) el plan correspondiente para cada árbol atribuído.

- Ignorar el problema y comenzar a evaluar inicialmente los atributos que no dependan de otros atributos, marcarlos como computados, luego evaluar los atributos que dependan solamente de ellos y así sucesivamente.

  En el caso de existir dependencias cíclicas, el proceso no podrá continuar con la evaluación (en cuyo caso emitirá un mensaje de error en caso de que se realice detección de ciclos, o se producirá un deadlock). Esta estrategia de evaluación se conoce comúnmente como evaluación bajo demanda o por necesidad.

En una gramática de atributos, se relaciona con cada símbolo de una gramática libre de contexto un conjunto de atributos. Cada regla o producción tiene asociados un conjunto de reglas semánticas que toman la forma de asignación a atributos de valores denotados por la aplicación de una función, la cual puede tomar como argumentos instancias de atributos pertenecientes a los símbolos que aparecen en la producción.

Las reglas semánticas inducen dependencias entre los atributos que ocurren en la producción. El orden de evaluación es implícito (si existe) y queda determinado por las dependencias entre las instancias de los atributos.

Una regla semántica se podrá evaluar cuando las instancias de los atributos que aparecen como sus argumentos estén evaluadas. Un evaluador de gramáticas de atributos debe tener en cuenta las dependencias entre las instancias de atributos para seguir un orden consistente de evaluación de los mismos.

Si una GA contiene dependencias circulares no podría ser evaluada ya que no podría encontrarse un orden de evaluación.

Herramientas

- YACC (Yet Another Compiler-Compiler) YACC es un generador de parsers que permite asociar atributos a los símbolos y acciones a cada producción de la gramática. Acepta gramáticas LALR, l-atribuidas y los parsers son shift-reduce. YACC ejecuta las acciones asociadas a cada regla en el momento de la reducción del símbolo que precede la acción. En una especificación YACC es dificultoso utilizad atributos heredados aunque se puede simular por el hecho que permite la declaración de variables globales con lo cual es posible realizar cualquier tipo de acción semántica.
- AntLR
- JavaCC
- JavaCUP
- ELI