Capítulo 5: Traducción Dirigida por Sintaxis

Javier Carvajal
Universidad de Costa Rica,
Escuela de Ciencias de la Computación e Informática,
San José, Costa Rica,
francisco.carvajal@ecci.ucr.ac.cr

and

Sergio Fonseca
Universidad de Costa Rica,
Escuela de Ciencias de la Computación e Informática,
San José, Costa Rica,
sergio.fonseca@ecci.ucr.ac.cr

Resumen

El presente artículo es básicamente un resumen del capítulo 5 del libro Autómatas y Compiladores de Alfred V. Aho, en el cual se trata sobre la traducción dirigida por sintaxis, y para abordar mejor el tema se definen un conjunto de conceptos que ayudan a comprender mejor lo que es en sí la traducción dirigida por sintaxis y detalladamente describir cómo es posible para un compilador llevarla a cabo, tomando en cuenta todos los pasos que sigue.

Palabras clave: **traducción, atributo heredado, atributo sintetizado, análisis sintáctico, reglas semánticas, descendente.**

1. Introducción

Un compilador utiliza gramáticas independientes de contexto como guía para la traducción de lenguajes, si logra de alguna manera asociar información a las distintas construcciones del lenguaje de programación proporcionando atributos a los símbolos de la gramática, puede entonces calcular sus valores mediante reglas semánticas.

Aunque hay dos maneras de hacer la asociación entre reglas semánticas y producciones, que son mediante definiciones dirigidas por sintaxis y esquemas de traducción, nos interesa particularmente ésta última, pues con éstas se trabaja más pues se puede definir el orden en que se ejecutan las acciones y las evaluaciones de los atributos.

La idea es brindar al lector una pequeña reseña de lo que el compilador hace desde que toma una cadena de entrada hasta que la lleva a evaluar las reglas semánticas, que es una parte importante del proceso de compilación en un lenguaje de programación.

2. Definición Dirigida por Sintaxis

Para traducir una construcción de un lenguaje de programación, un compilador puede necesitar tener en cuenta muchas características, además del código generado para la construcción. Una definición dirigida por sintaxis es un formalismo para especificar las traducciones para las construcciones en función de atributos asociados con sus componentes sintácticos.

Utiliza una gramática independiente de contexto para especificar la estructura sintáctica de la entrada, la idea es asociar con cada símbolo de la gramática un conjunto de atributos (que luego veremos que pueden ser sintetizados o heredados) y además a cada producción un conjunto de reglas semánticas

para calcular los valores de los atributos asociados con los símbolos que aparecen en esa producción. La definición dirigida por sintaxis consiste en sí entonces de la gramática y el conjunto de reglas semánticas.

En una definición dirigida por sintaxis, para cada producción gramatical $A \rightarrow \alpha$ se asocia un conjunto de reglas semánticas de la forma b:=f(c1,c2,…,ck) donde f es una función y b es un atributo sintetizado de A o un atributo heredado de uno de los símbolos gramaticales de la parte derecha de la producción y c1,c2,…,ck son atributos que pertenecen a los símbolos gramaticales de la producción. Se dice que b depende de c1,c2,…,ck.

3. Atributos

Un atributo puede representar cualquier cosa, un nombre, una cadena, un número, un tipo, una posición de memoria, su valor en un nodo se define mediante una regla semántica asociada a la producción usada en dicho nodo.

En una definición dirigida por sintaxis se asume que los terminales sólo tienen atributos sintetizados, ya que la definición no proporciona ninguna regla semántica para los terminales. El analizador léxico es el que proporciona generalmente los valores para los atributos de los terminales.

3.1. Atributos Sintetizados

Se puede decir que un atributo es sintetizado si su valor en un nodo del árbol de análisis sintáctico se determina a partir de los valores de atributos de los hijos de ese nodo (como decir de abajo hacia arriba). Se pueden calcular mediante un solo recorrido ascendente del árbol de análisis sintáctico, lo que es muy deseable.

3.2. Atributos Heredados

Un atributo heredado es uno cuyo valor en un nodo de un árbol de análisis sintáctico está definido a partir de los atributos en el padre y los hermanos de dicho nodo. Éstos sirven para expresar la dependencia de una construcción de un lenguaje de programación en el contexto en el que aparece.

4. Grafos de dependencias

Si un atributo b en un nodo de un árbol de análisis sintáctico depende de un atributo c, entonces se debe evaluar la regla semántica para b en ese nodo después de la regla semántica que define a c. las interdependencias entre los atributos heredados y sintetizados en los nodos de un árbol de análisis sintáctico se pueden representar de manera muy natural mediante un grafo dirigido llamado grafo de dependencias.

5. Evaluación de las Reglas Semánticas

Un ordenamiento de evaluación busca que las aristas vayan en dirección a los nodos que aparecen después en el ordenamiento. Todo ordenamiento de un grafo de dependencias da un orden válido en el que se pueden evaluar las reglas semánticas asociadas a los nodos del árbol de análisis sintáctico.

La traducción especificada por una traducción dirigida por sintaxis se puede precisar así. Se utiliza la gramática para construir un árbol de análisis sintáctico para la entrada. El grafo de dependencias se construye a partir del mismo. De éste se obtiene un orden para evaluar las reglas semánticas. La evaluación de las reglas sintácticas en este orden produce la traducción de la cadena de entrada.

5.1. Métodos para evaluar las Reglas Semánticas

5.1.1 Con árbol de Análisis Sintáctico

En el momento de la compilación, si el grafo de dependencias obtenido a partir del árbol de análisis sintáctico no contiene un ciclo, se obtiene un ordenamiento para cada entrada.

5.1.2. Basados en Reglas

En el momento de la construcción del compilador, las reglas semánticas asociadas con las producciones son analizadas. Para cada producción el orden de evaluación de los atributos asociados queda predeterminado.

5.1.3. “Sin recuerdo”

Escoge un orden sin considerar las reglas semánticas. Limita la clase de definiciones dirigidas por sintaxis que pueden implantarse.

Los métodos “sin recuerdo” y basados en reglas no necesitan construir de manera explicita el grafo de dependencias en el momento de la compilación, así que pueden ser más eficaces en el uso del tiempo y espacio del compilador.

6. Construcción de Árboles Sintácticos

El uso de árboles de análisis sintáctico como representación intermedia permite que la traducción se separe del análisis sintáctico. Las rutinas de traducción invocadas durante el análisis sintáctico deben activarse con dos clases de limitaciones. La primera, una gramática que resulte adecuada para le análisis sintáctico puede no reflejar la estructura jerárquica natural de las construcciones del lenguaje. La segunda, el método de análisis sintáctico restringe el orden en que se consideran los nodos de un árbol de análisis sintáctico. Pues este orden puede no coincidir con el orden en que se va disponiendo de la información sobre una construcción.

6.1. Arboles Sintácticos

Un árbol sintáctico es una forma condensada de un árbol de análisis sintáctico, útil para representar construcciones de lenguajes. Los operadores y las palabras no aparecen como hojas sino más bien están asociadas con el nodo interior que sería el padre de dichas hojas en el árbol análisis sintáctico.

La construcción de un árbol sintáctico para una expresión es similar a la traducción de la expresión a una forma posfija. Se construyen subárboles para las subexpresiones creando un nodo para cada operador y para cada operando. Los hijos de un nodo de un operador son las raíces de los nodos que representan las subexpresiones que constituyen los operandos de dicho operador.

Se puede implantar cada nodo en un árbol sintáctico como un registro con varios campos. En el nodo para un operador, un campo identifica el operador y el resto de los campos contienen apuntadores a los nodos de los operandos. El operador a menudo se denomina etiqueta del nodo. Cuando se usan para traducir los nodos de un árbol sintáctico pueden tener más campos para guardar valores (o apuntadores a los mismos) de los atributos asociados al nodo.

7. Grafos dirigidos acíclicos para expresiones

Sirve para identificar expresiones comunes. Tiene un nodo para cada subexpresión de la expresión, un nodo interior representa un operador y sus hijos representan los operandos. La única diferencia es que en un grafo dirigido acíclico que representa una subexpresión que aparece más de una vez, tendrá una cantidad de padres equivalente a la cantidad de veces que aparece.

8. Evaluación ascendente de definiciones con atributos sintetizados

Los atributos se pueden evaluar con un analizador sintáctico ascendente conforme la entrada es analizada. El analizador sintáctico puede conservar en su pila los valores de los atributos sintetizados asociados con los símbolos gramaticales. Siempre que se haga una reducción se calculan los valores de los nuevos atributos sintetizados a partir de los atributos que están en la pila para los símbolos gramaticales del lado derecho de la producción con la que se reduce.

8.1. Atributos sintetizados en la pila del analizador sintáctico

A partir de una definición con atributos sintetizados, el generador de analizadores sintácticos puede construir un traductor que evalúe los atributos conforme analiza la entrada. Un analizador

sintáctico ascendente utiliza una pila para guardar información acerca de los subárboles que ya han sido analizados. Se pueden utilizar campos adicionales en la pila del analizador para guardar los valores de los atributos sintetizados.

9. Definiciones con atributos por la izquierda

Cuando la traducción tiene lugar durante el análisis sintáctico, el orden de evaluación de los atributos va unido al orden en que, por el método de análisis sintáctico, se crean los nodos del árbol de análisis sintáctico. Un orden natural que caracteriza es el orden de evaluación en profundidad. Aunque de hecho no se construye el árbol de análisis sintáctico, es útil estudiar la traducción durante el análisis sintáctico considerando la evaluación en profundidad de los atributos en los nodos de un árbol de análisis sintáctico.

Se les llama "por la izquierda" porque la información de los atributos parece fluir de izquierda a derecha. Las definiciones con atributos por la izquierda incluyen todas las definiciones dirigidas por sintaxis basadas en gramáticas LL(1). Toda definición con atributos sintetizados es una definición con atributos por la izquierda.

10. Esquemas de traducción

Un esquema de traducción es una gramática independiente de contexto en la que se asocian atributos con los símbolos gramaticales y se insertan acciones semánticas encerradas entre llaves { } dentro de los lados derechos de las producciones. Los esquemas de traducción pueden tener tanto atributos sintetizados como heredados.

Cuando se diseña un esquema de traducción, se deben respetar algunas limitaciones para asegurarse de que el valor de un atributo esté disponible cuando una acción se refiera a él. Estas limitaciones, motivadas por las definiciones con atributos por la izquierda, garantizan que las acciones no hagan referencia a un atributo que aún no haya sido calculado. El ejemplo más sencillo ocurre cuando sólo se necesitan atributos sintetizados, en este caso, se puede construir el esquema de traducción creando una acción que conste de una asignación para cada regla semántica y colocando esta acción al final del lado derecho de la producción asociada.

11. Traducción Descendente

Se trabaja con esquemas de traducción en lugar de hacerlo con definiciones dirigidas por sintaxis, así que se puede ser explícito en cuanto al orden en que tienen que lugar las acciones y las evaluaciones de los atributos.

11.1. Eliminación de la recursividad izquierda de un esquema de traducción

Como la mayoría de los operadores aritméticos son asociativos por la izquierda, es natural utilizar gramáticas recursivas por la izquierda para las expresiones. La transformación se aplica a esquemas de traducción con atributos sintetizados.

Para el análisis sintáctico descendente, se supone que una acción se ejecuta en el mismo momento en que se expandiría un símbolo en la misma posición. Un atributo heredado de un símbolo debe ser calculado por una acción que aparezca antes que el símbolo, y un atributo sintetizado del no terminal de la izquierda se debe calcular después de que hayan sido calculados todos los atributos de los que depende.

11.2. Diseño de un traductor predictivo

Un analizador sintáctico predictivo es un tipo especial de analizador sintáctico descendente recursivo en el que el símbolo del preanálisis determina sin ambigüedad el procedimiento seleccionado para cada no terminal. La secuencia de procedimientos llamados en el procesamiento de la entrada define implícitamente un árbol de análisis sintáctico para la entrada.

La idea es generalizar la construcción de analizadores sintácticos predictivos para implantar un esquema de traducción basado en una gramática adecuada para el análisis sintáctico descendente.

Un atributo heredado de un símbolo debe ser calculado por una acción que aparezca antes que el símbolo, y un atributo sintetizado del no terminal de la izquierda se debe calcular después de que hayan sido calculados todos los atributos de los que depende.

12. Evaluación ascendente a atributos heredados

Existe un método, con el cual se puede implementar cualquier definición de una sintaxis que se base en atributos L, es decir, heredados por la izquierda en gramáticas LL(1) y muchos (no todos) basados en LR(1). Este método consiste en primero tomar el esquema de traducción orientada a sintaxis que posee las acciones semánticas incrustadas y remover dichas acciones, luego se debe insertar en su lugar un no terminal distinto para cada una de las producciones, el cual funciona como marca. Finalmente se agrega una producción extra a partir del no terminal marcador, que derive a épsilon y la acción a ejecutar (la acción removida anteriormente) [1].

Un ejemplo de dicho cambio a realizar, se presenta en la figura siguiente en donde se muestra el esquema original a) y se indica la modificación que se debe realizar b) usando como marcador un no terminal M y N en donde ambos esquemas de traducción aceptan el mismo lenguaje.

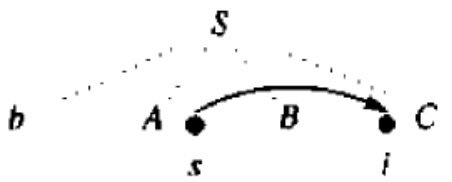
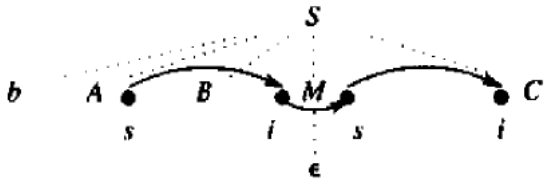
a) Esquema de traducción por orientado a sintaxis
E → T R
R → + T {printf('+')} R | - T {printf('-')} R | ε
T → num {printf(num.val)}

b) Modificación utilizando a M y N como no terminales marcadores
E → T R
R → + T M R | - T N R | ε
T → num {printf(num.val)}
M → ε {printf('+')}
N → ε {printf('-')}

El objetivo del análisis ascendente es realizar reducción, por ejemplo con una producción A → XY, se pretende reemplazar su parte derecha por el no terminal que se encuentra a en la parte izquierda de la producción, esto se puede llevar a acabo, debido a que si el valor Y posee un valor heredado, este estaría determinado por el valor estático de X el cual se encuentra en la pila [1] Se puede determinar que el valor de X.s se encuentra en la pila debido a que se realiza un análisis de abajo hacia arriba y se analizará primero el valor de X el cual determina a valor heredado de Y ya que este último depende de X, como se mostraría en su grafico de dependencia.

El comportamiento de la gramática modificada es el siguiente: el marcador de cada producción toma el valor estático del no terminal que posee a la izquierda como su valor heredado y sintetizado, lo cual permite que el no terminal que se encuentra a la derecha del marcador tome como valor heredado el recién adquirido valor sintetizado del marcador como se muestra en las siguientes figuras.

Producción Original.	Reglas Semánticas.	Producción Modificada.	Reglas Semánticas.
S → aAC S → bABC C →c	C.i = A.s C.i = A.s C.s = g(C.i)	S → aAC S → bABC C →c M → ε	C.i = A.s C.i = A.s C.s = g(C.i) M.s = M.i = A.s

Producción Original.	Dependencias Modificadas.
	

Copia del valor de un atributo a través de un marcador [1].

Gracias al trabajo realizado por las marcas (no terminales) es posible evaluar atributos L durante el análisis sintáctico LR, debido a que solo existe una única producción para cada marcado la gramática continua siendo LL y por ende LR [1], al agregar los marcadores no se produce ningún conflicto. Sin embargo, no se puede decir lo mismo de todas las gramáticas LR por que algunas si podrían presentar problemas de conflictos. A continuación se presenta un ejemplo de una gramática LR que presenta problemas a la hora de realizar el análisis sintáctico debido a que en la última de sus producciones no existe un no terminal con el cual se pueda determinar cual es el valor heredado de L.count a imprimir y con el cual se conoce el total de Ls generadas por S.

Producción.	Reglas Semánticas.
S → L	L.count = 0
L → L_1 1	L.count = L_1.count + 1
L → ε	print(L.count)

Problema con la traducción dirigida por sintaxis [1].

A veces es posible cambiar los valores heredados por valores sintetizados al hacer una pequeña modificación en la gramática, este cambio nos permite trabajar de manera mas eficiente ya que los valores de un atributo para determinado terminal se pueden obtener de manera directa en vez de tener que heredarlo. Ejemplo de la caracterisitica anterior se persenta en el lenguaje de programación de Pascal [1], como se muestra en la figura siguiente donde se trata la producción D que permite definir una variable L con su respectivo tipo T.

Producciones que podría contener la gramática de Pascal y en la cual se utilizarían atributos derivados.	Producciones para el lenguaje Pascal que emplea modificaciones con el fin de utilizar atributos sintetizados.
D → L : T T → integer | char L → L ‘,’ id | L	D → id L L → ‘,’ id L | : T T → integer | char

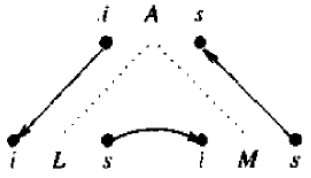
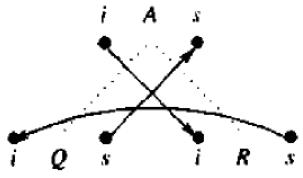
Gramáticas que generan parte del lenguaje de Pascal [1].

La modificación anterir se realiza con el fin de que el tipo de una variable este en el subárbol que genera L, de manera que L posea un atributo de tipo sintetizado que almacena el tipo de la variable del identificador generado por L [1], en vez de tener que heredarlo.

13. Evaluadores recursivos

Existen funciones recursivas que permiten recorrer un árbol de construido en base a una definición dirigida por la sintaxis. Las funciones recorren cada uno de los hijos para un determinado nodo, en algún orden, no necesariamente de izquierda a derecha [1]. Mediante la producción de un nodo la función puede determinar cuales son sus hijos, y retornar un valor para un no terminal A que puede recibir como parámetro algún valor heredado. El valor de retorno es utilizado para calcular el valor del no terminal del nodo en el nivel inmediato superior del árbol. El código correspondiente a cada producción simula la asociación de las reglas semánticas con dicha producción [1], las reglas son aplicadas conforme se calculen los valores heredados que cada no terminal recibe como parámetro.

En la figura que se muestra a continuación se señala la forma en que los nodos hijos de un nodo A del árbol son visitados, basándose en la dependencia del no terminal A.

Producciones.	Reglas Semanticas.	Recorrido para la producción A → L M.	Recorrido para la producción A → Q R.
A → L M A → Q R	L.i = l(A.i) M.i = m(L.s) A.s = f(M.s) R.i = r(A.i) Q.i = q(R.s) A.s = f(Q.s)		

Producciones y reglas semánticas para el no terminal A [1].

Con respecto a al simbología del a figura anterior A.i indica el valor heredado que debe poseer A y A.s representa su valor sintetizado. Note que el orden es determinado por el valor de dependencia.

14. Espacio para el valor de los atributos en tiempo de compilación

El espacio en número de registros que utilizaran los valores de los atributo se puede calcular en tiempo de compilación basándose en el árbol de análisis sintáctico. El tiempo de vida para un atributo inicia cuando es primeramente computado y termina cuando todos los valores que dependen de este son calculados [1], es decir cuando dicho valor es usado por última vez. Lo que resulta de importancia con respecto a la característica anterior es poder mantener ese espacio solo cuando sea necesario y una vez que termine el tiempo de vida para un atributo poder asignar dicho espacio a otro atributo con el cual su tiempo de vida no se traslape con el atributo anterior y de esta manera aprovechar mejor los recursos y cantidad de registros a utilizar.

A continuación se muestra un algoritmo de asignación de registros para el almacenamiento de valores para cada no terminal [1].

```
Para cada nodo m
    Para cada nodo n cuyo tiempo de vida termine con la evaluación de M
        Marque el registro n
    Si existe un registro r marcado entonces
        Desmárquelo
        Evalúe m en el registro r
        Retorne el registro a la piscina
    Si no
        Evalúe m en un registro de la piscina
    /*Las acciones que utilicen el valor de m se pueden insertar en esta sección */
```

```
        Si el tiempo de vida de m finalizó
                Retorne el registro m a la piscina de registros
Fin
```

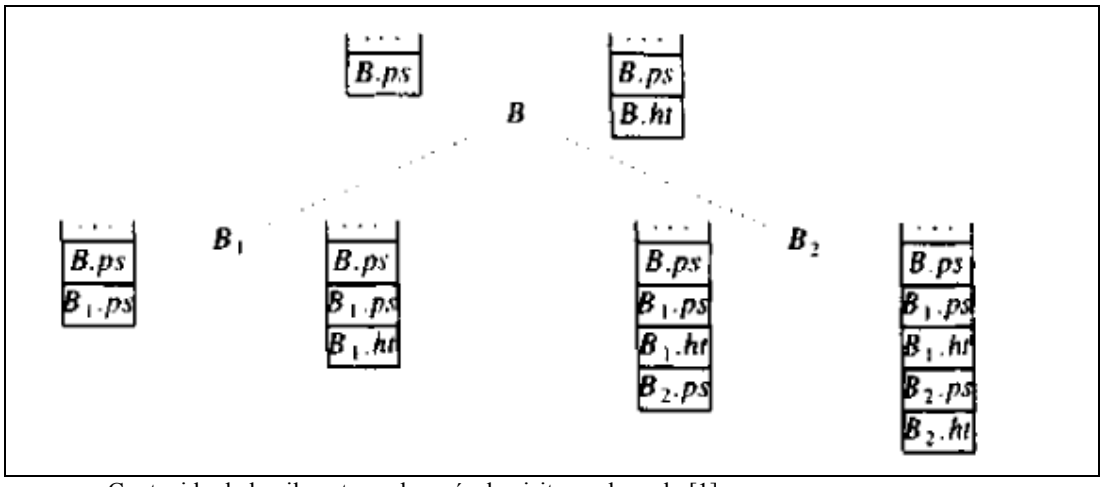
Se puede evitar replicas innecesarias de datos en los registros al detectar reglas semánticas de la forma b = c, debido a que ya existe el valor de c en algún registro no es necesario copiarlo en otro registro, por lo tanto cuando se trata de calcular un nodo m se puede chequear si este ya se encuentra definido mediante una instrucción de copia.

15. Asignación de espacio en tiempo de construcción - compilación

Se puede utilizar una pila para almacenar los valores de cada atributo, incluso no hay necesidad de preocuparse por las replicas innecesarias si los valores se guardan en la pila. Gracias a esta propiedad se puede predecir el tiempo de vida para los valores, sin tomar en cuenta que estos pueden estar compartidos con otras variables, lo que facilita la predicción.

Por ejemplo si se tiene una producción A → B C, para el nodo de A se analizarían tanto su hijo B como C, y una vez finalizado el calculo de A sus valores (los valores de B y C) ya no serian necesarios debido a que el padre de A no puede acceder de forma directa a los valores de B y C, por lo tanto el tiempo de vida para estos dos valores concluiría al calcular el valor de A [1].

Con el fin de Calcular el valor de un no terminal se debe calcular cada uno de los valores que se encuentren en la parte derecha de su producción, por lo tanto se debe considerar la cantidad de atributos para cada no terminal, con el fin de determinar la posición en la pila de cada uno de los valores de sus atributos. Por ejemplo si se posee una producción B → B1 B2 la pila variaría a la hora de realizar el recorrido por el árbol de la siguiente manera:



Contenido de la pila antes y después de visitar cada nodo [1].

16. Análisis de las definiciones dirigidas por sintaxis

Las funciones para un no terminal mapen los valores de los atributos heredados de un nodo con atributos sintetizados. Mientras que para los terminales se debe realizar una función por aparte, es decir distinta de la de los no terminales, sin embargo los terminales se pueden considerar como un solo grupo y ser evaluados con una simple función [1], la agrupación de dichos atributos esta determinada por las dependencias que se presentan con el conjunto de reglas semánticas especificadas en la definición orientada a la sintaxis.

La traducción dirigida a sintaxis es motivada por la sobrecarga de identificadores los cuales tiene un conjunto de posibles tipos y por lo tanto una expresión debe de elegir uno de esos tipos para cada subexpresión que involucre a dicho identificador [1]. Para solucionar dicho problema se debe de realizar

un análisis ascendente para determinar los valores sintetizados y un análisis descendente para seleccionar el tipo adecuado para la expresión. Es decir, la principal implementación del análisis de las traducciones dirigidas a sintaxis es el chequeo de tipos.

Existen definiciones orientadas a sintaxis que se llaman circulares debido a que su grafo de dependencia posee ciclos y no existe ninguna forma para poder calcular los valores de los atributos en el ciclo.

17. Conclusiones

Las traducciones dirigidas por la sintaxis basan su análisis en dos tipos de atributos, lo sintetizados, estos pueden obtenerlos de manera inmediata al calcular los valores de los nodos hijos; y los atributos heredados, los cuales están definidos por los nodos padres o hermanos. Para realizar un análisis sintáctico se basan en la creación de un grafo de dependencias en el cual a la hora de recorrerlo se evalúan cada una de las acciones semánticas que se tienen asignadas para cada nodo respetivo. Este tipo de traducción es útil para determinar el tipo de una variable en una expresión, es decir, el chequeo de tipos; y verificar que este corresponda al que la expresión puede poseer. Se puede trabaja tanto con esquemas de traducción como con definiciones dirigidas por sintaxis, realizando un análisis descendente o ascendente respectivamente, para este ultimo se puede evaluar cualquier gramática LL, sin embargo, no se puede decir lo mismo para las LR por que pueden existir conflictos.

18. Bibliografía

1. Aho, A.V., Sethi, R and Ullman, J. D. *Compilers: Principles, Techniques and Tools*. Pearson Education, primera edición, 1986.