Resumen capitulo 4

nicolas Taveras (1-18-0704)

**UNIVERSIDAD TECNOLOGICA DE SANTIAGO  
Sistema Corporativo  
UTESA**

Facultad de ingeniería y Arquitectura  
Carrera de Ingeniería en Sistemas

Logo

Description automatically generated

**Tema**

Resumen Capitulo 4

**PRESENTADO A:**

Iván Mendoza

**PRESENTADO POR:**

Ramon Nicolas Taveras

1-18-0704

11/06/2023

Santiago, Rep. Dom.

Tabla de contenido

[GitHub Link 1](#_Toc137419090)

[Código y resultados en Consola 2](#_Toc137419091)

[Resumen Capitulo 4 3](#_Toc137419092)

[Analizadores sintácticos 3](#_Toc137419093)

[Gramáticas libres de contexto 4](#_Toc137419094)

[Derivaciones 5](#_Toc137419095)

[Árboles de análisis sintáctico 6](#_Toc137419096)

[Ambigüedad 7](#_Toc137419097)

[Análisis sintáctico descendente y ascendente 8](#_Toc137419098)

[Diseño de gramáticas 9](#_Toc137419099)

[Analizadores sintácticos de descenso recursivo 10](#_Toc137419100)

[Analizadores sintácticos LL(1) 11](#_Toc137419101)

[Análisis sintáctico de desplazamiento-reducción 12](#_Toc137419102)

[Prefijos viables 13](#_Toc137419103)

[Elementos válidos 14](#_Toc137419104)

[Analizadores sintácticos LR 15](#_Toc137419105)

[Analizadores sintácticos LR simples 16](#_Toc137419106)

[Analizadores sintácticos LR canónicos 17](#_Toc137419107)

[Analizadores sintácticos LR con lectura anticipada 18](#_Toc137419108)

[Análisis sintáctico ascendente de gramáticas ambiguas 19](#_Toc137419109)

[Yacc 20](#_Toc137419110)

# GitHub Link

\

<https://github.com/nicotav/analizador-sintactico>

# A screenshot of a computer program Description automatically generated with medium confidenceCódigo y resultados en Consola

También hay otro archivo llamado sintacticoLR que utiliza este mismo método para realizar el análisis sintáctico. Para más información ir al link de GitHub.

# Resumen Capitulo 4

## Analizadores sintácticos

Un analizador sintáctico, también conocido como parser, desempeña un papel crucial en el proceso de análisis de un lenguaje. Recibe como entrada los tokens generados por el analizador léxico y utiliza una gramática libre de contexto para comprender la estructura sintáctica de la secuencia de tokens. El objetivo del analizador sintáctico es construir un árbol de análisis sintáctico que represente la jerarquía y la relación entre los elementos sintácticos de la entrada.

El proceso de construcción del árbol de análisis sintáctico implica la aplicación de reglas gramaticales para combinar los tokens y formar las estructuras sintácticas definidas por la gramática. El árbol de análisis sintáctico puede interpretarse de dos formas: de manera figurada o de forma literal. En la interpretación figurada, el árbol de análisis sintáctico se construye paso a paso, siguiendo las reglas de derivación de la gramática. Cada nodo del árbol representa un no terminal de la gramática y sus hijos representan los símbolos terminales o no terminales que se derivan a partir de ese no terminal. Este enfoque permite visualizar y comprender la estructura de la secuencia de tokens y cómo se combinan para formar construcciones más complejas.

Por otro lado, en la interpretación literal, el árbol de análisis sintáctico se construye como una representación directa de la estructura sintáctica de la entrada. En este caso, cada nodo del árbol representa un elemento sintáctico, como una expresión, una declaración o una estructura de control. Los tokens se asocian directamente con los símbolos terminales correspondientes en el árbol. Esta representación literal del árbol de análisis sintáctico es útil para comprender la semántica y el significado de la entrada en un contexto más amplio.

El árbol de análisis sintáctico es una herramienta valiosa para comprender la estructura de un programa o una expresión en un lenguaje de programación. Permite identificar errores sintácticos, como incompatibilidades de tipos o construcciones incorrectas, y también puede ser utilizado en etapas posteriores del proceso de compilación, como la generación de código intermedio o la optimización.

## Gramáticas libres de contexto

Una gramática es una herramienta fundamental en el estudio de la sintaxis de los lenguajes formales. Esta define las reglas y estructuras que determinan cómo se pueden construir las cadenas válidas en un lenguaje.

Una gramática está compuesta por tres conjuntos principales de símbolos: los símbolos terminales, los no terminales y las producciones.

Los símbolos terminales son los elementos básicos o unidades fundamentales del lenguaje. Representan los elementos concretos que forman parte de las cadenas válidas en el lenguaje en cuestión. Estos símbolos no pueden ser sustituidos por otros símbolos en el proceso de derivación.

Los no terminales, por otro lado, representan las construcciones sintácticas o estructuras más complejas del lenguaje. Estos símbolos pueden ser sustituidos por otros símbolos a través de las producciones de la gramática.

Las producciones son las reglas que definen cómo se pueden construir las cadenas válidas en el lenguaje. Cada producción consta de un encabezado y un cuerpo. El encabezado es un no terminal y representa la construcción que se va a sustituir, mientras que el cuerpo es una cadena de símbolos gramaticales que indica cómo se puede construir o expandir ese no terminal en particular.

En el proceso de derivación, se aplican las producciones de la gramática de manera recursiva para generar nuevas cadenas a partir de los no terminales. Se comienza con el símbolo inicial de la gramática y se sustituyen los no terminales de acuerdo con las producciones hasta obtener una cadena formada únicamente por símbolos terminales.

## Derivaciones

La derivación es un proceso fundamental en el análisis sintáctico, que consiste en reemplazar repetidamente un no terminal por el cuerpo de una de sus producciones de acuerdo con las reglas de una gramática. Este proceso nos permite generar secuencias de símbolos que son válidas en el lenguaje definido por la gramática.

Cuando llevamos a cabo la derivación, podemos elegir si siempre reemplazamos el no terminal más a la izquierda (derivación por la izquierda) o más a la derecha (derivación por la derecha) en cada paso. La elección de derivación por la izquierda o por la derecha puede afectar la estructura y el orden de los símbolos en la derivación resultante.

En la derivación por la izquierda, siempre reemplazamos el no terminal más a la izquierda en cada paso. Esto implica que en cada etapa de la derivación, se reemplaza el símbolo más a la izquierda en la cadena de símbolos. En otras palabras, la sustitución se realiza desde la parte más izquierda de la cadena hacia la derecha.

Por otro lado, en la derivación por la derecha, siempre reemplazamos el no terminal más a la derecha en cada paso. En cada etapa de la derivación, se reemplaza el símbolo más a la derecha en la cadena de símbolos. La sustitución se realiza desde la parte más derecha de la cadena hacia la izquierda.

Es importante tener en cuenta que la elección de derivación por la izquierda o por la derecha puede afectar la forma y el orden de los árboles de análisis sintáctico-resultantes. Una derivación por la izquierda generará un árbol de análisis sintáctico donde las expansiones de los no terminales ocurren desde la raíz hacia las hojas, mientras que una derivación por la derecha generará un árbol de análisis sintáctico donde las expansiones ocurren desde las hojas hacia la raíz.

La elección entre derivación por la izquierda o por la derecha depende del contexto y de la gramática en cuestión. Algunas gramáticas se diseñan naturalmente para ser derivadas por la izquierda, mientras que otras se diseñan para ser derivadas por la derecha. La elección también puede tener implicaciones en la eficiencia y la complejidad de los algoritmos de análisis sintáctico utilizados para procesar la gramática.

## Árboles de análisis sintáctico

Un árbol de análisis sintáctico es una representación gráfica de una derivación en la cual cada nodo representa un no terminal y sus hijos representan los símbolos utilizados para reemplazar ese no terminal en la derivación. Estos árboles son útiles para visualizar la estructura jerárquica de una cadena de terminales de acuerdo con las reglas de una gramática.

En un árbol de análisis sintáctico, la raíz del árbol corresponde al símbolo inicial de la gramática, y cada nodo no terminal representa un paso en la derivación. Los hijos de un nodo no terminal son los símbolos que se utilizan para expandir ese no terminal en la derivación. Los nodos hoja del árbol representan los símbolos terminales de la cadena de entrada.

Es importante destacar que existe una correspondencia de uno a uno entre los árboles de análisis sintáctico, las derivaciones por la izquierda y las derivaciones por la derecha de la misma cadena de terminales. Una derivación por la izquierda implica que en cada paso se reemplaza el lado más izquierdo de la producción, mientras que una derivación por la derecha implica que en cada paso se reemplaza el lado más derecho de la producción. Ambos tipos de derivaciones generan el mismo árbol de análisis sintáctico para una cadena de entrada dada.

Esta correspondencia entre árboles de análisis sintáctico y derivaciones es útil en el proceso de análisis sintáctico, ya que permite comprender la estructura y la relación entre los símbolos no terminales y terminales en la gramática. Los árboles de análisis sintáctico también son utilizados para verificar la validez sintáctica de una cadena de entrada y para identificar posibles errores o ambigüedades en la gramática.

## Ambigüedad

Una gramática se considera ambigua cuando una cadena de terminales puede tener dos o más árboles de análisis sintáctico distintos o, en otras palabras, cuando existen múltiples formas de derivar la cadena tanto por la izquierda como por la derecha. La ambigüedad en una gramática puede dar lugar a diferentes interpretaciones o significados para una misma cadena, lo cual puede ser problemático en el proceso de análisis sintáctico.

En muchos casos, especialmente en aplicaciones prácticas, se prefiere trabajar con gramáticas libres de ambigüedad para evitar confusiones o inconsistencias en la interpretación del lenguaje. Por lo tanto, es común realizar modificaciones en una gramática ambigua para convertirla en una gramática sin ambigüedad que genere el mismo lenguaje.

Existen varias técnicas para eliminar la ambigüedad en una gramática. Una de las técnicas más utilizadas es la eliminación de la recursividad por la izquierda y la factorización por la izquierda. La recursividad por la izquierda ocurre cuando un no terminal puede derivar directamente a sí mismo, lo cual puede dar lugar a múltiples derivaciones por la izquierda. Al eliminar la recursividad por la izquierda, se asegura que solo se realicen derivaciones por la derecha, lo que contribuye a la desambiguación de la gramática. La factorización por la izquierda consiste en identificar producciones que tienen un prefijo común en el cuerpo y reescribir esas producciones de manera que se elimine la ambigüedad.

Sin embargo, es importante destacar que en algunas ocasiones las gramáticas ambiguas pueden conducir a la generación de analizadores sintácticos más eficientes. Esto se logra mediante la incorporación de ciertos trucos o técnicas específicas en el diseño del analizador sintáctico. Estos trucos aprovechan la ambigüedad controlada de la gramática para simplificar la estructura del analizador y mejorar su rendimiento. Aunque estos analizadores pueden ser más complejos de implementar, en determinados casos pueden ofrecer ventajas en términos de velocidad o consumo de recursos.

## Análisis sintáctico descendente y ascendente

Los analizadores sintácticos se clasifican generalmente en dos categorías principales: analizadores sintácticos descendentes y analizadores sintácticos ascendentes. Esta clasificación se basa en la dirección en la que se construye el árbol de análisis sintáctico.

Los analizadores sintácticos descendentes se caracterizan por trabajar de arriba hacia abajo. Comienzan con el símbolo inicial de la gramática y construyen el árbol de análisis sintáctico partiendo de la parte superior. En este enfoque, se realiza una expansión de los no terminales en la gramática hasta llegar a las hojas correspondientes a los símbolos terminales. Los analizadores sintácticos descendentes pueden implementarse utilizando diferentes técnicas, como el análisis sintáctico descendente recursivo y el análisis sintáctico descendente predictivo (LL).

El análisis sintáctico descendente recursivo se basa en procedimientos recursivos que corresponden a los no terminales de la gramática. Cada procedimiento es responsable de analizar su no terminal correspondiente y decidir qué producción aplicar. Este enfoque es intuitivo y fácil de entender, pero puede tener dificultades con gramáticas ambiguas o con la necesidad de realizar retroceso (backtracking) en caso de elecciones incorrectas.

Por otro lado, el análisis sintáctico descendente predictivo (LL) se basa en una tabla de análisis sintáctico predictivo que proporciona, para cada no terminal y cada símbolo de preanálisis, la elección de la producción correcta. Estas tablas se construyen a partir de una gramática LL(1), que es una gramática en la que se puede elegir la producción correcta con solo analizar el siguiente símbolo de entrada. El análisis sintáctico LL es eficiente y se utiliza ampliamente en compiladores y herramientas de procesamiento de lenguaje.

En contraste, los analizadores sintácticos ascendentes construyen el árbol de análisis sintáctico de abajo hacia arriba. Comienzan con los símbolos terminales que forman las hojas del árbol y construyen el árbol hacia arriba hasta llegar al símbolo inicial de la gramática. Los analizadores sintácticos ascendentes más comunes son los analizadores sintácticos LR, que utilizan una tabla de análisis sintáctico LR para guiar las reducciones y desplazamientos en función del contenido de la pila y el siguiente símbolo de entrada. Los analizadores sintácticos LR son poderosos y pueden manejar gramáticas más complejas que los analizadores sintácticos descendentes, pero su implementación puede ser más compleja.

## Diseño de gramáticas

En el diseño de gramáticas para el análisis sintáctico descendente, a menudo nos enfrentamos a desafíos adicionales en comparación con las gramáticas utilizadas por los analizadores sintácticos ascendentes. Esto se debe a que en el análisis sintáctico descendente, es necesario evitar la recursividad por la izquierda y la factorización por la izquierda, lo que puede complicar el proceso de diseño.

La recursividad por la izquierda es una situación en la que un no terminal deriva a una cadena que comienza con el mismo no terminal. Esta forma de recursividad puede causar problemas en el análisis sintáctico descendente, ya que puede conducir a una recursión infinita y dificultar la determinación de qué producción aplicar. Para eliminar la recursividad por la izquierda, se pueden aplicar técnicas como la eliminación directa de recursividad o la transformación en recursividad por la derecha. Estas técnicas reescriben las producciones de manera que él no terminal recursivo aparezca en la parte derecha de las producciones, evitando así la recursión por la izquierda.

Otro desafío común es la factorización por la izquierda, que ocurre cuando hay varias producciones para el mismo no terminal que tienen un prefijo común en el cuerpo. Esto puede llevar a conflictos durante el análisis sintáctico, ya que no se puede determinar con certeza qué producción aplicar basándose solo en el siguiente símbolo de entrada. Para abordar este problema, se puede realizar una factorización por la izquierda, que consiste en dividir las producciones en grupos más pequeños y factorizar el prefijo común en una producción separada. Esto permite una elección clara y sin ambigüedades durante el análisis sintáctico.

El proceso de eliminación de la recursividad por la izquierda y la factorización por la izquierda puede ser desafiante y requerir un análisis detallado de la gramática. Es importante encontrar un equilibrio entre la expresividad y la simplicidad de la gramática, ya que una gramática demasiado compleja puede dificultar la comprensión y el mantenimiento del analizador sintáctico descendente.

## Analizadores sintácticos de descenso recursivo

Los analizadores sintácticos predictivos utilizan un procedimiento separado para cada no terminal de la gramática. Estos procedimientos son responsables de analizar la entrada correspondiente al no terminal y tomar decisiones sobre qué producción aplicar. Cada procedimiento examina el siguiente símbolo de entrada y utiliza esta información para determinar la producción adecuada.

Durante el análisis, los terminales en el cuerpo de una producción se relacionan con los símbolos de la entrada en el momento oportuno. Esto implica comparar el símbolo de entrada actual con los terminales en el cuerpo de la producción y avanzar al siguiente símbolo de entrada si hay una coincidencia.

Por otro lado, cuando un no terminal aparece en el cuerpo de una producción, se realiza una llamada al procedimiento correspondiente a ese no terminal. Esta llamada permite analizar la entrada asociada a ese no terminal y decidir qué producción aplicar para expandirlo.

En algunos casos, puede ocurrir que se elija la producción incorrecta inicialmente. En tales situaciones, el analizador puede necesitar realizar un rastreo hacia atrás para intentar con otra producción. Esto implica retroceder en la entrada y probar diferentes producciones hasta encontrar una que sea compatible con la entrada restante. El rastreo hacia atrás puede ser necesario cuando hay ambigüedad en la gramática o cuando se encuentran conflictos durante el análisis.

El uso de procedimientos separados para cada no terminal permite una organización clara y modular del análisis sintáctico. Cada procedimiento se encarga de analizar un aspecto específico de la gramática y toma decisiones basadas en el siguiente símbolo de entrada. Esto facilita el diseño y la implementación de analizadores sintácticos predictivos.

Sin embargo, el rastreo hacia atrás puede ser costoso en términos de rendimiento, ya que implica revisar múltiples producciones y retroceder en la entrada. En algunos casos, se pueden aplicar técnicas de optimización para evitar o reducir la necesidad de rastreo hacia atrás.

## Analizadores sintácticos LL(1)

Una gramática LL(1) es aquella en la cual se puede determinar la producción correcta para expandir un no terminal específico con solo analizar el siguiente símbolo de entrada. Estas gramáticas son especialmente útiles en el diseño de analizadores sintácticos predictivos, ya que permiten construir una tabla de análisis sintáctico que proporciona la elección adecuada de producción para cada no terminal y símbolo de preanálisis.

La tabla de análisis sintáctico predictivo es una estructura de datos que se utiliza para guiar el análisis sintáctico en tiempo real. Por cada combinación de no terminal y símbolo de preanálisis, la tabla indica qué producción debe ser aplicada. Esto se logra examinando el siguiente símbolo de entrada y consultando la tabla correspondiente.

La construcción de la tabla de análisis sintáctico predictivo se basa en la observación de que, en una gramática LL(1), no puede haber ambigüedad o conflictos en la elección de las producciones. Cada combinación de no terminal y símbolo de preanálisis debe tener una única producción asociada.

Sin embargo, en ocasiones puede ocurrir que no exista una producción legítima para cierta combinación de no terminal y símbolo de preanálisis, lo que puede indicar un error en la entrada. Para facilitar la corrección de errores, se pueden incluir rutinas de error en la tabla de análisis sintáctico predictivo. Estas rutinas de error pueden estar presentes en algunas o en todas las entradas de la tabla que no tienen una producción legítima. Al activarse, estas rutinas pueden proporcionar mensajes de error útiles para guiar al programador o al usuario en la identificación y corrección del error.

La propiedad de ser LL(1) es altamente deseable en el diseño de gramáticas y analizadores sintácticos, ya que garantiza un análisis sintáctico eficiente y sin ambigüedades. Sin embargo, no todas las gramáticas pueden ser convertidas a una forma LL(1). En casos en los que una gramática no es LL(1), se pueden aplicar técnicas de reescritura o factorización para resolver los conflictos y hacerla compatible con el análisis sintáctico predictivo.

## Análisis sintáctico de desplazamiento-reducción

Los analizadores sintácticos ascendentes, también conocidos como analizadores de reducción-desplazamiento o analizadores bottom-up, son utilizados para analizar gramáticas libres de contexto. Estos analizadores operan tomando decisiones basadas en el siguiente símbolo de entrada, también conocido como símbolo de anticipación, y el contenido actual de la pila. Durante el análisis sintáctico ascendente, el analizador se esfuerza por construir la derivación inversa de la entrada, es decir, la secuencia de producciones utilizadas para generar la cadena de entrada. El objetivo es reducir la pila desde la parte superior hasta que solo quede un símbolo no terminal en la pila, que corresponderá al símbolo inicial de la gramática.

En cada paso del análisis, el analizador debe tomar una decisión: desplazar el siguiente símbolo de entrada hacia la pila o reducir algunos símbolos en la parte superior de la misma. La elección se basa en el símbolo de anticipación y el contenido actual de la pila.

Cuando se decide desplazar, el analizador toma el siguiente símbolo de entrada y lo coloca en la pila. Esto indica que se ha reconocido ese símbolo en la entrada y se avanza hacia la construcción de la derivación.

Por otro lado, cuando se realiza una reducción, el analizador toma una producción de la gramática que coincide con una parte de la pila y la sustituye por el símbolo no terminal correspondiente al encabezado de esa producción. Esta reducción refleja que se ha reconocido una secuencia de símbolos en la entrada que coincide con la parte derecha de la producción, y se reemplaza por el símbolo no terminal correspondiente a la parte izquierda de la producción.

La elección entre desplazamiento y reducción se realiza en función de la estructura de la gramática y el contenido de la pila. El analizador busca patrones que coincidan con las producciones de la gramática y realiza reducciones cuando encuentra una coincidencia.

El proceso de reducción permite que el analizador construya progresivamente la derivación inversa de la entrada, reduciendo símbolos en la pila hasta que solo quede el símbolo inicial. En cada paso, se utilizan las producciones de la gramática para reescribir la pila y acercarse a la construcción de la derivación completa.

## Prefijos viables

En el análisis sintáctico de desplazamiento-reducción, la pila siempre contiene un prefijo viable durante el proceso de análisis. Un prefijo viable es una porción de la frase derecha que se encuentra a la derecha, pero no más allá del final del mango en la derivación actual.

El mango se refiere a la subcadena que fue introducida en el último paso de la derivación por la derecha de la forma de frase actual. Representa la parte de la frase que ya ha sido analizada y reconocida correctamente.

La pila en el análisis sintáctico de desplazamiento-reducción se utiliza para mantener un registro de los símbolos que se han encontrado hasta el momento durante el análisis. Estos símbolos corresponden al prefijo viable, es decir, a la porción de la frase derecha que se ha analizado exitosamente.

A medida que se procesa la entrada, se realizan desplazamientos (shifts) y reducciones (reductions) en el análisis sintáctico. Durante el desplazamiento, se toma un símbolo de la entrada y se coloca en la pila, lo que refleja que ese símbolo ha sido reconocido correctamente hasta ese punto. Durante la reducción, se reemplazan varios símbolos en la pila por un símbolo no terminal, indicando que se ha reconocido una producción completa de la gramática.

La pila se actualiza continuamente a medida que se realizan desplazamientos y reducciones, y siempre mantiene un prefijo viable. Esto significa que los símbolos en la pila corresponden a una porción válida de la frase derecha y están correctamente alineados con el mango, que es la parte analizada hasta el momento.

El seguimiento y mantenimiento de un prefijo viable en la pila es crucial para asegurar un análisis sintáctico correcto. Permite al analizador reconocer la estructura de la frase y realizar las acciones adecuadas en función de los elementos en la pila y los símbolos de entrada.

## Elementos válidos

En el contexto de los analizadores sintácticos LR, un elemento se refiere a una producción gramatical con un punto en alguna posición dentro del cuerpo de la producción. Este punto indica la posición actual en la derivación de esa producción y representa el progreso del analizador sintáctico durante el análisis.

Un elemento se considera válido para un prefijo viable si la producción asociada a ese elemento se utiliza para generar la parte "mango" de la derivación actual. El mango es la secuencia de símbolos que han sido derivados hasta el punto actual en el análisis sintáctico.

Además, para que un elemento sea válido para un prefijo viable, el prefijo viable debe incluir todos los símbolos a la izquierda del punto del elemento, pero no debe incluir los símbolos que están debajo del punto. Esto significa que el analizador sintáctico ha reconocido exitosamente la parte izquierda del elemento en la entrada y está esperando analizar los símbolos que siguen a ese punto.

Al considerar los elementos válidos para un prefijo viable, el analizador sintáctico toma decisiones sobre las reducciones y desplazamientos en función de la información proporcionada por estos elementos. Por ejemplo, si un elemento válido para el prefijo viable permite una reducción, el analizador puede aplicar esa reducción y reemplazar el prefijo viable con el símbolo no terminal correspondiente a la parte izquierda de la producción.

La noción de elementos válidos y prefijos viables es fundamental en los analizadores sintácticos LR, ya que ayuda a determinar el siguiente movimiento a realizar durante el análisis sintáctico. Al construir conjuntos de elementos válidos para cada estado del analizador, se puede construir una tabla de análisis sintáctico que especifica las acciones (desplazamientos o reducciones) a tomar en función del símbolo de entrada y el estado actual del analizador.

## Analizadores sintácticos LR

Cada uno de los diversos tipos de analizadores sintácticos LR opera construyendo conjuntos de elementos válidos, también conocidos como estados LR, para todos los prefijos viables posibles en la gramática. Estos estados LR representan las diferentes configuraciones de análisis en el proceso de reconocimiento de una cadena de entrada.

Durante el análisis sintáctico LR, se lleva un registro de los estados en una pila. Cada estado en la pila corresponde a un preﬁjo-viable que se ha encontrado hasta el momento. A medida que se procesa la entrada, los analizadores sintácticos LR utilizan el conjunto de elementos válidos para guiar la toma de decisiones entre desplazamientos (shifts) y reducciones (reductions).

Cuando se encuentra un elemento válido con el punto en el extremo derecho del cuerpo de una producción, es decir, un elemento que está listo para ser reducido, el analizador sintáctico LR tiene la opción de realizar una reducción. Por otro lado, si el símbolo de anticipación se encuentra justo a la derecha del punto en algún elemento válido, el analizador puede elegir desplazar el símbolo de anticipación hacia la pila.

En general, en un analizador sintáctico LR se prefiere la reducción sobre el desplazamiento cuando hay un elemento válido que puede ser reducido. Esto se debe a que una reducción representa la aplicación de una producción de la gramática y permite construir la estructura sintáctica de manera progresiva.

Por otro lado, el desplazamiento implica tomar un símbolo de entrada y colocarlo en la pila para su posterior análisis. Esto sucede cuando el símbolo de anticipación se encuentra justo a la derecha del punto en algún elemento válido, lo que indica que ese símbolo es esperado en la entrada.

El uso de conjuntos de elementos válidos y el seguimiento de los estados en la pila permiten que el analizador sintáctico LR tome decisiones informadas durante el análisis. Estas decisiones se basan en la estructura y las reglas de la gramática, así como en la posición de los elementos y los símbolos de anticipación en el análisis.

## Analizadores sintácticos LR simples

En un analizador sintáctico SLR (Simple LR), se realiza una reducción cuando se encuentra un elemento válido con un punto en el extremo derecho de una producción y el símbolo de anticipación se encuentra en el conjunto de símbolos que pueden seguir al encabezado de esa producción en alguna forma de frase.

El enfoque SLR se basa en la construcción de una tabla de análisis sintáctico que contiene información sobre los movimientos de desplazamiento (shift) y reducción (reduce) del analizador. Cada celda de la tabla indica la acción a seguir para un estado y un símbolo de entrada particular.

Para que el método SLR sea aplicable, es necesario que la gramática cumpla con ciertas condiciones. Es decir, no debe haber conflictos de acciones de análisis sintáctico en la gramática. Esto significa que no puede haber situaciones en las que existan dos producciones diferentes que permitan realizar una reducción, ni puede haber opciones ambiguas entre reducir o desplazar.

La construcción de una tabla SLR implica calcular conjuntos de elementos para cada estado, los cuales representan las posibles configuraciones de análisis en ese estado. Estos conjuntos se utilizan para determinar las acciones de análisis sintáctico en la tabla.

Si no hay conflictos de acciones de análisis sintáctico en la gramática, es decir, si para ningún conjunto de elementos y ningún símbolo de anticipación hay opciones ambiguas de reducción o desplazamiento, entonces el método SLR puede aplicarse correctamente para realizar el análisis sintáctico.

El enfoque SLR es considerado una versión más simple de los analizadores sintácticos LR (Left-to-right, Rightmost derivation), ya que no considera toda la información de anticipación disponible en el análisis. Sin embargo, a pesar de su simplicidad, los analizadores SLR son ampliamente utilizados debido a su eficiencia y facilidad de implementación.

## Analizadores sintácticos LR canónicos

Los analizadores sintácticos LALR (Look-Ahead LR) son una variante de los analizadores sintácticos LR canónicos que ofrecen muchas ventajas similares a los analizadores sintácticos SLR (Simple LR). La principal característica de los analizadores LALR radica en la combinación de estados que tienen los mismos "corazones" o núcleos, es decir, conjuntos de elementos, ignorando los conjuntos asociados de símbolos de anticipación.

Esta combinación de estados en los analizadores LALR permite reducir la cantidad total de estados en comparación con los analizadores sintácticos LR canónicos, y es igual al número de estados en el analizador sintáctico SLR. Esto significa que los analizadores LALR son más compactos y requieren menos recursos de memoria en comparación con los analizadores sintácticos LR canónicos.

Además, una ventaja significativa de los analizadores LALR radica en su capacidad para eliminar algunos conflictos de acciones de análisis sintáctico-presentes en los analizadores sintácticos SLR. Estos conflictos de acciones pueden surgir cuando el analizador se enfrenta a decisiones ambiguas durante el análisis sintáctico, lo que puede llevar a problemas de desplazamiento/reducción o reducción/reducción.

Mediante la combinación de estados y la reducción de la cantidad de información asociada con los conjuntos de símbolos de anticipación, los analizadores LALR pueden resolver algunos de estos conflictos de manera más eficiente. Esto se logra al agrupar estados con núcleos similares y permitir una mayor generalización en las decisiones de análisis sintáctico.

Como resultado, los analizadores sintácticos LALR se han convertido en el método más utilizado para el análisis sintáctico en muchos compiladores y herramientas relacionadas. Proporcionan un equilibrio entre la potencia de los analizadores LR canónicos y la simplicidad de los analizadores SLR, ofreciendo una buena eficiencia en el uso de recursos y una resolución mejorada de conflictos.

## Analizadores sintácticos LR con lectura anticipada

Los analizadores sintácticos LALR (Look-Ahead LR) son una variante de los analizadores sintácticos LR canónicos que ofrecen muchas ventajas similares a los analizadores sintácticos SLR (Simple LR). La principal característica de los analizadores LALR radica en la combinación de estados que tienen los mismos "corazones" o núcleos, es decir, conjuntos de elementos, ignorando los conjuntos asociados de símbolos de anticipación.

Esta combinación de estados en los analizadores LALR permite reducir la cantidad total de estados en comparación con los analizadores sintácticos LR canónicos, y es igual al número de estados en el analizador sintáctico SLR. Esto significa que los analizadores LALR son más compactos y requieren menos recursos de memoria en comparación con los analizadores sintácticos LR canónicos.

Además, una ventaja significativa de los analizadores LALR radica en su capacidad para eliminar algunos conflictos de acciones de análisis sintáctico-presentes en los analizadores sintácticos SLR. Estos conflictos de acciones pueden surgir cuando el analizador se enfrenta a decisiones ambiguas durante el análisis sintáctico, lo que puede llevar a problemas de desplazamiento/reducción o reducción/reducción.

Mediante la combinación de estados y la reducción de la cantidad de información asociada con los conjuntos de símbolos de anticipación, los analizadores LALR pueden resolver algunos de estos conflictos de manera más eficiente. Esto se logra al agrupar estados con núcleos similares y permitir una mayor generalización en las decisiones de análisis sintáctico.

Como resultado, los analizadores sintácticos LALR se han convertido en el método más utilizado para el análisis sintáctico en muchos compiladores y herramientas relacionadas. Proporcionan un equilibrio entre la potencia de los analizadores LR canónicos y la simplicidad de los analizadores SLR, ofreciendo una buena eficiencia en el uso de recursos y una resolución mejorada de conflictos.

## Análisis sintáctico ascendente de gramáticas ambiguas

En muchas situaciones importantes, como el análisis sintáctico de expresiones aritméticas, es común encontrarse con gramáticas ambiguas. Sin embargo, gracias a la capacidad de incorporar información adicional, como la precedencia de los operadores, es posible resolver los conflictos que surgen durante el análisis sintáctico.

El análisis sintáctico LR (Left-to-right, Rightmost derivation) es una técnica ampliamente utilizada para realizar el análisis sintáctico de gramáticas libres de contexto. A través de la construcción de una tabla de análisis, el análisis sintáctico LR permite identificar los movimientos de desplazamiento (shift) y reducción (reduce) necesarios para reconocer una cadena de entrada y construir la estructura sintáctica correspondiente.

Cuando se trabaja con gramáticas ambiguas, los conflictos entre desplazar y reducir, o entre reducciones diferentes, pueden surgir. Estos conflictos ocurren cuando la gramática permite múltiples interpretaciones posibles para una cadena de entrada.

Sin embargo, las técnicas de análisis sintáctico LR tienen la capacidad de extenderse y abordar gramáticas ambiguas al aprovechar información adicional, como la precedencia de los operadores. Esta información se utiliza para establecer reglas claras y deterministas para resolver los conflictos. Por ejemplo, en el caso de las expresiones aritméticas, es común asignar precedencias a los operadores y especificar si una reducción tiene prioridad sobre un desplazamiento.

Al utilizar la información de precedencia de operadores, el analizador sintáctico LR puede tomar decisiones adecuadas para resolver los conflictos de manera consistente. Esto permite obtener un análisis sintáctico correcto y determinista, incluso en presencia de gramáticas ambiguas.

## Yacc

El generador de analizadores sintácticos Yacc (Yet Another Compiler Compiler) es una herramienta ampliamente utilizada para generar analizadores sintácticos en diversos lenguajes de programación. Su función principal es recibir una gramática, que puede ser (posiblemente) ambigua, junto con la información de resolución de conflictos, y construir los estados del LALR (Look-Ahead LR).

El proceso de construcción de los estados LALR implica generar un conjunto de elementos de análisis conocidos como "items" que representan los posibles estados de una gramática. Estos estados contienen información sobre las producciones gramaticales, los símbolos de entrada y las posiciones dentro de las producciones. Utilizando estos items, Yacc crea un conjunto de estados que describen las diferentes configuraciones de análisis posibles en el proceso de análisis sintáctico.

Una vez que se han generado los estados LALR, Yacc produce una función en el lenguaje de destino que utiliza estos estados para realizar un análisis sintáctico ascendente, también conocido como análisis LR. El análisis sintáctico ascendente se basa en la construcción de una pila (stack) y un buffer de entrada para procesar los tokens de entrada y reducirlos a producciones gramaticales.

Durante el análisis, cuando ocurre una reducción, es decir, cuando se encuentra una coincidencia entre los elementos de la pila y una producción gramatical, Yacc llama a una función asociada a esa reducción. Esta función puede ser personalizada por el programador y se utiliza para realizar acciones semánticas relacionadas con la reducción específica que ha ocurrido.

La capacidad de definir acciones semánticas en Yacc permite que el analizador sintáctico realice tareas adicionales más allá de la simple validación de la estructura sintáctica. Por ejemplo, puede utilizarse para construir un árbol de análisis sintáctico o para ejecutar acciones que generen código intermedio o realicen verificaciones semánticas más complejas.