Instituto Tecnológico de Costa Rica

Escuela de Computación

IC5701 Compiladores e Intérpretes

Apuntes de clase
Fecha: 26 de abril del 2017

Realizado por:
Adrián Jesús Álvarez Calderón
2014071059

Profesor:
Francisco José Torres Rojas

I Semestre 2017

Contenido

… Final del Análisis Léxico

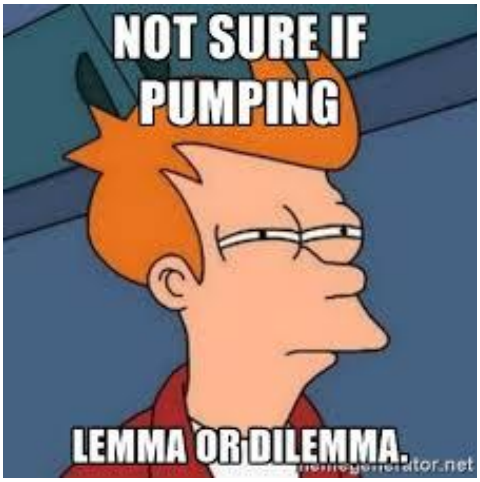
Pumping Lemma

Introducción:

Considere el siguiente lenguaje:

$$Sea\ \mathcal{L}\ el\ lenguaje\ sobre \sum = \{a, b\}\ definido\ como\ \{a^n b^n | n \geq 0\}$$

Diseñe un DFA que reconozca $\mathcal{L}$:



Es complicado diseñar un autómata de **estados finitos** que reconozca este lenguaje. Recuerde que podríamos decir que un lenguaje es regular si existe algún DFA (NFA, NFA- ε) o expresión regular que lo reconozca, empero, aunque esto es cierto, se requiere de algún argumento más sólido para sustentar la contraposición de esta afirmación, algo que sea demostrable.

Por lo que intuitivamente pensaríamos que no existe un autómata, pero no podemos darnos el lujo de que esa sea una justificación para decir que el lenguaje no es regular, puede que su manquedad contribuya a que no puede concebir un autómata para el lenguaje y tal vez éste si exista. Así que, según F.T, encontramos el primer lenguaje satánico.

Lenguajes No Regulares:

Como la intuición lo indica, existen lenguajes que no son regulares. Haciendo spoiler, el lenguaje anteriormente descrito efectivamente es un lenguaje no regular, por lo ahora podemos creer que realmente existen los lenguajes no regulares.
Esto incluye una implicación especial y es que entonces no existe algún DFA (NFA, NFA- ε) o expresión regular que lo reconozca. Pero para demostrar que un lenguaje no es regular se requiere algo más que esta afirmación para indicarlo, por lo tanto, se requiere de una demostración completa, pero para eso existe *Pumping Lemma*.

Aceptando una hilera de longitud n:

$Sea\ M = (Q, \sum, \delta\ , q_0, F)\ un\ DFA$

Siendo Q el conjunto finito de estados, ∑ el alfabeto del autómata, δ: Qx∑ -> Q la función de transición, q_0 el estado inicial del autómata, y F un subconjunto de Q el cual representa todos los estados de aceptación existentes en el autómata.

Cuando se tiene un n símbolos, se llega a visitar $n+1$ estados y estos no necesariamente diferentes. Considere el caso de épsilon y el autómata sí acepta épsilon, si la hilera no tiene símbolos se llegan a visitar 1 símbolo, razón por la cual se da la visita del n (según la cantidad de símbolos) más el estado inicial.

$Sea\ w = w_1, w_2, \dots, w_n\ una\ hilera\ de\ longitud\ n\ sobre\ \sum$

La definición de aceptación de una hilera corresponde a que M acepta w, si existe una secuencia de estados $r_0, r_1, r_2, \dots, r_n$ (n+1 estados) de manera tal que:

1. Todos los estados r_i pertenecen a Q.
2. $r_0 = q_0$ El primer estado es el de aceptación
3. $\delta(r_i, w_{i+1}) = r_{i+1}\ para\ i = 0, \dots, n-1$
4. $r_n \in F$ El último estado es un estado de aceptación

Lenguajes Finitos e Infinitos:

Se concibe de manera trivial que todo lenguaje finito como regular. No obstante, si un lenguaje fuera "no regular", debe ser infinito como axioma, esto claramente no implica que un lenguaje por ser infinito es necesariamente no regular, pues existen lenguajes regulares que son infinitos, los cuales hemos visto en clase algunos ejemplares de estos. Por lo que los lenguajes infinitos representan un tema de interés nuestro. Estos lenguajes infinitos poseen hileras de longitud arbitraria, por ende, de longitud infinita.

Lenguajes Regulares Infinitos:

$Sea\ \mathcal{L}\ un\ lenguaje\ regular\ infinito$

Como $\mathcal{L}$ es infinito entonces tiene hileras de longitud arbitraria, y como $\mathcal{L}$ es regular se sabe que existe un DFA (NFA, NFA- ε) o expresión regular que lo reconozca. Este DFA se caracteriza por tener k estados siendo este k finito. ¿Es posible que existan hileras de $\mathcal{L}$ que longitud $n \geq k$?, claro, pues $\mathcal{L}$ es infinito.

Aceptando una hilera de longitud $n \geq k$:

$Sea\ \mathcal{L}\ un\ lenguaje\ regular\ infinito\ reconocido\ por\ un\ DFA\ de\ k\ estados$

$Sea\ w\ \in\ \mathcal{L}\ una\ hilera\ w, n = |w|\ \ tal\ que\ n \geq k$

Cuando se llega a procesar y aceptar a la hilera w, dado un DFA que reconoce el lenguaje $\mathcal{L}$ ve van a llegar a visitar $n + 1$ estados, dado que $n + 1 > k$, entonces se visita por lo menos 2 veces un mismo estado, esto es por el principio del palomar.

Esto se da ya que se si se llega a visitar $n+1$ estados, siendo $n \geq k$ implica que mínimo se pasó dos veces por un mismo estado, pero pueden ser más veces.

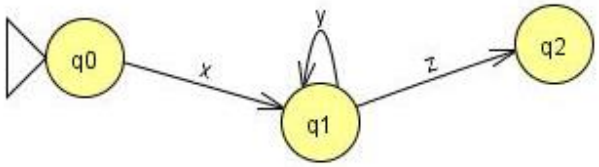
Hileras de longitud $n \geq k$ y ciclos en DFA:

$Sea\ \mathcal{L}\ un\ lenguaje\ regular\ infinito\ reconocido\ por\ un\ DFA\ de\ k\ estados$

$Sea\ w\ \in \mathcal{L}, una\ hilera\ w, n = |w|\ tal\ que\ n \geq k$

Ya que $w\ \in \mathcal{L}$, existe una ruta en el DFA que va desde el estado inicial hasta un estado de aceptación, proceso en el cual se consume los símbolos de w. Este proceso incluye que mínimo se pasó 2 veces por un mismo estado. Con esta información se puede llegar a concluir que esta ruta contiene por lo menos un ciclo en el grafo del DFA que reconoce dicho lenguaje.

Cuando se procesa un $w\ \in \mathcal{L}$, la ruta tiene un ciclo si $|w| > k$. En este DFA se pueden distinguir tres partes o componentes principales, que son consecutivos y son la representación de w. Siendo que $w = x\ y\ z\ /\in\ \mathcal{L}$



Como se aprecia en este ejemplo muy simple, y sin importar la cantidad de estados, se aprecia la existencia de una primera parte x, una segunda par y que es cíclica y una parte z, estas partes de manera consecutiva, que representan al diagrama.

"Y aquí viene la magia", F.T. El secreto y detalle más relevante es que el componente "y" puede estar, no estar o bien estar muchas veces. Por lo que:

Dado que $w = x\ y\ z\ \in \mathcal{L}$,

Si se quita "y" sigue perteneciendo a $\mathcal{L}$ pero también llega a un estado de aceptación.

Si se pone n veces y, por ejemplo $w = x\ y\ ...\ y\ z$, sigue perteneciendo a $\mathcal{L}$, no obstante, también logra llegar a un estado de aceptación.

Pumping Lemma:

Por fin se va a definir formalmente, no sin antes recordar que era.

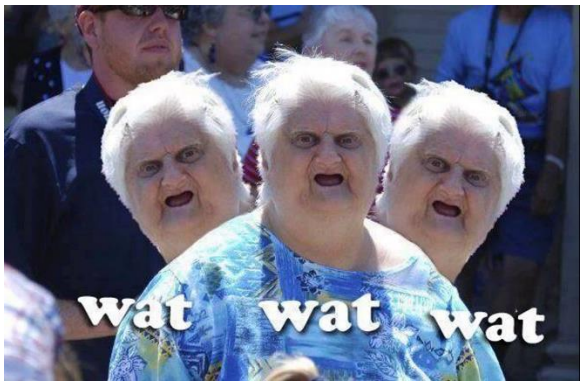
Provee un mecanismo para demostrar y determinar que un lenguaje no es regular.

Sea $\mathcal{L}$ un lenguaje regular reconocido por un DFA de k estados.

Sea $w \in \mathcal{L}$ una hilera w, $n = |w|$ tal que $n \geq k$. Entonces w puede ser separada en tres piezas, $w = x\ y\ z$ tales que se cumple que:

1. $|y| > 0$
2. $|xy| \leq k$
3. $x\ y^i\ z \in \mathcal{L}, tal\ que\ i \geq 0$

Con lo que podemos concluir que todo lenguaje regular infinito satisface el Pumping Lemma.

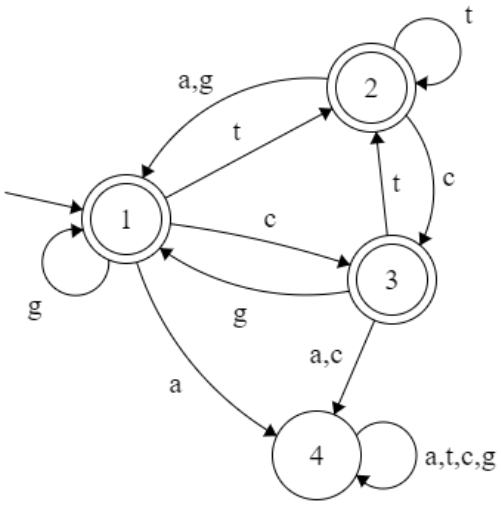


Entonces, si los lenguajes regulares infinitos satisfacen el Pumping Lemma, ¿cómo carajos esto nos va a ayudar a demostrar que un lenguaje no es regular?

Pronto lo descubriremos, sino es que ya lo hicimos. Primeros exploremos un poco el Pumping Lemma en un lenguaje regular.

Ejemplo:

Sea $\mathcal{L}$ el lenguaje sobre $\sum = \{a, t, c, g\}$ de hileras que no contengan la subhilera "cc" y donde totda A es precedida de una T.



Entonces, note que por ejemplo, la hilera “TCTAGGC” pertenece a $\mathcal{L}$ y tiene 7 símbolos, por lo que la hilera es mayor a la cantidad de símbolos en el alfabeto (7>4). Si llegamos a procesar esta hilera podemos observar que se recorre (inicialmente) el estado 2 antes del cuarto símbolo. Por lo que podríamos dividir la hilera de la siguiente manera.

T C T A G G C

x y z

¿Qué hileras también pertenecen a $\mathcal{L}$?

- TAGGC
- TCTAGGC
- TCTCTAGGC
- TCT…CTAGGC

Tigres y Rayas:

- Suponga que todos los tigres tienen rayas
- Si alguien nos describe un animal que no estamos viendo y nos dice que "tiene rayas". ¿Podemos afirmar que ese animal es un tigre?
  No, puede ser una cebra.
- Si alguien nos describe un animal que no estamos viendo y nos dice que "no tiene rayas". ¿Podemos afirmar con seguridad?
  Este animal no es un tigre ni una cebra, o cualquier otro animal que cuente con la característica de tener rayas.

Tigres, Rayas, Lenguajes Regulares y Pumping Lemma:

Si un lenguaje es descrito y nos dicen que "No cumple el Pumping Lemma", ¿Qué podemos afirmar? Se puede indicar con toda seguridad que **el lenguaje no es regular**.

Si un lenguaje es descrito y nos dice que "cumple el Pumping Lemma", ¿Podemos afirmar que es un lenguaje Regular?, No necesariamente, pues **existen algunos lenguajes no regulares que si satisfacen el Pumping Lemma**.

¿Cómo se usa el Pumping Lemma?:

Para demostrar que un lenguaje $\mathcal{L}$ no es regular se va a demostrar por contradicción.

1. Para lo cual se debe suponer que $\mathcal{L}$ es regular y que hay un DFA con k estados que lo reconoce.
2. Creamos una hilera $w \in \mathcal{L}$ es decir que satisfaga el lenguaje, cuya longitud sea mayor que k.
3. Debemos demostrar que al hacerle un $pump\ up$ o $pump\ down$ para cualquier subhilera "y" dentro de w como se describe en el $Pumping\ Lemma$, se genera al menos una hilera que **no pertenezca a** $\mathcal{L}$.

4. Con lo cual encontramos que $\mathcal{L}$ no satisface el Pumping Lemma.
5. Por lo que podemos concluir que no es un lenguaje regular.

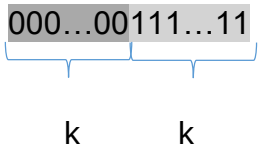
Ejemplo 1:

Sea $\mathcal{L}$ el lenguaje sobre $\sum = \{0,1\}$ definido como $\{0^n 1^n | n \geq 0\}$. Demuestre por contradicción que $\mathcal{L}$ no es regular.

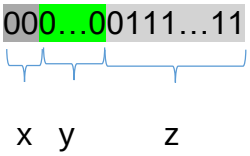
Solución:

Suponga que $\mathcal{L}$ es regular y que es reconocido por un DFA de k estados.

Sea $w = 0^k 1^k$ que satisface al lenguaje $\mathcal{L}$. Lo se vería así:



Pero nótese que $|w| > k$, por lo que cumple con el Pumping Lemma. Dada esta consideración, se puede afirmar que w se puede separar en $x\ y\ z$.



Lo que plantea la regla de que y estará formada solo por 0's. Esto plantea una situación problemática, dado que si hacemos un $pump\ up$ de y dos o más veces generamos una hilera más ceros que unos, la cual no pertenece a $\mathcal{L}$. ⇒⇐

$\mathcal{L}$ no cumple el Pumping Lemma, entonces $\mathcal{L}$ **no es regular**.

Ejemplo 2:

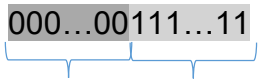
Sea $\mathcal{L}$ el lenguaje sobre $\sum = \{0,1\}$ de hileras con la misma cantidad de 0's y 1's. Demuestre por contradicción que $\mathcal{L}$ no es regular.

Solución:

Suponga que $\mathcal{L}$ es regular y que es reconocido por un DFA de k estados.

• Nótese que este este ejemplo es muy diferente que el anterior, reconoce hileras como 010011, pero se puede resolver como el pasado dado que el ejemplo anterior es un subconjunto de este ejemplo. •

Sea $w = 0^k 1^k$ que satisface al lenguaje $\mathcal{L}$. Lo se vería así:



k k

Pero nótese que $|w| > k$, por lo que cumple con el Pumping Lemma. Dada esta consideración, se puede afirmar que w se puede separar en $x\ y\ z$.

x y z

Lo que plantea la regla de que y estará formada solo por 0's. Esto plantea una situación problemática, dado que si hacemos un $pump\ up$ de y dos o más veces generamos una hilera más ceros que unos, la cual no pertenece a $\mathcal{L}$. ⇒⇐

$\mathcal{L}$ no cumple el Pumping Lemma, entonces $\mathcal{L}$ **no es regular**.

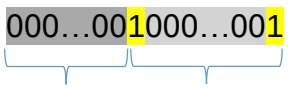
Ejemplo 3:

Sea $\mathcal{L}$ el lenguaje sobre $\sum = \{0,1\}$ definido como $\{ss \mid s \in \{0,1\}^*\}$. Demuestre por contradicción que $\mathcal{L}$ no es regular.

Solución:

Suponga que $\mathcal{L}$ es regular y que es reconocido por un DFA de k estados.

Sea $w = 0^k 10^k 1$ que satisface al lenguaje $\mathcal{L}$. Lo se vería así:



k k

Pero nótese que $|w| > k$, por lo que cumple con el Pumping Lemma. Dada esta consideración, se puede afirmar que w se puede separar en $x\ y\ z$.

000…001000…001

x y z

Lo que plantea la regla de que y estará formada solo por 0's. Esto plantea una situación problemática, dado que si hacemos un $pump\ up$ de y dos o más veces generamos una hilera más ceros que unos, la cual no pertenece a $\mathcal{L}$. ⇒⇐

$\mathcal{L}$ no cumple el Pumping Lemma, entonces $\boldsymbol{\mathcal{L}}$ **no es regular**.

Ejemplo 4:

Sea $\mathcal{L}$ el lenguaje sobre $\sum = \{0\}$ definido como $\{0^n \mid n\ es\ un\ cuadrado\ perfecto\}$. Demuestre por contradicción que $\mathcal{L}$ no es regular.

Reconoce cantidades de ceros como 1, 4, 9, 16, 25, …

Solución:

Suponga que $\mathcal{L}$ es regular y que es reconocido por un DFA de k estados.

Sea $w = 0^{i^2}$ que satisface al lenguaje $\mathcal{L}$. Lo se vería así:

0000…00000000000

k^2

Pero nótese que $|w| > k$, por lo que cumple con el Pumping Lemma. Dada esta consideración, se puede afirmar que w se puede separar en $x\ y\ z$.

0000…00000000000

x y z

Según las reglas $0 < |y| \leq k$ y dado que $|w| = k^2$, pero también se puede despejar el siguiente resultado:

$$|w'| = |w| + |y|$$

$$k^2 + 0 < k^2 + |y| \leq k^2 + k$$

$$k^2 < |w'| < \boldsymbol{k^2 + k} < k^2 + 2k + 1$$

Este último paso se debe a que $k^2, (k+1)^2 = k^2 + 2k + 1$.

Pero sucede que $|xy^2z|$ no es un cuadrado perfecto, entonces xy^2z no pertenece a $\mathcal{L}$. ⇒⇐

$\mathcal{L}$ no cumple el Pumping Lemma, entonces **$\mathcal{L}$ no es regular**.

Ejemplo 5:

Sea $\mathcal{L}$ el lenguaje sobre $\sum = \{0,1\}$ definido como $\{0^m 1^n \mid m > n\}$. Demuestre por contradicción que $\mathcal{L}$ no es regular.

Solución:

Suponga que $\mathcal{L}$ es regular y que es reconocido por un DFA de k estados.

Sea $w = 0^{k+1}1^k$ que satisface al lenguaje $\mathcal{L}$. Lo se vería así:

$$\underbrace{000000\ldots}_{k+1}\underbrace{00011111\ldots111}_{k}$$

Pero nótese que $|w| > k$, por lo que cumple con el Pumping Lemma. Dada esta consideración, se puede afirmar que w se puede separar en $x\ y\ z$.

$$\underbrace{000000\ldots}_{x}\underbrace{00}_{y}\underbrace{011111\ldots111}_{z}$$

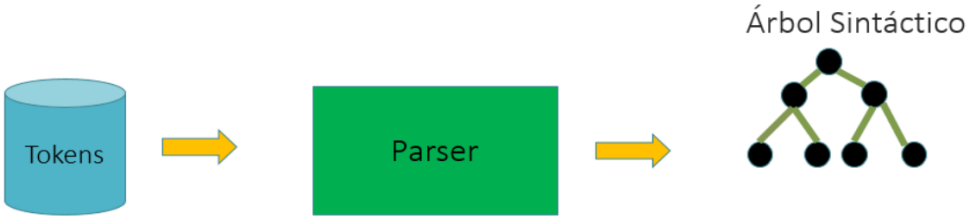
Lo que plantea la regla de que y estará formada solo por 0's. Esto plantea una situación problemática, dado que si hacemos un $pump\ down$ es decir que no exista el "y" de y dos o más veces generamos una hilera más ceros que unos, la cual no pertenece a $\mathcal{L}$. $\Rightarrow\Leftarrow$

$\mathcal{L}$ no cumple el Pumping Lemma, entonces **$\mathcal{L}$ no es regular**.

Análisis Sintáctico:

Introducción:

Recordemos un poco que recibe y que retorna el analizador sintáctico.

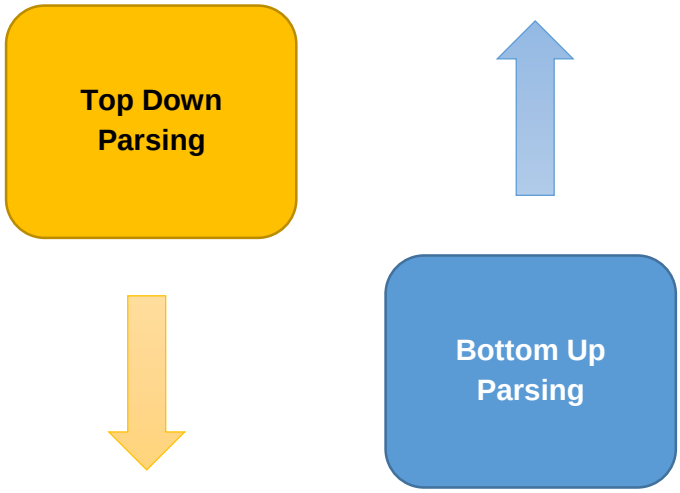


Pero además recordar que pertenece al front-end de un compilador, y es quien tiene relación con el scanner.

Su objetivo es revisar que la sintaxis (estructura) de un programa esté correcta. Se le conoce como analizador sintáctico o Parser. Toma la secuencia de tokens como entrada y verifica que correspondan a construcciones válidas, al final genera un árbol sintáctico. Este analizador requiere de una gramática para lograr definir esa sintaxis o estructura. Existen múltiples algoritmos posibles de parsing.

Tipos de Parsing:

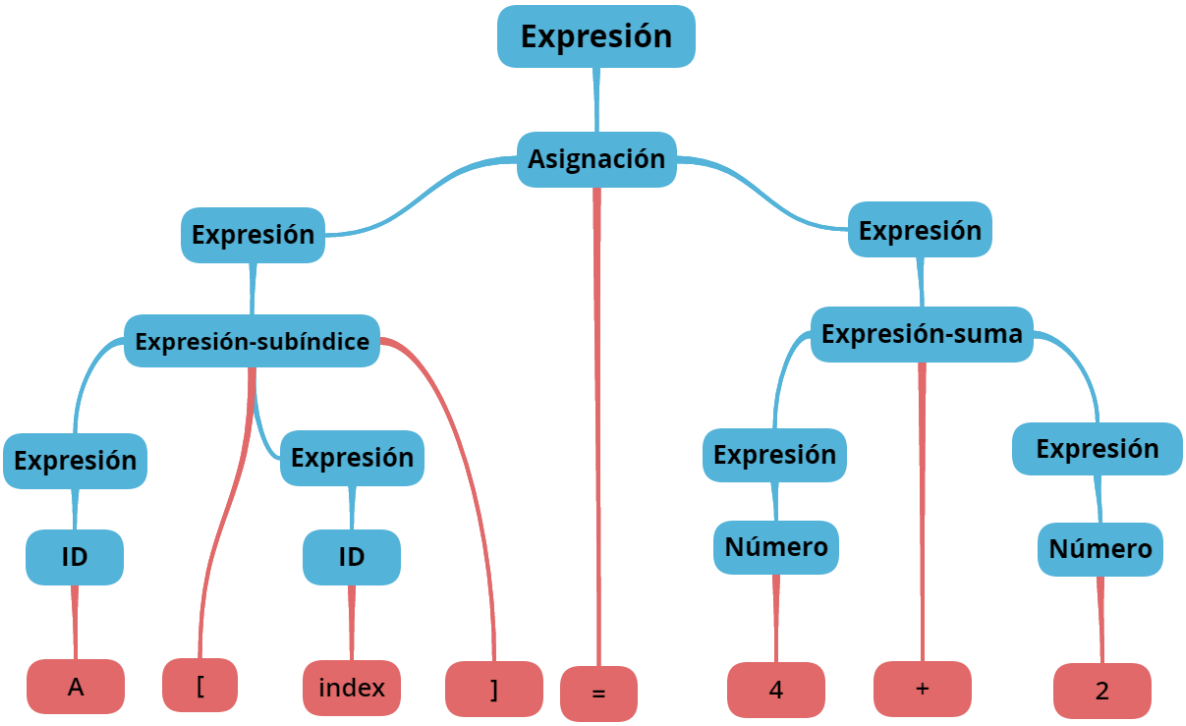
Existen dos grandes familias de parsers:



De los cuales, los parsers Top Down son más fáciles que los Bottom Up que tienden a ser más complicados, pero curiosamente son los Bottom Up los más utilizados.

Árbol Sintáctico:

La expresión: $A\ [\ index\ ]\ =\ 4\ +\ 2$ genera un árbol sintáctico como este:



Errores de Sintaxis:

El manejo de errores es muy difícil complejo en el análisis sintáctico, mientras los errores léxicos son fácilmente manejados. Un Parser hace diferentes cosas frente a un error (triple R):

- Reportar (Report): Indicar la existencia del error, que puede incluir información adicional para que el desarrollador sepa donde está ubicado.
- Recuperar (Recover): “Omitir la existencia de un error” para seguir buscando más errores.
- Reparar (Repair): Un compilador puede llegar a reparar código, pero uno como desarrollador no quiere dejar que eso suceda, dado que pueden llegar a existir múltiples maneras de hacerlo.

Reportar:

Busca errores sintácticos de manera automática con diversos algoritmos de parsing. No resulta difícil saber que hay errores, lo complicado en realidad es redactarlos con cierta información que hay que conseguir. Lo ideal sería:

Localizar el error:

- Posicionar exactamente el token ofensivo resulta difícil
- Puede que la ausencia de un token sea lo que genere el error.

Mensaje de error:

- En el peor de los casos: “Error de sintaxis”
- Realizar una redacción apropiada de los errores resulta ser muy difícil.

Recuperar:

Dado un único error, ya toda la hilera está mala, por lo que a su vez todo el programa está malo. Esto hace que sea impráctico que se reporte sólo el primer error porque si el programa es muy grande hay que corregir y volver a compilar para verificar si hay más errores. Razón por la cual el Parser debe de recuperarse para seguir analizando el resto del programa.

Esto se logra avanzando hasta un punto seguro que cubra el ámbito del error, y continuar con el proceso de parsing. Este proceso de “saltar a un punto seguro” puede conllevar múltiples problemas delicados, como por ejemplo saltar demasiado, y si no es lo suficiente se pueden generar cascada de errores; o bien la aplicación de técnicas de ad hoc (“macheteado”), que son muy específicas para un caso.