Tarea 1 Introducción

Ian Israel García Vázquez

26 de septiembre de 2021

1. Introducción

- Investigue y enuncie una definición formal de gramática. Sea $\Sigma = \{a, b\}$ y $L = \{w \in \Sigma^* \text{ — } w$ contiene exáctamente dos b$\}$. Defina una gramática que reconozca a L y muestre una derivación de una cadena en L. Observemos que el lenguaje luce como sigue:

$$L = a^*ba^*ba^*$$

Una propuesta de gramática podría verse como sigue:

$$S \longrightarrow bb|AbAbA$$
$$A \longrightarrow a|aA|\epsilon$$

1.1. Derivación

Sea la cadena aaaba, su derivación corresponde a:

$$S \longrightarrow AbAbA \longrightarrow aAbAbA \longrightarrow aaAbAbA \longrightarrow aaabAbA \longrightarrow aaab\epsilon bA \longrightarrow aaabbA \longrightarrow aaabba$$

- Lea la sección 1.2 del libro [1] (disponible en el archivero de nuestra página web) y entregue un breve comentario (máximo 1 cuartilla) discutiendo las ventajas y desventajas de la semántica informal.

Como lo hace notar el principio de la lectura puede en realidad verse algo difuso el entendimiento de las máquinas abstracas (y es que en general lo es) perosu comprensión también implica la del lenguaje de programación mismo, y si bien la formalidad a veces nos encuadra para lograr seguir la vía correcta, es cierto que esta conlleva muchas veces un nivel de abstracción más alto del que hemos desarrollado o del que podemos desarrollar en transcursos de tiempo corto, entonces dadas las características de aprendizaje y adquisión de conocimientos respectos a cualquier problema, la intuición por experiencia, generar ejemplos y la semántica informal facilitan la comprensión en la mayoría de los niveles de abstracción, y permiten tener cercanía con el objeto de abstracción en cuestión, pensando que puede ser una máquina abstraca, un compilador o cualquier otro que pudiera también ser considerado de bajo nivel; entonces la semántica informal aparece como un fluido que permite minimizar el rozamiento con la baja abstracción, quizá ahorrar un poco de perturbación mental, dejando que el pensamiento humano normal trabaje bajo el entendimiento más cercano que pueda tener, pero a su vez el hecho de no abarcar en su totalidad la formalidad del mismo, lleva a generar *huecos*, a los que llamamos ambigüedad, esta ambigüedad está definida por ALGOL 60 como:
"*Es ambiguo si dos implementaciones si dos implementaciones que generan el mismo comportamiento son consistentes con su definición en el Reporte*"
El problema con la ambigüedad no es únicamente teórico, desafortunadamente, dada su naturaleza, al traducirse a una implementación, es decir, al hacer código con ambigüedad en el lenguaje,

el programa se hace susceptible a errores, como un auto cuyo volante es defectuoso y gira ocasionalmente al lado contrario al que es accionado y en otras ocasiones ni siquiera actúa aunque se haya turnado en alguna dirección. Si bien en un desarrollo el asunto pareciera no ser tan dramático, es claro como en un principio el código necesitará correciones, traducido en tiempo y dinero, pero si software con este tipo de fallos controlarán al mundo como lo hacen hoy y no existiera algún plan contra catástrofes, es claro que nuestra vida sería incertidumbre absoluta y pendería sobre el hilo de la suerte.

Finalmente bajo esta perspectiva queda aún la discusión si bien la semántica informal ahorra en un principio esfuerzo mental o facilita la labor de comprensión, el reparar los huecos que por su naturaleza pudieran generarse, bajo los sesgos naturales de quienes desarrollan estas soluciones, pudiera ser equiparable al tiempo de aprendizaje y comprensión casi total en su forma de semántica formal, la verdadera solución corresponde a aquella que disminuya el tiempo de comprensión y acción, contenga poca ambigüedad y permita crear soluciones e implementaciones robustas.

- Escriba la tabla 1.1 de la página 11 de [1] en español.

Algol 60
s:=p[0] := n+1+s
n:=n+1
A:=B/C-v-q x S
S[v, k+2]:=3 -arctan(s x zeta)
V:=$Q > Y \ \wedge Z$

4.2.3 Semántica
a asignación de declaraciones sirve para dar el valor de una expresión a una o varias variables o identificadores de procedimientos. La asignación para un identificador de procedimiento puede ocurrir únicamente dentro de la estructura de un procedimiento definiendo el valor de un designador de función. El proceso generalmente será concebido por ocurrir en los siguientes tres pasos:

4.2.3.1 Cualquier expresión de subíndice que se encuentre en la parte izquierda es evaluada en secuencia de izquierda a derecha.

4.2.3.2 La expresión de la declaración es evaluada.

4.2.3.3 El valor de la expresión es asignado a todas las variables del lado izquierdo con cualquier expresión de subínidice, teniendo valores semejantes a los evaluados en el paso *4.2.3.1.*

- Averigüe en que consiste el fenómeno conocido como expresión else colgante (en inglés dangling else) incluyendo un ejemplo sencillo. ¿Sucede este fenómeno en los lenguajes actuales de programación?

  Para el planteamiento debemos considerar una grámatica como la siguiente

$$\text{SENTENCIA} \longrightarrow \text{if CONDICION then SENTENCIA}$$

$$\text{SENTENCIA} \longrightarrow \text{IF CONDICION then SENTENCIA else SENTENCIA}$$

$$\text{SENTENCIA} \longrightarrow \text{OTRAS SENTENCIAS}$$

  Se considera que existen lenguajes que son intrínsecamente ambiguos, esto lenguajes independientes del contexto tales que, cualquier gramática que los genere es ambigua. Sin embargo, lo normal es que siempre se pueda encontrar una gramática que no sea ambigua y que genere el mismo lenguaje. Existe una ambiguedad en la parte de las sentencias condicionales de un lenguaje. Consideremos la grámatica anterior, la cual es por su naturaleza , ambigua, ya que para una instrucción como :

$$\text{if } B_1 \text{ then if } B_2 \text{ then } S_3 \text{ else } S_4$$

existen dos ramificaciones que le otorgan un significado diferente, tal que

$$\text{if } B_1 \text{ then } (\text{ if } B_2 \text{ then } S_3 \text{ else } S_4)$$

y por otro lado:

$$\text{if } B_1 \text{ then } \ (\text{if } B_2 \text{ then } S_3) \text{ else } S_4$$

claramente estos otorgarán resultados diferentes a los que podrían esperarse, sin embargo la parentización más utilizada es aquella que relaciona un *else* con el *if* más cercano [1]. Por lo revisado en [2], si bien este era un problema común en compiladores se ha logrado eliminar la ambigüedad en varios lenguajes actuales cambiando la gramática por una que restringa la ambiguëdad según lo necesitado, incluso gracias a la utilización de $\{\cdots\}$ que permiten definir el rango de un if, o en otros casos con los lenguajes que utilizan identación, con gramáticas no ambiguas o con un alcance expresado con mayor número de tabulaciones.

2. Referencias

[1]Rodriguez, C., 2021. La ambiguedad de las sentencias if-then-else. [online] Campusvirtual.ull.es. Available at: ¡https://campusvirtual.ull.es/ocw/pluginfile.php/2208/mod_resource/content/0/perlexamples/node178.html 26 September 2021].

[2]Wikies.wiki. 2021. (Dangling else problem) - wikipe.wiki. [online] Available at: ¡https://wikies.wiki/wiki/en/Dangling 26 September 2021].