Universidad Simón Bolívar
Traductores e Interpretadores (CI-3725)
Enero – Marzo 2016
Profesor: Ricardo Monascal

Guillermo Betancourt, carnet 11-10103
Gabriel Giménez, carnet 12-11006

Proyecto – BOT

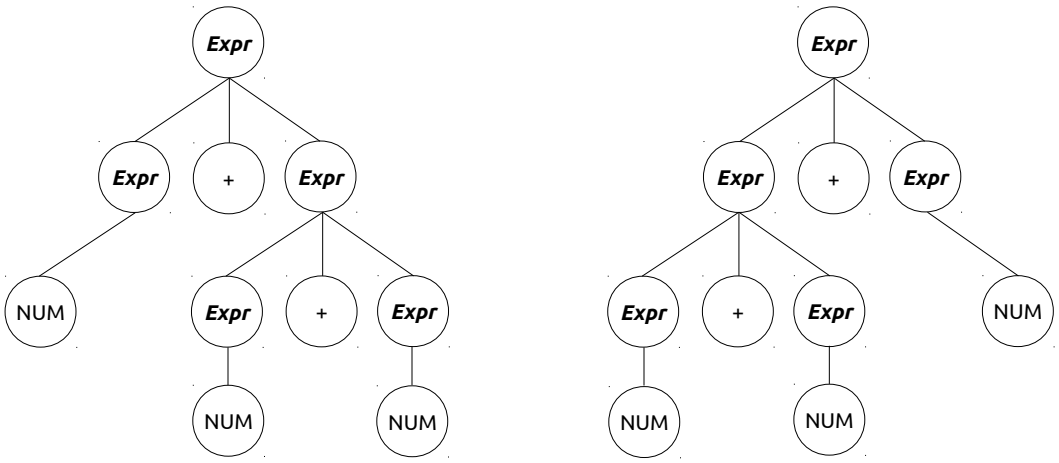
Revisión teórico-práctica, segunda entrega

1. ***Sea*** G_1 ***la gramática*** $(\{Expr\},\{+,NUM\},P_1,Expr)$ ***, con*** P_1 ***compuesto por las siguientes producciones:***

$$Expr \rightarrow Expr + Expr \mid NUM$$

a) Muestre que la frase $NUM+NUM+NUM$ ***de*** G_1 ***es ambigua.***

Para mostrar que la frase dada es ambigua, basta encontrar dos árboles de derivación, más izquierdos o más derechos, tales que ambos árboles representen la misma frase. Consideremos los dos siguientes árboles de derivación más izquierdos.



Ambos árboles de derivación representan la frase $NUM+NUM+NUM$ de G_1 , por lo tanto, la gramática es ambigua.

b) Dé una gramática no ambigua $Izq(G_1)$ ***que genere el mismo lenguaje que*** G_1 ***y que asocie las expresiones aritméticas generadas hacia la izquierda. Dé también una gramática*** $Der(G_1)$ ***con las mismas características pero que asocie hacia la derecha.***

Dada la ambigüedad de asociación de la gramática G_1 , procedemos a desambiguarla mediante parentización. Consideremos la siguiente gramática para $Izq(G_1)$ con la firma $(\{Expr,T\},\{+,NUM,(,)\},P_{izq},Expr)$, siendo P_{izq} el conjunto de producciones:

$$Expr \rightarrow T + Expr \mid NUM$$

$$
\begin{aligned}
T &\rightarrow (Expr) \\
&\mid NUM
\end{aligned}
$$

Parentizando las sumas por la izquierda, se soluciona la ambigüedad por asociación. De manera similar generaremos una gramática $Der(G_1)$.

Dada la ambigüedad de asociación de la gramática G_1 , procedemos a desambiguarla mediante parentización. Consideremos la siguiente gramática para $Der(G_1)$ con la firma $(\{Expr, T\}, \{+, NUM, (,)\}, P_{der}, Expr)$, siendo P_{der} el conjunto de producciones:

$$
\begin{aligned}
Expr &\rightarrow Expr + T \\
&\mid NUM \\
T &\rightarrow (Expr) \\
&\mid NUM
\end{aligned}
$$

c) ¿Importa si se asocian las expresiones hacia la izquierda o hacia la derecha? Considere el caso del operador "–" o el caso del operador "/".

Específicamente para el operador suma no es relevante la asociatividad a la izquierda o a la derecha, suponiendo que NUM es un número no negativo. Sin embargo, para los operador "–" y "/" se requiere que se especifique si es asociativo a la izquierda o a la derecha, dado que dependiendo de la asociación pueden haber resultados distintos para una misma expresión. Por ejemplo, para el operador "–" se tiene que $1-(0-1)=2$ y $(1-0)-1=0$, mientras que para el operador "/" se tiene que $8/(4/2)=4$ y $(8/4)/2=1$.

2. *En BOT la secuenciación, o composición secuencial, es un operador binario infijo sobre instrucciones. Este operador es en realidad virtual, ya que no es representado por ninguna secuencia de símbolos en particular. Sin embargo, a efectos de esta pregunta, denotaremos la secuenciación por el símbolo ";". Suponga que para el manejo de este operador se decide utilizar la gramática G_2 definida como $(\{Instr\}, \{;, IS\}, P_2, Instr)$, con P_2 compuesto por las siguientes producciones.*

$$
\begin{aligned}
Instr &\rightarrow Instr\,;\,Instr \\
&\mid IS
\end{aligned}
$$

Por conveniencia, momentáneamente se ignora el resto de los constructores de instrucciones compuesta de BOT, y se simplifica a las instrucciones simples con el símbolo terminal IS .

a) ¿Presenta G_2 los mismos problemas de ambigüedad que G_1 ? ¿Cuáles son las únicas frases no ambiguas de G_2 ?

Sí, en efecto, para n instrucciones simples separadas por $n-1$ operadores ";", con $n \geqslant 3$, se presentas problemas de ambigüedad por asociación, por ejemplo, $IS;IS;IS$ puede ser interpretado como $(IS;IS);IS$ y $IS;(IS;IS)$, similarmente con mayor cantidad de instrucciones simples.

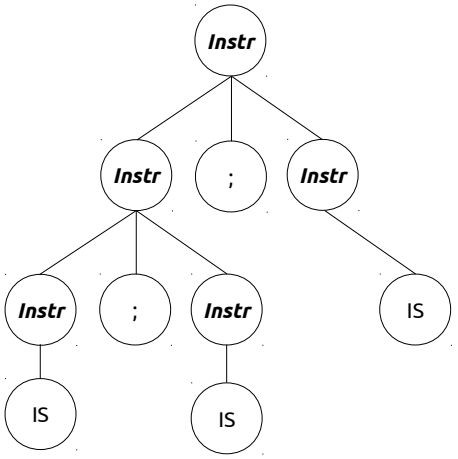
Las únicas frases no ambiguas en G_2 son IS y $IS;IS$.

b) ¿Importa si la ambigüedad se resuelve con asociación hacia la izquierda o hacia la derecha?

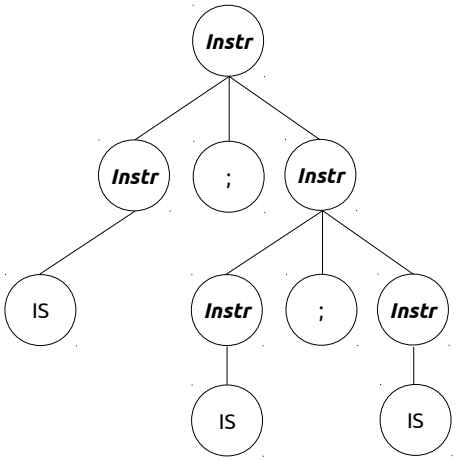
Para efecto de reconocimiento de las instrucciones, no importa cómo se resuelva la ambigüedad. Sin embargo, para efectos de eficiencia, pudiese convenir algún alguna de las dos desambiguaciones.

c) Dé una derivación "más a la izquierda" (leftmost) y una derivación "más a la derecha" (rightmost) de G_2 ***para la frase*** $IS;IS;IS$ ***.***

Derivación leftmost



Derivación rightmost



3. Consideramos ahora los constructores de instrucciones condicionales BOT, pero ignorando momentáneamente el uso que la sintaxis de éstos hace de los "end" y manteniedo el operador ";" propuesto en la pregunta anterior.

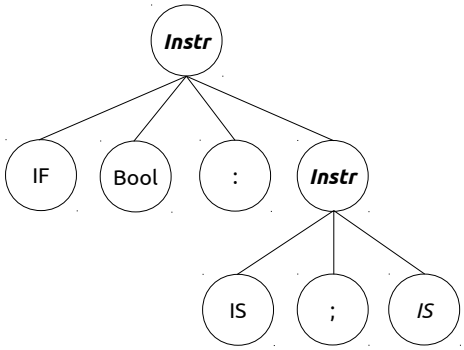
Sea G_3 ***la gramática*** $(\{Instr\},\{;,IS,:,ELSE,Bool,IS\},P_3,Instr)$ ***, con*** P_3 ***compuesto por:***

$$
\begin{aligned}
Instr \rightarrow\ & Instr\ ;\ Instr \\
\mid\ & IF\ Bool\ :\ Instr \\
\mid\ & IF\ Bool\ :\ Instr\ \ ELSE\ :\ Instr \\
\mid\ & IS
\end{aligned}
$$

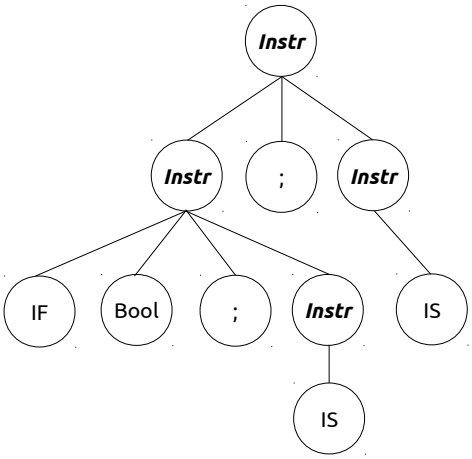
a) Note que G_3 ***mantiene el mismo problema que*** G_2 ***, i.e. frases con más de una ocurrencia de ";" también pueden ser ambiguas. Sea*** f ***la frase*** $IF\ Bool\ :\ IS;IS$ ***. Muestre que*** f ***es una frase ambigua de*** G_3 ***.***

Sea la frase $f = IF\ Bool\ :\ IS;IS$ presenta una ambigüedad de asociación, pues puede ser interpretada como $(IF\ Bool\ :\ IS);IS$, o bien $IF\ Bool\ :\ (IS;IS)$. Mostremos entonces que f es ambigua dando dos árboles de derivación *leftmost* diferentes.

$f = IF\,Bool : (IS ; IS)$



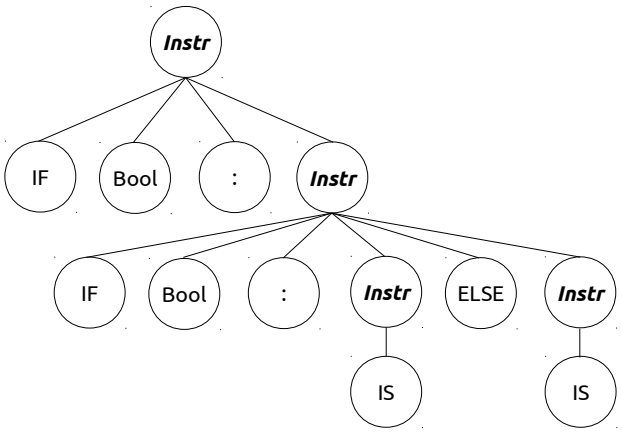
$f = (IF\,Bool : IS) ; IS$



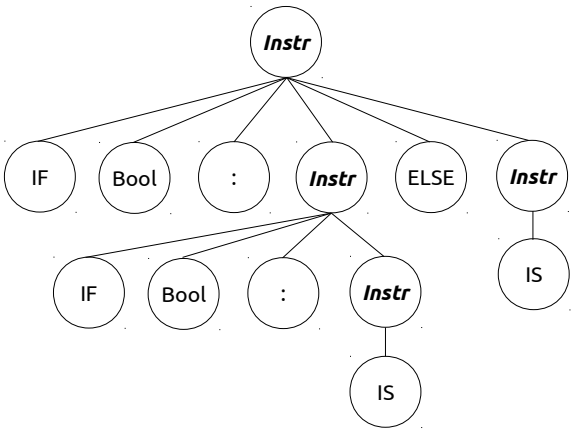
b) Dé una frase g de G_3 sin ocurrencias de ";" que sea ambigua, y muestre que lo es.

Consideremos la frase $g = IF\,Bool : IF\,Bool : IS\ \ ELSE : IS$, dicha frase es ambigua por asociación, pues se puede interpretar como $IF\,Bool : (IF\,Bool : IS\ \ ELSE : IS)$, o bien $IF\,Bool : (IF\,Bool : IS)\ ELSE : IS$. Esta ambigüedad es conocida como "the dangling-else ambiguity". Ahora mostremos que la frase es ambigua dando dos árboles de derivación *leftmost* diferentes.

$g = IF\,Bool : (IF\,Bool : IS\ \ ELSE : IS)$



$g = IF\,Bool : (IF\,Bool : IS)\ ELSE : IS$



c) En lenguajes como Java, C y C++ se hace uso de los símbolos "{" y "}" ("begin" y "end" en Pascal) para diferenciar las dos posibles interpretaciones de la frase f . Lo mismo ocurre con la frase g . ¿Cómo se escribirían las dos interpretaciones de f y las dos interpretaciones de g usando las llaves como separadores?

Dadas $f = IF\,Bool : IS ; IS$ y $g = IF\,Bool : IF\,Bool : IS\ \ ELSE : IS$ frases ambiguas en G_3 , procedemos a escribir sus respectivas interpretaciones con la notación sugerida.

La interpretación $f = IF\,Bool : (IS ; IS)$ se escribiría como sigue:

```
IF Bool : {
     IS;
     IS
}
```

La interpretación $f = (IF\ Bool\ :\ IS);IS$ se escribiría como sigue:

```
IF Bool : {
     IS
};
IS
```

La interpretación $g = IF\ Bool\ :(IF\ Bool\ :\ IS\ \ ELSE\ :\ IS)$ se escribiría como sigue:

```
IF Bool : {
     IF Bool : {
          IS
     }
     ELSE : {
          IS
     }
}
```

Finalmente, la interpretación $g = IF\ Bool\ :(IF\ Bool\ :\ IS)\ ELSE\ :\ IS$ se escribiría como sigue:

```
IF Bool : {
     IF Bool : {
          IS
     }
}
ELSE : {
     IS
}
```

d) En BOT, que utiliza los terminadores "end" en la sintaxis de sus condicionales, ¿cómo se escribirían las dos interpretaciones de f y las dos interpretaciones de g ?

Dadas $f = IF\ Bool\ :\ IS;IS$ y $g = IF\ Bool\ :\ IF\ Bool\ :\ IS\ \ ELSE\ :\ IS$ frases ambiguas en G_3 , procedemos a escribir sus respectivas interpretaciones con los terminadores "*end*" del lenguaje BOT (solamente se utiliza en terminador indicado, no se reescriben las intepretaciones de las frases con la sintáxis completa del lenguaje BOT).

La interpretación $f = IF\ Bool\ :\ (IS;IS)$ se escribiría como sigue:

```
IF Bool :
     IS;
     IS
end
```

La interpretación $f = (IF\ Bool\ :\ IS);IS$ se escribiría como sigue:

```
IF Bool :
     IS
end;
IS
```

La interpretación $g = IF\ Bool\ :(IF\ Bool\ :\ IS\ \ ELSE\ :\ IS)$ se escribiría como sigue:

```
IF Bool :
     IF Bool :
          IS
     ELSE :
          IS
     end
end
```

Finalmente, la interpretación $g = IF\ Bool\ :(IF\ Bool\ :\ IS)\ ELSE\ :\ IS$ se escribiría como sigue:

```
IF Bool :
     IF Bool :
          IS
     end
ELSE :
     IS
end
```