Norelys Rumbos 12-10941
David Atias 12-10771

INFORME PROYECTO 2: ANÁLISIS SINTÁCTICO

TRADUCTORES E INTERPRETADORES

El presente informe tiene como finalidad aclarar algunos detalles en cuanto al código y a la implementación del lexer. A continuación se enumeran las mismas:

1. Para modelar el árbol sintáctico abstracto, creamos la clase base ArbolBinario que utilizamos para modelar aquellas construcciones del lenguaje que se derivan en máximo dos sub-árboles y un valor, que puede ser, por ejemplo, la operación entre dos expresiones.

2. Además, creamos la clase Arbol_Secuenciación que representa un árbol general (consideramos la secuenciación como una instrucción que se expande en tantos sub-árboles como instrucciones formen parte de ella.

3. Implementamos la clase ArbolRepDet que representa una repetición determinada, la cual no caía en ninguno de los dos casos anteriores.

4. Todas las construcciones del lenguaje, salvo la repetición determinada y la secuenciación, son representadas por árboles binarios.

5. Creamos un diccionario que toma como clave el nombre de la estructura y como valor una lista de 3 strings que describen cómo se interpreta el valor, hijo izquierdo e hijo derecho de cada una de estas, y con ello, generamos una clase para cada estructura con un método que convierte el árbol en un string, tomando en cuenta la descripción de cada atributo del árbol de acuerdo al diccionario.

REVISIÓN TEÓRICA-PRÁCTICA

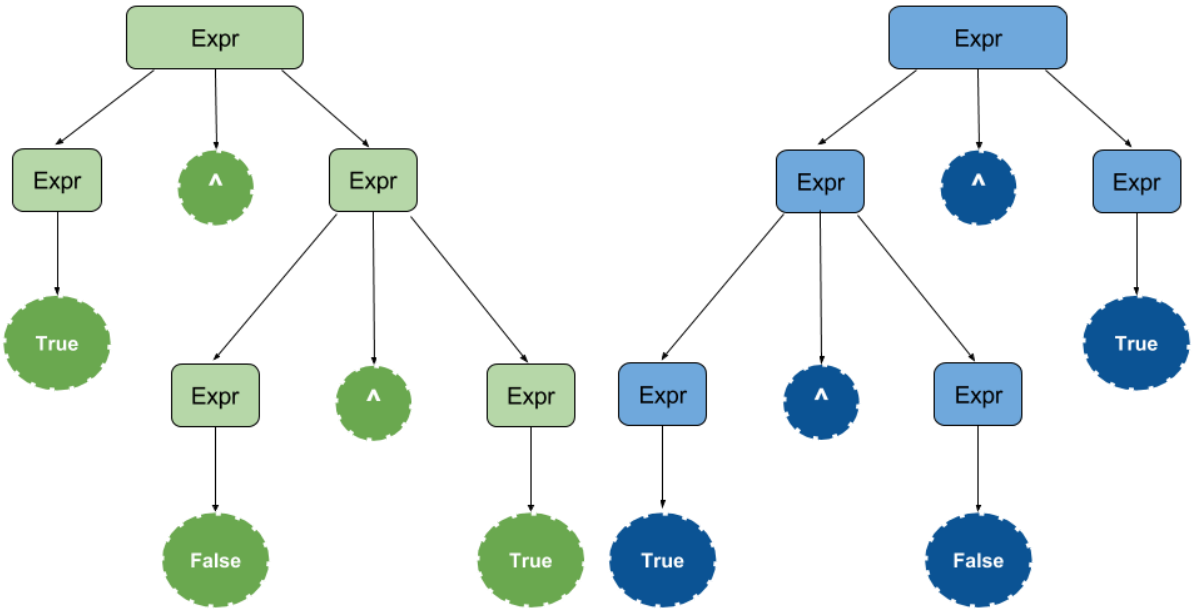
1) Sea G1 la gramática ({ Expr } , { + , true, false } , P1 , Expr) , con P1 compuesto por las siguientes producciones:

Expr⟶ Expr ∧ Expr

Expr⟶ true

Expr⟶ false

a) True /\ False /\ True es ambigua?



b) Gramática no ambigua que genere el mismo lenguaje que G1

Gramática Izquierda:

Expr ⟶ Expre /\ T | T

T ⟶ True | False

Gramática Derecha

Expr ⟶ T /\ Expr | T

T ⟶ True | False

c) ¿Importa si se asocian las expresiones hacia la izquierda o hacia la derecha?

Si importa ya que con los operadores que no son asociativos como el caso de la implicación el resultado cambiará dependiendo del camino tomado.

Por ejemplo con False ⇒ True ⇒ False

Asociando por la izquierda:

(False ⇒ True) ⇒ False

True ⇒ False

False

Asociando por la derecha:

False ⇒ (True ⇒ False)

False ⇒ False

True

2) En Neo 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 G2 definida como:

({ Instr } , { ; , IS } , P2 , Instr) , con P2

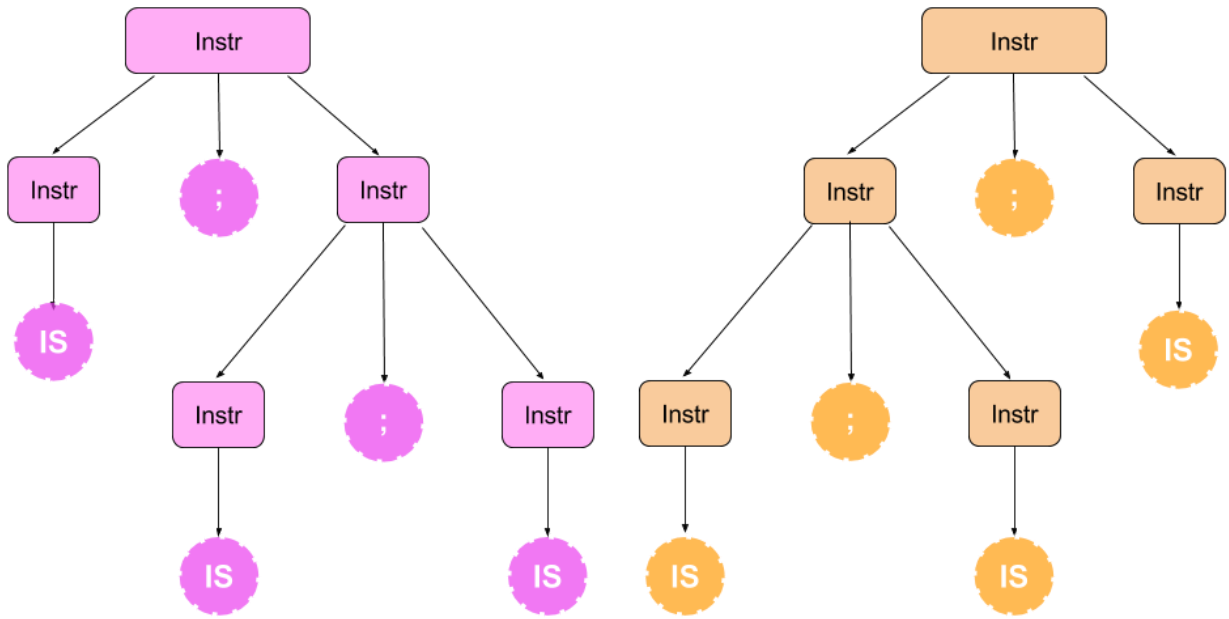
compuesto por las siguientes producciones:

Instr→ Instr ; Instr

Instr→ IS

a) ¿Presenta G2 los mismos problemas de ambigüedad que G1? ¿Cuáles son las únicas frases no ambiguas de G2?

Si, por ejemplo la frase IS;IS;IS



Las frases no ambiguas son las que tienen menos de dos " ; ". Es decir: IS (con cero ;) e IS;IS (con un ;)

b) Importa si la ambigüedad se resuelve asociando hacia la izquierda o hacia la derecha

En este caso no importa si se asocia hacia la izquierda o hacia la derecha porque sin importar cómo se haga, el orden de ejecución de las instrucciones será siempre el mismo. Es decir si se asocian las dos últimas [IS;(IS;IS)] se

ejecuta en primer lugar el IS que está fuera de los paréntesis, luego la primera de los paréntesis y por ultimo la ultima instrucción. De forma análoga ocurriría con (IS;IS);IS, ya que ese es el orden ejecución y no lo altera la parentización.

c) De una derivación más izquierda y una más derecha de G2 para la frase IS;IS;IS

Derivación más izquierda:

Instr
→ **Instr** ; Instr
→ **Instr** ; Instr ; Instr
→ IS ; **Instr** ; Instr
→ IS ; IS ; **Instr**
→ IS ; IS ; IS

Derivación más derecha:

Instr
→ Instr ; **Instr**
→ **Instr** ; IS
→ Instr ; **Instr** ; IS
→ **Instr** ; IS ; IS
→ IS ; IS ; IS

3) Consideremos ahora los constructores de instrucciones condicionales de Neo, pero ignorando momentáneamente el uso que la sintaxis de éstos hace de los “end” y manteniendo el operador “ ; ” propuesto en la pregunta anterior. Sea G3 la gramática:

({ Instr } , { ; , IF , : , OTHERWISE , Bool , IS } , P3 , Instr)

con P3 compuesto por:

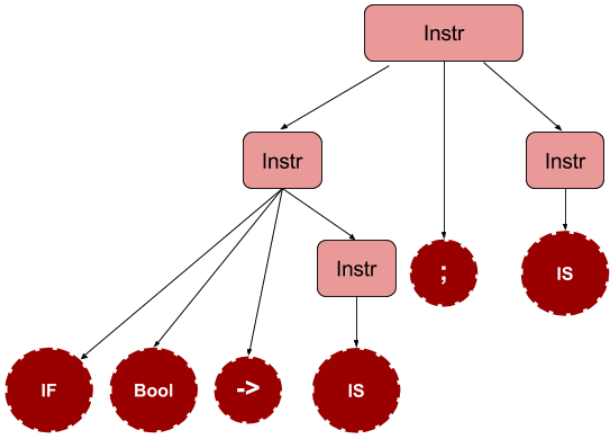
Instr ⟶ Instr ; Instr

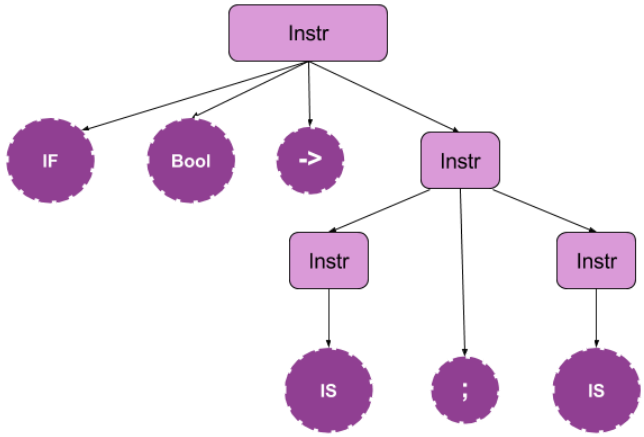
Instr ⟶ IF Bool − > Instr

Instr ⟶ IF Bool − > Instr OTHERWISE − > Instr

Instr ⟶ IS

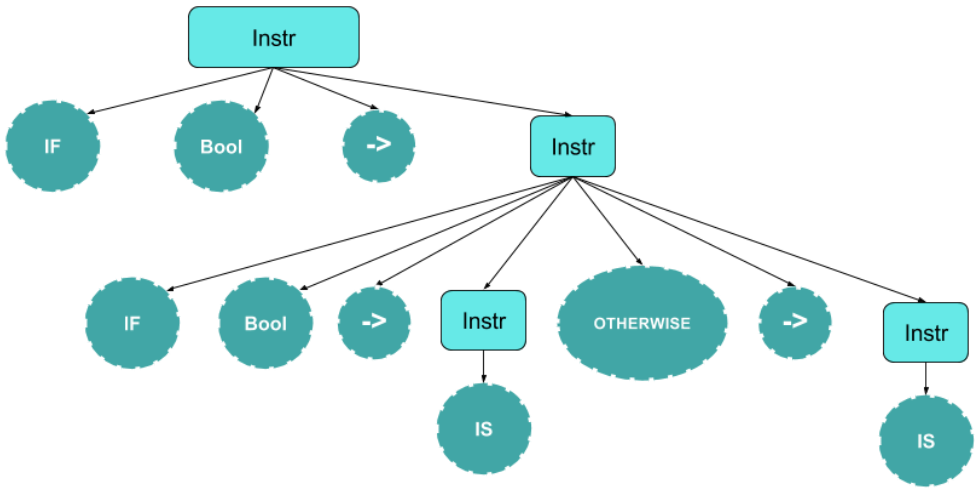
a) Sea f la frase “IF Bool − > IS ; IS ”. Muestre que f es una frase ambigua de G3

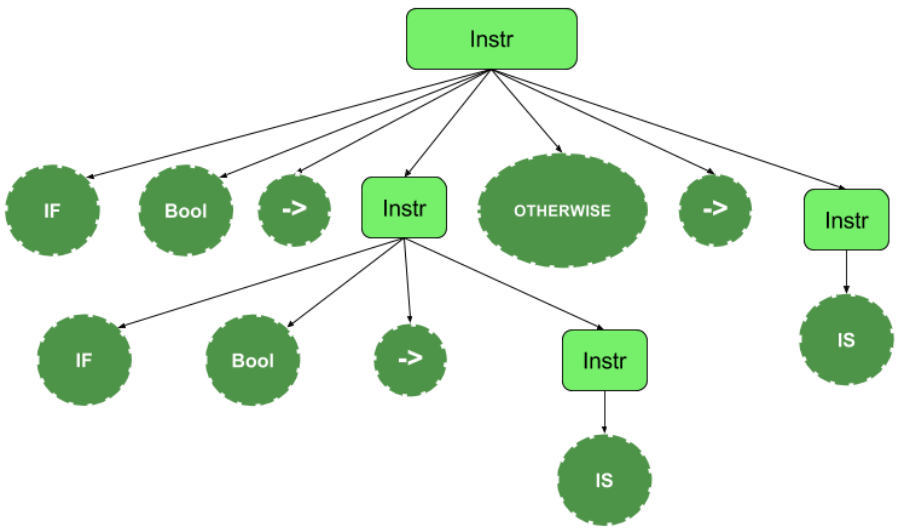




b) Dé una frase g de G3 sin ocurrencias de “ ; ” que sea ambigua, y muestre que lo es.

g: IF Bool -> IF Bool -> IS OTHERWISE -> IS





c) Cómo se escribirían las dos interpretaciones de f y las dos interpretaciones de g usando las llaves como separadores?

Frase f:

1. IF Bool -> {IS} ; IS
2. IF Bool -> {IS ; IS}

Frase g:

1. IF Bool -> **{**IF Bool -> {IS} OTHERWISE -> {IS}**}**
2. IF Bool -> **{**IF Bool -> {IS}**}** OTHERWISE -> {IS}

d) En Neo, 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?

Frase f:

1. IF Bool -> IS **end** ; IS
2. IF Bool -> IS ; IS **end**

Frase g:

1. IF Bool -> IF Bool -> IS OTHERWISE -> IS **end end**
2. IF Bool -> IF Bool -> IS **end** OTHERWISE -> IS **end**