G→ADUN

Ejercicio 6: No Terminales Extraños

Ejercicio 7: Gramáticas S, Q y LL(1)

Conceptos a trabajar:

- No terminales muertos
- Producciones muertas o inalcanzables

Proyecto:

A través de GADUN, en la carpeta raíz de la herramienta busque en la sección de *"GRAMÁTICAS_EJERCICIOS"* el proyecto *"ejemplo gramática S.json"*, *"Identificación de gramática 1.json"* y *"Identificación de gramática 2.json"*. Si no encuentra el directorio o el archivo por favor descargue los de REPOGADUN.

Introducción:

Una **gramática libre de contexto** (o **de contexto libre**) es una gramática formal en la que cada regla de producción es de la forma:
$V \rightarrow w$
Donde V es un símbolo no terminal y *w* es una cadena de terminales y/o no terminales. El término *libre de contexto* se refiere al hecho de que el no terminal V puede siempre ser sustituido por *w* sin tener en cuenta el contexto en el que ocurra.

Dentro del estudio de estas, es común encontrarse con cierto tipo de gramáticas distintivas, entre ellas están:

- **Gramáticas - S:** Poseen las siguientes restricciones en sus producciones:
    - El lado derecho de cada producción inicia con un símbolo terminal
    - Si dos producciones tienen el mismo elemento no terminal izquierdo, entonces sus lados derechos deben iniciar con diferente símbolo terminal.

    Ejemplo: Identificar el tipo de gramática que forman las siguientes producciones:

1. <S> -> ab<R>
2. <S> -> b<R>b<S>
3. <R> -> a
4. <R> -> b <R>

Podemos observar que todas las producciones de la gramática inician con un terminal para su lado derecho (a o b) y las producciones que tienen el mismo no terminal en el lado izquierdo (1 y 2 o 3 y 4) empiezan su lado derecho con un terminal distinto (a o b); se cumplen las condiciones mencionadas y concluimos que la gramática es de tipo S.

Adicionalmente, se ha recreado este ejemplo dentro del archivo de nombre *ejemplo gramática S.json*

Para el estudio de las siguientes gramáticas, es necesario tener en cuenta el concepto de **primero,** el de **siguiente** y el del **conjunto selección** de una producción.

- El conjunto **PRIMERO(α)**, siendo α una producción, es el conjunto de Terminales que pueden aparecer de primeros en cadenas derivadas de α.

- Si A es un símbolo No Terminal de una gramática, **SIGUIENTE(A)** es el conjunto de Terminales (incluyendo el símbolo de fin de cadena) que pueden aparecer justo después de A en alguna producción derivada del símbolo inicial.

- El **conjunto selección** de una producción es la unión de los conjuntos primero y siguiente, es decir:
    - conjunto selección = {siguiente} U {primero}

- **Gramáticas - Q:** Es una gramática libre de contexto que tiene las siguientes condiciones:

    - El lado derecho de cada producción puede ser nulo o iniciar con un símbolo terminal

    - Producciones con el mismo lado izquierdo tienen un conjunto de selección diferente. Es decir, la intersección entre los conjuntos de selección debe ser vacía:

    - {c} Intersección {a, b}= {} *donde a,b,c son terminales*

- **Gramáticas - LL(1):** Se trata de un tipo de gramáticas libres de contexto, no ambiguas, que cumplen con las siguientes condiciones:

    - El lado derecho de cada producción puede comenzar con un Terminal, un No Terminal o incluso ser la secuencia nula

    - Las producciones que tengan el mismo símbolo del lado izquierdo tiene conjuntos de selección disjuntos, es decir, que no tienen ningún elemento en común.

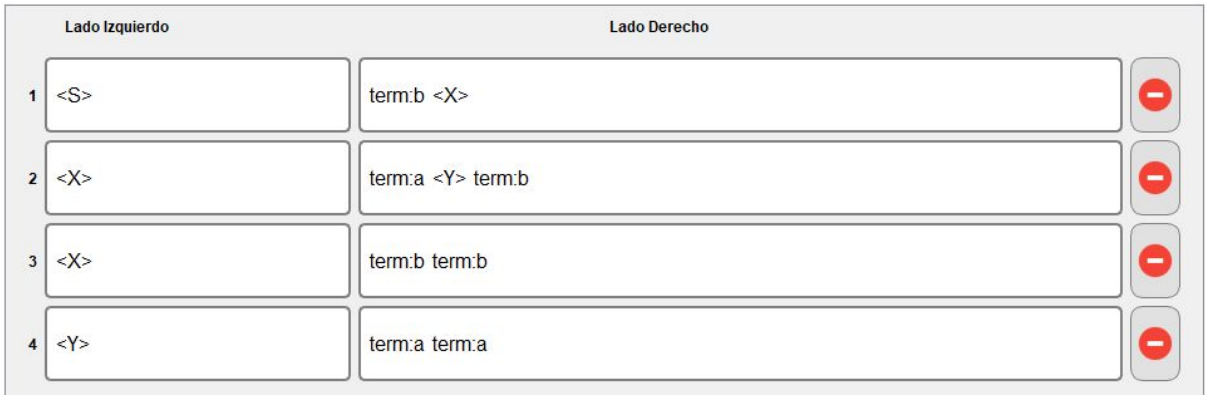
  Nota: Si observamos las definiciones, podemos ver que las gramáticas LL(1) son parecidas a las Q; cambian en el hecho de que estas primeras permiten inicializar el lado derecho de las producciones con un no terminal. Aparte, también se resalta el hecho de que las gramáticas LL(1) **no** deben ser ambiguas.

  La **ambigüedad** dentro de las gramáticas refiere a un problema o situación en el que nuestra gramática posee dos caminos o *derivaciones* que terminan en el mismo resultado. Esto introduce cierto grado de de indeterminismo a nuestra estructura y por eso se evita a toda costa. La herramienta GADUN se encargará de informarnos cuando tengamos producciones y gramáticas que produzcan ambigüedad.

Proceso

1. Abre el proyecto de nombre *Identificación de gramática 1.json* indicado para esta guía. Como primer ejercicio, se tratará de comprobar si la gramática dada cumple con lo necesario para ser de tipo Q.

2. Dirígete a la sección del analizador sintáctico, veremos una lista de producciones que describen una gramática libre de contexto que reconoce cadenas de a’s y b’s en cierto orden:



	Lado Izquierdo	Lado Derecho
1	<S>	term:b <X>
2	<X>	term:a <Y> term:b
3	<X>	term:b term:b
4	<Y>	term:a term:a

3. Para identificar si se trata de una gramática de tipo Q, debemos comprobar:

    a. El lado derecho de cada producción sea nulo o que inicie con un símbolo terminal
    b. Producciones con el mismo lado izquierdo tienen un conjunto de selección diferente.

    Si observamos cada producción, veremos que cada lado derecho de las producciones empieza con un elemento de tipo terminal, por lo que se cumple la condición a.

4. Como ya comprobamos la condición a, queda verificar si el conjunto selección de las producciones que tienen el mismo lado izquierdo es el mismo para ambas (producciones 2 y 3).

    Para lo anterior haremos uso de la tabla de control. Ubicándonos dentro del analizador sintáctico, presionamos el botón *Validar Producciones*, acto seguido nos ubicamos en el tab de tabla de control.

Producciones

Tabla de Control

	term:a	term:b	¬
<S>	Error	P:1	Error
<X>	P:2	P:3	Error
<Y>	P:4	Error	Error
term:b	Error	PA	Error
term:a	PA	Error	Error
$	Error	Error	Aceptado

Aquí podremos comprobar fácilmente si las producciones que 2 y 3 tienen un distinto conjunto selección, simplemente las seleccionamos ya sea de la lista de la izquierda o en su respectiva casilla de la tabla (P:2,P:3).

	term:a	term:b	¬
<S>	Error	P:1	Error
<X>	P:2	P:3	Error
<Y>	P:4	Error	Error
term:b	Error	PA	Error
term:a	PA	Error	Error

Una vez seleccionada la producción dirigimos nuestra atención hacia el tab de selección, aquí encontraremos los conjuntos selección para las dos producciones:

Producción	<X> → term:a <Y> term:b
Conjunto Selección	['term:a']

SELECCION(2) = PRIMERO(term:a <Y> term:b)
PRIMERO(term:a <Y> term:b) = (term:a)
SELECCION(2) = (term:a)

conjunto selección para producción 2

Producción	<X> → term:b term:b
Conjunto Selección	['term:b']

SELECCION(3) = PRIMERO(term:b term:b)
PRIMERO(term:b term:b) = (term:b)
SELECCION(3) = (term:b)

conjunto selección para producción 3

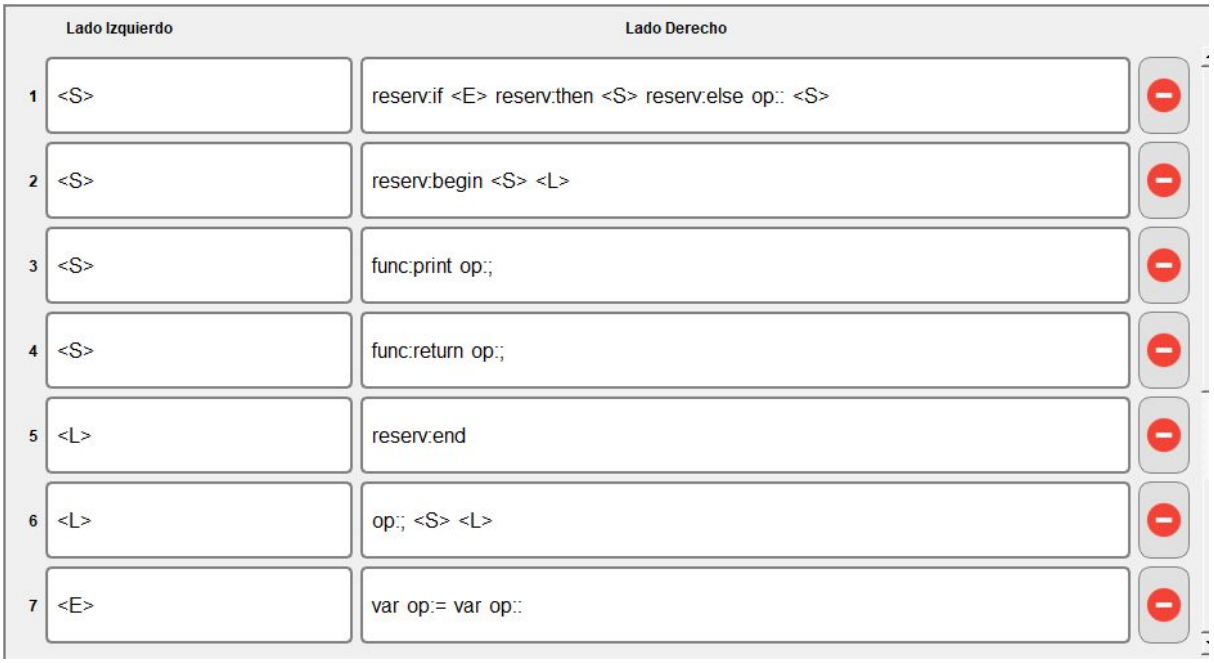
Al comparar los resultados, podemos observar que el conjunto selección para la producción 2 es {a} y para la producción 3 es {b}, como la intersección de ambos conjuntos nos da como resultado un conjunto vacío {}, se cumple la condición b y se concluye que la gramática dada es de tipo Q.

5. Para la siguiente parte de la guía usaremos el proyecto de nombre *Identificación de gramática 2.json*. Comprobaremos si la gramática cumple con las condiciones para ser de tipo LL(1)

6. Dirígete a la sección del analizador sintáctico, veremos una lista de producciones que describen una gramática libre de contexto que reconoce ciertas líneas de código como la siguiente:

```
if x = y:
then begin
print;
end
else:
return;
```

Así mismo, vemos las producciones generadas para el problema:



	Lado Izquierdo	Lado Derecho
1	<S>	reserv:if <E> reserv:then <S> reserv:else op:: <S>
2	<S>	reserv:begin <S> <L>
3	<S>	func:print op:;
4	<S>	func:return op:;
5	<L>	reserv:end
6	<L>	op:; <S> <L>
7	<E>	var op:= var op::

7. Para identificar si se trata de una gramática de tipo LL(1), debemos comprobar:

    a. El lado derecho de cada producción puede ser iniciado con cualquier tipo de símbolo (terminal, no terminal o incluso puede ser vacío)
    b. Las producciones que tengan el mismo símbolo del lado izquierdo tiene conjuntos de selección disjuntos, es decir, que no tienen ningún elemento en común.

    El apartado a se cumplirá independientemente de cómo esté estructurada nuestra gramática. Es aquí donde se resalta la distinción mencionada anteriormente con las gramáticas Q, puesto que las LL(1) permiten la inicialización del lado derecho con no terminales, cosa que no ocurre con las gramáticas Q.

8. Como ya se cumple con la condición a, queda comprobar si las producciones que tienen el mismo lado izquierdo poseen un conjunto selección distinto.

    Si observamos la gramática veremos que las producciones 1,2,3 y 4 comienzan con el No Terminal <S>.

    Por otra parte, las producciones 5 y 6 también repiten el lado izquierdo, en este caso es <L>

Compararemos los conjuntos selección de la misma manera que lo hicimos para la gramática Q, presionamos el botón de *Validar Producciones*, acto seguido nos ubicamos en el tab de tabla de control.

9. Para las producciones que empiezan con el no terminal <S> vemos que:

Producción	<S> → reserv:if <E> reserv:then <S> reserv:else op:: <S>
Conjunto Selección	['reserv:if']

PRIMERO(reserv:if <E> reserv:then <S> reserv:else op:: <S>) = (reserv:if)
SELECCION(1) = (reserv:if)

conjunto selección para Producción #1

Producción	<S> → reserv:begin <S> <L>
Conjunto Selección	['reserv:begin']

SELECCION(2) = PRIMERO(reserv:begin <S> <L>)
PRIMERO(reserv:begin <S> <L>) = (reserv:begin)
SELECCION(2) = (reserv:begin)

conjunto selección para Producción #2

Producción	<S> → func:print op:;
Conjunto Selección	['func:print']

SELECCION(3) = PRIMERO(func:print op:;)
PRIMERO(func:print op:;) = (func:print)
SELECCION(3) = (func:print)

conjunto selección para Producción #3

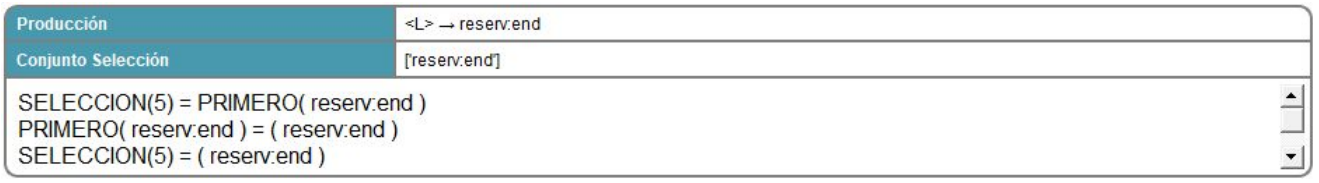
Producción	<S> → func:return op:;
Conjunto Selección	['func:return']

SELECCION(4) = PRIMERO(func:return op:;)
PRIMERO(func:return op:;) = (func:return)
SELECCION(4) = (func:return)

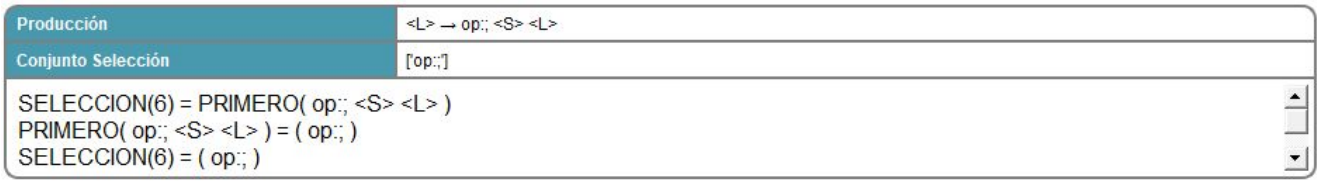
conjunto selección para Producción #4

Como vemos, ningún conjunto selección es igual o tiene elementos en común, hasta ahora la gramática cumple con la condición.

10. Para las producciones que empiezan con el no terminal <L> vemos que:



conjunto selección para Producción #5



conjunto selección para Producción #6

Como vemos, ningún conjunto selección es igual o tiene elementos en común. Como ya descartamos todos los casos donde se repite el primer No Termina y, además, no recibimos ninguna advertencia de ambigüedad de parte del software, concluimos que efectivamente se trata de una gramática de tipo LL(1).

Ejercicio de refuerzo:

1. Dentro de la carpeta de *"OTRAS_GRAMÁTICAS"* busque el ejercicio *"gramática libre de contexto 1.json"* que permite reconocer secuencias de la forma **aacbb.** Determine el tipo de gramática a la que pertenece.