Universidad Simón Bolívar
Coordinación de Ingeniería de la Computación
Curso: Traductores E Interpretadores
Profesor: Ricardo Monascal

INFORME PROYECTO 3
(Traductores E Interpretadores)

Autores:
Arleyn Goncalves 10-10290
Francisco Sucre 10-10717

Sartenejas, Marzo del 2016

INFORME DE IMPLEMENTACIÓN DE PROYECTO

Herramientas utilizadas

- Python 3.5
- Eclipse ver 4.5 (Mars) como IDE para el desarrollo del proyecto, con el complemento "PyDev" para integrar el IDE con el idioma Python.
- Paquete PLY de Python con algunas modificaciones hechas para cumplir con lo necesario del proyecto.
- GitHub como herramienta de control de versiones, utilizando Egit (Complemento de Git para eclipse), RabbitSVN(Linux) y TortoiseGit (Windows) como herramientas para manejar el repositorio.

Detalle de la implementación

El analizador de contexto se encuentra integrado al parser, se construyeron dos clases nuevas una llamada SymbolTable para crear la tabla de simbolos de nuestro programa y una clase simbolo para crear los símbolos correspondientes. En detalle la tabla de simbolo tiene dos atributos un diccionario cumpliendo las funciones de una tabla de hash y un atributo operlevel que apunta a la tabla de simbolos de nivel superior, en los nuevos niveles se crean cada vez que el parser consigue una incorporación de alcance así como también generan niveles para las definiciones de los comportamientos de un bot.

Cuando nuestro parser se encuentra con un identificador empieza a buscarlo por la tabla de nivel inferior y al no encontrarlo en dicha tabla buscará en las tablas de nivel superior, si el simbolo no se encuentra en ninguna tabla el parseo rechazado y se enviará un mensaje de error.

Instrucciones De Uso

- Abrir una ventana de terminal.
- Trasladarse a la carpeta de scripts.
- Se debe verificar que el archivo de texto a ser leído debe encontrarse en la carpeta input_case ubicada en la raíz del proyecto.
- En Linex escribir ./ContBot <texto> donde <texto> es el nombre del archivo donde se encuentra el código a analizar.
- La pantalla del terminal se vaciara y el programa imprimirá el resultado.

TEORICO - PRACTICO

1. Sea G1i la gramática recursiva-izquierda ({ S } , { a } , { S → Sa , S → λ } , S) y sea G1d la gramática recursiva-derecha ({ S } , { a } , { S → aS , S → λ } , S). Ambas generan el lenguaje denotado por la expresión regular a*, i.e. el lenguaje L(a*)

   (a) Muestre que ambas gramáticas son LR(1) y construye sus analizadores sintácticos

   Tenemos que G1i:

   S → Sa

   S → λ

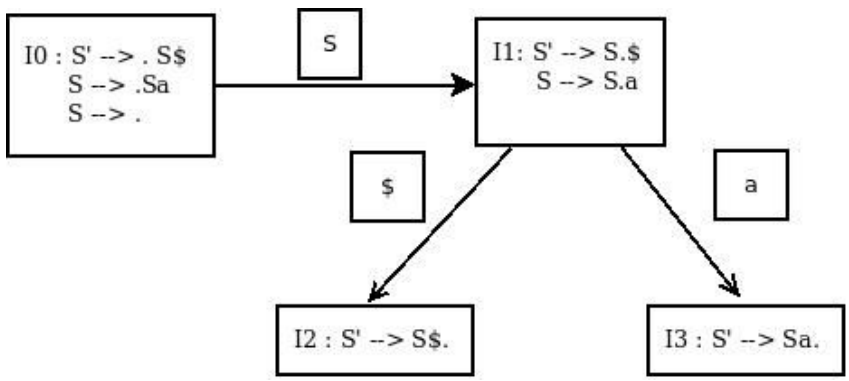
   Le agregamos el valor inicial:

   S' → S (i)

   S → Sa (ii)

   S → λ (iii)

   Realizamos el automata de prefijos viables:



   Calculamos los first and follow

	First	Follow
S'	λ	$
S	λ	$ a

Construir la tabla de parsing SLR(1)

	Acciones		Go to	
	$	a	S’	S
I0	reducir(iii)			1
I1	shitf(2)	shift(3)		
I2	aceptar			
I3	reducir(ii)			

Podemos concluir que en esta gramática se presenta un shift-reduce en I0.

Tenemos que G1d:

$S \rightarrow aS$

$S \rightarrow \lambda$

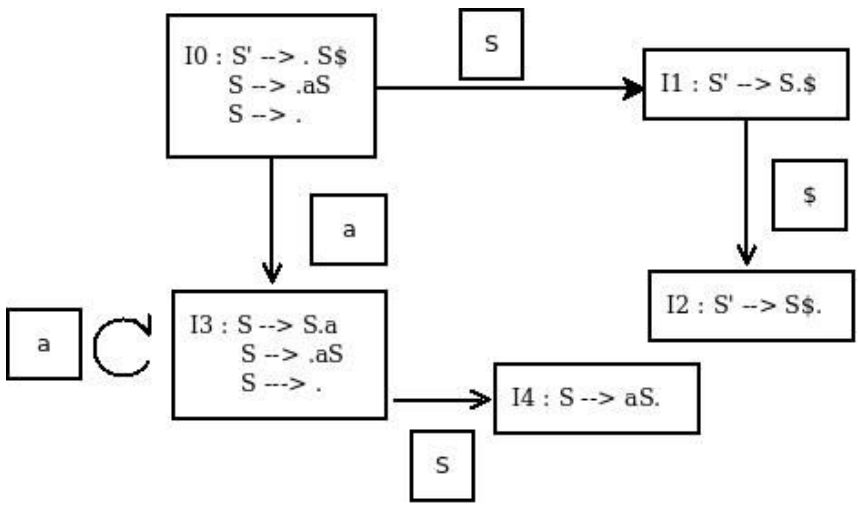
Le agregamos el valor inicial:

$S' \rightarrow S$ (i)

$S \rightarrow aS$ (ii)

$S \rightarrow \lambda$ (iii)

Realizamos el automata de prefijos viables:



Calculamos los first and follow

	First	Follow
S’	λ a	$
S	λ a	$

Construir la tabla de parsing SLR(1)

	Acciones		Go To	
	$	a	S’	S
I0	shift(3)	reduce(iii)		1
I1		shift(2)		
I2	aceptar			
I3	shift(3)	reduce(iii)		4
I4	reduce(i)			

Podemos concluir que en esta gramática se presenta un shift-reduce en I0 y I3.

(b) Compare la eficiencia de ambos analizadores en términos de espacio, i.e. los tamaños de sus tablas y la cantidad de pila utilizada para reconocer cada frase de L(a*), y de tiempo, i.e. cantidad de movimientos realizados por el autómata de pila para reconocer cada frase de L(a*).

Para comparar la eficiencia de ambos analizadores se usó el reconocedor en tres palabras “aaa”, “aaaaa” y “aaaaaaa”. Al terminar los tres ejemplos mencionados anteriormente se pudo observar que la cantidad de movimientos en ambos analizadores son el mismo entonces el tiempo también es igual, la diferencia es que la cantidad de pila usada G1d es mayor que la usada en G1i, entonces podemos concluir que G1i es más eficiente.

2. Sea G2 la gramática ({ Instr } , { ; , IS} , P2 , Instr), con P2 compuesto:

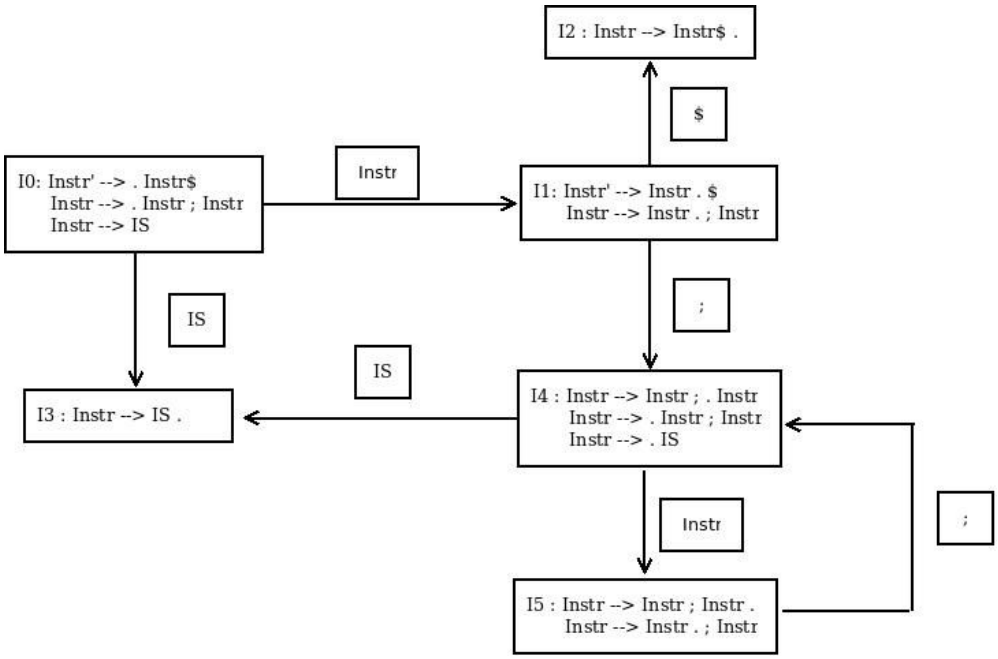
Instr → Instr ; Instr
Instr → IS

(a) Muestre que G2 no es una gramática LR(1), intentando construir un analizador sintáctico LR(1) para ella y consiguiendo que tal analizador tendría un conflicto.

Primero le agregamos el valor inicial:

Instr' → Instr (i)
Instr → Instr ; Instr (ii)
Instr → IS (iii)

Realizamos el automata de prefijos viables:

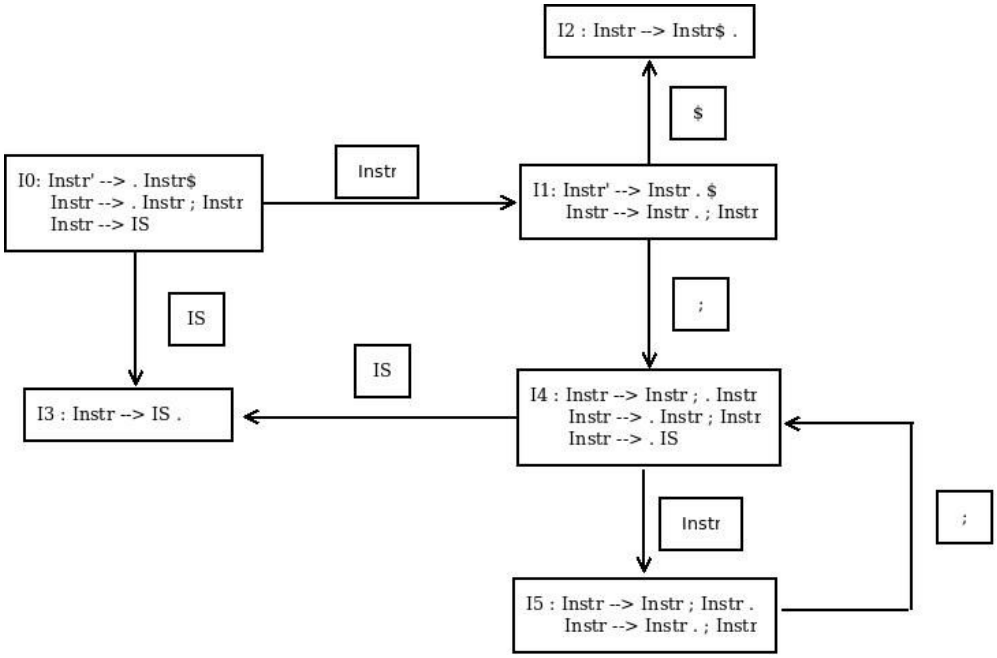


Al realizar el autómata de prefijos viables se puede observar que el I5 se presenta un problema de shift-reduce. En la pregunta (b) se terminará de construir el analizador sintáctico.

(b) Construya el analizador sintáctico, especificando cuál es el conflicto y de que tipo (shift - reduce o reduce -reduce)

Primero le agregamos el valor inicial:

Instr' → Instr (i)
Instr → Instr ; Instr (ii)
Instr → IS (iii)



Realizamos el automata de prefijos viables:

Se puede observar que se presenta un problema de shift - reduce en I5, entonces se construye la tabla de los first and follow

	First	Follow
Instr’	IS	$
Instr	IS	$;

Se construye la tabla de parser SLR(1):

	Acciones			Go To	
	IS	;	$	Instr’	Instr
I0	shift(3)				1
I1		shift(4)	shift(2)		
I2	aceptar				
I3	reduce(iii)				
I4	shift(3)				5
I5		shift(4) o	reduce(ii)		

		reduce(ii)			

En la tabla SLR(1), se puede observar que el problema que se presenta de shift-reduce, se puede solucionar de dos maneras, aplicando un shift(4) o un reduce(ii)

(c) Considere las dos alternativas de eliminación del conflicto (i.e. en favor del shift o en favor del reduce en caso de un conflicto shift-reduce, o en favor de producción o de otra en caso de un conflicto reduce-reduce). Muestre, para ambas alternativas de eliminación del conflicto, la secuencia de reconocimiento de la frase IS ; IS ; IS dando como salida la secuencia de producciones reducidas ¿A qué corresponde cada una de las alternativas: a asociar el operador de secuenciación hacia la izquierda o hacia la derecha?

- Resolvemos el problema con shift(4), el operador de secuenciación está a la derecha

Pila	Entrada	Acción
I0	IS ; IS ; IS	avanzar(3)
I3 I0	; IS ; IS	reduce(iii)
I1 I0	; IS ; IS	avanzar(4)
I4 I1 I0	IS ; IS	avanzar(3)
I3 I4 I1 I0	; IS	reduce(iii)
I5 I4 I1 I0	; IS	avanzar(4)
I4 I5 I4 I1 I0	IS	avanzar(3)
I3 I4 I5 I4 I1 I0	$	reducir(iii)
I5 I4 I5 I4 I1 I0	$	reducir(ii)
I5 I4 I1 I0	$	reducir(ii)
I1 I0	$	avanzar(2)
I2 I1 I0	$	aceptar

- Resolvemos el problema con reduce(ii),el operador de secuenciación está a la izquierda.

Pila	Entrada	Acción
I0	IS ; IS ; IS	avanzar(3)
I3 I0	; IS ; IS	reduce(iii)
I1 I0	; IS ; IS	avanzar(4)
I4 I1 I0	IS ; IS	avanzar(3)
I3 I4 I1 I0	; IS	reduce(iii)
I5 I4 I1 I0	; IS	reduce(ii)
I1 I0	; IS	avanzar(4)
I4 I1 I0	IS	avanzar(3)
I3 I4 I1 I0	$	reducir(iii)
I5 I4 I1 I0	$	reducir(ii)
I1 I0	$	avanzar(2)
I2 I1 I0	$	aceptar

(d) En la etapa II se concluyó que era indiferente resolver esta ambigüedad hacia la izquierda o hacia la derecha. Compare ahora la eficiencia de ambas alternativas, en términos de la cantidad de pila y del tiempo que se utiliza para reconocer frases de la forma IS (; IS)^n con n un número natural. ¿Cuál alternativa conviene entonces utilizar?

Para comparar la eficiencia de ambos analizadores se usó el reconocedor en dos palabras “IS ; IS ; IS”,” y “IS ; IS ; IS ; IS ; IS”. En ambos caso se pudo observar que la cantidad de movimientos fueron iguales lo cual implica que el tiempo de ejecución también es igual.

La diferencia que se presenta es que en el analizador con el operador con secuenciación a la derecha primero empila hasta consumir la frase y después desempila hasta que se acepta la frase, mientras que en el operador con secuenciación a la izquierda el analizador va empilando y desempilando una y otra vez hasta que acepta la frase. Entonces podemos decir que el operador con secuenciación a la izquierda es más eficiente debido a que usa menos espacio en la pila.