Instituto Tecnológico de Costa Rica

Compiladores e intérpretes

I Sem

Documentación del Parser

Profesora: Erika Marín Schumann

Estudiantes:

Ariel Araya Corrales

Luis Adrián Badilla Godínez

Juan Diego Bejarano Pino

Cartago, Costa Rica

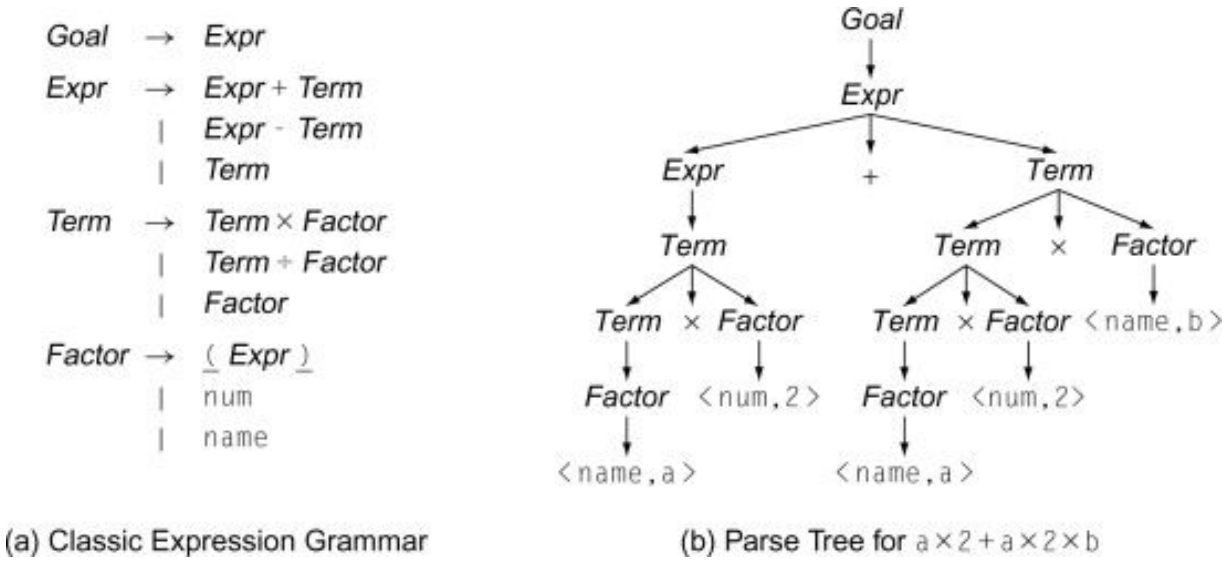
Julio, 2020

Índice

Introducción

El parseo es una parte sumamente importante del proceso de compilación, ya que es el paso que hace el análisis sintáctico del código, y nos da el camino hacia el análisis semántico del lenguaje, parsing es la manera de decir el análisis estructural del lenguaje, donde se validan todos los aspectos como el orden, formas de declaración de estructuras, como las variables, bloques de código como if, for y while. A partir del análisis sintáctico se puede generar un árbol de sintaxis, el cual se usa para, a partir de las decisiones que se dejan claras en este, buscar los alcances de cada contrato, función o bloque, y hacer los análisis semánticos, para luego traducir este código al lenguaje final (ensamblador, otro lenguaje o a una versión más nueva del programa).

Para una gramática, como se mencionó anteriormente se crea un parse tree, el cual si se sigue se puede construir cualquier string que pertenece a la gramática, diciéndonos si algo ingresado al compilador es válido o no, y si este no es válido, donde se encuentra el problema; una vez solucionado los problemas, se puede pasar a los siguientes pasos de la compilación.



(a) Classic Expression Grammar (b) Parse Tree for a×2+a×2×b

Figura 1: Ejemplo de Parse tree

Estrategia de solución

Este proyecto encuentra su base en el anterior, el scanner para el lenguaje de programación Solidity. Para este se presenta el siguiente detalle de lo necesario con respecto a la lógica para lograr el parser o analizador sintáctico del lenguaje Solidity.

Lógica

Lo primer es crear otro archivo .jflex en el cual se va a copiar y pegar la definición del scanner, solo que este en vez de utilizar el enumerador Tokens, utilizará la clase Symbols que se encuentra en java_cup.runtime. Esto para que se pueda conectar con cup. Lo segundo es crear el .cup con los detalles necesarios. Con el .cup y el .jflex creados se modifica la clase main, que se encarga de generar los archivos Parser.java, ScannerF.java (el anterior que ya teníamos para mostrárselo al usuario) y ScannerC.java (el que utilizará cup para generar el análisis sintáctico). Una vez listo estos archivos se deben escribir las reglas de la gramática del lenguaje para completar el .cup. Por último, desarrollar el despliegue de errores del cup.

Análisis de resultados

En esta sección se presenta una tabla con el detalle de la completitud de las tareas que se deben cumplir a nivel funcional.

Tarea	Porcentaje completado(%)	Detalle
Creación del archivo .cup y el .jflex para el cup.	100%	
Modificación de la clase main para que genere los archivos.	100%	
Creación de todas las reglas de la gramática libre de contexto para el lenguaje Solidity.	...	....
Modificar la interfaz para que nos sea posible visualizar los dos análisis.	100%	
Crear una forma de visualizar los errores.	90%	Se puede visualizar los errores, solo que hay que fijarse desde el token que sigue, no dice exactamente cual es el error encontrado

Lecciones aprendidas

En el transcurso de este proyecto se aprendieron varias cosas importantes, de varios aspectos del desarrollo.

En la parte Técnica del proyecto, podemos rescatar que JCup no está bien documentado, en lo absoluto, es un problema al momento de resolver problemas, desde errores ambiguos retornados al momento de compilar, hasta falta completa de documentación en la cual se puede ver exactamente cómo hacer las cosas, en vez de tener que ir a videos de Terceros para revisar dichas situaciones, los cuales pueden ser extremadamente desactualizados. Lo cual nos enseña que antes de decidir por una tecnología, buscar alternativas, en este caso se pudo haber usado JBison, que tiene una relación más estrecha a JFlex.

Otro punto importante a rescatar entre la parte técnica es el orden de la construcción de la gramática, la cual debería iniciar por las reglas y producciones más sencillas o aquellas que son directamente terminales, posteriormente aquellas reglas que indiquen listas o repeticiones de estas y por último las reglas más complejas hasta el inicio del árbol.

Por otro lado tenemos las lecciones personales, este es un proyecto el cual toma bastante tiempo, por lo cual una mejor comunicación y organización con más tiempo ayudaría mucho al proceso de desarrollo ya que se empezó a trabajar hasta que quedo poco tiempo al momento de entregar el proyecto.

Casos de pruebas

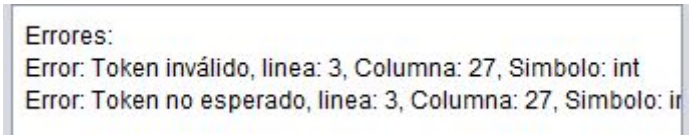
Para los casos de prueba, se proveyeron varios archivos de prueba, estos tienen el nombre PruebaEstructura, PruebaEstructurasControl, PruebaExpresiones, PruebaFunciones y PruebaVariables, los cuales tienen pruebas dependiendo de los nombres de cada uno. El objetivo es detectar los errores señalados, así como no resaltar errores inexistentes.

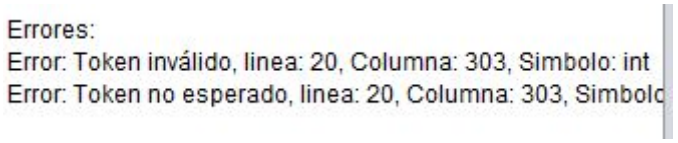
Pruebas Estructura

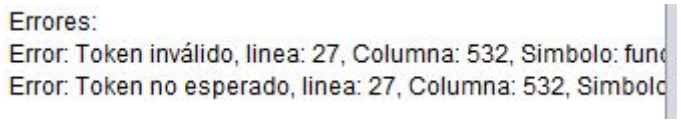
Salidas esperadas

Errores en las líneas: 3, 20 y 26

Salidas obtenidas







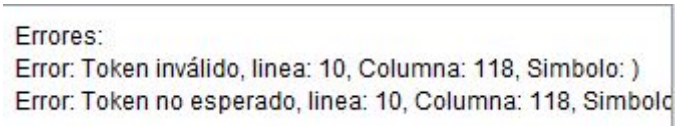
En la última figura, tenemos el error en una línea 27 en vez de la 26, esto es esperado, ya que al no encontrar el cierre de función, pone el marcador de error una línea adelante, cuando encuentra algo que no espera.

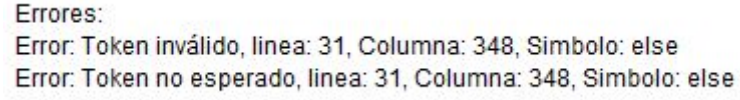
Pruebas Estructuras de control

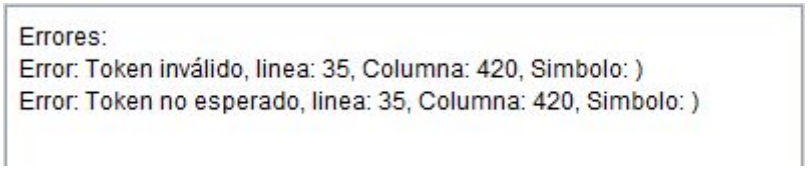
Salidas esperadas

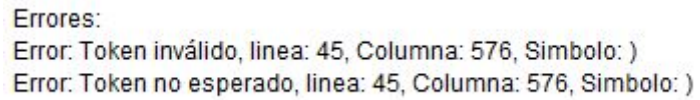
Errores en las líneas: 10, 31, 35, 45 y 54

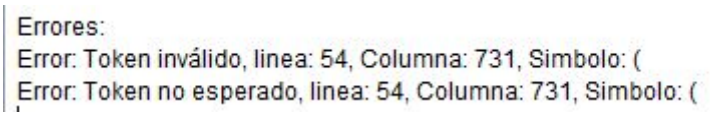
Salidas obtenidas











El parser encontró sin problemas todos los errores especificados en el archivo de prueba, sin embargo se encontraron 2 errores que no debieron haber sido considerados como esto, los cuales se pueden denotar de la siguiente manera:

```
Errores:
Error: Token inválido, linea: 49, Columna: 654, Simbolo: (
Error: Token no esperado, linea: 49, Columna: 654, Simbolo: (
```

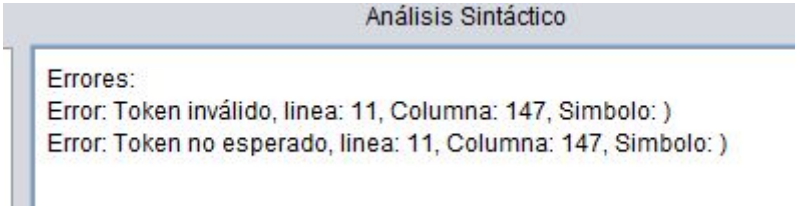
```
Errores:
Error: Token inválido, linea: 56, Columna: 762, Simbolo: (
Error: Token no esperado, linea: 56, Columna: 762, Simbolo: (
```

Pruebas Expresiones

Salidas esperadas

Errores en las líneas: 23 y 27

Salidas obtenidas



Error en la línea 11 (x();)ya que el programa no espera una función sin parámetros.

```
Errores:
Error: Token inválido, linea: 21, Columna: 369, Simbolo: (
Error: Token no esperado, linea: 21, Columna: 369, Simbolo: (
```

Error en la línea 21 (a = 345 +8 * a + (5-3);) ya que no espera un paréntesis como expresión.

```
Errores:
Error: Token inválido, linea: 25, Columna: 459, Simbolo: a
Error: Token no esperado, linea: 25, Columna: 459, Simbolo: a
```

Error esperado en la línea 23 (a = a + 1), como no se cierra el punto y coma, da el error en la siguiente línea en la que hay código.

```
Errores:
Error: Token inválido, linea: 26, Columna: 499, Simbolo: this
Error: Token no esperado, linea: 26, Columna: 499, Simbolo: this
```

Error en la línea 26 (a= id.this();), ya que el programa no espera que se acceda a un valor this de un identificador.

```
Errores:
Error: Token inválido, linea: 27, Columna: 519, Simbolo: ;
Error: Token no esperado, linea: 27, Columna: 519, Simbolo: ;
```

Error esperado en la línea 27 (a = a. ;), ya que no se especifica el atributo o función al que se accede del identificador,

Pruebas Funciones

Salidas esperadas

Errores en las líneas: 6, 14 y 21

Salidas obtenidas

```
Errores:
Error: Token inválido, linea: 6, Columna: 65, Simbolo: x
Error: Token no esperado, linea: 6, Columna: 65, Simbolo: x
```

```
Errores:
Error: Token inválido, linea: 14, Columna: 161, Simbolo: ,
Error: Token no esperado, linea: 14, Columna: 161, Simbolo
```

```
Errores:
Error: Token inválido, linea: 21, Columna: 278, Simbolo: )
Error: Token no esperado, linea: 21, Columna: 278, Simbolo
```

Pruebas Variables

Salidas esperadas

Errores en las líneas: 7, 12, 15, 18, 21, 28 y 35

Salidas obtenidas

```
Errores:
Error: Token inválido, linea: 7, Columna: 102, Simbolo: id3
Error: Token no esperado, linea: 7, Columna: 102, Simbolo: id3
```

```
Errores:
Error: Token inválido, linea: 11, Columna: 139, Simbolo: ,
Error: Token no esperado, linea: 11, Columna: 139, Simbolo: ,
```

```
Errores:
Error: Token inválido, linea: 13, Columna: 178, Simbolo: int
Error: Token no esperado, linea: 13, Columna: 178, Simbolo: int
```

```
Errores:
Error: Token inválido, linea: 13, Columna: 185, Simbolo: ,
Error: Token no esperado, linea: 13, Columna: 185, Simbolo: ,
```

```
Errores:
Error: Token inválido, linea: 14, Columna: 207, Simbolo: private
Error: Token no esperado, linea: 14, Columna: 207, Simbolo: private
```

```
Errores:
Error: Token inválido, linea: 15, Columna: 228, Simbolo: public
Error: Token no esperado, linea: 15, Columna: 228, Simbolo: public
```

```
Errores:
Error: Token inválido, linea: 18, Columna: 267, Simbolo: b;//error
Error: Token no esperado, linea: 18, Columna: 267, Simbolo: b;//error

Errores:
Error: Token inválido, linea: 21, Columna: 303, Simbolo: ;
Error: Token no esperado, linea: 21, Columna: 303, Simbolo: ;

Errores:
Error: Token inválido, linea: 25, Columna: 348, Simbolo: ,
Error: Token no esperado, linea: 25, Columna: 348, Simbolo: ,

Errores:
Error: Token inválido, linea: 28, Columna: 380, Simbolo: {
Error: Token no esperado, linea: 28, Columna: 380, Simbolo: {


Errores:
Error: Token inválido, linea: 29, Columna: 400, Simbolo: ,
Error: Token no esperado, linea: 29, Columna: 400, Simbolo: ,

Errores:
Error: Token inválido, linea: 34, Columna: 461, Simbolo: ,
Error: Token no esperado, linea: 34, Columna: 461, Simbolo: ,

Errores:
Error: Token inválido, linea: 36, Columna: 532, Simbolo: int
Error: Token no esperado, linea: 36, Columna: 532, Simbolo: int

Errores:
Error: Token inválido, linea: 36, Columna: 539, Simbolo: ,
Error: Token no esperado, linea: 36, Columna: 539, Simbolo: ,
```

Los errores presentes en las líneas 11, 13, 25, 29, 34 y 36 corresponden a que parte de lo que no se pudo implementar la declaración múltiple de variables. También hay errores indicados una línea después de que aparecen, esto ya que al no encontrar el cierre de sentencia, pone el marcador de error una línea adelante, cuando encuentra algo que no espera.

Manual de usuario

La aplicación es sencilla dado a que es una modificación de la anterior. En este caso lo que se le despliega al usuario es una ventana que contiene, dos paneles y tres botones. Los cuales tienen las siguientes funcionalidades:

- Análisis Léxico: El panel izquierdo y el botón superior izquierdo son los utilizados por esta funcionalidad. Al usuario a dar clic el programa desplegará un selector de archivos para que seleccione el archivo con código fuente en Solidity para que lo pueda analizar léxicamente y que despliegue los resultados en el panel izquierdo.
- Análisis Sintáctico: En este caso se utilizan el panel derecho y el botón inferior derecho. Al usuario se le notificará los errores sintácticos en caso de haber. Si no solo se le desplegará un mensaje de análisis finalizado.

Bitácora de trabajo

En esta sección se muestran los registros de todas las actividades de trabajo.

Fecha y hora inicio	12/7/2020, 21:03	Participantes
Detalle	Primera revisión del documento de especificación del proyecto.	Adrián Badilla

Fecha y hora inicio	13/7/2020, 15:55	Participantes
Detalle	Consulta a la profesora sobre corrección necesaria en el Scanner.	Adrián Badilla

Fecha y hora inicio	14/7/2020, 11:18	Participantes
Detalle	Corrección del archivo .jflex para que	Adrián Badilla

	cada token de tipo reservada, unit o transaction se vean individualmente. Creación del .cup y la primera regla de la CFG.	

Fecha y hora inicio	15/7/2020, 10:38	Participantes
Detalle	Modificación del main. Conexión entre el jflex nuevo y el cup. Creación de una clase SError para el manejo de errores en una lista tanto en el .cup como en el .jflex.	Adrián Badilla

Fecha y hora inicio	16/7/2020, 10:38	Participantes
Detalle	Detalles en el main, la interfaz y la conexión. Aclaración al grupo de trabajo qué falta (Crear las demás reglas y ver cómo se hace para desplegar los errores según la implementación de la lista)	Adrián Badilla

Fecha y hora inicio	17/7/2020, 9:23	Participantes
Detalle	Creación del documento de google para la documentación.	Adrián Badilla

Fecha y hora inicio	20/7/2020, 11:50	Participantes
Detalle	Revisión de Especificación a detalle para generación de gramática	Juan Diego Bejarano

Fecha y hora inicio	20/7/2020, 12:30	Participantes
Detalle	Creación de la primera sección de la gramática, Manejo de Variables, tipos de datos y Asignaciones	Juan Diego Bejarano

Fecha y hora inicio	21/7/2020, 13:00	Participantes
Detalle	Administración de Repositorio, Merges de Branches a Master	Juan Diego Bejarano

Fecha y hora inicio	22/7/2020, 13:00	Participantes
Detalle	Arreglos en código de error de sintaxis, Agregados los structs parámetros y base de funciones	Juan Diego Bejarano

Fecha y hora inicio	22/7/2020, 17:00	Participantes
Detalle	Arreglo de funcion, soluciones de bugs y agregado el lookback para	Juan Diego Bejarano, Adrian Badilla

	errores	

Fecha y hora inicio	23/7/2020, 13:00	Participantes
Detalle	Eliminación de variables usadas, análisis de errores shift / reduce	Juan Diego Bejarano, Adrian Badilla

Fecha y hora inicio	24/7/2020, 15:00	Participantes
Detalle	Debugeo de gramáticas	Juan Diego Bejarano, Ariel Araya

Fecha y hora inicio	25/7/2020, 13:00	Participantes
Detalle	Eliminación de errores shift / reduce	Juan Diego Bejarano, Ariel Araya

Fecha y hora inicio	25/7/2020, 17:00	Participantes
Detalle	Arreglos de Gramáticas	Juan Diego Bejarano, Ariel Araya, Adrian Badilla

Fecha y hora inicio	25/7/2020, 20:30	Participantes
Detalle	Pruebas finales	Juan Diego Bejarano, Adrian Badilla, Ariel Araya

Fecha y hora inicio	25/7/2020, 23:00	Participantes
Detalle	Finalización de Documentación	Juan Diego Bejarano, Adrian Badilla, Ariel Araya

Bibliografía

JCup y JFlex | analizador sintáctico con java (explicación paso a paso). Charlead (Director). (2019a, May 29,).[Youtube] Recuperado de https://www.youtube.com/watch?v=4Z6Tnit810Y&t=1452s

JFlex | analizador léxico con java (explicación paso a paso). Charlead (Director). (2019b, Feb 17,).[Youtube] Recuperado de https://www.youtube.com/watch?v=5mIRrn2yEe8&t=297s

Jflex y cup Ejemplo1 parte 1. Codigo, 6. 1. (Director). (2016a, Sep 4,).[Youtube] Recuperado de https://www.youtube.com/watch?v=RnxEgEYnp9o&t=769s

Jflex y cup Ejemplo1 parte 2. Codigo, 6. 1. (Director). (2016b, Sep 4,).[Youtube] Recuperado de https://www.youtube.com/watch?v=AQkd5AwjV1E

CUP 0.11b. Recuperado de http://www2.cs.tum.edu/projects/cup/

JFlex. Recuperado de https://www.jflex.de/index.html

Klein, G., Rowe, S. & Décamps, R. (2020). JFlex user's manual. Recuperado de https://jflex.de/manual.html#CUPWork

Solidity a fondo. Recuperado de

https://solidity-es.readthedocs.io/es/latest/solidity-in-depth.html