Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 - Traductores e Interpretadores
Enero - Marzo 2016

Etapa III
Análisis de Contexto

Esta tercera etapa del proyecto consiste en realizar *análisis de contexto* sobre programas escritos en `BOT`. El *Árbol Sintáctico Abstracto (AST)*, implementado en la etapa anterior, deberá ser aumentado con información de contexto (por ejemplo, de las variables y sus tipos). En el proceso se debe verificar ahora los errores estáticos que puedan estar presentes (por ejemplo, errores de tipo como: `3 + true`).

Concretamente usted deberá implementar (en conjunto con lo realizada para las dos etapas anteriores), la verificación de errores estáticos. Esto incluye la creación de una tabla de símbolos para almacenar las variables que han sido declaradas, sus tipos y sus valores. Los errores estáticos que deben manejarse son los siguientes:

- Utilización de variables que no hayan sido declaradas.
- Redeclaración la misma variable.
- Utilización de la palabra reservada `me`, fuera de un comportamiento.
- Errores de tipo (Por ejemplo: Intentar sumar un entero y un booleano).

La tabla de símbolos a utilizar debe ser implementada con una estructura eficiente, como una tabla de hash. Por motivo de las instrucciones de incorporación de alcance, la tabla de símbolos debe ser jerárquica. Esto es, si un nombre no se encuentra en la tabla de símbolos local, debe buscarse en la tabla de símbolos inmediatamente superior. Sólo si, al recorrer todas las tablas de símbolos no se encuentra el nombre, es que debe reportarse que no ha sido declarado. Las redeclaraciones de variables sólo pueden ocurrir en un mismo nivel (dondo no se han incorporado nuevos alcances). Si un nombre es usado en diferentes niveles, el del nivel más interno esconde la definición del nivel mas externo en la extensión de su alcance.

Como referencia para la implementación de buenas tablas de símbolos jerárquicas, se recomienda revisar el capítulo 2, sección 7 de [ALSU07].

Al igual que para la entrega pasada, si el programa analizado presenta errores léxicos debe mostrarlos *todos*. Si presenta un error sintáctico o un error de contexto (descritos anteriormente) debe imprimir solamente el primero (sin importar si es sintáctico o de contexto). Si el programa analizado no presenta errores, debe imprimir el árbol sintáctico abstracto asociado al mismo.

Revisión Teórico-Práctica En la Etapa II, se analizó el manejo de gramáticas ambiguas por la herramienta sintáctica de se elección. Esto se hizo en términos de "conflictos como indicadores de ambigüedad" y la resolución de éstos por medio de del estilo "`%left`, `%right`, `%nonassoc`" (o similares, dependiendo del lenguaje). Ahora se desea que Ud. profundice el estudio de tales conflictos en el contexto de análisis sintáctico LR(1). Además, se desea que Ud. haga algunos análisis de eficiencia en este mismo contexto. Empezaremos por esto último y luego procederemos al análisis del manejo de ambigüedad.

Nota: Las clases de teoría necesarias para resolver esta revisión teórico-práctica serán después del parcial. Sin embargo, si quieren adelantar trabajo les recomiendo revisar [Her06a] y [Her06b].

1. Sea $G1_i$ la gramática recursiva-izquierda $(\{S\}, \{a\}, \{S \to Sa, S \to \lambda\}, S)$ y sea $G1_d$ la gramática recursiva-derecha $(\{S\}, \{a\}, \{S \to aS, S \to \lambda\}, S)$. Ambas generan el lenguaje denotado por la expresión regular a^*, i.e. el lenguaje $\mathcal{L}(a^*)$.

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

   (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 $\mathcal{L}(a^*)$, y de tiempo, i.e. cantidad de movimientos realizados por el autómata de pila para reconocer cada frase de $\mathcal{L}(a^*)$.

   Nota: Un análisis <u>serio</u> de eficiencia debe hacerse en términos de órdenes de complejidad, e.g. mediante notación $O(f)$.

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

   $Instr \to Instr\ ;\ Instr$
   $Instr \to IS$

   (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.

   (b) A pesar de que la gramática $G2$ no es LR(1), se puede construir un analizador sintáctico LR(1) con conflictos para ella (lo cual corresponde a un autómata de pila no-determinístico). Construya tal analizador sintáctico, especificando cuál es el conflicto y de qué tipo (i.e. *shift-reduce* o *reduce-reduce*) es.

   (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 una 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?

   (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?

   Nota: Recuerde que un análisis <u>serio</u> de eficiencia debe hacerse en términos de órdenes de complejidad, e.g. mediante notación $O(f)$.

Entrega de la Implementación Ud. debe entregar un correo electrónico que contenga:

- El código fuente en el lenguaje y la herramienta de su elección, entre los permitidos, de su analizador sintáctico, aumentado con análisis de contexto. Todo el código debe estar debidamente documentado. El analizador deberá ser ejecutado con el comando "`./ContBot` ⟨*Archivo*⟩", por lo que es posible que tenga que incorporar un script a su entrega que permita que la llamada a su programa se realice de esta forma. ⟨*Archivo*⟩ tendrá el programa escrito en `BOT` que se analizará sintácticamente y por contexto. Este archivo tendrá extensión ‘‘`.bot`’’, sin embargo no tiene que verificar esto.

- Un <u>breve</u> informe explicando la formulación/implementación de su analizador de contexto y justificando todo aquello que Ud. considere necesario. Además el informe debe contener sus respuestas a la revisión teórico–práctica.

Nota: Es importante que su código pueda ejecutarse en las máquinas del LDC, pues es ahí y únicamente ahí donde se realizará su corrección.

Referencias Bibliográficas

[ALSU07] A. Aho, M. Lam, R. Sethi & J. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 2007, Segunda Edición.

[Her06a] E. Hernandez-Novich *Guía de LR(0)*. Disponible aqui.

[Her06b] E. Hernandez-Novich *Guía de LR(1)*. Disponible aquí.

Fecha de Entrega: Viernes, 4 de Marzo (Semana 11), hasta las 11:59 pm.
Valor: 8%.

S. Ancelmo, C. Ferreira y R. Monascal / Enero 2016