IIC2323 — Construcción de Compiladores
Agú: Descripción Sintáctica

Semestre 1/2016

1. Notación

En el documento, el lenguaje es definido utilizando gramáticas libres de contexto, donde en el lado derecho de las producciones se utilizan expresiones regulares con la siguiente notación:

1. $[\varphi]$ corresponde a $\varphi + \varepsilon$, es decir, que φ es opcional,
2. $\{\varphi\}$ corresponde a φ^*, es decir que φ se repite una cantidad arbitraria de veces,
3. $\varphi|\psi$ es equivalente a $\varphi + \psi$, es decir que se utiliza φ o ψ
4. (φ) corresponde a φ.

Por ejemplo, la gramática:

$$
\begin{aligned}
identifier &\rightarrow [\ \texttt{\$}\ ] letter \{letter | digit\} \\
letter &\rightarrow\ \texttt{a} \mid \texttt{b} \mid \texttt{c} \\
digit &\rightarrow\ \texttt{0} \mid \texttt{1}
\end{aligned}
$$

indica que un identificador comienza opcionalmente con $, continúa con una letra, y termina con una cantidad arbitraria de letras y dígitos. Algunos identificadores válidos según esta son: $a101b, b, y $c0, mientras que 1a00c y a$01 no lo son.

1.1. Tokens

- Cada keyword y símbolo es un token. En la gramática se representan por su valor literal, usualmente en mayúsculas.
- `ID` representa a un identificador, y `TYPE_ID` a un identificador de tipo.
- Las expresiones literales son: `INT` , `CHAR` , `STRING` , y `BOOL` , para valores enteros, caracteres, strings, y booleanos respectivamente.
- Los separadores se indican con `SEP` .
- Los comentarios y espacios en blanco no son tokens.

En general, los *no_terminales* se distinguen de los `terminales` de forma tipográfica.

2. Elementos sintácticos

2.1. Símbolos especiales

Estos se utilizan sólo para evitar ambigüedades en este documento, y corresponden a los tokens respectivos.

$$\begin{aligned} LP &\rightarrow \texttt{(} \\ RP &\rightarrow \texttt{)} \\ LCB &\rightarrow \{ \\ RCB &\rightarrow \} \end{aligned}$$

2.2. Programa

Un programa es una secuencia de declaraciones. El no terminal *program* funciona como símbolo inicial.

$$program \rightarrow \{ decl \texttt{ SEP} \}$$

2.3. Tipos

Los tipos pueden ser simples (`TYPE_ID`), o "firmas" de funciones:

$$\begin{aligned} type &\rightarrow \texttt{TYPE\_ID} \\ &\rightarrow type \texttt{->} type \\ &\rightarrow LP\ type\ RP \end{aligned}$$

2.4. Declaraciones

Las declaraciones definen elementos fundamentales del lenguaje.

$$\begin{aligned} decl &\rightarrow alias\_decl \\ &\rightarrow var\_decl \\ &\rightarrow const\_decl \\ &\rightarrow func\_decl \\ &\rightarrow assignment \\ &\rightarrow classdef \\ &\rightarrow while\_stmt \\ &\rightarrow expression \end{aligned}$$

2.5. Declaración de alias

Un alias permite agregar un nombre a un tipo.

$$alias\_decl \rightarrow \texttt{alias TYPE\_ID =} type$$

2.6. Declaración de variables

$$var\_decl \rightarrow \texttt{var ID :}\ type[\texttt{=}\ expression]$$

2.7. Declaración de constantes

$$const\_decl \rightarrow \texttt{let ID :}\ type\ \texttt{=}\ expression$$

2.8. Asignación de expresiones

$$\begin{aligned} assignment &\rightarrow \texttt{set ID \{ ID \} =}\ expression \\ &\rightarrow \texttt{set \$ ID \{ ID \} =}\ expression \\ &\rightarrow \texttt{\$ ID <-}\ expression \end{aligned}$$

2.9. Declaración de clases

$$\begin{aligned} classdef &\rightarrow [\ \texttt{mut}\ ]\ \texttt{class TYPE\_ID}\ [constructor\_args]\ \texttt{=}\ (classdef\_body | class\_ctor) \\ classdef\_body &\rightarrow LCB\{classdef\_body\_decl\ \texttt{SEP}\ \}RCB \\ classdef\_body\_decl &\rightarrow const\_decl | var\_decl | func\_decl \\ constructor\_args &\rightarrow LP[constr\_arg\_dec\{\ \texttt{,}\ constr\_arg\_dec\}]RP \\ constr\_arg\_dec &\rightarrow [\ \texttt{var} \mid \texttt{val}\ ]\ \texttt{ID :}\ type \\ class\_ctor &\rightarrow\ \texttt{TYPE\_ID}\ LP[expression\{\ \texttt{,}\ expression\}]RP \end{aligned}$$

2.10. Declaración de funciones

$$\begin{aligned} func\_decl &\rightarrow\ \texttt{let ID}\ func\_arg\{func\_arg\}[\texttt{:}type]\ \texttt{=}\ body \\ body &\rightarrow expression | block\_expression \\ func\_arg &\rightarrow\ \texttt{ID :}\ type \end{aligned}$$

2.11. While

$$\begin{aligned} while\_stmt &\rightarrow\ \texttt{while}\ expression\ do\_expression \\ do\_expression &\rightarrow\ \texttt{do}\ expression \\ &\rightarrow\ \texttt{do}\ block\_expression \\ &\rightarrow\ \texttt{do}\ assignment \end{aligned}$$

2.12. Expresiones

2.12.1. If-then-else y when

$$
\begin{aligned}
\textit{expression} \rightarrow\ & \texttt{if}\ \textit{expression}\ \texttt{then}\ \textit{body}\ \texttt{else}\ \textit{body} \\
\rightarrow\ & \texttt{when}\ \textit{expression}\ \texttt{DO}\ \textit{expression} \\
\rightarrow\ & \texttt{when}\ \textit{expression}\ \texttt{DO}\ \textit{block\_expression}
\end{aligned}
$$

2.12.2. Paréntesis y operaciones

$$
\begin{aligned}
\textit{expression} \rightarrow\ & \textit{LP}\ \textit{expression}\ \textit{RP} \\
\rightarrow\ & \texttt{-}\ \textit{expression} \\
\rightarrow\ & \texttt{+}\ \textit{expression} \\
\rightarrow\ & \texttt{not}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{/}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{*}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{+}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{-}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{\%}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{==}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{/=}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{<}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{>}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{<=}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{>=}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{\&\&}\ \textit{expression} \\
\rightarrow\ & \textit{expression}\ \texttt{||}\ \textit{expression}
\end{aligned}
$$

2.12.3. Construcción de objetos nuevos

$$
\textit{expression} \rightarrow\ \texttt{TYPE\_ID}\ \textit{LP}\ [\textit{expression}\{\ \texttt{,}\ \textit{expression}\}]\ \textit{RP}
$$

2.12.4. Acceso directo a atributos

$$
\textit{expression} \rightarrow\ \texttt{\$ ID}
$$

2.12.5. Llamada a funciones o acceso a atributos

$$
\textit{expression} \rightarrow \textit{expression}\ \textit{expression}\{\textit{expression}\}
$$

2.12.6. Acceso a variables y expresiones literales

$$
\begin{aligned}
\textit{expression} &\rightarrow \texttt{ID} \\
&\rightarrow \texttt{INT} \\
&\rightarrow \texttt{CHAR} \\
&\rightarrow \texttt{STRING} \\
&\rightarrow \texttt{BOOL}
\end{aligned}
$$

2.12.7. Expresiones Let

$$
\begin{aligned}
\textit{expression} &\rightarrow \texttt{let}\ \textit{let\_expr}\{\ \texttt{,}\ \textit{let\_expr}\}\ \texttt{in}\ \textit{body} \\
\textit{let\_expr} &\rightarrow \texttt{ID}\ \texttt{:}\ \textit{type}\ \texttt{=}\ \textit{expression}
\end{aligned}
$$

2.13. Bloques

$$
\begin{aligned}
\textit{block\_expression} &\rightarrow \textit{LCB}\ \{\textit{block\_statement}\ \texttt{SEP}\ \}\ \textit{RCB} \\
\textit{block\_statement} &\rightarrow \textit{var\_decl} \\
&\rightarrow \textit{const\_decl} \\
&\rightarrow \textit{while\_stmt} \\
&\rightarrow \textit{assignment} \\
&\rightarrow \textit{expression}
\end{aligned}
$$

3. Precedencia de operadores y asociatividad

operadores	
`not + -` (unarios)	más fuerte
`* / %`	
`+ -` (binarios)	
`< <= == /= >= >`	
`&& ||`	más débil

Todos los operadores anteriores son asociativos por la izquierda. El uso de `->` es asociativo por la derecha.