Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3032 - Construcción de Compiladores

Aspectos Léxicos y Sintácticos YAPL

Clase Main

- Todo programa en YAPL debe de contener una clase Main.
- La clase Main debe contener un método main sin parámetros formales.
- La clase Main no puede heredar de ninguna otra clase.
- Le ejecución de un programa en YAPL, comienza evaluando `(new Main).main()`

Consideraciones léxicas

Las unidades léxicas de YAPL se pueden agrupar en números enteros, identificadores de objetos, identificadores de tipos, cadenas, palabras reservadas, caracteres especiales y espacios en blanco [con caracteres especiales incluidos].

- **Enteros**. Cadenas no vacías de dígitos 0-9.
- **Identificadores**. Cadenas de uno o más caracteres [letras, dígitos, _], a excepción de palabras reservadas.
- Los identificadores de tipos comienzan con letra mayúscula.
- Los identificadores de objetos comienzan con letra minúscula.
- Existen dos identificadores especiales, **self** y **SELF_TYPE**, que no son tratados como palabras reservadas.
- Caracteres especiales son los operadores unarios, binarios, de asignación y de agrupación tradicionales.

- **Cadenas**. Secuencia de caracteres encerrados entre comillas dobles ("")
- Las cadenas pueden contener las siguientes secuencias de escape: \b, \t, \n, \f
- Una cadena no puede contener el marcador de EOF o el carácter null (\0)

- **Comentarios.** Los comentarios en YAPL pueden comenzar con la secuencia "–-" hasta el fin de línea. Otra forma de comentarios se puede dar encerrando caracteres entre la secuencia (* … *)

- **Palabras reservadas.** Las palabras reservadas de YAPL son: **class, else, false, fi, if, in, inherits, isvoid, loop, pool, then, while, new, not, true.**
- Las palabras reservadas son *case insensitive*, es decir tanto **class,** como **CLASS** son válidas. (a excepción de las constantes **true** y **false**, que deben ir siempre en minúsculas)

- **Espacios en blanco** es cualquier secuencia de los siguientes caracteres: espacio en blanco [ASCII 32], \n [ASCII 10], \f [ASCII 12], \r [ASCII 13], \t [ASCII 9], \v [ASCII 11]

Consideraciones sintácticas

Esta especificación no se encuentra en una forma pura BNF, y se agregan algunas notaciones similares a expresiones regulares por conveniencia y facilidad de escritura. Al trasladarse a una herramienta generadora de analizadores léxicos y sintácticos, esta debe de ser escrita en la sintaxis esperada por la herramienta.

Se presenta la gramática de YAPL:

$$
\begin{array}{rcl}
\textit{program} & ::= & [\![\textit{class};]\!]^{+} \\
\textit{class} & ::= & \textbf{class}\ \text{TYPE}\ [\textbf{inherits}\ \text{TYPE}]\ \{\ [\![\textit{feature};]\!]^{*}\} \\
\textit{feature} & ::= & \text{ID}(\ [\ \textit{formal}\ [\![, \textit{formal}]\!]^{*}\ ]\ ) : \text{TYPE}\ \{\ \textit{expr}\ \} \\
 & | & \text{ID} : \text{TYPE}\ [\ \texttt{<-}\ \textit{expr}\ ] \\
\textit{formal} & ::= & \text{ID} : \text{TYPE} \\
\textit{expr} & ::= & \text{ID}\ \texttt{<-}\ \textit{expr} \\
 & | & \textit{expr}[@\text{TYPE}].\text{ID}(\ [\ \textit{expr}\ [\![, \textit{expr}]\!]^{*}\ ]\ ) \\
 & | & \text{ID}(\ [\ \textit{expr}\ [\![, \textit{expr}]\!]^{*}\ ]\ ) \\
 & | & \textbf{if}\ \textit{expr}\ \textbf{then}\ \textit{expr}\ \textbf{else}\ \textit{expr}\ \textbf{fi} \\
 & | & \textbf{while}\ \textit{expr}\ \textbf{loop}\ \textit{expr}\ \textbf{pool} \\
 & | & \{\ [\![\textit{expr};]\!]^{+}\} \\
 & | & \textbf{let}\ \text{ID} : \text{TYPE}\ [\ \texttt{<-}\ \textit{expr}\ ]\ [\![, \text{ID} : \text{TYPE}\ [\ \texttt{<-}\ \textit{expr}\ ]]\!]^{*}\ \textbf{in}\ \textit{expr} \\
 & | & \textbf{new}\ \text{TYPE} \\
 & | & \textbf{isvoid}\ \textit{expr} \\
 & | & \textit{expr} + \textit{expr} \\
 & | & \textit{expr} - \textit{expr} \\
 & | & \textit{expr} * \textit{expr} \\
 & | & \textit{expr} / \textit{expr} \\
 & | & \tilde{}\,\textit{expr} \\
 & | & \textit{expr} < \textit{expr} \\
 & | & \textit{expr} <= \textit{expr} \\
 & | & \textit{expr} = \textit{expr} \\
 & | & \textbf{not}\ \textit{expr} \\
 & | & (\textit{expr}) \\
 & | & \text{ID} \\
 & | & \text{integer} \\
 & | & \text{string} \\
 & | & \textbf{true} \\
 & | & \textbf{false}
\end{array}
$$

Precedencia de operadores

La precedencia de los operadores dentro de YAPL se resume en la siguiente tabla [de mayor a menor precedencia]:

.
@
~
Isvoid
*** /**
+ -
<=, <, =
not
<-

Referencias

YAPL se encuentra basado en el lenguaje COOL y este documento se basa principalmente en el manual de COOL. Las personas involucradas en el desarrollo de COOL son Manuel Fahndrich, David Gay, Douglas Hauge, Megan Jacoby, Tendo Kayiira, Carleton Miyamoto, y Michael Stoddart.