Universidad de los Andes

ISIS-1106 Lenguajes y máquinas
TP 1
Fecha: September 23, 2021

Task 1. Para este taller vamos a construir un lexer y parser para reconocer programas de Mozart, un lenguaje de programación multi paradigma. En Mozart las variables están definidas como conjuntos de caracteres (únicamente letras del alfabeto) que comienzan con mayúscula. Adicionalmente Mozart utiliza símbolos, constantes que representan su propio valor, los cuales se definen como conjuntos de caracteres (únicamente letras del alfabeto) que comienzan con minúscula. Los procedimientos en Mozart definen un nombre (como una variable) y el conjunto de parámetros (variables separadas por espacios) delimitados por un `proc` y un `end`. Las instrucciones básicas que se pueden realizar en el lenguaje consisten en asignaciones (*e.g.,* `A = 2`, asigna dos a una variable `A`) o llamados a procedimientos, delimitados por corchetes (*e.g.,* {`Square 5 R`} llama el procedimiento `Square`) o cambios de linea. Además el lenguaje también tiene condicionales, listas (delimitadas por `[]` y con valor inicial para la lista vacia `nil`) y definiciones de variables (delimitadas por `local ... in`).

Usted debe proveer un programa de JavaCC, procesar los programas escritos en Mozart. Puede encontrar algunos ejemplos de programas aceptados por el lenguaje a continuación.

Programas ejemplo.

```
local Square X R in
   proc {Square X R} R = X*X end
   X=5
   {Square X R}
   {Browse R}
end
```

Snippet 1: Definición de un procedimiento

```
local SumSqr Lst R in
  proc {SumSqr Lst R}
    if Lst = nil then 0
    else
      Car|Cdr = List
      R = Car*Car + {SumSqr Cdr R}
    end
  end
  {Show {SumSqr [1 2 3 4 5]}}
end
```

Snippet 2: Programa recursivo

Gramática de Mozart.

$$
\begin{aligned}
S &\rightarrow S|SS \\
&\rightarrow Intconst \\
&\rightarrow Symbol \\
&\rightarrow String \\
&\rightarrow Var = S \\
&\rightarrow Var \\
&\rightarrow [] \\
&\rightarrow S + S \,|\, S - S \,|\, S * S \,|\, S \,/\, S \,|\, S > S \,|\, S = S \\
&\rightarrow local\ Params\ in\ S\ end \\
&\rightarrow if\ B\ then\ S\ else\ S\ end \\
&\rightarrow proc\ \{Var\ Params?\}\ S\ end \\
&\rightarrow \{Var\ Vals?\} \\
B &\rightarrow S|S > S \,|\, S = S|S >= S \\
Var &\rightarrow (A|...|Z)(a|..|Z) * \\
Params &\rightarrow Var|Var\ Parms \\
Vals &\rightarrow (Var|Intconst|Symbol|String)|(Var|Intconst|Symbol|String)Vals \\
Intconst &\rightarrow (0|...|9) \,|\, (0|...|9)Intconst \,|\, nil \\
String &\rightarrow "(a|...|Z) * " \\
Symbol &\rightarrow "(a|...|z) * "
\end{aligned}
$$

Note que el símbolo distinguido de la gramática es S

```
local Subtract Val Amount Main R Res in
  proc {Subtract Val Amount R}
    if Val >= Amount then
      R = Val - Amount
    else
      "Only positive numbers allowed"
    end
  end

  proc {Main}
    R = {Subtract 9 2}
    Res = {Subtract 5 28}
    {Show R}
    {Show Res}
  end
  {Main}
end
```

Snippet 3: Llamados de funciones program