

Nombres : Javier Fernando González Guzmán.

Laura Daniela Molina Villar.

1. Partiendo del ejemplo de calculadora de BISON implemente una calculadora que opere funciones trigonométricas (ej,  $\sin(x)$ ,  $\cos(x)$ , entre otras) y funciones matemáticas como (ej.  $\sqrt{x}$ ). Explique ¿Cada función debe tratarse como un nombre particular o los nombres de las funciones pueden tratarse como identificadores?, implemente la solución más adecuada.

Las funciones que incluimos en la calculadora son  $\sin$ ,  $\cos$ ,  $\tan$ ,  $\sinh$ ,  $\cosh$ ,  $\tanh$ ,  $\sqrt{x}$ ,  $\log$ , y la exponencial ( $e$ ).

Consideramos cada función como un nombre particular teniendo en cuenta las siguientes observaciones.

(i) Cada función tiene sus propiedades y restricciones que es mejor mirar de manera particular, una de las restricciones importantes es el dominio, en la implementación tenemos en cuenta fuertemente, por ejemplo para evaluar tangente en un número  $x$ , debe suceder  $\cos(x) \neq 0$ , ahora, teniendo en cuenta que la calculadora tiene un margen de error en cálculos, habrá error cuando  $|\cos(x)| < \delta = 0.0001$ .

(ii) Al tratar las funciones con nombres particulares queda claro cuáles están disponibles y como se utilizan.

(iii) Tratar las funciones como identificadores dificultaría observar las propiedades particulares mencionadas.

3. Especifique en BISON la siguiente porción de gramática que modela una estructura condicional:

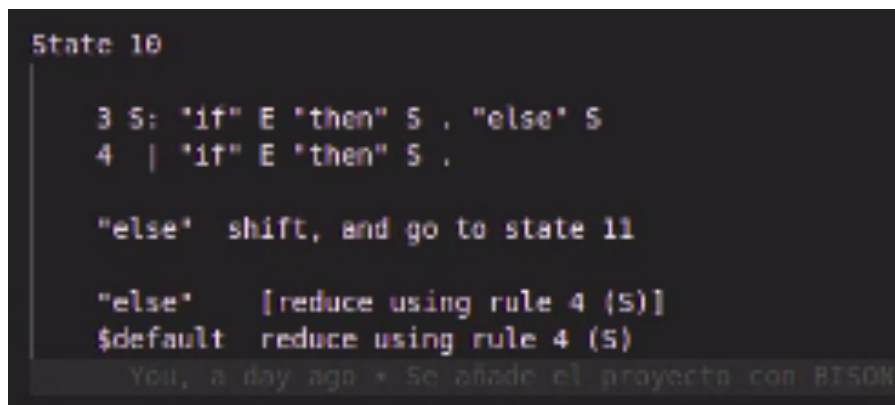
```
Z -> S
S -> if E then S else S
S -> if E then S
S -> inst
E -> id
```

- a. Genere el reporte de la máquina de estados.

Utilizamos el comando `bison -v expr.y`

- b. Verifique si existen conflictos shift-reduce. Si existe un conflicto explique cuáles pueden ser sus consecuencias.

Sí, existe un conflicto shift-reduce en el siguiente estado



```
State 10
  3 S: "if" E "then" S , "else" S
  4 | "if" E "then" S .

"else" shift, and go to state 11

"else" [reduce using rule 4 (S)]
$default reduce using rule 4 (S)
```

observamos que tenemos en la pila la cadena "if" E "then" S, y consideramos que procede a leer un else, la opción shift envía el else a la pila para intentar derivarlo más adelante, la opción reduce permite identificar que "if" E "then" S se deriva de S y realiza el cambio en la pila.

- c. Explique como se puede eliminar este conflicto en BISON.

Vamos a priorizar el uso del shift, este envía el else a la pila y lo explicará con la regla (3), pues es la única que tiene "else", entonces requerimos que se use. La opción de priorizar el reduce no funciona, pues nunca se eliminarían los "else" de la pila.

4. Considere la siguiente gramática para XML

```
L -> E L
|
E -> A L B
|   ident
A -> < ident >
B -> </ ident >
```

- Escriba los archivos lex y bison para reconocer esta gramática.
- Agregue reglas para verificar que los tags de apertura y cierre se refieren a los mismos identificadores.

Proponemos la siguiente gramática que no tiene conflictos shift-reduce

```
XML -> start EXP stop
EXP -> XML | str | EXP XML
```

start y stop son regex de la forma  $\langle [a-zA-z]^+ \rangle$  y  $\langle / [a-zA-z]^+ \rangle$  respectivamente, y str  $[a-zA-z ]^+$ , es decir cadenas con letras y espacios, es decir, oraciones.

La verificación de que los tags start y stop coinciden no se hizo modificando la gramática sino directamente con funcionalidades de Bison.

Nota: El código lee programas unilínea, es decir programas como  $\langle a \rangle \text{hola} \langle b \rangle \text{mundo} \langle /b \rangle \langle /a \rangle$ , y no como:

```
<a> hola
      <b>
        mundo
      </b>
    </a>
```