Memoria de la Práctica de Procesadores de Lenguajes

Diego José Abengózar Vilar, Alejandro García Castellanos,
Ignacio Javier Encinas Ramos

Grupo 82

January 18, 2020

Índice

1 Diseno del Analizador Semántico

1.1 Implementación del EdT

```
0. P' → MM1 P {liberaTS(TSG)}
1. P → D P1 {Aux[ntope].tipoRet = Aux[tope].tipoRet}
2. P → F P1 {Aux[ntope].tipoRet = Aux[tope].tipoRet}
3. P → S P1 {Aux[ntope].tipoRet = if(Aux[tope-1].tipoRet = tipo_vacio)then
                                        Aux[tope].tipoRet
                                    else error(1)}
4. D → var MM2 T id MM8 ; {InsertarTipoTS(Aux[tope-2].posi, Aux[tope-3].tipo)
                           if(TS_actual = TSG) then
                               InsertarDespl(Aux[tope-1].posi, desplG)
                               desplG = desplG + Aux[tope-3].tamano
                           else
                               InsertarDespl(Aux[tope-2].posi, desplL)
                               desplL = desplL + Aux[tope-3]}
5. T → int {Aux[ntope].tipo=entero; Aux[ntope].tamano = 1}
6. T → string {Aux[ntope].tipo=cadena; Aux[ntope].tamano = 64}
7. T → boolean {Aux[ntope].tipo=logico; Aux[ntope].tamano = 1}
8. F → function MM3 T1 id MM4 ( A ) MM5 { C } {if (Aux[tope-1].tipoRet != Aux[tope-7]
    then error(1); TS_actual = TSG; LiberarTS(TSL) }
9. T1 →λ {Aux[ntope].tipo = tipo_vacio}
10. T1 →T{Aux[ntope].tipo = Aux[tope].tipo }
11. A → T id {InsertarTipoTS(Aux[tope-1].posi, Aux[tope-2].tipo) ;
     InsertarDesplTS(Aux[tope-1].posi, desplL);
     desplL = desplL + Aux[tope-2].tamano} K
     {Aux[ntope].tipo = if(Aux[tope].tipo == tipo_vacio)
                            then Aux[tope-2].tipo}
                        else
                            Aux[tope].tipo.push(Aux[tope-2].tipo)
     }
12. A → λ {Aux[ntope].tipo = tipo_vacio}
13. K → λ {Aux[ntope].tipo = tipo_vacio}
14. K → , T id {InsertarTipoTS(Aux[tope-1].posi, Aux[tope-2].tipo);
     InsertarDesplTS(Aux[tope-1].posi, desplL);
     desplL = desplL + Aux[tope-2].tamano} K1
     {K.tipo = if(Aux[tope].tipo == tipo_vacio) then
                    NuevaPila(Aux[tope-2].tipo)
                else
                    Aux[tope].tipo.push(Aux[tope-2].tipo)
      }
15. C → {S.func = C.func} S {C1.func = C.func}C1
        {Aux[ntope].tipoRet=
        if(Aux[tope-1].tipoRet == Aux[tope].tipoRet) then
            Aux[tope-1].tipoRet
        else if(Aux[tope-1].tipoRet == tipo_vacio) then
            Aux[tope-1].tipoRet
        else if(Aux[tope].tipoRet == tipo_vacio) then
            Aux[tope-1].tipoRet}
        else
            error(2)
     }
17. C → λ{Aux[ntope].tipoRet = tipo_vacio}
18. S→ id L E ; {Aux[ntope].tipo =
     if(BuscaTipoTS(Aux[tope-3].posi)==(Aux[tope-1].tipo)
     AND (Aux[tope-1].tipo != tipo_error))then
```

```
        tipo_ok
     else
        error(3)}
19. S→ id (M) E ; {Aux[ntope].tipo =
     if(BuscaTipoTS(Aux[tope-4].posi)== ParFunc(Aux[tope-2].tipo , t)
        then tipo_ok
     else
        error(4)}
20. S → print (E) ; {Aux[ntope].tipo =
     if(Aux[tope-2].tipo == entero || Aux[tope-2].tipo == cadena)
        then tipo_ok
     else
        error(5)
        }
21. S → input(id); {Aux[ntope].tipo =
     if(BuscaTipoTS(Aux[tope-2].posi == entero
        || Aux[tope-2].tipo == cadena)) then tipo_ok
     else
        error(6)
     }
22. S → if(E) {S1.func = S.func} S1 {Aux[ntope].tipo =
     if(Aux[tope-2].tipo == logico) then Aux[tope].tipo
     else
        error(7)
     }
23. S → return X;{Aux[ntope].tipo =
     if(Aux[ntope].func) then
        if(Aux[tope-1].tipo != tipo.error) then tipo_ok
        else
            error(8)
     else
        error(9)
     }
24. L → |= {}
25. L→ = {}
26. M→ EQ {Aux[ntope].tipo =
     if(Aux[tope-1].tipo != tipo_error
     AND Aux[tope].tipo != tipo_error)
        then if(Aux[tope].tipo == tipo_vacio)
                then Aux[tope-1].tipo
             else
                Aux[tope].tipo.push(Aux[tope-1].tipo)
     else
        error(10)
    }
27. M → λ {Aux[ntope].tipo = tipo_vacio}
28. Q → λ {Aux[ntope].tipo = tipo_vacio}
29. Q → ,EQ-1{if(Aux[tope-1].tipo != tipo_error
                  AND Aux[tope].tipo != tipo_error)
                    then if(Aux[tope].tipo == tipo_vacio)
                            then NuevaPila(Aux[tope-1].tipo)
                         else Aux[tope].tipo.push(Aux[tope-1].tipo)
                  else
                    error(11)
                  }
30. S-1 →{ {S2.func = S1.func} S2}
     {G.func=S1.func} G {Aux[ntope].tipo =
                            if(Aux[tope-2].tipo != tipo_error)
```

```
                                    if(Aux[tope].tipo != tipo_error)
                                        then Aux[tope-2].tipo
                                    else error(13)
                                else error(12)
    }
31. S1 → { {S2.func=s1.func} S {Aux[ntope].tipo=Aux[tope].tipo}
32. G → else{ {S2.func=G.func} S2}{Aux[ntope].tipo=Aux[tope-1].tipo}
33. G → λ {Aux[ntope].tipo = tipo_vacio}
34. X → E {Aux[ntope].tipo = Aux[tope].tipo}
35. X → λ {Aux[ntope].tipo = tipo_vacio}
36. E → E1 < U {Aux[ntope].tipo =
                    if(Aux[tope-2].tipo = Aux[tope].tipo = entero)
                       then logico
                    else
                       error(14)
                   }
37. E → U {Aux[ntope].tipo = Aux[tope].tipo}
38. U → u1 + R {Aux[ntope].tipo =
                   if(Aux[tope-2].tipo = Aux[tope].tipo = entero)
                       then entero
                   else
                       error(15)
                   }
39. U→ R {Aux[ntope].tipo = Aux[tope].tipo}
40. R -> !V {Aux[ntope].tipo =
             if(Aux[tope].tipo = logico) then logico
             else error(16)
             }
41. R → V {Aux[ntope].tipo = Aux[tope].tipo}
42. V → (E) {Aux[ntope].tipo = Aux[tope-1].tipo}
43. V → id {Aux[ntope].tipo = BuscaTipoTS(Aux[tope].posi)}
44. V → id(M) {Aux[ntope].tipo =
                  if(BuscaTipoTS(Aux[tope].posi) == ParFunc(Aux[tope-1].tipo, t))
                     then t
                  else
                     error(17)
                 }
45. V → ent{Aux[ntope].tipo = entero}
46. V → cadena{Aux[ntope].tipo = cadena}
47. S2 →{S.func = S2.func} S {S'2.func = S2.func} S'2 {
             Aux[ntope] = if(Aux[tope-1].tipo != tipo_error) then Aux[tope].tipo
                  else
                     error(18)
             }
48. S2 →{S.func = S2.func} S {Aux[ntope].tipo = Aux[tope].tipo}
49. P → λ {}
```