**Analizador Sintáctico Descendiente**

**Jaime Sáez de Buruaga Brouns**

**Julia Miguélez Fernández-Villacañas**

1. **Especificación mediante gramática incontextual**

**TABLA DE OPERADORES:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Tipo** | **Prioridad** | **Asociatividad** |
| ***+*** | *Binario infijo* | *0* | *Asociativo a izquierdas* |
| ***-*** | *Binario infijo* | *0* | *Asociativo a izquierdas* |
| ***and*** | *Binario infijo* | *1* | *Asociativo a derechas* |
| ***or*** | *Binario infijo* | *1* | *No asocia* |
| ***>*** | *Binario infijo* | *2* | *No asocia* |
| ***<*** | *Binario infijo* | *2* | *No asocia* |
| ***>=*** | *Binario infijo* | *2* | *No asocia* |
| ***<=*** | *Binario infijo* | *2* | *No asocia* |
| ***!=*** | *Binario infijo* | *2* | *No asocia* |
| ***==*** | *Binario infijo* | *2* | *No asocia* |
| ***\**** | *Binario infijo* | *3* | *Asociativo a izquierdas* |
| ***/*** | *Binario infijo* | *3* | *Asociativo a izquierdas* |
| ***-*** | *Uniario prefijo* | *4* | *Asocia* |
| ***not*** | *Uniario prefijo* | *4* | *No asocia* |

**NOTA:** ¿ID, REAL, ENT quedan dentro o fuera de la definición de la gramática?

*ID = LET [ LET | DIG | \_ ]\**

*LET = [ a | A | b | B | … | z | Z ]*

*DIG = [ 0 | 1 | … | 9 ]*

*REAL = [+ | - | ε ] [ ENTDEC | ENTEXP | ENTDECEXP ]*

*ENT = [+ | - | ε ] POS (DIG)\**

*POS = [ 1 | 2 | … | 9 ]*

*DEC = .ENT*

*EXP = [e | E] ENT*

*G = (Vt, Vn, P, S)*

*Vt = [a..z, 0..9, \_, int, real, bool, true, false, E, +, -, ., and, or, >, <, >=, <=, ==, !=, \*, /, not, ;]*

*Vn = [S, Sd, Si, D, T, I, E0, E1, E2, E3, E4, E5, OP]*

*S = S*

*P = {*

*S → Sd && Si,*

*Sd → D (; D)\*,*

*D → T ID,*

*T → int | real | bool,*

*Si → I (; I)\*,*

*I → ID = E0,*

*E0 → E0 + E1 | E0 – E1 | E1*

*E1 → E2 and E1 | E2 or E2 | E2*

*E2 → E3 OP E3 | E3*

*E3 → E3 \* E4 | E3 / E4 | E4*

*E4 → -E4 | not E5 | E5*

*E5 → (E0) | E0 | ID |* *REAL | ENT | true | false*

*OP → < | > | <= | >= | == | !=*

*}*

1. **Transformaciones necesarias para LL(1) equivalente.**
   1. **Eliminación de recursión por la izquierda**

*G = (Vt, Vn, P, S)*

*Vt = [a..z, 0..9, \_, int, real, bool, true, false, E, +, -, ., and, or, >, <, >=, <=, ==, !=, \*, /, not, ;]*

*Vn = [S, Sd, D, T, Si, I, E0, E0’, E1, E2, E3, E3’, E4, E5, OP]*

*S = S*

*P = {*

*S → Sd && Si,*

*Sd → D | D; Sd,*

*D → T ID,*

*T → int | real | bool,*

*Si → I | I ; Si*

*I → ID = E0,*

*E0 → E1 E0’*

*E0’ → + E1 E0’ | - E1 E0’ | ε*

*E1 → E2 and E1 | E2 or E2 | E2*

*E2 → E3 OP E3 | E3*

*E3 → E4 E3’*

*E3’ → \* E4 E3 | / E4 E3 | ε*

*E4 → -E4 | not E5 | E5*

*E5 → (E0) | E0 | ID | REAL | ENT | true | false*

*OP → < | > | <= | >= | == | !=*

*}*

* 1. **Eliminación de factores a la izquierda**

*G = (Vt, Vn, P, S)*

*Vt = [a..z, 0..9, \_, int, real, bool, true, false, E, +, -, ., and, or, >, <, >=, <=, ==, !=, \*, /, not, ;]*

*Vn = [S, Sd, D, DEC, T, Si, I, INS, E0, E0’, E1, EE1, E2, EE2, E3, E3’, E4, E5, OP]*

*S = S*

*P = {*

*S → Sd && Si,*

*Sd → D | DEC*

*D → T ID,*

*DEC →* *ε | ; D DEC*

*T → int | real | bool,*

*Si → I | INS*

*I → ID = E0,*

*INS → ε | ; I SINS*

*E0 → E1 E0’*

*E0’ → + E1 E0’ | - E1 E0’ | ε*

*E1 → E2 EE1*

*EE1 → and E1 | or E2 | ε*

*E2 → E3 EE2*

*EE2 → OP E3 | E3*

*E3 → E4 E3’*

*E3’ → \* E4 E3 | / E4 E3 | ε*

*E4 → -E4 | not E5 | E5*

*E5 → (E0) | E0 | ID | REAL | ENT | true | false*

*OP → < | > | <= | >= | == | !=*

*}*

1. **No terminales: *primeros* y *salientes***

|  |  |  |
| --- | --- | --- |
| **Productor** | **Primeros** | **Siguientes** |
| *S* | *int, real, bool* | *Ø* |
| *Sd* | *int, real, bool* | *&&* |
| *D* | *int, real, bool* | *;, &&* |
| *DEC* | *;* | *&&* |
| *T* | *int, real, bool* | *ID* |
| *Si* | *ID* | *Ø* |
| *I* | *ID* | *;* |
| *INS* | *;* | *Ø* |
| *E0* | *-, not, (, ID, REAL, ENT, true, false* | *), ;* |
| *E0’* | *+, -* | *), ;* |
| *E1* | *-, not, (, ID, REAL, ENT, true, false* | *), +, -, ;* |
| *EE1* | *and, or* | *), +, -, ;* |
| *E2* | *-, not, (, ID, REAL, ENT, true, false* | *and, or, ), +, -, ;* |
| *EE2* | *<, >, <=, >=, !=, ==* | *and, or, ), +, -, ;* |
| *E3* | *-, not, (, ID, REAL, ENT, true, false* | *), >, <, >=, <=, !=, ==, and, or, +, -, ;* |
| *E3’* | *\*, /* | *), >, <, >=, <=, !=, ==, and, or, +, -, ;* |
| *E4* | *-, not, (, ID, REAL, ENT, true, false* | *), \*, /, <, >, <=, >=, !=, ==, and, or, +, -, ;* |
| *E5* | *(, ID, REAL, ENT, true, false* | *), \*, /, <, >, <=, >=, !=, ==, and, or, +, -, ;* |
| *OP* | *<, >, <=, >=, ==, !=* | *-, not, (, ID, REAL, ENT, true, false* |

1. **Reglas: *directores***

|  |  |
| --- | --- |
| **Productor** | **Directores** |
| *S → Sd && Si* | *int, real, bool* |
| *Sd → D DEC* | *int, real, bool* |
| *D → T ID* | *int, real, bool* |
| *DEC → ε* | *&&* |
| *DEC → ; D DEC* | *int, real, bool* |
| *T → int* | *ID* |
| *T → real* | *ID* |
| *T → bool* | *ID* |
| *I → ID = E0* | *;* |
| *INS → ε* | *-* |
| *INS → ; I INS* | *;* |
| *E0 → E1 E0’* | *-, not, (, ID, INT, REAL, true, false* |
| *E0’ → + E1 E0’* | *+* |
| *E0’ → - E1 E0’* | *-* |
| *E0’ → ε* | *), -, ;* |
| *E1 → E2 EE1* | *-, not, (, ID, INT, REAL, true, false* |
| *EE1 → and E1* | *and* |
| *EE1 → or E2* | *or* |
| *EE1 → ε* | *+, -, ), -, ;* |
| *E2 → E3 EE2* | *-, not, (, ID, INT, REAL, true, false* |
| *EE2 → OP E3* | *<, >, <=, >=, ==, !=* |
| *EE2 → ε* | *and, or, +, -, ), -, ;* |
| *E3 → E4 E3’* | *-, not, (, INT, REAL, true, false* |
| *E3’ → \* E4 E3’* | *\** |
| *E3’ → / E4 E3’* | */* |
| *E3’ → ε* | *<, >, <=, >=, ==, !=, and, or, +, -, ), -, ;, not, (, ID, INT, REAL, true, false* |
| *E4 → - E4* | *-* |
| *E4 → not E5* | *not* |
| *E4 → E5* | *(, ID, INT, REAL, true, false* |
| *E5 → (E0)* | *(* |
| *E5 → ID* | *ID* |
| *E5 → REAL* | *REAL* |
| *E5 → ENT* | *ENT* |
| *E5 → true* | *True* |
| *E5 → false* | *False* |
| *OP → <* | *<* |
| *OP → >* | *>* |
| *OP → <=* | *<=* |
| *OP → >=* | *>=* |
| *OP → ==* | *==* |
| *OP → !=* | *!=* |