Gramatica LL1

* **Program → StatementList**
* **Modifier → "FLOAT"**

**| "BOOLEAN"**

**| "INTEGER"**

**| "PARAM"**

**| "VECTOR" "[" "NUM" "]"**

**| "TYPE" "(" "ID" ")"**

* **IdList → "ID" IdListCompound**
* **IdListCompound → "," IdListCompound | ε**
* **ModifierList → Modifier ModifierListMultiple**
* **ModifierListMultiple → Modifier ModifierListMultiple | ε**
* **NameDecl → ModifierList IdList**
* **Fragment → "FRAGMENT" FragmentBody**
* **FragmentBody → ModifierList FragmentId**

**| "ID" FragmentDestiny**

**| "IF" "(" Expression")" StatementList StatementElse StatementListOptional "ENDFRAGMENT"**

**|Fragment StatementListOptional "ENDFRAGMENT"**

**| "READ" "(" ExpressionList ")" ";" StatementListOptional "ENDFRAGMENT"**

**| "WHILE" "(" Expression ")" StatementList StatementListOptional "ENDFRAGMENT"**

**| "SELECT" "(" Expression ")" "FRAGMENT" CaseBlock "ENDFRAFMENT"**

**| StatementListOptional "ENDFRAGMENT"**

* **EndFragmentLiteral → StatementList "ENDFRAGMENT"**

**| "LITERAL" "ENDFRAGMENT"**

* **EndFragmentNum → StatementList "ENDFRAGMENT"**

**| "NUM" "ENDFRAGMENT"**

* **FragmentId → "ID" FragmentIdList**
* **FragmentDestiny → StatementDestiny StatementListOptional "ENDFRAGMENT"**

**| ";" EndFragmentLiteral**

* **FragmentIdList → ";" EndFragmentNum**

**| "." Name NameCompound ";" EndFragmentNum**

* **StatementList → Statement StatementListOptional**
* **StatementListOptional → StatementList | ε**
* **Statement → "ID" StatementDestiny**

**| "IF" "(" Expression")" StatementList StatementElse**

**| NameDecl ";"**

**| Fragment**

**| "READ" "(" ExpressionList ")" ";"**

**| "WHILE" "(" Expression ")" StatementList**

**| "SELECT" "(" Expression ")" "FRAGMENT" CaseBlock "ENDFRAGMENT"**

**| "BREAK" ";"**

**| "WRITE" "(" ExpressionList ")" ";"**

* **StatementElse → "ELSE" StatementList | ε**
* **StatementDestiny → DestinyName "=" Expression ";"**

**| "(" ExpressionList ")" ";"**

* **Destiny → "ID" DestinyName**
* **DestinyName → "[" Expression "]" DestinyNameCompound**

**| NameCompound**

**| ε**

* **DestinyNameCompound → NameCompound**

**| ε**

* **CaseBlock → "CASE" Expression ":" StatementList CaseBlock**

**| "DEFAULT" ":" StatementList CaseBlock**

* **ExpressionList → Expression ExpressionListCompound**
* **ExpressionListCompound → "," Expression ExpressionListCompound | ε**
* **Expression → Primary**

**| UnaryOp Expression**

**| ExpressionOr**

* **ExpressionOr → ExpressionAnd Or**
* **Or → "||" ExpressionAnd Or | ε**
* **ExpressionAnd → ExpressionLTEqual And**
* **And → "&&" ExpressionLTEqual And | ε**
* **ExpressionLTEqual → ExpressionLT LTEqual**
* **LTEqual → "<=" ExpressionLT LTEqual | ε**
* **ExpressionLT → ExpressionGTEqual LT**
* **LT → "<" ExpressionGTEqual LT | ε**
* **ExpressionGTEqual → ExpressionGT GTEqual**
* **GTEqual → ">=" ExpressionGT GTEqual | ε**
* **ExpressionGT → ExpressionDiferent GT**
* **GT → ">" ExpressionDiferent GT | ε**
* **ExpressionDiferent → ExpressionEqual Diferent**
* **Diferent → "!=" ExpressionEqual Diferent | ε**
* **ExpressionEqual → ExpressionSub Equal**
* **Equal → "==" ExpressionSub Equal | ε**
* **ExpressionSub → ExpressionSum Sub**
* **Sub → "-" ExpressionSum Sub | ε**
* **ExpressionSum → ExpressionDiv Sum**
* **Sum → "+" ExpressionDiv Sum | ε**
* **ExpressionDiv → ExpressionMult Div**
* **Div → "/" ExpressionMult Div | ε**
* **ExpressionMult → Primary Mult**
* **Mult → "\*" Primary Mult | ε**
* **Primary → "ID" PrimaryExpression**

**| "(" Expression ")"**

**| "TRUE"**

**| "FALSE"**

**| "LITERAL"**

* **PrimaryExpression → "(" ExpressionList ")"**

**| NameExpression**

* **Name → "ID" NameExpression**
* **NameExpression → NameCompound**

**| "[" Expression "]" NameCompound**

* **NameCompound → "." Name NameCompound**

**| ε**

* **UnaryOp → "-" | "!" | "+"**

Sem Ambiguidade

Foram criados novos estados substituindo as chamadas do BinOp no Expression e separado os Expressions por ordem decrescente de prioridade dos operadores.

**Ordem dos Operadores:**

| **||** | **&&** |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **<=** | **<** | **>=** | **>** | **!=** | **==** |
| **-** | **+** |  |  |  |  |
| **/** | **\*** |  |  |  |  |

**Novas chamadas:**

* **Expression → Primary | UnaryOp Expression | ExpressionOr**
* **ExpressionOr → ExpressionOr “||” ExpressionAnd**

**| ExpressionAnd**

* **ExpressionAnd → ExpressionAnd “&&” ExpressionLTEqual**

**| ExpressionLTEqual**

* **ExpressionLTEqual → ExpressionLTEqual “<=” ExpressionLT**

**| ExpressionLT**

* **ExpressionLT → ExpressionLT “<” ExpressionGTEqual**

**| ExpressionGTEqual**

* **ExpressionGTEqual → ExpressionGTEqual “>=” ExpressionGT**

**| ExpressionGT**

* **ExpressionGT → ExpressionGT “>” ExpressionDiferent**

**| ExpressionDiferent**

* **ExpressionDiferent → ExpressionDiferent “!=” ExpressionEqual**

**| ExpressionEqual**

* **ExpressionEqual → ExpressionEqual “==” ExpressionSub**

**| ExpressionSub**

* **ExpressionSub → ExpressionSub “-” ExpressionSum**

**| ExpressionSum**

* **ExpressionSum → ExpressionSum “+” ExpressionDiv**

**| ExpressionDim**

* **ExpressionDiv → ExpressionDiv “/” ExpressionMult**

**| ExpressionMult**

* **ExpressionMult → ExpressionMult “\*” Primary**

**| Primary**

Sem Recursão

Primeiro foi achar os pontos que tem recursão a esquerda que são:

* **ModifierList → Modifier**

**|ModifierList Modifier**

* **ExpressionList → Expression**

**|ExpressionList "," Expression**

* **Name → "ID"**

**| "ID" "[" Expression "]"**

**| Name "." Name**

* **ExpressionOr → ExpressionOr “||” ExpressionAnd**

**| ExpressionAnd**

* **ExpressionAnd → ExpressionAnd “&&” ExpressionLTEqual**

**| ExpressionLTEqual**

* **ExpressionLTEqual → ExpressionLTEqual “<=” ExpressionLT**

**| ExpressionLT**

* **ExpressionLT → ExpressionLT “<” ExpressionGTEqual**

**| ExpressionGTEqual**

* **ExpressionGTEqual → ExpressionGTEqual “>=” ExpressionGT**

**| ExpressionGT**

* **ExpressionGT → ExpressionGT “>” ExpressionDiferent**

**| ExpressionDiferent**

* **ExpressionDiferent → ExpressionDiferent “!=” ExpressionEqual**

**| ExpressionEqual**

* **ExpressionEqual → ExpressionEqual “==” ExpressionSub**

**| ExpressionSub**

* **ExpressionSub → ExpressionSub “-” ExpressionSum**

**| ExpressionSum**

* **ExpressionSum → ExpressionSum “+” ExpressionDiv**

**| ExpressionDiv**

* **ExpressionDiv → ExpressionDiv “/” ExpressionMult**

**| ExpressionMult**

* **ExpressionMult → ExpressionMult “\*” Primary**
  + **| Primary**

Seguindo a fórmula foi retirado a recursão:

**Fórmula:**

**A → Aα | β**

**↓**

**A → βA’**

**A’ → αA’ | ε**

**Novos Estados:**

* **ModifierList → Modifier ModifierList'**
* **ModifierList' → Modifier ModifierList' | ε**

* **ExpressionList → Expression ExpressionList'**
* **ExpressionList' → "," Expression ExpressionList' | ε**
* **Name → "ID" Name' | "ID" "[" Expression "]" Name'**
* **Name' → "." Name Name' | ε**
* **ExpressionOR → ExpressionAnd ExpressionOR'**
* **ExpressionOR' → "||" ExpressionAnd ExpressionOR' | ε**
* **ExpressionAnd → ExpressionLTEqual ExpressionAnd'**
* **ExpressionAnd' → "&&" ExpressionLTEqual ExpressionAnd' | ε**

* **ExpressionLTEqual → ExpressionLT ExpressionLTEqual'**
* **ExpressionLTEqual' → "<=" ExpressionLT ExpressionLTEqual' | ε**
* **ExpressionLT → ExpressionGTEqual ExpressionLT'**
* **ExpressionLT' → "<" ExpressionGTEqual ExpressionLT' | ε**
* **ExpressionGTEqual → ExpressionGT ExpressionGTEqual'**
* **ExpressionGTEqual' → ">=" ExpressionGT ExpressionGTEqual' | ε**
* **ExpressionGT → ExpressionDiferent ExpressionGT'**
* **ExpressionGT' → ">" ExpressionDiferent ExpressionGT' | ε**

* **ExpressionDiferent → ExpressionEqual ExpressionDiferent'**
* **ExpressionDiferent' → "!=" ExpressionEqual ExpressionDiferent' | ε**
* **ExpressionEqual → ExpressionSub ExpressionEqual'**
* **ExpressionEqual' → "==" ExpressionSub ExpressionEqual' | ε**
* **ExpressionSub → ExpressionSum ExpressionSub'**
* **ExpressionSub' → "-" ExpressionSum ExpressionSub' | ε**
* **ExpressionSum → ExpressionDiv ExpressionSum'**
* **ExpressionSum' → "+" ExpressionDiv ExpressionSum' | ε**
* **ExpressionDiv → ExpressionMult ExpressionDiv'**
* **ExpressionDiv' → "/" ExpressionMult ExpressionDiv' | ε**
* **ExpressionMult → Primary ExpressionMult'**
* **ExpressionMult' → "\*" Primary ExpressionMult' | ε**

E aqui está os novos estados renomeados:

* **ModifierList → Modifier ModifierListRepl**
* **ModifierListRepl → Modifier ModifierListRepl | ε**
* **ExpressionList → ExpressionOr ExpressionListRepl**
* **ExpressionListRepl → "," ExpressionOr ExpressionListRepl | ε**
* **Name → "ID" NameRepl | "ID" "[" Expression "]" NameRepl**
* **NameRepl → "." Name NameRepl | ε**

* **ExpressionOr → ExpressionAnd Or**
* **Or → "||" ExpressionAnd Or | ε**
* **ExpressionAnd → ExpressionLTEqual And**
* **And → "&&" ExpressionLTEqual And | ε**
* **ExpressionLTEqual → ExpressionLT LTEqual**
* **LTEqual → "<=" ExpressionLT LTEqual | ε**
* **ExpressionLT → ExpressionGTEqual LT**
* **LT → "<" ExpressionGTEqual LT | ε**
* **ExpressionGTEqual → ExpressionGT GTEqual**
* **GTEqual → ">=" ExpressionGT GTEqual | ε**
* **ExpressionGT → ExpressionDiferent GT**
* **GT → ">" ExpressionDiferent GT | ε**
* **ExpressionDiferent → ExpressionEqual Diferent**
* **Diferent → "!=" ExpressionEqual Diferent | ε**
* **ExpressionEqual → ExpressionSub Equal**
* **Equal → "==" ExpressionSub Equal | ε**

* **ExpressionSub → ExpressionSum Sub**
* **Sub → "-" ExpressionSum Sub | ε**
* **ExpressionSum → ExpressionDiv Sum**
* **Sum → "+" ExpressionDiv Sum | ε**
* **ExpressionDiv → ExpressionMult Div**
* **Div → "/" ExpressionMult Div | ε**
* **ExpressionMult → Primary Mult**
* **Mult → "\*" Primary Mult | ε**

Sem Não Determinismo

Primeiro foi localizar os locais com não determinismo direto que são:

* **Id\_List → "ID"**

**| ”ID" "," Id\_List**

* **Fragment → "FRAGMENT" "ID" ";" StatmentList "ENDFRAGMENT"**

**| "FRAGMENT" StatmentList "ENDFRAGMENT"**

**| "FRAGMENT" ModifierList "ID" ";" StatmentList "ENDFRAGMENT"**

**| "FRAGMENT" "ID" ";" "LITERAL" "ENDFRAGMENT"**

**| "FRAGMENT" ModifierList "ID" ";" "NUM" "ENDFRAGMENT"**

* **StatementList → Statement**

**|StatementList Statement**

* **Statement → NameDecl ";"**

**| Destiny "=" Expression ";"**

**| Calfunc ";"**

**| Fragment**

**| "IF" "(" Expression")" StatementList "ELSE" StatementList**

**| "IF" "(" Expression")" StatementList**

**| "READ" "(" ExpressionList ")" ";"**

**| "WHILE" "(" Expression ")" StatementList**

**| "SELECT" "(" Expression ")"**

**| "FRAGMENT" CaseBlock "ENDFRAFMENT"**

**| "BREAK" ";"**

**| "WRITE" "(" ExpressionList ")" ";"**

* **Destiny → "ID"**

**| "ID" "[" Expression "]" Name**

**| Name**

* **Name → "ID"**

**| "ID" "[" Expression "]"**

**| Name "." Name**

Aqui está os estados após aplicar a fórmula:

A Fórmula:

**A → αβ | αγ**

**↓**

**A → α A’**

**A’ → β | γ**

Novos Estados:

* **IdList → "ID" IdList'**
* **IdList' → "," IdList' | ε**
* **Fragment → FRAGMENT Fragment1**
* **Fragment’ → ModifierList Fragment2**

**| "ID" ";" Fragment3**

**| StatmentList "ENDFRAGMENT"**

* **Fragment’’ → ID ";" Fragment4**
* **Fragment’’’ → StatmentList "ENDFRAGMENT"**

**| "LITERAL" "ENDFRAGMENT"**

* **Fragment’’’’ → NUM "ENDFRAGMENT"**

**| StatmentList "ENDFRAGMENT"**

* **StatementList → Statement StatementList'**
* **StatementList' → StatementList | ε**
* **statement → "IF" "(" Expression")" StatementList Statement'**

**| NameDecl ";"**

**| Destiny "=" Expression ";"**

**| Calfunc ";"**

**| Fragment**

**| "READ" "(" ExpressionList ")" ";"**

**| "WHILE" "(" Expression ")" StatementList**

**| "SELECT" "(" Expression ")"**

**| "FRAGMENT" CaseBlock "ENDFRAFMENT"**

**| "BREAK" ";"**

**| "WRITE" "(" ExpressionList ")" ";"**

* **statement' → "ELSE" StatementList | ε**
* **Destiny → "ID" Destiny' | Name**
* **Destiny' → "[" Expression "]" | ε**
* **Name → "ID" Name'**
* **Name' → NameRepl | "[" Expression "]" Name Repl**
* **NameRepl → "." Name NameRepl | ε**

Depois foi localizar os não determinismo indireto que são:

* **Fragment’ → ModifierList Fragment2**

**| "ID" ";" Fragment3**

**| StatmentList "ENDFRAGMENT"**

O Não Determinismo Indireto e entre o modifierList e StatementList -> Statement -> NameDecl “;” - > ModifierList IdList. Sendo o não determinismo no ModifierList

* **Destiny → "ID" Destiny' | Name**

O Não Determinismo Indireto e entre o “ID” e Name -> "ID" Name'. Sendo o não determinismo no “ID”

* **Primary → Name | Calfunc | "(" Expression ")"**

**| "ID" "(" ExpressionList ")" | "TRUE" | "FALSE" | "LITERAL"**

O Não Determinismo Indireto e entre o name -> "ID" Name' , Callfunc -> “ID” (ExpressionList) e “ID” (ExpressionList). Sendo o não determinismo no “ID”

* **statement → "IF" "(" Expression")" StatementList Statement' | NameDecl ";" | Destiny "=" Expression ";" | Calfunc ";" | Fragment | "READ" "(" ExpressionList ")" ";" | "WHILE" "(" Expression ")" StatementList | "SELECT" "(" Expression ")" "FRAGMENT" CaseBlock "ENDFRAGMENT" | "BREAK" ";" | "WRITE" "(" ExpressionList ")" ";"**

O Não Determinismo Indireto e entre o Destiny -> "ID" Destiny' -> e Calfunc. Sendo o não determinismo no “ID”

Após aplicarmos a fórmula aqui estão os novos estados: