**Rapport Mini-projet de Compilation**

1. Nettoyage de grammaire
2. **Eliminer la récursivité à gauche**

Les règles de production contenant une récursivité à gauche :

* **Dcl → Dcl var Liste\_id : Type ; | ε**

Se transforme en :

* Dcl → Dcl’ : on peut l’éliminer
* Dcl’→ var Liste\_id : Type ; Dcl’ | ε
* **Liste\_id → id | Liste\_id , id**

Se transforme en :

* Liste\_id → id Liste\_id’
* Liste\_id’ → , id Liste\_id’ | ε
* **Liste\_inst → I | Liste\_inst ; I**

Se transfore en :

* Liste\_inst → I Liste\_inst’
* Liste\_inst’ → ; I Liste\_inst’ | ε
* **Exp\_simple → Terme | Exp\_simple opadd Terme**

Se transforme en :

* Exp\_simple → Terme Exp\_simple’
* Exp\_simple’ → opadd Terme Exp\_simple’ | ε
* **Terme → Facteur | Terme opmul Facteur**

Se transforme en :

* Terme → Facteur Terme’
* Terme’ → opmul Facteur Terme’ | ε

1. **Eliminer l’ambiguïté**

* **Exp → Exp\_simple | Exp\_simple oprel Exp\_simple**
* Exp **→** Exp\_simple Exp’
* Exp’ **→** ε | oprel Exp\_simple
* La grammaire deviant :
* P → **program** id ; Dcl Inst\_composée .
* Dcl → **var** Liste\_id : Type ; Dcl | ε
* Liste\_id → id Liste\_id’
* liste\_id’ → , id Liste\_id’ | ε
* Liste\_inst → I Liste\_inst’
* Liste\_inst’ → ; I Liste\_inst’ | ε
* Type → **integer** | **char**
* Inst\_composée → **begin** Liste\_inst **end**
* I → Id := Exp\_simple | **if** Exp **then** I **else** I |

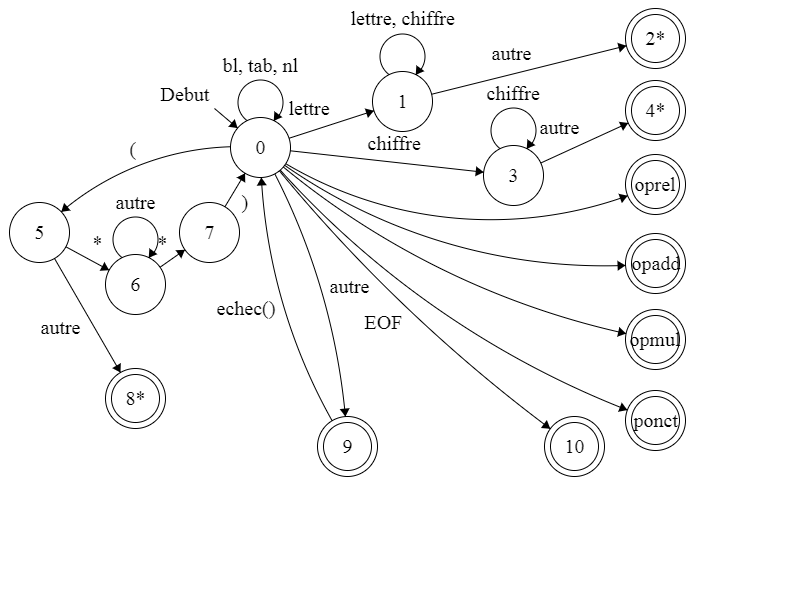
**while** Exp **do**  I

**read**(id) | **readln**(id) | **write**(id) | **writeln**(id)

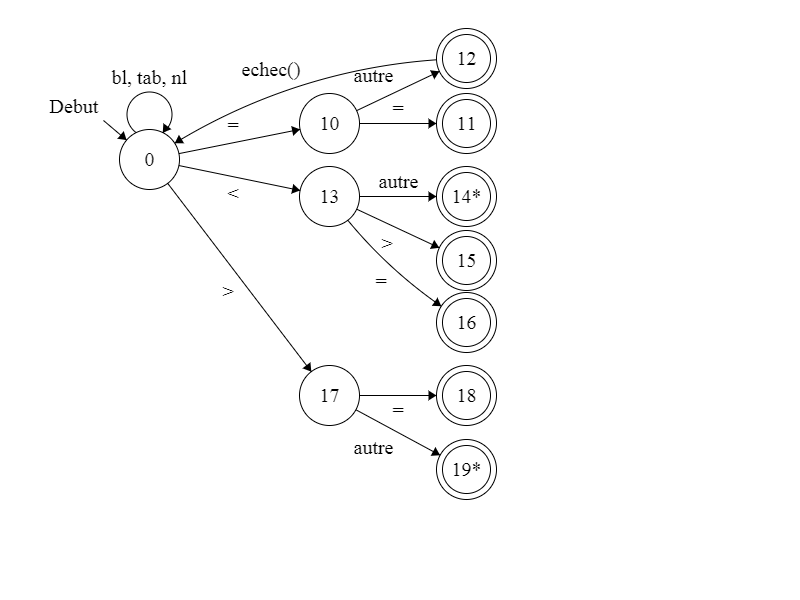
* Exp **→** Exp\_simple Exp’
* Exp’ **→** ε | oprel Exp\_simple
* Exp\_simple → Terme Exp\_simple’
* Exp\_simple’ → opadd Terme Exp\_simple’ | ε
* Terme → Facteur Terme’
* Terme’ → opmul Facteur Terme’ | ε
* Facteur → id | nb | (Exp)

1. Analyse lexicale
2. **Définition des unités lexicales**

|  |  |  |  |
| --- | --- | --- | --- |
| UL | Code | UL | Code |
| **Program** | 0 | Pv ‘ ; ’ | 16 |
| **Var** | 1 | Pt ‘. ‘ | 17 |
| **Integer** | 2 | Dp ‘: ’ | 18 |
| **Char** | 3 | Vg ‘,’ | 19 |
| **Begin** | 4 | Opaff ‘:=’ | 20 |
| **End** | 5 | Opmul | 21 |
| **If** | 6 | Oprel | 22 |
| **Then** | 7 | Opadd | 23 |
| **Else** | 8 | Nb | 24 |
| **while** | 9 | Po ‘(‘ | 25 |
| **Do** | 10 | Pf ‘)’ | 26 |
| **Read** | 11 |  |  |
| **Readln** | 12 |  |  |
| **Write** | 13 |  |  |
| **Writeln** | 14 |  |  |
| id | 15 |  |  |

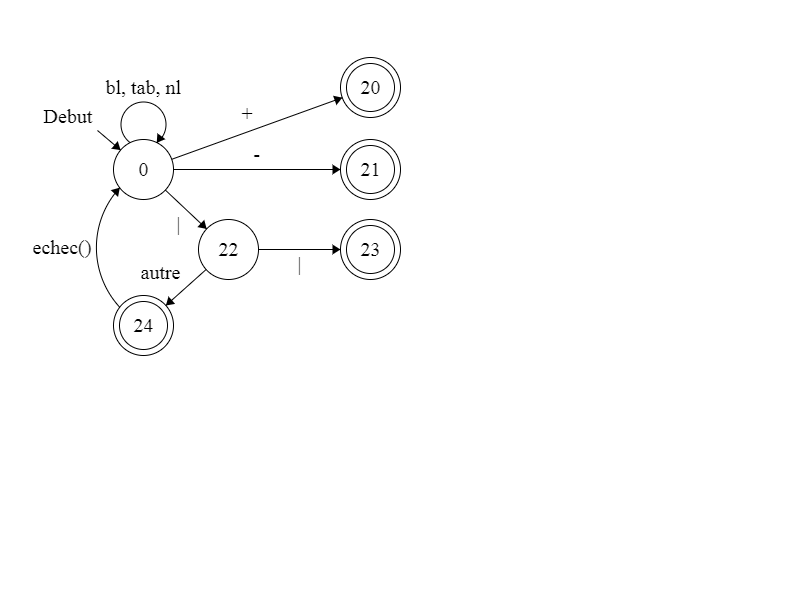
****

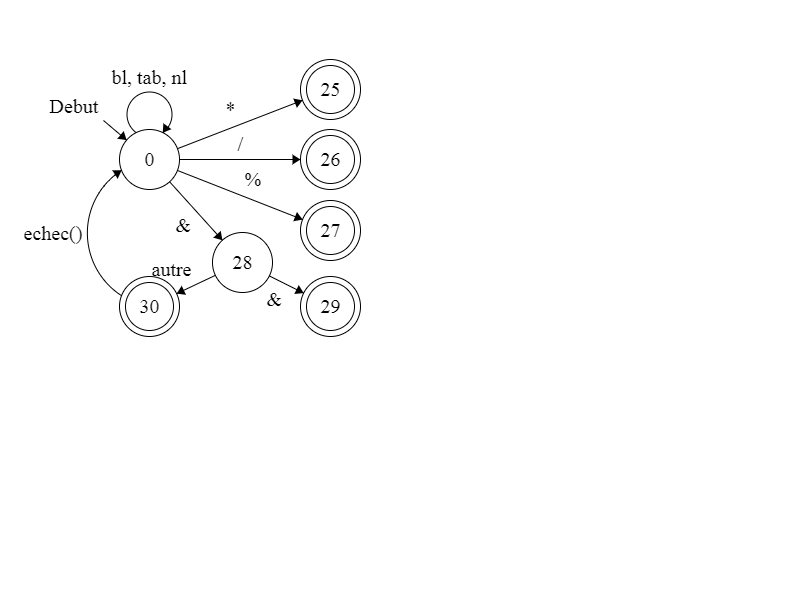
1. **Automate à état fini**

****

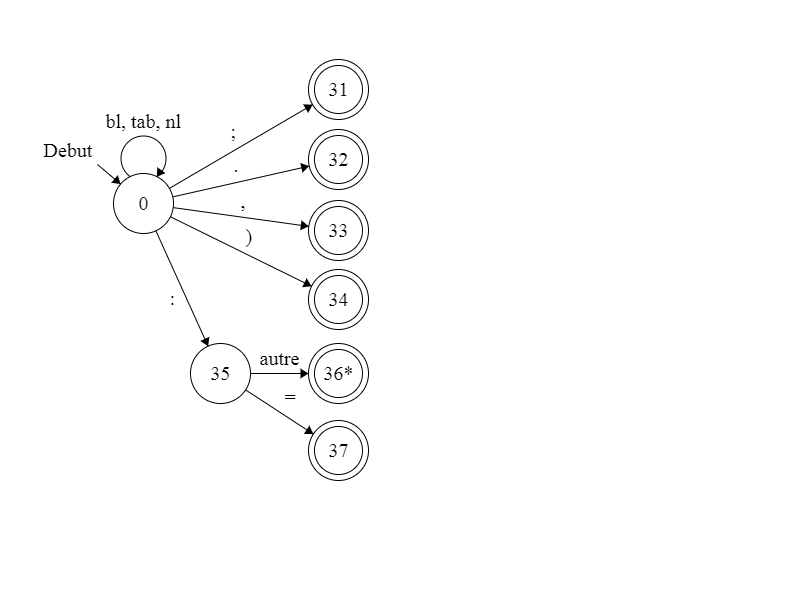
# ! 10 become 38

1. **Oprel**

1. **Opadd**

****

1. **Opmul**



1. **Ponct**
2. Controleur sémantique et reprise sur erreur
3. **Contrôle sémantique sur :**

* Déclaration mal placée.
* Ré déclaration de variables. (pas encore implémenté)
* Contrôle de type.

1. **Reprise sur erreur**
2. *Les premiers*
3. *Les suivants*