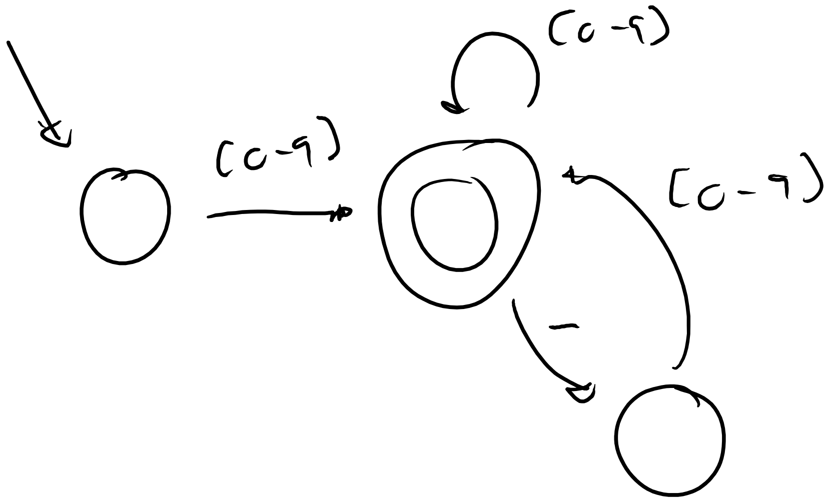
1) Um autômato finito não consegue contar o número de '=' que abre o comentário para casar com os '=' que fecham, logo uma expressão regular também não pode expressar.

2) $[0-9]+(\_[0-9]+)^*$



3) ⓐ Primeiro fragmento não é LL(1) pois duas primeiras alternativas de STMT começam com <u>id</u>.
Segundo fragmento é LL(1)

FIRST+(STMT → var ...) = {var}

FIRST+(STMT → VAR ...) = {id}

FIRST+(STMT → '{' ...) = {'{'}

FIRST+([':' TYPE] → ':' ...) = {':'}

FIRST+([':' TYPE] → ε) = {'=', ';', '}',
'{', id, var}

FIRST+(['=' EXP] → '=' ...) = {'='}

FIRST+(['=' EXP] → ε) = {';', '}', '{',
id, var}

FIRST+([';'] → ';') = {';'}

FIRST+([';'] → ε) = {'}', '{', id, var}

FIRST+({STMT} → STMT ...) =
{var, id, '{'}

FIRST+({STMT} → ε) =
{'}'}

FIRST+(['<' TARGS '>'] → '<') =
{'<'}

FIRST+(['<' TARGS '>'] → ε) =

$\{ '=', ';', '[', \text{var}, \text{id} \}$

$\text{FIRST+}(\{'.' \text{ id}\} \to '.' \text{ id}) = \{'.'\}$

$\text{FIRST+}(\{'.' \text{ id}\} \to \varepsilon) = \{'='\}$

Não há interseção entre os FIRST+ para o mesmo não-terminal.

(b) Esquerda (com retrocesso)

```
STMT() {
    atual = pos
    try {
        TYPE()
        match(id)
        atual' = pos
        try {
            match('=')
            EXP()
        } catch(Falha)
            pos = atual'
        }
        match(';')
    } catch(Falha) {
```

```
pos = atual
try {
    VAR()
    match('=')
    EXP()
    match(';')
} catch(Falha) {
    pos = atual
    match('{')
    while(true) {
        atual' = pos
        try { STMT() }
        catch (Falha) {
            pos = atual'
            break
        }
    } //while
  } //try
} //try
} //STMT()
```

```
TYPE() {
    match(id)
    atual = pos
    try {
        match('<')
        TARGS()
        match('>')
    } catch(Falha) {
        pos = atual
    }
}

VAR() {
    match(id)
    while(true) {
        atual = pos
        try {
            match('.')
            match(id)
        } catch(Falha) {
                pos = atual
                break
            }
        }
}
```

Direita (preditivo)

```
START() {
    if (la == var) {
        match(var)
        match(id)
        if (la == ':') {
            match(':')
            TYPE()
        }
        if (la == '=') {
            match('=')
            EXP()
        -
```

```
            }
            if (la == ';') match(';')
        } else if (la == id) {
            VAR()
            match('=')
            Exp()
            if( la == ';') match(';')
        } else {
            match('{')
            while(la != '}') {
                STMT()
            }
            match('}')
        }
}

TYPE() {
    match(id)
    if( la == '<') {
        match('<')
        Tparcs()
        match('>')
    }
}
var() {
    match(id)
    while( la == '.') {
        match('.')
        match(id)
    }
}
```