A linguagem LOOP

A linguagem LOOP é uma linguagem de programação muito simples que possui apenas dois tipos de instrução

```
X ← <expr aritmética>;          Loop X
                                    <bloco de comandos>
                                EndLoop
```

Isto é, um comando de atribuição e um laço de repetição.

O número de voltas do laço é determinado pelo valor da variável `X` no momento em que o programa chega ao laço.

Modificações no valor dessa variável dentro do bloco de comandos não afetam o número de voltas do laço.

Por exemplo, o programa abaixo troca os valores de `X` e `Y`, caso o primeiro seja menor do que o segundo

```
Z  <--  Y - X;
LOOP Z
   X  <--  X + 1;
   Y  <--  Y - 1;
ENDLOOP
```

Note que a coisa funciona porque na situação emq em `Z` vale zero ou um número negativo, o programa não entra no laço.

Certo.

Agora imagine que um programa em LOOP é uma palavra — (*i.e., uma sequência de caracteres*)

Daí, a gente pode definir a linguagem

ProgLOOP = palavras que correspondem a programas válidos em LOOP

Por exemplo, o programa acima é uma palavra de ProgLOOP.

Mas o seguinte programa não é

```
LOOP X
   Y  <--  X + * 41;
ENDLOOP
LOOP Y
ENDLOOP
LOOP Z
```

Quer dizer, nós temos 3 problemas aqui

- a expressão aritmética no comando de atribuição não é válida
- o segundo laço não tem um bloco de comandos
- as quantidades de termos `LOOP` e `ENDLOOP` não são iguais

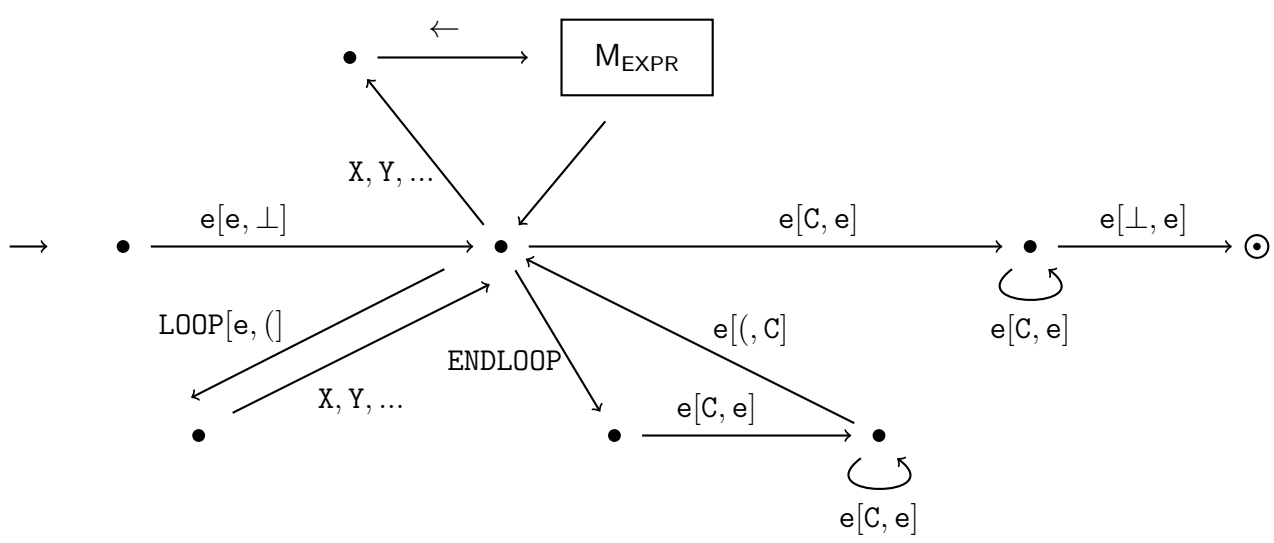
A tarefa agora é

→ *Construir um autômato de pilha que reconhece a linguagem* ProgLOOP

Bom, é bem fácil ver que os termos `LOOP` e `ENDLOOP` fazem o papel de abre-parênteses e fecha-parênteses.

E nós podemos reutilizar o autômato que reconhece EXPR para verificar se os comandos de atribuição são válidos.

Daí, a coisa fica assim



◇