Teoria dos Autômatos e Linguagens Formais

aula 18: O padrão recursivo

1 Introdução

Considere a seguinte palavra de PARBAL

$$( ) ((( ))) (( ) ( ))$$

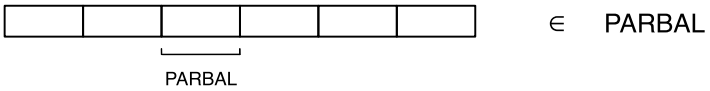
A primeira observação é que ela pode ser decomposta em 3 partes

$$( ) \;\Big|\; ((( ))) \;\Big|\; (( ) ( ))$$

onde cada parte também é uma palavra de PARBAL.

Essa observação mostra que existe um padrão repetitivo na linguagem PARBAL

→ *Nós podemos formar palavras de* PARBAL *fazendo concatenações arbitrárias de palavras de* PARBAL



A seguir, nós observamos que a primeira parte da decomposição acima é a palavra mais simples da linguagem PARBAL: ( ) — (*com a exceção da palavra vazia, claro*)

Por outro lado, a segunda parte da decomposição

$$((( )))$$

é uma palavra complexa que não pode ser decomposta como uma concatenação de palavras mais simples de PARBAL.

O que está acontecendo aqui?

Bom, o que está acontecendo é que existe um outro tipo de padrão nas linguagens reconhecidas por autômatos de pilha: os *padrões recursivos*.

Mas, o que é isso?

Examinando novamente a segunda parte da decomposição

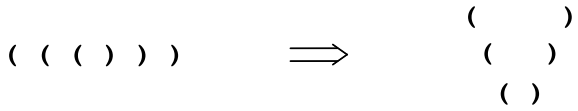
$$((( )))$$

nós vemos claramente que há alguma forma de repetição ali — (*uma repetição controlada*)

Quer dizer, nós podemos descrever a situação da seguinte maneira

- primeiro aparecem 3 símbolos abre parênteses,
  e depois aparecem 3 símbolos fecha parênteses

Mas, nós também podemos decompor essa palavra em 3 palavras mais simples de PARBAL, da seguinte maneira



Quer dizer, nós podemos imaginar que existem duas maneiras de combinar palavras simples de PARBAL para formar palavras mais complexas

A primeira forma corresponde a um *padrão repetitivo*, e a segunda corresponde a um *padrão recursivo*.

É interessante observar que a última parte da decomposição é formada por uma combinação dos dois tipos de padrão

(( ) ( ))

2 O padrão recursivo

Todas as linguagens que nós vimos nas aulas 16 e 17 envolvem o padrão recursivo.

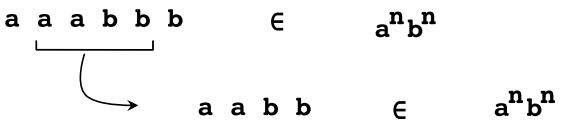
Vejamos.

a. A linguagem $a^n b^n$

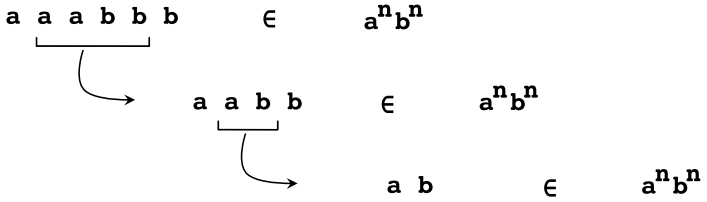
Note que no interior da seguinte palavra de $a^n b^n$

a a a b b b

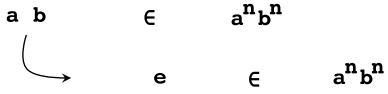
nós podemos encontrar uma outra palavra de $a^n b^n$



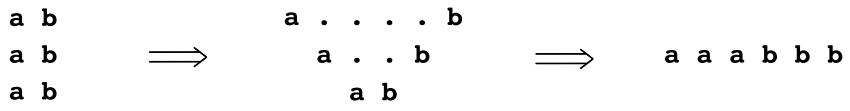
E que dentro dessa outra palavra nós podemos encontrar mais uma[1]



[1] Algumas pessoas ainda gostam de ver uma quarta palavra de $a^n b^n$ dentro dessa última:



Quer dizer, a palavra aaabbb é o resultado da composição recursiva de 3 palavras ab



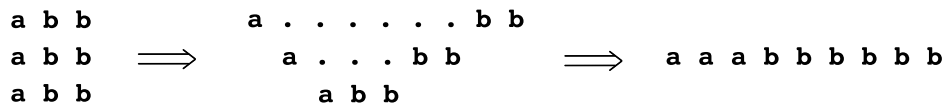
◇

b. A linguagem $a^n b^{2n}$

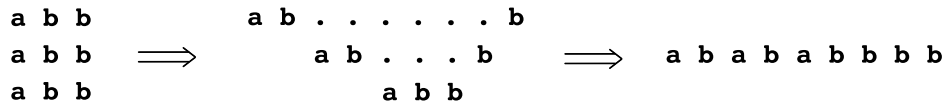
Agora não é difícil ver que uma palavra de $a^n b^{2n}$ como

a a a b b b b b b

pode ser vista como a composição recursiva de 3 blocos abb



Mas, vendo as coisas desse modo, nós podemos pensar em uma outra forma de composição recursiva



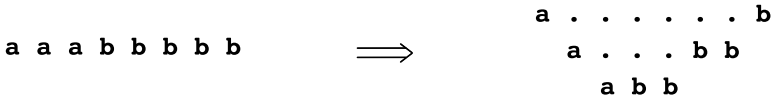
que corresponde à linguagem $(ab)^n b^n$.

◇

c. A linguagem $a^n b^m$, onde $n \leq m \leq 2n$

Dessa vez, nós temos uma composição recursiva envolvendo dois tipos de bloco: ab e abb.

Por exemplo,



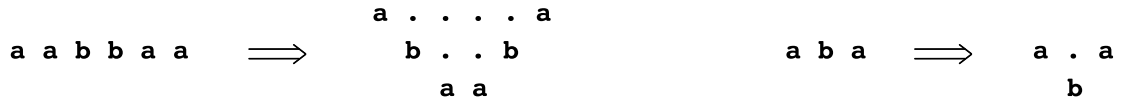
Mas, essa não é a única decomposição possível, pois nós também podemos ter

◇

d. PAL

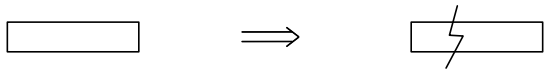
As palavras da linguagem PAL são formadas pela composição recursiva de 4 tipos de blocos: aa, bb, a, b.

Por exemplo



A novidade desse exemplo são os blocos de tamanho 1, que funcionam como uma espécie de condição de parada para a composição recursiva.

Quer dizer, nós podemos ver a composição recursiva como um procedimento onde um bloco básico é quebrado em duas partes



e daí um outro bloco é colocado entre as duas partes

Mas, um bloco de tamanho 1 não pdoe ser quebrado.

Logo, quando nós decidimos utilizar um bloco de tamanho 1, a composição recursiva deve parar.

◇

e. BAL

Ao contrário dos exemplos anteriores, BAL é uma linguagem que envolve tanto o padrão repetitivo como o padrão recursivo.

Por exemplo, a seguinte palavra de BAL

```
abbbbaaaaabb
```

pode ser decomposta como a concatenação de 3 palavras mais simples de BAL

```
ab | bbbaaa | aabb
```

onde cada uma dessas palavras é

- uma palavra mais simples possível (não vazia) de BAL
- ou uma composição recursiva

Por outro lado, a seguinte palavra

```
aababababb
```

não pode ser decomposta como uma concatenação de palavras mais simples de BAL, mas corresponde a uma composição recursiva com um padrão repetitivo no seu interior

```
a..................b
  ab | ab | ab | ab
```

No caso geral, as concatenações e as composições recursivas podem se alternar no processo de formação da palavra.

Por exemplo

$$\mathtt{a\,b} \;\Big|\; \mathtt{b}.....\mathtt{a} \;\Big|\; \mathtt{b}.....\mathtt{a}$$
$$\mathtt{b}....\mathtt{a} \qquad \mathtt{a\,b} \mid \mathtt{a\,b}$$
$$\mathtt{b\,a}$$

◇

f. Complicando as coisas

As construções realizadas por meio de padrões repetitivos e padrões recursivos podem ficar mais complicadas.

Por exemplo, consider a linguagem

$$\mathtt{a}^{\mathtt{m}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{m+n}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{n}}$$

À primeira vista, isso não parece envolver nem repetição e nem recursão.

Mas, uma pequena esperteza deixa as coisas mais claras.

Veja o que acontece quando nós quebramos o bloco de a's mais interno da seguinte maneira

$$\mathtt{a}^{\mathtt{m}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{m}} \mid \mathtt{a}^{\mathtt{n}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{n}}$$

Quer dizer, agora nós vemos que a linguagem consist na concatenação de dois blocos construídos de maneira recursiva.

E nós podemos imaginar a possibilidade de uma concatenação arbitrária de blocos desse tipo, o que nos daria a linguagem

$$\left(\,\mathtt{a}^{\mathtt{n}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{n}}\,\right)^{*}$$

Mas, e essa linguagem aqui

$$\mathtt{a}^{\mathtt{m}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{n}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{m+n}}$$

O que nós podemos fazer nesse caso?

Bom, aplicando a esperteza do exemplo anterior, nós obtemos

$$\mathtt{a}^{\mathtt{m}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{n}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{n}} \mid \mathtt{a}^{\mathtt{m}}$$

A seguir, separando as coisas da seguinte maneira

$$\mathtt{a}^{\mathtt{m}}\,\mathtt{b}\,\left(\,\mathtt{a}^{\mathtt{n}}\,\mathtt{b}\,\mathtt{a}^{\mathtt{n}}\,\right)\mathtt{a}^{\mathtt{m}}$$

Nós vemos um padrão recursivo no interior de um outro padrão recursivo.

◇

3 Reconhecendo padrões recursivos

Ao identificar o padrão recursivo de uma linguagem fica fácil construir um autômato de pilha que reconhece as suas palavras.

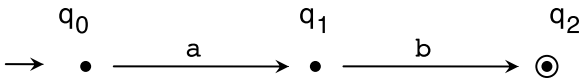
Vejamos.

Considere a linguagem $a^n b^n$ outra vez.

O bloco básico dessa linguagem é ab.

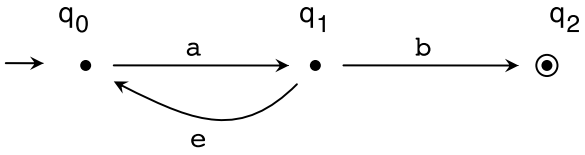
E o padrão recursivo consiste na possibilidade de introduzir outros blocos no interior de um bloco básico.

Daí que, para obter o autômato de pilha, nós começamos com o caminho

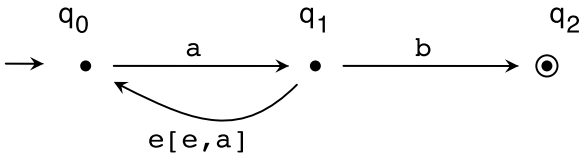


O estado q_2 foi marcado como final porque ab é uma palavra da linguagem.

A seguir, nós colocamos a transição que permite iniciar a leitura de um novo bloco no meio da leitura de um bloco básico

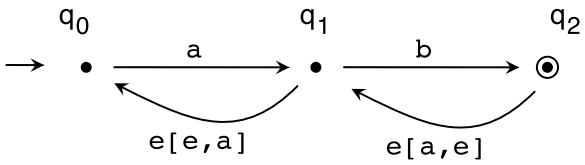


Mas, a leitura do bloco que foi interrompida precisa ser completada mais tarde, e para lembrar disso nós colocamos um a na pilha



Em algum momento a recursão chega ao fim, quando nós lemos uma bloco ab completo (e alcançamos o estado q_2).

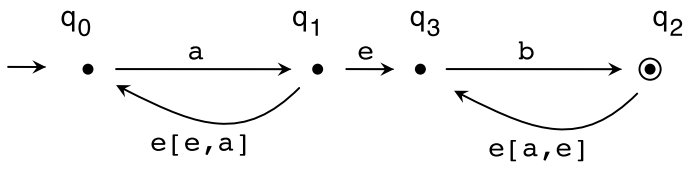
Nessa hora, é preciso retomar a leitura dos blocos que foram interrompidos, o que pode ser feito da seguinte maneira



Mas esse autômato ainda tem um pequeno defeito: note que ele aceita a palavra aababb, por exemplo.

O problema é que, no momento em que nós retomamos a leitura dos blocos interrompidos ainda é possível voltar ao estado inicial para iniciar a leitura de novos blocos ab.

O diagrama abaixo resolve esse problema

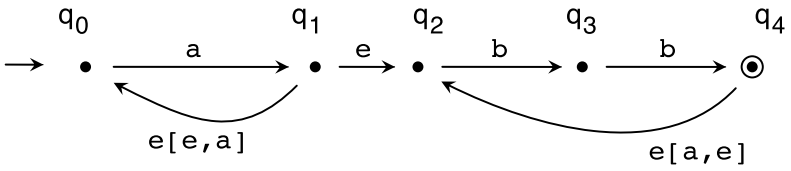


A seguir nós vamos ver mais exemplos.

Exemplos

a. A linguagem $a^n b^{2n}$ (continuação)

Com base nas ideias que foram apresentadas no exemplo anterior, é fácil construir um autômato de pilha para $a^n b^{2n}$

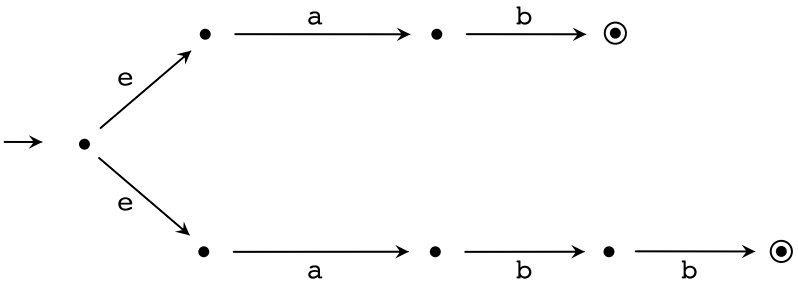


◇

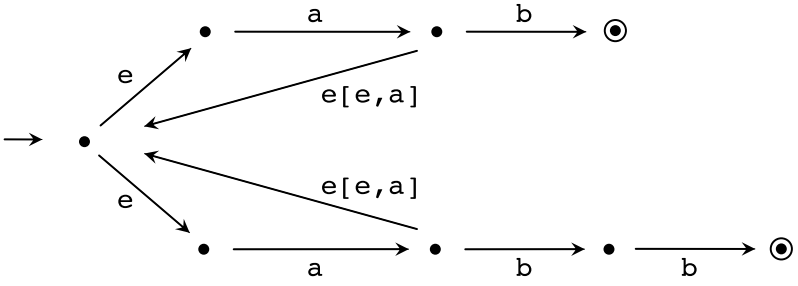
b. A linguagem $a^n b^m$, onde $n \leq m \leq 2n$ (continuação)

Como vimos na seção anterior, essa linguagem possui dois blocos básicos: `ab` e `abb`.

Logo, nós vamos ter dois caminhos no autômato



Em ambos os caminhos nós podemos interromper a leitura do bloco no meio, para dar início à leitura de um novo bloco



E a ideia é que, ao alcançarmos o fim de ambos os caminhos, nós devemos retomar a leitura dos blocos que foram interrompidos.

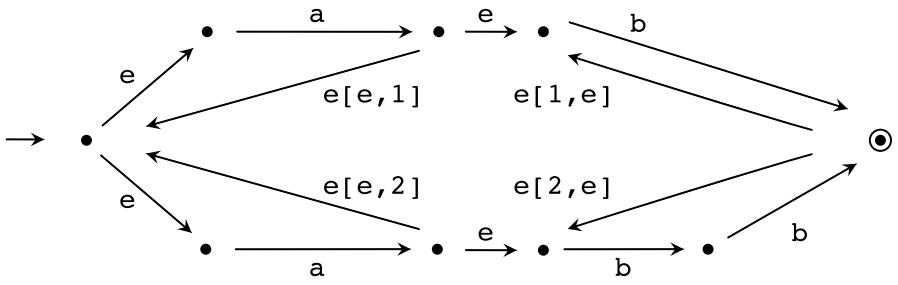
Mas, aqui nós temos um pequeno problema

- *Como é que nós podemos saber se a leitura que foi interrompida era de um bloco* `ab` *ou* `abb`*?*

A solução é colocar essa informação na pilha.

Quer dizer, ao invés de empilhar um `a` em ambos os casos, nós podemos empilhar, por exemplo, os símbolos `1` ou `2` para indicar qual caminho foi interrompido.

Abaixo nós temos o autômato de pilha que implementa essa ideia, onde os dois estados finais foram combinados em um só



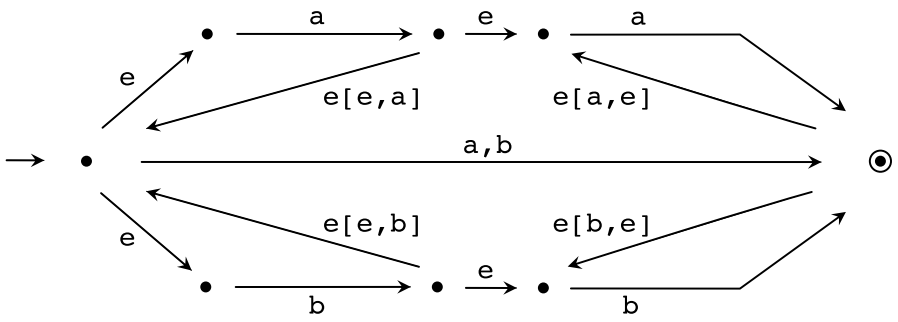
◇

c. PAL (continuação)

A linguagem PAL possui 4 blocos básicos: `aa`, `bb`, `a`, `b`.

A leitura dos dois primeiros pode ser interrompida, e a leitura dos dois últimos indica o fim da recursão.

Abaixo nós temos o autômato de pilha que reconhece a linguagem PAL



◇

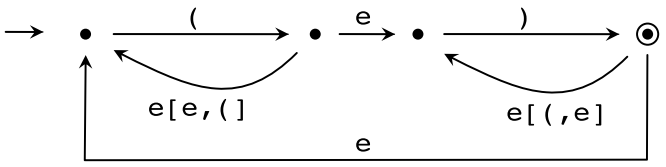
d. PARBAL (continuação)

A linguagem PARBAL só possui um bloco básico: ( ).

Mas, ela também envolve padrões repetitivos.

Quer dizer, quando nós chegamos ao final de um bloco nós podemos iniciar a leitura de um novo bloco.

Para implementar essa possibilidade, basta conectar o estado final ao estado inicial do autômato (*Examine a computação desse autômato sobre a palavra* ( ) ((( ))) (( ) ( )) .)



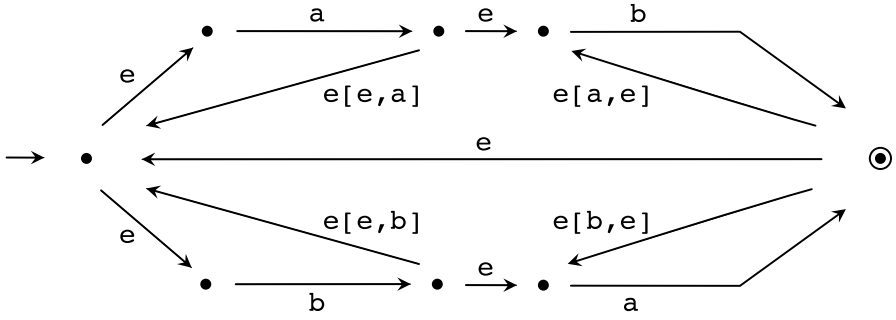
◇

e. BAL (continuação)

A linguagem BAL possui dois blocos básicos: `ab` e `ba`.

E ela também envolve padrões repetitivos.

Abaixo nós temos o autômato de pilha que reconhece a linguagem BAL



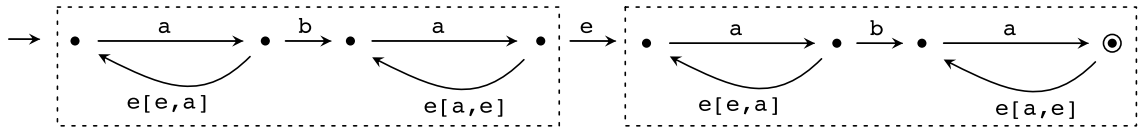
◇

f. Complicando as coisas (continuação)

Como vimos na seção anterior, a ideia é ver essa linguagem como a concatenação de duas linguagens definidas por um padrão recursivo

$$\mathtt{a}^n\,\mathtt{b}\,\mathtt{a}^n \mid \mathtt{a}^m\,\mathtt{b}\,\mathtt{a}^m$$

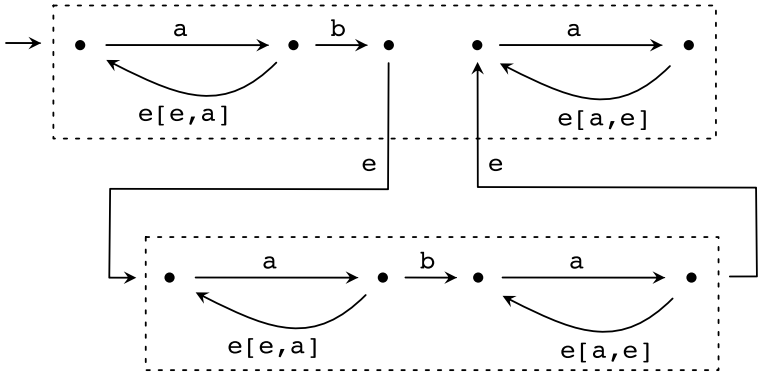
Daí que o autômato de pilha que reconhece essa linguagem pode ser construído como a concatenação de dois autômatos de pilha para $\mathtt{a}^n\mathtt{b}\mathtt{a}^n$.



Como vimos na seção anterior, a ideia é ver essa linguagem como palavras definidas por um padrão recursivo no interior de uma palavra definida por um padrão recursivo

$$\mathtt{a}^n\,\mathtt{b}\,\left(\,\mathtt{a}^m\,\mathtt{b}\,\mathtt{a}^m\,\right)\,\mathtt{a}^n$$

Daí que o autômato de pilha que reconhece essa linguagem pode ser construído como um autômato de pilha que interrrompe a sua execução para iniciar a computação de um segundo autômato de pilha, e, quando a computação do segundo autômato termina, a execução do primeiro é retomada



◇

f. EXPR

Lembre que EXPR é o conjunto das expressões aritméticas completamente parentizadas (envolvendo as operações + e ∗).

Por exemplo,

$$(\,4\,+\,7\,)$$

$$(\,(\,18\,+\,5\,)\,*\,2\,)$$

$$(\,(\,5\,+\,7\,)\,*\,(\,15\,+\,2\,)\,)$$

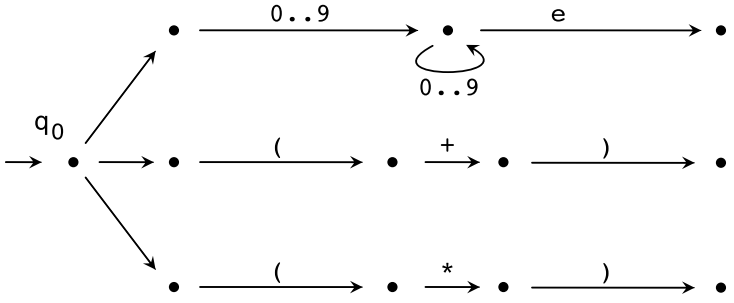
Nós podemos descrever o padrão recursivo da linguagem EXPR da seguinte maneira

`<num>` $\qquad$ (`<expr>` + `<expr>`) $\qquad$ (`<expr>` ∗ `<expr>`)

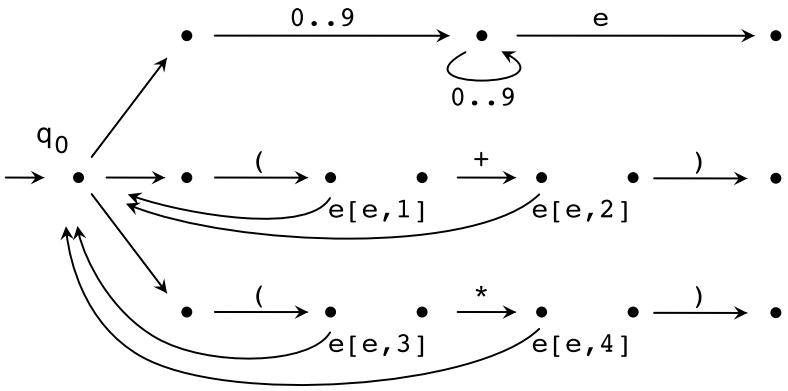
Quer dizer, um expressão aritmética é:

- um número
- ou uma construção com parênteses e o operador +, onde os operandos são expressões
- ou uma construção com parênteses e o operador ∗, onde os operandos são expressões

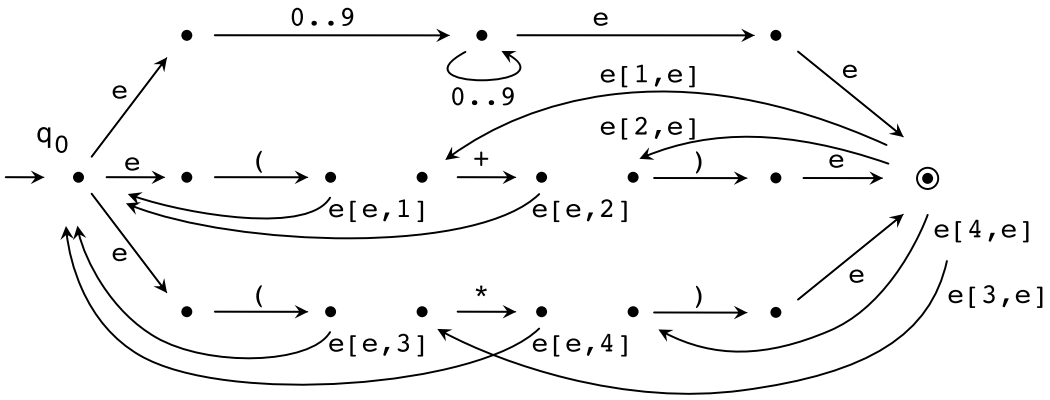
Logo, nós podemos começar a construção do autômato com os seguintes caminhos



A seguir, nós introduzimos transições que permitem iniciar a leitura de uma nova expressão no meio de uma expressão



Finalmente, nós adicionamos um estado final que retoma a leitura de expressões interrompidas, removendo símbolos da pilha.



◇