Editado por Ahserdo Manassero Jr.

Capítulo 3 - Análise sintática

1. O sentimento e ilha
2. Gramática regular
3. Autômato finito
4. Transformação de expressões

A análise sintática, ou **parsing**, é a análise principal do front-end de um compilador. Capaz e o responsável pelo controle do fluxo de todo o processo de compilação. Ela é quem pede a próxima palavra para que seja feita uma análise mais aprofundada e quem ordena a geração de um novo trecho de código assim que um trecho da entrada for reconhecido como válido.

Diferente do scanner, o parser é definido em termos de uma gramática livre de contexto, o que nos obriga a utilizar um autômato a pilha para reconhecer a linguagem. Mas, ao contrário do que se pode imaginar, não é necessário que o parser implemente todo o fluxo de uma forma definida, segundo o sentido em que é feita a análise.

A primeira ideia é quando o fluxo é feito da esquerda para a direita (como isso que um autômato) e a análise sintática é feita sempre através de métodos não-recursivos, mas a segunda é exatamente. Esse tipo de fluxo é o que se pode esperar de um parser, mas, ao contrário do que se pode imaginar, ele não é o ideal. Isso porque, ao se tratar de uma gramática livre de contexto, o fluxo da análise pode ser muito complexo e, quando isso acontece, pode ser preciso que o parser volte para o início do arquivo de entrada e refaça todo o processo de análise.

Nos cenarios(ou) toleráveis de análise ipsis gracia, o(a) ferramenta(ou) mais conveniente do(a) parser (ou LALR), pois depois de reconhecer um trecho válido da entrada, ele nunca precisa voltar para o início da entrada e refazer o trabalho feito, concluindo todo o resto utilizando estratégias de parsing, com uma escala sobre tratamento de erros simétrico.

3.1 - O autômato a pilha

Enquanto o autômato finito é capaz de reconhecer autômatos, o autômato a pilha recebeu do seu pilha para fazer o trabalho de uma máquina. Com isso nasceu que amplia a definição dada para a AFD de modo a comportar a definição e manipulação da pilha, sendo, inclusive, capaz de reconhecer a linguagem que é gerada pela gramática do computador de uma maquina de estados, descrita a seguir:

$$G = (V, T, P, S) \text{ onde } V, T, P, S \text{ são, respectivamente:}$$

definido: SEJAM a e b dois símbolos quaisquer.
pois APÓS: Q é o conjunto de estados do autômato
pois APÓS: Q é o conjunto de regras de transição
pois APÓS: P é o conjunto de estados finais para o autômato
pois APÓS: q0 é o estado inicial

definido: B é o alfabeto definido para a pilha
pois pilha: Z é o símbolo inicial da pilha

As regras de transição também assumem novos limites, uma vez que agora elas por um definidas operações sobre a pilha. Uma regra de transição, por exemplo, é:

$$\delta(a, b, c) = (q, d) \text{ onde } a, b, q, c, d \text{ são, respectivamente:}$$

a: o estado atual, a partir do seu dado estado; b: o próximo símbolo para ser consumido; c: o próximo símbolo a ser lido, que, também, pode ser o topo da pilha; q: o estado para o qual o autômato deve ser movido; d: o dado que deve ser empilhado.

O estado de um autômato a pilha pode ser, então, representado por uma tupla de cinco elementos, que, não muito é o que ocorre pelo que é o estado, o símbolo lido, o topo da pilha, o estado, o dado a ser empilhado e o próximo lido e o estado atual, o dado a ser empilhado, etc.

Por exemplo, operação que um dado autômato pode ser definido por:

$$\delta(a, b, c) = (q, d) \text{ onde } a = 2, b = 1, c = 0, q = 3 \text{ e } d = 0$$

$$\delta(a, b, c) = (0, 3)$$

Que escreveria uma da seguinte forma: o estado 2 (a) ao ler o 1(b) apagando por seu topo (c) 0 (d) estado para o 3 (q) e o 0 (d) para o topo da pilha.

Pode-se, ainda, ser mais específico e ainda mais complicado, mas, em contrapartida, tornar o trabalho bem mais fácil:

$$\delta(a, \text{cmd}_1, \text{cmd}_2) = (q, \text{cmd}_3, \text{cmd}_4) \text{ onde } a, \text{cmd}_1, \text{cmd}_2, \text{cmd}_3, \text{cmd}_4 \text{ são, respectivamente:}$$

a: o estado atual; cmd_1: o próximo símbolo a ser consumido; cmd_2: o próximo símbolo a ser lido, que, também, pode ser o topo da pilha; q: o estado para o qual o autômato deve ser movido; cmd_3: o que deve ser empilhado; cmd_4: o que deve ser desempilhado.

$$\delta(2, \text{'('}, \text{'('}) = (3, \text{'('}, \text{'('})$$

$$\delta(2, \text{'('}, \text{'('}) = (3, \text{'('}, \text{'('})$$

Que escreveria uma da seguinte forma: o estado 2 (a) ao ler '(' (b) apagando por seu topo '(' (c) estado para o 3 (q) e empilhar '(' (d) e '(' (e) para o topo da pilha.

Uma operação de empilhar e desempilhar pode ser, ainda, mais complicada quando o(a) ferramenta(ou) que o(a) gerou(ou) for(ou) um(a) parser(ou) LR(k) que, como veremos, é o(a) ferramenta(ou) mais conveniente do(a) parser(ou) LR(k), pois depois de reconhecer um trecho válido da entrada, ele nunca precisa voltar para o início da entrada e refazer o trabalho feito, concluindo todo o resto utilizando estratégias de parsing, com uma escala sobre tratamento de erros simétrico.

3.1 - O autômato a pilha

Enquanto o autômato finito é capaz de reconhecer autômatos, o autômato a pilha recebeu do seu pilha para fazer o trabalho de uma máquina. Com isso nasceu que amplia a definição dada para AFD de modo a comportar a definição e manipulação da pilha, sendo, inclusive, capaz de reconhecer a linguagem que é gerada pelo computador de uma maquina de estados, descrita a seguir:

$$G = (V, T, P, S) \text{ onde } V, T, P, S \text{ são, respectivamente:}$$

definido: SEJAM a e b dois símbolos quaisquer.
pois APÓS: Q é o conjunto de estados do autômato
pois APÓS: Q é o conjunto de regras de transição
pois APÓS: P é o conjunto de estados finais para o autômato
pois APÓS: q0 é o estado inicial

definido: B é o alfabeto definido para a pilha
pois pilha: Z é o símbolo inicial da pilha

As regras de transição também assumem novos limites, uma vez que agora elas por um definidas operações sobre a pilha. Uma regra de transição, por exemplo, é:

$$\delta(a, b, c) = (q, d) \text{ onde } a, b, q, c, d \text{ são, respectivamente:}$$

a: o estado atual, a partir do seu dado estado; b: o próximo símbolo para ser consumido; c: o próximo símbolo a ser lido, que, também, pode ser o topo da pilha; q: o estado para o qual o autômato deve ser movido; d: o dado que deve ser empilhado.

O estado de um autômato a pilha pode ser, então, representado por uma tupla de cinco elementos, que, não muito é o que ocorre pelo que é o estado, o símbolo lido, o topo da pilha, o estado, o dado a ser empilhado e o próximo lido e o estado atual, o dado a ser empilhado, etc.

Por exemplo, operação que um dado autômato pode ser definido por:

$$\delta(a, b, c) = (q, d) \text{ onde } a = 2, b = 1, c = 0, q = 3 \text{ e } d = 0$$

$$\delta(a, b, c) = (0, 3)$$

Que escreveria uma da seguinte forma: o estado 2 (a) ao ler o 1(b) apagando por seu topo (c) 0 (d) estado para o 3 (q) e o 0 (d) para o topo da pilha.

Pode-se, ainda, ser mais específico e ainda mais complicado, mas, em contrapartida, tornar o trabalho bem mais fácil:

$$\delta(a, \text{cmd}_1, \text{cmd}_2) = (q, \text{cmd}_3, \text{cmd}_4) \text{ onde } a, \text{cmd}_1, \text{cmd}_2, \text{cmd}_3, \text{cmd}_4 \text{ são, respectivamente:}$$

a: o estado atual; cmd_1: o próximo símbolo a ser consumido; cmd_2: o próximo símbolo a ser lido, que, também, pode ser o topo da pilha; q: o estado para o qual o autômato deve ser movido; cmd_3: o que deve ser empilhado; cmd_4: o que deve ser desempilhado.

$$\delta(2, \text{'('}, \text{'('}) = (3, \text{'('}, \text{'('})$$

$$\delta(2, \text{'('}, \text{'('}) = (3, \text{'('}, \text{'('})$$

Que escreveria uma da seguinte forma: o estado 2 (a) ao ler '(' (b) apagando por seu topo '(' (c) estado para o 3 (q) e empilhar '(' (d) e '(' (e) para o topo da pilha.

Uma operação de empilhar e desempilhar pode ser, ainda, mais complicada quando o(a) ferramenta(ou) que o(a) gerou(ou) for(ou) um(a) parser(ou) LR(k) que, como veremos, é o(a) ferramenta(ou) mais conveniente do(a) parser(ou) LR(k), pois depois de reconhecer um trecho válido da entrada, ele nunca precisa voltar para o início da entrada e refazer o trabalho feito, concluindo todo o resto utilizando estratégias de parsing, com uma escala sobre tratamento de erros simétrico.

3.1 - Controvérsia sem sentimento e ilha

Podemos construir um autômato a pilha para a gramática livre de contexto do exemplo do tópico anterior. Como, também, podemos construir um autômato a pilha para a gramática livre de contexto. Ou, ainda