Não determinismo: AFN

Como vimos, a definição da função de transição em AFDs como uma função total confere a esses autômatos seu determinismo nas computações. Ou seja, conseguimos traçar perfeitamente as computações realizadas para processar uma dada palavra.

☆ Em algumas situações, contudo, essa restrição é muito rigorosa e dificulta a construção de um autômato para uma linguagem. Se abrirmos mão de sempre haver uma única transição correspondente para cada par (estado, símbolo), ou seja, possibilitarmos que múltiplas (incluindo nenhuma) transição seja feita sob esse par, então teremos várias possíveis computações para uma dada palavra em um autômato. Isso confere a essa máquina o que chamamos de **não determinismo**.

☆ No caso de autômatos não determinísticos, existem diversas computações possíveis para uma mesma palavra. Esses autômatos passam a ser chamados de **autômatos finitos não determinísticos (AFN)**. De maneira intuitiva, uma palavra será reconhecida por um AFN se existir uma computação na qual a palavra é integralmente consumida (processada) e a máquina para em estado final. Note que podem existir outras computações em que a palavra não é reconhecida, mas é suficiente que ela seja reconhecida em uma para que seja aceita.

Exemplo: O diagrama abaixo apresenta um AFN.

Veja que existem duas transições sob o símbolo 0 no estado e_1. Nesse caso, ao processar a palavra 1010, por exemplo, fazemos uma transição para e_1 consumindo o símbolo 1. Em seguida, temos duas alternativas: (1) fazemos a transição que leva de novo a e_1; ou (2) transitamos para o estado e_2. O fato de termos alternativas é que confere o não determinismo ao autômato. Já não podemos traçar com certeza qual foi a transição escolhida. De qualquer forma, sabemos que se a escolha for pela segunda transição, então não conseguimos consumir toda a palavra, já que não existem transições sob qualquer símbolo no estado e_2. Por outro lado, se escolhermos a primeira, então consumimos toda a palavra e por fim transitamos sob o último 0 para e_2 , reconhecendo a palavra. Ou seja, existe uma computação que leva ao reconhecimento da palavra. Logo, ela é aceita pelo autômato. De fato, qualquer palavra que termine em 0 é aceita por esse autômato.

O AFN pode ser interpretado como uma máquina "massivamente paralela" em que todas as computações são feitas ao mesmo tempo e se uma terminar em aceitação a máquina aceita a palavra. Ou ainda pode-se dizer que ela possui um oráculo que é capaz de adivinhar qual a transição que leva a aceitação da palavra.

☆ Formalmente, um autômato finito não determinístico é uma quíntupla $(Q, \Sigma, \delta, I, F)$:

- Q é um conjunto finito não vazio de estados;
- Σ é o alfabeto da linguagem
- $I \subseteq Q$ é um conjunto de estados iniciais
- $F \subseteq Q$ é um conjunto de estados finais
- $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ é uma função total, a função de transição

Essa definição apresenta duas principais diferenças com a definição de AFDs. A primeira em relação à possibilidade de se iniciar a computação em mais de um estado. Alguns textos restringem os AFN a ter somente um estado inicial como os AFDs. Essa flexibilização, no entanto, não confere maior poder computacional aos AFNs. Ela apenas aumenta seu poder expressivo (facilita a construção de autômatos). A veracidade dessa afirmação será evidenciada mais a frente no curso. Mas, intuitivamente, podemos substituir os vários estados iniciais por um único com múltiplas transições.

A segunda diferença diz respeito à função de transição. Note que agora as transições são para conjuntos de estados. Ou seja, como vimos, dado um par (estado, símbolo) podemos transitar para um conjunto de outros estados, inclusive para nenhum (conjunto vazio). No diagrama de estados, essas transições são representadas por múltiplas arestas, uma para cada estado do conjunto resultante, tal como no exemplo acima.

Exemplo: O AFN do exemplo anterior é $M = (\{e_1, e_2\}, \{0,1\}, \delta, \{e_1\}, \{e_2\})$. A função de transição é dada na tabela

δ	0	1
e_1	$\{e_1, e_2\}$	$\{e_1\}$
e_2	$\emptyset$	$\emptyset$

Para definir formalmente a linguagem reconhecida por um AFN, precisamos definir a função de transição estendida tal como fizemos nos AFDs. Dado um AFN $M = (Q, \Sigma, \delta, I, F)$, **a função de transição estendida** $\hat{\delta}: \mathcal{P}(Q) \times \Sigma^* \rightarrow \mathcal{P}(Q)$ é definida como a seguir:

- $\hat{\delta}(X, \lambda) = X, \quad para \quad todo \quad X \subseteq Q$
- $\hat{\delta}(X, ay) = \hat{\delta}\left(\bigcup_{e \in X} \delta(e, a), y\right), \quad para \quad X \subseteq Q, \quad a \in \Sigma, \quad y \in \Sigma^*$

Intuitivamente, se quisermos saber quais estados podem ser alcançados ao computar uma palavra a partir de um conjunto de estados, precisamos fazer uma transição a partir de cada estado desse conjunto e repetir esse processo recursivamente dos estados alcançados nessa primeira transição.

Exemplo: Considerando o AFN do exemplo anterior e a palavra 1010. Veja a computação da função de transição estendida.