Definição de Gramáticas Irrestritas [1]

As produções de uma gramática têm a forma:

$$(V \cup \Sigma)^+ \rightarrow (V \cup \Sigma)^*$$

sendo que o lado esquerdo possui no mínimo uma variável (elemento de V). Os outros tipos de gramáticas consideradas (linear a direita, livre de contexto, sensível ao contexto) restringem a forma das produções. Uma gramática *irrestrita*, não.

Vai-se tentar mostrar que as gramáticas irrestritas são equivalentes às Máquinas de Turing. Lembre-se que:

- ◊ Uma linguagem é *recursivamente enumerável* se existe uma máquina de Turing que aceita toda cadeia da linguagem, e não aceita cadeias que não pertencem à linguagem.
- ◊ "Não aceita" *não* é o mesmo que "rejeita" – a máquina de Turing poderia entrar num *loop* infinito e nunca parar para aceitar *ou* rejeitar a cadeia.

Planeja-se mostrar que as linguagens geradas pelas gramáticas irrestritas são precisamente as linguagens recursivamente enumeráveis.

Das Gramáticas para as Máquinas de Turing

Teorema: Qualquer linguagem gerada por uma gramática irrestrita é recursivamente enumerável.

Isto pode ser provado da seguinte forma:

1. Se existe um procedimento para enumerar as cadeias de uma linguagem, então a linguagem é recursivamente enumerável.
2. Existe um procedimento para enumerar todas as cadeias em qualquer linguagem gerada por uma gramática irrestrita.
3. Portanto, qualquer linguagem gerada por uma gramática irrestrita é recursivamente enumerável.

Um argumento para o item (1) acima. Prova-se que a linguagem é recursivamente enumerável construindo uma Máquina de Turing para aceitar qualquer cadeia *w* da linguagem.

---

[1] Tradução de David Matuszek: http://www.seas.upenn.edu/~cit596/notes/dave/ungram1.html

- ◊ Construa uma máquina de Turing que "gere" as cadeias da linguagem em alguma ordem sistemática.
- ◊ Construa uma segunda máquina de Turing que compara sua entrada a *w* e aceita sua entrada se as duas cadeias são idênticas.
- ◊ Construa uma máquina de Turing composta que incorpora as duas máquinas acima, usando a saída da primeira como entrada para a segunda.

Agora, gera-se sistematicamente todas as cadeias da linguagem. Para outros tipos de gramáticas, gera-se as cadeias menores antes; não se sabe como fazer isso com uma gramática irrestrita, porque algumas produções poderiam encurtar a forma sentencial. Pode levar um milhão de passos para derivar λ.

Ao invés disto, ordena-se as cadeias *de derivação mais curta antes*. Primeiro, considera-se todas as cadeias que podem ser geradas a partir de S em um passo de derivação e verifica-se se as mesmas são compostas inteiramente de terminais (pode-se fazer isso porque há apenas um número finito de produções). Então considera-se todas as cadeias que podem ser derivadas em dois passos, e assim por diante.

Das Máquinas de Turing para as Gramáticas

Mostrou-se que uma Máquina de Turing pode fazer qualquer coisa que uma gramática irrestrita pode fazer. Agora, deve-se mostrar que uma gramática irrestrita pode fazer qualquer coisa que uma Máquina de Turing pode fazer. Isto pode ser feito usando uma gramática irrestrita para emular uma Máquina de Turing.

Lembre-se que uma *descrição instantânea* (DI) de uma Máquina de Turing é a cadeia

$$x_i \ldots x_j q_m x_k \ldots x_l$$

onde os x's são os símbolos na fita, q_m é o estado corrente e a cabeça de leitura/escrita está no quadrado que contém x_k (o símbolo que segue imediatamente q_m). Faz sentido que uma gramática, que é um sistema para reescrever cadeias, possa ser usada para manipular DIs, que são cadeias de símbolos.

Uma Máquina de Turing aceita uma cadeia *w* se

$$q_0 w \Rightarrow^* w_1 q_a w_2$$

para w_1, $w_2 \in \Sigma'$ e estado de aceitação q_a, enquanto uma gramática produz uma cadeia se

$$S \Rightarrow^* w.$$

Como a máquina de Turing começa com *w* e a derivação gramatical termina com *w*, a gramática construída funcionará "reversamente" quando comparada à Máquina de Turing.

As produções da gramática construída podem ser logicamente agrupadas em três conjuntos:

1. **Iniciação**. Estas produções constroem a cadeia ...B&$w_1q_aw_2$B... onde B indica um branco e & é uma variável especial usada para terminação;
2. **Execução**: Para cada regra de transição δ necessita-se uma produção correspondente;
3. **Limpeza**: A derivação deixará alguns símbolos q_0, B e & na cadeia (juntamente com a cadeia *w*), tal que são necessárias algumas produções adicionais para limpá-los.

Para os símbolos terminais Σ_G da gramática, usa-se o alfabeto de fita Σ_M da Máquina de Turing.

Para as variáveis V da gramática, usa-se:

- ◊ Um símbolo $q_i \in Q$ para cada estado da Máquina de Turing.
- ◊ B (branco) e & (usado para terminação).
- ◊ S (para símbolo inicial) e A (para iniciação).

Iniciação: É necessário gerar qualquer cadeia da forma

B...B&$w_1q_aw_2$B...B

Para gerar um número arbitrário de "brancos" em ambos os lados, usa-se as produções

$S \rightarrow BS \mid SB \mid \&A$

Agora, usa-se o A para gerar as cadeias w_1, $w_2 \in \Sigma'_M$, com um estado q_a em algum lugar no meio:

$A \rightarrow xA \mid Ax \mid q_a$, para todo $x \in \Sigma'_M$.

Execução: Para cada regra de transição δ necessita-se de uma produção correspondente. Para cada regra da forma

$$\delta(q_i, a) = (q_j, b, R)$$

usa-se uma produção

$$bq_j \rightarrow q_ia$$

e para cada regra da forma

$$\delta(q_i, a) = (q_j, b, L)$$

usa-se uma produção

$$q_jcb \rightarrow cq_ia$$

para todo $c \in \Sigma'_M$ (a assimetria é devido ao símbolo *à direita* de q ser o símbolo sob a cabeça de leitura/escrita da Máquina de Turing.)

Limpeza. Termina-se com uma cadeia que se parece com B...B&q_0wB...B, tal que são necessárias produções para se livrar de tudo menos do w:

$$B \rightarrow \lambda$$

$$\&q_0 \rightarrow \lambda$$