2
Gramáticas de Expressões de Parsing

Neste capítulo vamos rever a definição de Gramáticas de Expressões de Parsing (*Parsing Expression Grammars* — PEGs). Além disso, apresentaremos uma nova formalização de PEGs, baseada em semântica natural, que usaremos ao longo da tese.

Na seção 2.1 mostramos a formalização original de PEGs dada por Ford, e na seção 2.2 mostramos a relação usada por Ford para interpretar PEGs. Finalmente, na seção 2.3, apresentamos a nossa formalização de PEGs baseada em semântica natural.

2.1
Definição de PEGs

Segundo Ford [Ford, 2004], uma PEG é uma tupla $G = (V, T, P, p_S)$, onde V é um conjunto finito de não terminais (também chamados de variáveis), T é um conjunto finito de terminais, P é uma função de não terminais em expressões de parsing, e p_S é a expressão de parsing inicial.

Ao longo do texto, iremos usar também o termo *gramática* ao nos referirmos a uma PEG.

Uma expressão de parsing p pode ser definida indutivamente como mostrado a seguir, onde $a \in T$, $A \in V$, e p_1 e p_2 também são expressões de parsing:

$$p \;=\; \varepsilon \;\mid\; a \;\mid\; A \;\mid\; p_1\, p_2 \;\mid\; p_1 \,/\, p_2 \;\mid\; p_1^* \;\mid\; !p_1$$

Na definição acima, temos que ε é uma expressão que casa a cadeia vazia, a é um terminal, A é um não terminal, $p_1\, p_2$ é uma concatenação, $p_1 \,/\, p_2$ é uma escolha ordenada, p_1^* é uma repetição, e $!p_1$ é um predicado de negação.

A prioridade do operador * é a mais alta, seguida pela prioridade do operador !. A seguir, temos a prioridade da concatenação, e por último temos a prioridade da escolha ordenada. A concatenação e a escolha são operações associativas, de modo que podemos interpretar a escolha $p_1 \,/\, p_2 \,/\, p_3$ tanto como $p_1 \,/\, (p_2 \,/\, p_3)$ quanto como $(p_1 \,/\, p_2) \,/\, p_3$ sem alterar o seu resultado. Por motivos que veremos mais adiante, iremos considerar uma associatividade mais à direita para a concatenação e para a escolha.

Seguindo Ford [Ford, 2004], esse conjunto básico de expressões de parsing que apresentamos não contém algumas expressões que vimos na figura 1.1, tais como $\mathbf{.}$, $p?$, $p+$, e $\&p$, pois estas expressões podem ser vistas como um açúcar sintático. A seguir discutimos como podemos reescrever alguns dos açúcares sintáticos que aparecem na figura 1.1.

A expressão de parsing $\mathbf{.}$ casa qualquer terminal da entrada e pode ser reescrita como uma escolha ordenada de todos os elementos do conjunto de terminais.

Uma expressão da forma $\texttt{"}a_1 a_2 \cdots a_n\texttt{"}$, que casa cada terminal a_i em sequência, pode ser reescrita como uma concatenação $a_1\, a_2\, \cdots\, a_n$, enquanto que uma classe de caracteres $[a_1 a_2 \cdots a_n]$, que casa algum caractere a_i, pode ser reescrita como uma escolha ordenada $a_1 \,/\, a_2 \,/\, \cdots \,/\, a_n$.

A expressão de parsing $p?$ representa um casamento opcional e pode ser reescrita como $p \,/\, \varepsilon$. Já a expressão $p+$, que casa p uma ou mais vezes, pode ser reescrita como $p\, p^*$.

Por fim, a expressão de parsing $\&p$ é um predicado que casa quando p casa e falha quando o casamento de p falha. A expressão $\&p$ pode ser reescrita como $!!p$.

Ao longo do texto, usaremos G para representar uma gramática e p para representar uma expressão de parsing. Letras minúsculas do início do alfabeto, como a e b, serão usadas para representar terminais; letras minúsculas do final do alfabeto, como x e y, para representar cadeias (possivelmente vazias) de terminais; e letras maiúsculas do início do alfabeto, como A e B, para representar não terminais.

Dada uma PEG $G = (V, T, P, p_S)$, usaremos o termo produção ao nos referirmos a um par $(A, p) \in P$. Iremos representar uma produção (A, p) como $A \rightarrow p$.

Dada uma gramática $G = (V, T, P, p_S)$, usaremos a notação $G[p'_S]$ para representar uma nova gramática $G' = (V, T, P, p'_S)$, ou seja, G' é uma gramática com o mesmo conjunto de terminais, não terminais e produções de G, mas que possui p'_S como expressão de parsing inicial.

2.2 Interpretação de PEGs Usando a Relação $\Rightarrow_G$

Na formalização original de PEGs [Ford, 2004], o significado de uma gramática G é dado pela relação $\Rightarrow_G$, que é baseada no trabalho anterior de Birman [Birman, 1970, Birman e Ullman, 1973]. Dada uma gramática $G = (V, T, P, p_S)$, a relação $\Rightarrow_G$ associa pares da forma (p, x) com pares da forma (n, o), onde p é uma expressão de parsing, x é uma cadeia de entrada,

$n \geq 0$ é um contador de passos, e $o \in (T^* \cup \{\texttt{fail}\})$ representa o resultado de um casamento. Quando um casamento é bem sucedido, a saída o é um prefixo da entrada x. Caso contrário, a saída é `fail`.

Ford define a relação $\Rightarrow_G$ indutivamente da seguinte maneira:

1. **Cadeia Vazia:** $(\varepsilon,\, x) \Rightarrow_G (1,\, \varepsilon)$, para qualquer x.
2. **Terminal (caso de sucesso):** $(a,\, ax) \Rightarrow_G (1,\, a)$.
3. **Terminal (caso de falha):** $(a,\, bx) \Rightarrow_G (1,\, \texttt{fail})$ se $a \neq b$, e $(a,\, \varepsilon) \Rightarrow_G (1,\, \texttt{fail})$.
4. **Variável:** $(A,\, x) \Rightarrow_G (n+1,\, o)$ se $A \rightarrow p \in P$ e $(p,\, x) \Rightarrow_G (n,\, o)$.
5. **Concatenação (caso de sucesso):** se $(p_1,\, xyz) \Rightarrow_G (n_1,\, x)$ e $(p_2,\, yz) \Rightarrow_G (n_2,\, y)$, então $(p_1\, p_2,\, xyz) \Rightarrow_G (n_1 + n_2 + 1,\, xy)$.
6. **Concatenação (caso de falha 1):** se $(p_1,\, x) \Rightarrow_G (n_1,\, \texttt{fail})$, então $(p_1\, p_2,\, x) \Rightarrow_G (n_1 + 1,\, \texttt{fail})$.
7. **Concatenação (caso de falha 2):** se $(p_1,\, xy) \Rightarrow_G (n_1,\, x)$ e $(p_2,\, y) \Rightarrow_G (n_2,\, \texttt{fail})$, então $(p_1\, p_2,\, xy) \Rightarrow_G (n_1 + n_2 + 1,\, \texttt{fail})$.
8. **Escolha Ordenada (caso 1):** se $(p_1,\, xy) \Rightarrow_G (n_1,\, x)$, então $(p_1 \,/\, p_2,\, xy) \Rightarrow_G (n_1 + 1,\, x)$.
9. **Escolha Ordenada (caso 2):** se $(p_1,\, x) \Rightarrow_G (n_1,\, \texttt{fail})$ e $(p_2,\, x) \Rightarrow_G (n_2,\, o)$, então $(p_1 \,/\, p_2,\, x) \Rightarrow_G (n_1 + n_2 + 1,\, o)$.
10. **Repetição (caso de repetição):** se $(p,\, xyz) \Rightarrow_G (n_1,\, x)$ e $(p^*,\, yz) \Rightarrow_G (n_2,\, y)$, então $(p^*,\, xyz) \Rightarrow_G (n_1 + n_2 + 1,\, xy)$.
11. **Repetição (caso de terminação):** se $(p,\, x) \Rightarrow_G (n_1,\, \texttt{fail})$, então $(p^*,\, x) \Rightarrow_G (n_1 + 1,\, \varepsilon)$.
12. **Predicado de Negação (caso 1):** se $(p,\, xy) \Rightarrow_G (n,\, x)$, então $(!p,\, xy) \Rightarrow_G (n+1,\, \texttt{fail})$.
13. **Predicado de Negação (caso 2):** se $(p,\, x) \Rightarrow_G (n,\, \texttt{fail})$, então $(!p,\, x) \Rightarrow_G (n+1,\, \varepsilon)$.

A relação $\Rightarrow_G$ é então usada para definir uma nova relação $\Rightarrow_G^+$, que associa pares da forma $(p,\, x)$ com uma saída o, onde a definição de $\Rightarrow_G^+$ é dada a seguir:

$(p,\, x) \Rightarrow_G^+ o$ se e somente se existe um n tal que $(p,\, x) \Rightarrow_G (n,\, o)$

De acordo com Ford, podemos definir a linguagem de uma PEG da seguinte maneira:

Definição 2.2.1. *Dada uma PEG* $G = (V, T, P, p_S)$*, a sua linguagem consiste das cadeias* xy *tais que* $(p_S, xy) \Rightarrow_G^+ x$.

Ao longo do texto, usaremos $L(G)$ para representar a linguagem de uma PEG G.

Pela definição anterior, dada uma PEG G, se o casamento de w é bem sucedido em G, então $w \in L(G)$, mesmo que somente um prefixo de w tenha sido consumido nesse casamento.

2.3 Formalização de PEGs Usando Semântica Natural

Ao contrário de Ford, que usou as relações $\Rightarrow_G$ e $\Rightarrow_G^+$, usaremos a relação $\overset{\text{PEG}}{\rightsquigarrow}$ para dar significado a uma PEG. A definição de $\overset{\text{PEG}}{\rightsquigarrow}$ usa semântica natural, que é mais adequada para estudar a correspondência de PEGs com expressões regulares e CFGs pois torna mais explícitas as diferenças e semelhanças entre esses formalismos. Nos próximos capítulos da tese, iremos apresentar formalizações de expressões regulares e CFGs que também usam semântica natural e iremos estabelecer a correspondência de PEGs com expressões regulares e CFGs $LL(k)$-forte.

Definimos $\overset{\text{PEG}}{\rightsquigarrow}$ como uma relação $(G \times T^*) \times (T^* \cup \{\texttt{fail}\})$, onde $\overset{\text{PEG}}{\rightsquigarrow}$ relaciona uma gramática G e uma entrada xy ou com um sufixo y da entrada ou com `fail`. Usamos a notação $G\ xy \overset{\text{PEG}}{\rightsquigarrow} X$, onde $X \in (T^* \cup \{\texttt{fail}\})$, para indicar que $((G, xy), X) \in \overset{\text{PEG}}{\rightsquigarrow}$. A figura 2.1 apresenta a definição da relação $\overset{\text{PEG}}{\rightsquigarrow}$ usando semântica natural.

Uma diferença entre as relações $\Rightarrow_G$ e $\overset{\text{PEG}}{\rightsquigarrow}$ é que a relação $\Rightarrow_G$ nos dá como resultado de um casamento bem sucedido um prefixo da entrada e um contador de passos, enquanto que a relação $\overset{\text{PEG}}{\rightsquigarrow}$ nos dá um sufixo da entrada e uma árvore de prova. O fato da relação $\overset{\text{PEG}}{\rightsquigarrow}$ nos dar um prefixo da entrada ao invés de um sufixo não é relevante, pois ela poderia ser facilmente modificada para nos dar um sufixo da entrada. Já a ausência do contador de passos em $\overset{\text{PEG}}{\rightsquigarrow}$ se deve ao fato de que a altura da árvore de prova dada por essa relação pode ser usada como uma medida da complexidade de um casamento, o que torna o contador de passos desnecessário.

A seguir, discutimos as regras de $\overset{\text{PEG}}{\rightsquigarrow}$ e a sua correspondência com as regras da relação $\Rightarrow_G$.

A regra *empty*.1 trata do caso em que a expressão de parsing inicial da gramática representa a cadeia vazia. O casamento dessa expressão de parsing

Cadeia Vazia $\dfrac{}{G[\varepsilon]\ x \overset{\text{PEG}}{\rightsquigarrow} x}\,(\textbf{empty.1})$ **Variável** $\dfrac{G[P(A)]\ x \overset{\text{PEG}}{\rightsquigarrow} X}{G[A]\ x \overset{\text{PEG}}{\rightsquigarrow} X}\,(\textbf{var.1})$

Terminal $\dfrac{}{G[a]\ ax \overset{\text{PEG}}{\rightsquigarrow} x}\,(\textbf{char.1})$ $\dfrac{}{G[b]\ ax \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}}\,, b \neq a\,(\textbf{char.2})$ $\dfrac{}{G[a]\ \varepsilon \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}}\,(\textbf{char.3})$

Concatenação $\dfrac{G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y \quad G[p_2]\ y \overset{\text{PEG}}{\rightsquigarrow} X}{G[p_1\,p_2]\ xy \overset{\text{PEG}}{\rightsquigarrow} X}\,(\textbf{con.1})$ $\dfrac{G[p_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}}{G[p_1\,p_2]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}}\,(\textbf{con.2})$

Escolha Ordenada $\dfrac{G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y}{G[p_1\,/\,p_2]\ xy \overset{\text{PEG}}{\rightsquigarrow} y}\,(\textbf{ord.1})$ $\dfrac{G[p_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail} \quad G[p_2]\ x \overset{\text{PEG}}{\rightsquigarrow} X}{G[p_1\,/\,p_2]\ x \overset{\text{PEG}}{\rightsquigarrow} X}\,(\textbf{ord.2})$

Predicado de Negação $\dfrac{G[p]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}}{G[!p]\ x \overset{\text{PEG}}{\rightsquigarrow} x}\,(\textbf{not.1})$ $\dfrac{G[p]\ xy \overset{\text{PEG}}{\rightsquigarrow} y}{G[!p]\ xy \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}}\,(\textbf{not.2})$

Repetição $\dfrac{G[p]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}}{G[p^*]\ x \overset{\text{PEG}}{\rightsquigarrow} x}\,(\textbf{rep.1})$ $\dfrac{G[p]\ xyz \overset{\text{PEG}}{\rightsquigarrow} yz \quad G[p^*]\ yz \overset{\text{PEG}}{\rightsquigarrow} z}{G[p^*]\ xyz \overset{\text{PEG}}{\rightsquigarrow} z}\,(\textbf{rep.2})$

Figura 2.1: Definição da Relação $\overset{\text{PEG}}{\rightsquigarrow}$ Usando Semântica Natural

não consome nenhum terminal da entrada e sempre é bem sucedido. Essa regra corresponde à regra 1 da semântica de $\Rightarrow_G$.

As regras *char*.1, *char*.2 e *char*.3 tratam do caso em que a expressão de parsing inicial de G é um terminal. A regra *char*.1 corresponde à regra 2 da semântica de $\Rightarrow_G$, e as regras *char*.2 e *char*.3 correspondem à regra 3 de $\Rightarrow_G$.

Quando discutirmos a correspondência de PEGs com expressões regulares e CFGs lineares à direita usaremos **.** para representar uma expressão de parsing que casa qualquer terminal.

A regra *var*.1 trata do caso em que a expressão de parsing inicial é um não terminal A. Essa regra faz com que a expressão de parsing associada com A seja usada como expressão de parsing inicial da gramática. O resultado do casamento dessa expressão de parsing é então o resultado do casamento de A. A regra correspondente a *var*.1 na semântica de $\Rightarrow_G$ é a regra 4.

As regras *con*.1 e *con*.2 tratam da concatenação de expressões de parsing. No caso de uma concatenação $p_1\,p_2$, se o casamento de p_1 falha, então o casamento da concatenação falha (regra *con*.2). Se o casamento de p_1 é bem sucedido, o resultado da concatenação é o resultado do casamento de p_2 (regra *con*.1). A regra *con*.1 corresponde às regras 5 e 7 da semântica de Ford, e a regra *con*.2 corresponde à regra 6 da semântica de Ford.

$$\dfrac{\dfrac{\dfrac{}{G[a]\ bcd \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}}(\textbf{char.2}) \quad \dfrac{}{G[b]\ bcd \overset{\text{PEG}}{\rightsquigarrow} cd}(\textbf{char.1})}{G[a\ /\ b]\ bcd \overset{\text{PEG}}{\rightsquigarrow} cd}(\textbf{ord.2}) \quad \dfrac{}{G[c]\ cd \overset{\text{PEG}}{\rightsquigarrow} d}(\textbf{char.1})}{G[(a\ /\ b)\ c]\ bcd \overset{\text{PEG}}{\rightsquigarrow} d}(\textbf{con.1})$$

Figura 2.2: Exemplo de Árvore de Prova Usando a Relação $\overset{\text{PEG}}{\rightsquigarrow}$

As regras *ord*.1 e *ord*.2 tratam da escolha ordenada. No caso de uma escolha ordenada $p_1 \,/\, p_2$, se o casamento de p_1 é bem sucedido, então o casamento da escolha ordenada é bem sucedido (regra *ord*.1). Se o casamento da alternativa p_1 falha, então o resultado da escolha ordenada é o resultado do casamento da alternativa p_2 (regra *ord*.2). Temos que *ord*.1 é a regra correspondente da regra 8 de $\Rightarrow_G$, e que *ord*.2 corresponde à regra 9 de $\Rightarrow_G$.

As regras *not*.1 e *not*.2 tratam do predicado de negação. Se o casamento de uma expressão de parsing p falha, então o casamento de $!p$ é bem sucedido (regra *not*.1). De modo análogo, se o casamento de p é bem sucedido, então o casamento de $!p$ falha (regra *not*.2). O casamento de $!p$ não consome nenhum terminal da entrada. A regra *not*.1 corresponde à regra 13 de $\Rightarrow_G$, enquanto que *not*.2 corresponde à regra 12 de $\Rightarrow_G$.

Na seção 4.6.1, ao discutir a correspondência entre CFGs $LL(k)$-forte e PEGs, usaremos $\&p$ como um açúcar sintático para $!!p$. O casamento da expressão de parsing $\&p$ é bem sucedido quando o casamento de p é bem sucedido e falha quando o casamento de p falha. O casamento de $\&p$ não consome nenhum prefixo da entrada.

Por fim, as regras *rep*.1 e *rep*.2 tratam da repetição de uma expressão de parsing p. A regra *rep*.1 trata do caso em que o casamento de p falha. Nesse caso, o resultado do casamento de p^* é a entrada corrente. A regra *rep*.2 trata do caso em que p casa um prefixo da entrada. Nesse caso, o resultado do casamento de p^* é o resultado do casamento de p^* levando-se em conta o restante da entrada. A regra correspondente a *rep*.1 em $\Rightarrow_G$ é a regra 11, enquanto que a regra 10 de $\Rightarrow_G$ corresponde à regra *rep*.2.

Na figura 2.2 podemos ver o exemplo de uma árvore de prova usando as regras de relação $\overset{\text{PEG}}{\rightsquigarrow}$. Nesse exemplo, a cadeia de entrada é `bcd`, e a expressão de parsing inicial da gramática é $(a\ /\ b)\ c$. O casamento dessa expressão de parsing é bem sucedido para a entrada `bcd`, e o prefixo `bc` dessa entrada é casado.

No exemplo da figura 2.2, a gramática G não é relevante para o resultado do casamento, uma vez que a expressão de parsing inicial não possui não terminais. Quando uma expressão de parsing p não possui não terminais,

podemos omitir a gramática e representar o casamento de p simplesmente como $p\ xy \overset{\text{PEG}}{\rightsquigarrow} X$, dado que o resultado desse casamento não depende da gramática.

Dadas as definições das relações $\Rightarrow_G$ e $\overset{\text{PEG}}{\rightsquigarrow}$, podemos ver que todas as regras de $\Rightarrow_G$ possuem regras correspondentes em $\overset{\text{PEG}}{\rightsquigarrow}$ e vice versa. É possível então estabelecer uma correspondência entre um casamento em $\Rightarrow_G$ e um casamento em $\overset{\text{PEG}}{\rightsquigarrow}$, como dito a seguir:

Lema 2.3.1. *Dada uma PEG G e uma expressão de parsing p, temos que $\exists\, n \cdot (p,\, xy) \Rightarrow_G (n,\, o)$ se e somente se $G[p]\ xy \overset{\text{PEG}}{\rightsquigarrow} X$, onde $o = x \Leftrightarrow X = y$ e $o = \texttt{fail} \Leftrightarrow X = \texttt{fail}$.*

Demonstração. ($\Rightarrow$): A prova desta parte é por indução no número n dado pela relação $\Rightarrow_G$. Vamos estruturar nossa prova com base nas possíveis regras da relação $\Rightarrow_G$ que podem ter sido usadas.

Se a regra 1 foi usada, então $(\varepsilon,\, x) \Rightarrow_G (1,\, \varepsilon)$, e por *empty*.1 temos que $G[\varepsilon]\ x \overset{\text{PEG}}{\rightsquigarrow} x$.

Se a regra 2 foi usada, então $(a,\, ax) \Rightarrow_G (1,\, a)$, e por *char*.1 temos que $G[a]\ ax \overset{\text{PEG}}{\rightsquigarrow} x$.

Se a regra 3 foi usada, há dois subcasos dependendo se a entrada é ou não vazia. No primeiro subcaso, o casamento foi $(a,\, bx) \Rightarrow_G (1,\, \texttt{fail})$, onde $a \neq b$, e por *char*.2 concluímos que $G[a]\ bx \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$. No segundo subcaso, o casamento foi $(a,\, \varepsilon) \Rightarrow_G (1,\, \texttt{fail})$, e por *char*.3 concluímos que $G[a]\ \varepsilon \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$.

Se a regra 4 foi usada, temos que $(P(A),\, x) \Rightarrow_G (n,\, o)$. Pela hipótese de indução temos que $G[P(A)]\ x \overset{\text{PEG}}{\rightsquigarrow} X$, e por *var*.1 concluímos que $G[A]\ x \overset{\text{PEG}}{\rightsquigarrow} X$.

Se a regra 5 foi usada, temos que $(p_1,\, xyz) \Rightarrow_G (n_1,\, x)$ e que $(p_2,\, yz) \Rightarrow_G (n_2,\, y)$. Pela hipótese de indução temos que $G[p_1]\ xyz \overset{\text{PEG}}{\rightsquigarrow} yz$ e que $G[p_2]\ yz \overset{\text{PEG}}{\rightsquigarrow} z$, e por *con*.1 concluímos que $G[p_1\, p_2]\ xyz \overset{\text{PEG}}{\rightsquigarrow} z$.

Se a regra 6 foi usada, temos que $(p_1,\, x) \Rightarrow_G (n_1,\, \texttt{fail})$. Pela hipótese de indução temos que $G[p_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$, e por *con*.2 concluímos que $G[p_1\, p_2]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$.

Se a regra 7 foi usada, temos que $(p_1,\, xy) \Rightarrow_G (n_1,\, x)$ e que $(p_2,\, y) \Rightarrow_G (n_2,\, \texttt{fail})$. Pela hipótese de indução temos que $G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$ e que $G[p_2]\ y \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$. Assim, por *con*.2 concluímos que $G[p_1\, p_2]\ xy \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$.

Se a regra 8 foi usada, temos que $(p_1,\, xy) \Rightarrow_G (n_1,\, x)$. Pela hipótese de indução temos que $G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$, e por *ord*.1 concluímos que $G[p_1\, /\, p_2]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$.

Se a regra 9 foi usada, temos que $(p_1,\, x) \Rightarrow_G (n_1,\, \texttt{fail})$ e que $(p_2,\, x) \Rightarrow_G (n_2,\, o)$. Pela hipótese de indução temos que $G[p_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$ e que $G[p_2]\ x \overset{\text{PEG}}{\rightsquigarrow} X$, e por *ord*.2 concluímos que $G[p_1\, /\, p_2]\ x \overset{\text{PEG}}{\rightsquigarrow} X$.

Se a regra 10 foi usada, temos que $(p,\, xyz) \Rightarrow_G (n_1,\, x)$ e que $(p^*,\, yz) \Rightarrow_G (n_2,\, y)$. Pela hipótese de indução temos que $G[p]\ xyz \overset{\text{PEG}}{\rightsquigarrow} yz$ e que $G[p^*]\ yz \overset{\text{PEG}}{\rightsquigarrow} z$. Assim, por *rep*.2 concluímos que $G[p^*]\ xyz \overset{\text{PEG}}{\rightsquigarrow} z$.

Se a regra 11 foi usada, temos que $(p,\, x) \Rightarrow_G (n_1,\, \texttt{fail})$. Pela hipótese de indução temos que $G[p]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$, e por *rep*.1 concluímos que $G[p^*]\ x \overset{\text{PEG}}{\rightsquigarrow} x$.

Se a regra 12 foi usada, temos que $(p,\, xy) \Rightarrow_G (n,\, x)$. Pela hipótese de indução temos que $G[p]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$, e por *not*.2 concluímos que $G[!p]\ xy \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$.

Se a regra 13 foi usada, temos que $(p,\, x) \Rightarrow_G (n,\, \texttt{fail})$. Pela hipótese de indução temos que $G[p]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$, e por *not*.1 concluímos que $G[!p]\ x \overset{\text{PEG}}{\rightsquigarrow} x$.

($\Leftarrow$): A prova desta parte é por indução na altura da árvore de prova dada pela relação $\overset{\text{PEG}}{\rightsquigarrow}$. Vamos estruturar nossa prova com base nas possíveis regras da relação $\overset{\text{PEG}}{\rightsquigarrow}$ que podem ter sido usadas para concluir essa árvore de prova.

Se a regra *empty*.1 foi usada, então $p = \varepsilon$. Assim, pela regra 1 temos que $(\varepsilon,\, x) \Rightarrow_G (1,\, \varepsilon)$.

Se a regra *char*.1 foi usada, então $p = a$ e a entrada é da forma ax. Assim, pela regra 2 temos que $(a,\, ax) \Rightarrow_G (1,\, a)$.

Se a regra *char*.2 foi usada, então $p = a$ e a entrada é forma bx, onde $a \neq b$. Assim, pela regra 3 temos que $G[a]\ bx \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$.

Se a regra *char*.3 foi usada, então $p = a$ e a entrada é da forma ε. Assim, pela regra 3 temos que $(a,\, \varepsilon) \Rightarrow_G (1,\, \texttt{fail})$.

Se a regra *var*.1 foi usada, temos que $G[P(A)]\ x \overset{\text{PEG}}{\rightsquigarrow} X$. Pela hipótese de indução temos que $(P(A),\, x) \Rightarrow_G (n,\, o)$, e pela regra 4 concluímos que $(A,\, x) \Rightarrow_G (n+1,\, o)$.

Se a regra *con*.1 foi usada, temos que $G[p_1]\ xyz \overset{\text{PEG}}{\rightsquigarrow} yz$ e que $G[p_2]\ yz \overset{\text{PEG}}{\rightsquigarrow} X$. Pela hipótese de indução temos que $(p_1,\, xyz) \Rightarrow_G (n_1,\, x)$ e que $(p_2,\, yz) \Rightarrow_G (n_2,\, o)$. Há dois subcasos dependendo se o casamento de p_2 foi ou não bem sucedido. No primeiro subcaso, pela regra 5 concluímos que $(p_1\, p_2,\, xyz) \Rightarrow_G (n_1+n_2+1,\, xy)$. No segundo subcaso, pela regra 7 concluímos que $(p_1\, p_2,\, x) \Rightarrow_G (n_1 + n_2 + 1,\, \texttt{fail})$.

Se a regra *con*.2 foi usada, temos que $G[p_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$. Pela hipótese de indução temos que $(p_1,\, x) \Rightarrow_G (n_1,\, \texttt{fail})$, e pela regra 6 concluímos que $(p_1\, p_2,\, x) \Rightarrow_G (n_1 + 1,\, \texttt{fail})$.

Se a regra *ord*.1 foi usada, temos que $G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$. Pela hipótese de indução temos que $(p_1,\, xy) \Rightarrow_G (n_1,\, x)$, e pela regra 8 concluímos que $(p_1\, /\, p_2,\, x) \Rightarrow_G (n_1 + 1,\, x)$.

Se a regra *ord*.2 foi usada, temos que $G[p_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$ e que $G[p_2]\ x \overset{\text{PEG}}{\rightsquigarrow} X$. Pela hipótese de indução temos que $(p_1,\, x) \Rightarrow_G (n_1,\, \texttt{fail})$

e que $(p_2, x) \Rightarrow_G (n_2, o)$, e pela regra 9 concluímos que $(p_1 \,/\, p_2, x) \Rightarrow_G (n_1 + n_2 + 1, o)$.

Se a regra *rep*.1 foi usada, temos que $G[p]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$. Pela hipótese de indução temos que $(p, x) \Rightarrow_G (n_1, \texttt{fail})$, e pela regra 11 concluímos que $(p^*, x) \Rightarrow_G (n_1 + 1, \varepsilon)$.

Se a regra *rep*.2 foi usada, temos que $G[p]\ xyz \overset{\text{PEG}}{\rightsquigarrow} yz$ e que $G[p^*]\ yz \overset{\text{PEG}}{\rightsquigarrow} z$. Pela hipótese de indução sabemos que $(p, xyz) \Rightarrow_G (n_1, x)$ e que $(p^*, yz) \Rightarrow_G (n_2, y)$, e pela regra 10 concluímos que $(p^*, xyz) \Rightarrow_G (n_1 + n_2 + 1, xy)$.

Se a regra *not*.1 foi usada, temos que $G[p]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$. Pela hipótese de indução temos que $(p, x) \Rightarrow_G (n, \texttt{fail})$, e pela regra 13 concluímos que $(!p, x) \Rightarrow_G (n + 1, \varepsilon)$.

Finalmente, se a regra *not*.2 foi usada, temos que $G[p]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$. Pela hipótese de indução temos que $(p, xy) \Rightarrow_G (n, x)$, e pela regra 12 concluímos que $(!p, xy) \Rightarrow_G (n + 1, \texttt{fail})$. □

Após provar a correspondência entre as relações $\Rightarrow_G$ e $\overset{\text{PEG}}{\rightsquigarrow}$, vamos definir alguns conceitos de PEGs que são adaptações das definições dadas por Ford [Ford, 2004] e enunciar alguns lemas.

Primeiro, vamos definir os conceitos de gramática *livre de predicado* e *livre de repetição*.

Definição 2.3.1. *Uma gramática é livre de repetição se ela não possui nenhuma expressão de parsing* p^*.

Definição 2.3.2. *Uma gramática é livre de predicado se ela não possui nenhuma expressão de parsing* $!p$.

Agora, vamos definir quando uma escolha ordenada é disjunta:

Definição 2.3.3. *Dada uma PEG G, uma escolha* $p_1 \,/\, p_2$ *de G é disjunta se temos que* $G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y \Rightarrow G[p_2]\ xy \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$, *e que* $G[p_2]\ xy \overset{\text{PEG}}{\rightsquigarrow} y \Rightarrow G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$.

Como podemos ver na definição acima, quando uma escolha é disjunta o casamento de uma alternativa é bem sucedido para uma dada entrada somente quando o casamento da outra alternativa não é bem sucedido para essa entrada. Se uma escolha é disjunta, a ordem das alternativas não é relevante.

A seguir, apresentamos a nossa definição da linguagem de uma PEG:

Definição 2.3.4. *Dada uma PEG G, temos que uma cadeia* $x \in L(G)$ *se e somente se existe uma cadeia y tal que* $G\ xy \overset{\text{PEG}}{\rightsquigarrow} y$.

Dada uma PEG G, seja $L(G)$ a nossa definição da linguagem de G e seja $L^F(G)$ a definição correspondente dada por Ford, temos que as cadeias de $L(G)$ são prefixos das cadeias de $L^F(G)$.

Como podemos ver, a nossa definição de linguagem enfatiza o prefixo x da entrada que foi casado, enquanto que a definição de Ford dá uma importância maior ao sufixo y da entrada que não foi casado.

Geralmente, se mudarmos o sufixo y da entrada que não foi casado, podemos casar um prefixo diferente da entrada ou obtermos um casamento que não é bem sucedido. Por exemplo, dada uma PEG G cuja expressão de parsing inicial é $a\,!b$, temos que essa expressão casa apenas o primeiro terminal da entrada, mas o segundo terminal da entrada também é importante para determinar o resultado do seu casamento. Segundo a nossa definição, $L(G)$ seria apenas $\{\texttt{a}\}$, ao passo que $L^F(G)$ teria, além de `a`, cadeias como `aa` e `acaca`.

Há casos em que o sufixo y não é importante para determinar o resultado de um casamento. Por exemplo, dada uma PEG G cuja expressão de parsing inicial é a escolha disjunta $a \,/\, b$, podemos ver que apenas o primeiro terminal da entrada é importante para determinar o resultado do casamento dessa expressão. Nesse caso, temos que $L(G) = \{\texttt{a}, \texttt{b}\}$, enquanto que $L^F(G)$ teria cadeias como `acc` e `bcc`, onde o sufixo `cc` não é importante para determinar o resultado do casamento dessas cadeias.

A nossa definição da linguagem descrita por uma PEG é mais próxima da definição de linguagem que usaremos para expressões regulares e CFGs. No caso destes formalismos, o sufixo y não é importante para determinar o resultado de um casamento. Como veremos, quando uma CFG ou expressão regular casa um prefixo x de uma entrada xy, isso implica que para qualquer sufixo y' é possível casar o prefixo x de uma entrada da forma xy'.

Um outro conceito definido por Ford que iremos adaptar é o de gramática *completa*, que é dado a seguir:

Definição 2.3.5. *Dada uma PEG G, dizemos que G é completa se para toda cadeia w temos que G $w \overset{PEG}{\rightsquigarrow} X$, e portanto existe uma árvore de prova associada ao casamento de w em G através da relação $\overset{PEG}{\rightsquigarrow}$.*

Abaixo, temos o exemplo de uma PEG que não é completa:

$$A \rightarrow A\,a \,/\, b$$

Essa PEG não é completa pois durante o casamento de uma cadeia w, se substituirmos o não terminal A pela sua expressão de parsing associada iremos obter novamente esse mesmo não terminal A e iremos tentar casar a mesma

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots}{G[A]\ \ w \overset{\text{PEG}}{\rightsquigarrow}}}{G[A\,a]\ \ w \overset{\text{PEG}}{\rightsquigarrow}}}{G[A\,a\,/\,b]\ \ w \overset{\text{PEG}}{\rightsquigarrow}}}{G[A]\ \ w \overset{\text{PEG}}{\rightsquigarrow}}$$

Figura 2.3: Exemplo de Casamento Quando uma PEG não é Completa

entrada w. Dado que a semântica de $\overset{\text{PEG}}{\rightsquigarrow}$ é determinística, então esse processo se repetirá infinitas vezes e não irá resultar em uma árvore de prova, como podemos ver na figura 2.3:

No caso anterior, onde tentamos casar infinitas vezes o mesmo não terminal e a mesma entrada, dizemos que a gramática é *recursiva à esquerda*, ou possui produções recursivas à esquerda.

Outro exemplo de PEGs que não são completas são gramáticas que possuem expressões de parsing da forma p^*, onde p casa a cadeia vazia. Nesse caso, a expressão de parsing p pode casar a cadeia vazia infinitas vezes.

No seu artigo sobre PEGs [Ford, 2004], Ford apresenta uma relação que pode ser usada para determinar se uma gramática é ou não completa de acordo com a sua estrutura. Com base nessa relação, Ford mostrou que uma gramática é completa se ela não é recursiva à esquerda nem possui expressões p^* onde p casa a cadeia vazia.

A seguir, vamos enunciar um lema que relaciona a cadeia de entrada na parte de baixo de uma árvore de prova com a cadeia de entrada de uma subárvore:

Lema 2.3.2. *Dada uma PEG G, se existe um casamento $G\ x \overset{PEG}{\rightsquigarrow} X$ então para toda subárvore $G\ y \overset{PEG}{\rightsquigarrow} X$ desse casamento temos que y é um sufixo de x.*

Demonstração. A prova é por indução na altura da árvore de prova dada por $\overset{\text{PEG}}{\rightsquigarrow}$. Podemos ver que em todas as regras de $\overset{\text{PEG}}{\rightsquigarrow}$ este lema é verdadeiro. □

Por sua vez, o lema a seguir relaciona a cadeia resultante na parte de baixo de uma árvore de prova com a cadeia resultante de uma subárvore:

Lema 2.3.3. *Dada uma PEG G, se existe um casamento $G\ x \overset{PEG}{\rightsquigarrow} x'$ então para toda subárvore $G\ y \overset{PEG}{\rightsquigarrow} y'$ desse casamento temos que x' é um sufixo de y'.*

Demonstração. A prova é por indução na altura da árvore de prova dada por $\overset{\text{PEG}}{\rightsquigarrow}$. Podemos ver que em todas as regras de $\overset{\text{PEG}}{\rightsquigarrow}$ este lema é verdadeiro. □

No próximo capítulo, discutiremos a correspondência entre expressões regulares e PEGs, e no capítulo 4 abordaremos a correspondência entre CFGs lineares à direita e $LL(k)$-forte e PEGs.