4
Gramáticas Livres de Contexto e PEGs

Este capítulo apresenta uma nova formalização de Gramáticas Livres de Contexto (*Context-Free Grammars* —CFGs) e discute a correspondência de CFGs lineares à direita e $LL(k)$-forte com PEGs equivalentes.

Na próxima seção mostramos a definição usual de CFGs e na seção 4.2 apresentamos a nossa formalização de CFGs baseada em semântica natural, que é próxima da formalização de PEGs apresentada no capítulo 2. Em seguida, na seção 4.3, discutimos a interpretação de uma mesma gramática como uma CFG e como uma PEG. A seção 4.4 discute a correspondência entre CFGs lineares à direita e PEGs equivalentes. Na seção 4.5 abordamos a correspondência entre CFGs $LL(1)$ e PEGs, e mostramos que uma gramática $LL(1)$, com uma pequena restrição na ordem das alternativas de uma escolha, define a mesma linguagem quando interpretada como uma CFG e quando interpretada como uma PEG. A seção 4.6 estende essa discussão para gramáticas $LL(k)$-forte, e mostra que a partir de uma CFG $LL(k)$-forte podemos gerar uma PEG equivalente que possui a mesma estrutura da CFG.

4.1
Definição Usual de CFGs

Uma CFG é geralmente definida como uma tupla (V, T, P, S), onde V é um conjunto finito de variáveis ou não terminais, T é um conjunto finito de terminais, P é um conjunto finito de produções, onde uma produção é um par (A, α) que relaciona um não terminal $A \in V$ com uma cadeia α de símbolos da gramática (um símbolo da gramática pode ser um terminal ou um não terminal) e é representada como $A \rightarrow \alpha$, e S representa o não terminal inicial da gramática [Hopcroft e Ullman, 1979].

Dada uma CFG $G = (V, T, P, S)$, usamos a relação $\Rightarrow_G$ para substituir um não terminal A por uma cadeia β de símbolos da gramática, onde $A \rightarrow \beta \in P$. Dessa forma, se α e γ são cadeias de símbolos da gramática e $A \rightarrow \beta \in P$, então $\alpha A \gamma \Rightarrow_G \alpha\beta\gamma$.

A relação $\overset{*}{\Rightarrow}_G$ é o fecho reflexivo transitivo de $\Rightarrow_G$. Dada uma cadeia α de símbolos da gramática, dizemos que α gera (ou deriva) β se temos que

$\alpha \overset{*}{\Rightarrow}_G \beta$.

Dizemos que uma cadeia w pertence à linguagem de uma gramática G quando $S \overset{*}{\Rightarrow}_G w$.

4.2 Uma Nova Formalização de Linguagens Livres de Contexto usando Semântica Natural

Apresentaremos nesta seção uma nova formalização de Linguagens Livres de Contexto (*Context-Free Languages* — CFLs) que é baseada em semântica natural. Essa nova formalização de CFLs usa expressões de parsing e é próxima da formalização de PEGs apresentada no capítulo 2. O uso de uma formalização de CFLs que é próxima da formalização de PEGs nos ajudará a estabelecer a correspondência entre classes de linguagens definidas através de PEGs e através de CFGs.

Assim como em CFGs, na nova formalização de CFLs a descrição de uma linguagem é dada por uma tupla (V, T, P, p_S), onde V é um conjunto finito de não terminais, 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.

Diferentemente de CFGs, onde uma produção associa um não terminal com uma cadeia de símbolos da gramática, nessa nova formalização de CFLs temos uma função P que relaciona não terminais com expressões de parsing. Apesar dessa diferença, a descrição visual de algumas linguagens é bastante similar nos dois formalismos. Como exemplo, vamos analisar a descrição a seguir, que está na Forma de Backus-Naur (*Backus-Naur Form* — BNF) [Knuth, 1964],

$$A \rightarrow B\,C \qquad B \rightarrow a \,|\, b \qquad C \rightarrow c \,|\, d$$

Se a descrição acima é de uma CFG, temos o seguinte conjunto de produções:

$$P \;=\; \{\, A \rightarrow B\,C,\;\; B \rightarrow a,\;\; B \rightarrow b,\;\; C \rightarrow c,\;\; C \rightarrow d \,\}$$

Nesse caso, o símbolo | é apenas uma convenção sintática usada para separar cadeias de símbolos da gramática associadas a um mesmo não terminal.

No caso em que a descrição anterior é baseada em expressões de parsing, temos a seguinte definição da função P:

$$P(A) \;=\; B\,C \qquad P(B) \;=\; a \,|\, b \qquad P(C) \;=\; c \,|\, d$$

Nesse caso, o símbolo | é um operador binário.

Para simplificar a descrição de CFLs através de expressões de parsing, vamos assumir que a escolha e a concatenação são associativas à direita, de modo que representaremos como $p_1 \mid p_2 \mid \cdots \mid p_m$ a expressão de parsing $p_1 \mid (p_2 \mid (\cdots \mid p_m))$, e como $p_1\, p_2 \cdots p_n$ a expressão de parsing $p_1\, (p_2\, (\cdots\, p_n))$.

Apesar de usarmos expressões de parsing na nova formalização de CFLs, não daremos significado a expressões de parsing da forma $!p$, pois com predicados poderíamos definir linguagens que não são CFLs, e da forma p^*, pois ela não é necessária para definir CFLs.

Dada a descrição de uma CFL através de expressões de parsing é possível obter uma CFG, assim como a partir de uma CFG podemos obter a descrição de uma CFL através de expressões de parsing. A primeira dessas transformações é mais complicada, de modo que iremos discuti-la a seguir.

Uma vez que na nova formalização de CFLs temos uma expressão de parsing inicial p_S ao invés de um não terminal, no caso em que $p_S \notin V$ devemos modificar a descrição da CFL. Primeiro, adicionamos um novo não terminal A ao conjunto V. Segundo, adicionamos a produção $A \rightarrow p_S$. Finalmente, tornamos A a expressão de parsing inicial.

Feito isso, o próximo passo é eliminar as escolhas que são subexpressões de concatenações. Dado que é possível agrupar expressões de parsing com o uso de parênteses, podemos ter uma expressão de parsing em que uma concatenação possui uma escolha como subexpressão, como mostrado a seguir:

$$A \rightarrow (a \mid b)\, (c \mid d)$$

Acima, o não terminal A está associado a uma conjunção de disjunções, isto é, uma concatenação de escolhas. Para obtermos uma descrição baseada em expressões de parsing que não possui uma escolha como subexpressão de uma concatenação, devemos realizar uma transformação similar a que fazemos para converter uma descrição na Forma de Backus-Naur Estendida (*Extended Backus-Naur Form* — EBNF) [Wirth, 1977, ISO] em uma CFG. Durante essa transformação, devemos criar um novo não terminal para cada escolha que é subexpressão de uma concatenação. A seguir, reescrevemos o exemplo anterior após essa transformação:

$$A \rightarrow B\, C \qquad B \rightarrow a \mid b \qquad C \rightarrow c \mid d$$

Notem que usamos novos não terminais B e C para eliminar a conjunção de disjunções.

Embora intuitivamente as transformações anteriores sejam simples, a

formalização delas é trabalhosa. Como essa formalização não possui aspectos interessantes, preferimos discutir essas transformações somente de maneira informal.

Dado que visualmente a descrição de CFLs através de CFGs é praticamente idêntica à descrição de CFLs através de expressões de parsing, e que, como discutimos acima, é possível transformar uma descrição na outra, vamos cometer um abuso de linguagem e usar daqui em diante o termo CFG (ou gramática) ao nos referirmos à descrição de uma CFL através de expressões de parsing. Quando for necessário fazer uma distinção entre essas duas formalizações de CFLs, iremos usar os termos CFG usual e CFG descrita através de expressões de parsing.

A interpretação de uma CFG $G = (V, T, P, S)$ descrita através de expressões de parsing é feita pela relação $\overset{\text{CFG}}{\rightsquigarrow}$, cuja definição usa semântica natural e é apresentada na figura 4.1. De modo análogo a PEGs, dada uma CFG $G = (V, T, P, p_S)$, usamos a notação $G[p'_S]$ para representar uma nova gramática $G' = (V, T, P, p'_S)$.

Na figura 4.1, podemos ver que dada uma gramática e uma cadeia de entrada, a relação $\overset{\text{CFG}}{\rightsquigarrow}$ casa um prefixo da entrada de acordo com as produções da gramática. De maneira mais precisa, definimos $\overset{\text{CFG}}{\rightsquigarrow}$ como uma relação $(G \times T^*) \times T^*$, onde $\overset{\text{CFG}}{\rightsquigarrow}$ relaciona uma gramática G e uma entrada xy com um sufixo y da entrada. Usamos a notação $G\ xy \overset{\text{CFG}}{\rightsquigarrow} y$ para representar que $((G, xy), y) \in \overset{\text{CFG}}{\rightsquigarrow}$.

A relação $\overset{\text{CFG}}{\rightsquigarrow}$ não é uma função, pois ela não relaciona cada par (G, w) com um sufixo de w, e ela também pode relacionar um par (G, w) com diferentes sufixos de w.

A seguir, definimos a linguagem de uma CFG descrita através de expressões de parsing:

Definição 4.2.1. *Dada uma CFG G descrita através de expressões de parsing, a linguagem de G consiste das cadeias x tais que $G\ xy \overset{\text{CFG}}{\rightsquigarrow} y$, onde y é uma cadeia qualquer.*

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

Note na definição anterior que o sufixo y pode ser qualquer cadeia, pois não temos predicados em CFGs e portanto y não é usado para determinar o resultado de um casamento.

A seguir, vamos discutir as linguagens de uma CFG usual e da sua formalização correspondente descrita através de expressões de parsing.

Como vimos anteriormente, a partir de uma CFG $G = (V, T, P, S)$ descrita de forma usual é possível obter uma gramática $G' = (V, T, P', S)$

Cadeia Vazia $\dfrac{}{G[\varepsilon]\ x \overset{\text{CFG}}{\rightsquigarrow} x}\,(\textbf{empty.1})$ **Terminal** $\dfrac{}{G[a]\ ax \overset{\text{CFG}}{\rightsquigarrow} x}\,(\textbf{char.1})$

Variável $\dfrac{G[P(A)]\ xy \overset{\text{CFG}}{\rightsquigarrow} y}{G[A]\ xy \overset{\text{CFG}}{\rightsquigarrow} y}\,(\textbf{var.1})$ **Concatenação** $\dfrac{G[p_1]\ xyz \overset{\text{CFG}}{\rightsquigarrow} yz \quad G[p_2]\ yz \overset{\text{CFG}}{\rightsquigarrow} z}{G[p_1\,p_2]\ xyz \overset{\text{CFG}}{\rightsquigarrow} z}\,(\textbf{con.1})$

Escolha $\dfrac{G[p_1]\ xy \overset{\text{CFG}}{\rightsquigarrow} y}{G[p_1 \mid p_2]\ xy \overset{\text{CFG}}{\rightsquigarrow} y}\,(\textbf{choice.1})$ $\dfrac{G[p_2]\ xy \overset{\text{CFG}}{\rightsquigarrow} y}{G[p_1 \mid p_2]\ xy \overset{\text{CFG}}{\rightsquigarrow} y}\,(\textbf{choice.2})$

Figura 4.1: Definição da Relação $\overset{\text{CFG}}{\rightsquigarrow}$ Usando Semântica Natural

descrita através de expressões de parsing. Além disso, dada a definição de $\overset{\text{CFG}}{\rightsquigarrow}$ podemos afirmar que $S \Rightarrow_G x$ se e somente se $G'\ xy \overset{\text{CFG}}{\rightsquigarrow} y$.

Para provar formalmente essa correspondência precisamos formalizar a transformação de G em G', o que não iremos fazer. Contudo, dadas as regras da relação $\overset{\text{CFG}}{\rightsquigarrow}$, acreditamos que não é difícil ver que essa correspondência é verdadeira.

Uma consequência da correspondência entre $\Rightarrow_G$ e $\overset{\text{CFG}}{\rightsquigarrow}$ é que w pertence à linguagem de G se e somente se $w \in L(G')$.

Após mostrar a correspondência entre $\Rightarrow_G$ e $\overset{\text{CFG}}{\rightsquigarrow}$, vamos discutir quando uma gramática descrita através de expressões de parsing possui estrutura BNF e enunciar algumas propriedades de CFGs descritas através de expressões de parsing. Mais adiante, na seção 4.3, vamos estudar a correspondência entre as relações $\overset{\text{CFG}}{\rightsquigarrow}$ e $\overset{\text{PEG}}{\rightsquigarrow}$.

4.2.1 Estrutura BNF

A seguir, definimos quando uma CFG descrita através de expressões de parsing possui estrutura BNF:

Definição 4.2.2. *Uma CFG $G = (V, T, P, p_S)$ descrita através de expressões de parsing possui estrutura BNF se ela está de acordo com as seguintes restrições:*

1. *Nenhuma escolha de G é uma subexpressão de uma concatenação.*
2. *$p_S \in V$*
3. *Para toda escolha $p_1 \mid p_2$ de G onde uma alternativa casa a cadeia vazia e a outra não, temos que a alternativa p_2 é a que casa a cadeia vazia.*

A restrição 1 já foi discutida anteriormente, quando mostramos como transformar uma CFG descrita através de expressões em uma CFG usual. Se uma gramática G respeita essa restrição, ela possui uma representação direta na forma BNF.

A segunda restrição diz respeito à expressão de parsing inicial, que deve ser um não terminal. Essa restrição será útil ao provarmos a correspondência entre CFGs $LL(1)$ e $LL(k)$-forte e PEGs.

Por fim, a restrição 3 exige que quando somente uma alternativa de uma escolha casa a cadeia vazia, essa alternativa seja a última. Essa restrição não afeta a linguagem da CFG, pois a linguagem de uma CFG não muda quando na sua descrição BNF alteramos a ordem das alternativas de uma escolha. Apesar dessa restrição parecer um pouco artificial, veremos mais adiante que ela simplifica a discussão da correspondência entre CFGs $LL(1)$ com expressões ε e PEGs.

4.2.2 Propriedades de CFGs

Iremos definir dois lemas a respeito do casamento em uma gramática usando a semântica de $\overset{\text{CFG}}{\rightsquigarrow}$. O primeiro desses lemas possui lemas correspondentes em PEGs, mas o segundo não.

Lema 4.2.1. *Dada uma CFG G, se existe um casamento $G\ x \overset{\text{CFG}}{\rightsquigarrow} x'$ então para toda subárvore $G\ y \overset{\text{CFG}}{\rightsquigarrow} y'$ desse casamento temos que y é um sufixo de x e que x' é um sufixo de y'.*

Demonstração. A prova é por indução na altura da árvore de prova dada por $\overset{\text{CFG}}{\rightsquigarrow}$. Vamos provar que o lema é verdadeiro nos antecedentes (subárvores próprias) de todas as regras de $\overset{\text{CFG}}{\rightsquigarrow}$.

Quando a expressão de parsing é da forma ε ou da forma a, como as regras *empty*.1 e *char*.1 não possuem antecedentes o lema é trivialmente satisfeito.

Quando a expressão de parsing é da forma A, pela regra *var*.1 temos que o antecedente é $G[P(A)]\ y \overset{\text{CFG}}{\rightsquigarrow} y'$, onde $y = x$ e $y' = x'$, e pela hipótese de indução concluímos que o lema é verdadeiro nessa subárvore.

Quando a expressão de parsing é da forma $p_1\, p_2$, seja $x = tuv$, pela regra *con*.1 temos as subárvores $G[p_1]\ tuv \overset{\text{CFG}}{\rightsquigarrow} uv$ e $G[p_2]\ uv \overset{\text{CFG}}{\rightsquigarrow} v$. No caso da primeira subárvore, temos que tuv é um sufixo de tuv, e que v é um sufixo de uv, e pela hipótese de indução o lema é verdadeiro nas suas subárvores. Na caso da segunda subárvore, temos que uv é um sufixo de tuv, e que v é um sufixo de v, e pela hipótese de indução o lema é verdadeiro nas suas subárvores.

Quando a expressão de parsing é da forma $p_1 \mid p_2$, as regras associadas são *choice*.1 e *choice*.2. Se a regra *choice*.1 foi usada, temos uma subárvore

$G[p_1]\ \ y \overset{\text{CFG}}{\rightsquigarrow} y'$, onde $y = x$ e $y' = x'$, e pela hipótese de indução o lema é verdadeiro nas suas subárvores. O caso em que a regra *choice*.2 foi usada é análogo. □

O lema anterior é o correspondente para CFGs dos lemas 2.3.2 e 2.3.3, que são a respeito de PEGs. Por outro lado, o próximo lema não possui um lema correspondente em PEGs, por razões que discutiremos logo em seguida:

Lema 4.2.2. *Dada uma CFG G, temos que se $G\ \ xy \overset{CFG}{\rightsquigarrow} y$ então $\forall y' \cdot G\ \ xy' \overset{CFG}{\rightsquigarrow} y'$.*

Demonstração. A prova é por indução na altura da árvore de prova dada por $\overset{\text{CFG}}{\rightsquigarrow}$.

No caso de uma concatenação $p_1\, p_2$, somente a regra *con*.1 pode ter sido usada. Por essa regra sabemos que $G[p_1]\ \ x_1x_2y \overset{\text{CFG}}{\rightsquigarrow} x_2y$ e que $G[p_2]\ \ x_2y \overset{\text{CFG}}{\rightsquigarrow} y$, onde $x = x_1x_2$. Pela hipótese de indução temos que $G[p_1]\ \ x_1x_2y' \overset{\text{CFG}}{\rightsquigarrow} x_2y'$ e que $G[p_2]\ \ x_2y' \overset{\text{CFG}}{\rightsquigarrow} y'$. Assim, pela regra *con*.1 concluímos que $G[p_1\, p_2]\ \ x_1x_2y' \overset{\text{CFG}}{\rightsquigarrow} y'$.

A prova dos outros casos é similar. □

Dada uma PEG G, onde G é livre de predicado, em um primeiro momento poderíamos pensar em adaptar o lema 4.2.2 para PEGs. Contudo, essa adaptação do lema 4.2.2 não é possível. Para ver a razão disso, vamos considerar a PEG a seguir, onde S é a expressão de parsing inicial da gramática:

$$S \rightarrow A\,B \qquad A \rightarrow a\,b\,a \,/\, a \qquad B \rightarrow b$$

Dada a entrada `abc`, temos o casamento $G\ \ abc \overset{\text{PEG}}{\rightsquigarrow} c$, com subárvores $G[A]\ \ abc \overset{\text{PEG}}{\rightsquigarrow} bc$ e $G[B]\ \ bc \overset{\text{PEG}}{\rightsquigarrow} c$. Dada uma nova entrada `abac`, temos o casamento $G\ \ abac \overset{\text{PEG}}{\rightsquigarrow}$ `fail`, com subárvores $G[A]\ \ abac \overset{\text{PEG}}{\rightsquigarrow} c$ e $G[B]\ \ c \overset{\text{PEG}}{\rightsquigarrow}$ `fail`. Como podemos ver, ao mudarmos um sufixo da entrada, o não terminal A passou a casar um prefixo diferente, o que fez com que o casamento de B falhasse e com que os casamentos de $A\,B$ e de S também falhassem.

Em resumo, no caso de expressões regulares e CFGs os lemas 3.2.3 e 4.2.2, nos dizem que quando uma expressão regular ou CFG casa um prefixo x da entrada e deixa um sufixo y, podemos mudar esse sufixo y da entrada e ainda casar o mesmo prefixo x. Já no caso de PEGs, com base na discussão anterior podemos concluir que quando uma PEG casa um prefixo x da entrada e deixa um sufixo y, isso não implica que podemos mudar esse sufixo y para y' e obter um casamento bem sucedido para a entrada xy'.

4.3 Correspondência entre $\overset{\text{CFG}}{\rightsquigarrow}$ e $\overset{\text{PEG}}{\rightsquigarrow}$

Com base na definição de $\overset{\text{CFG}}{\rightsquigarrow}$ apresentada na seção anterior e na definição de $\overset{\text{PEG}}{\rightsquigarrow}$ apresentada na seção 2.3, vamos comparar as regras de $\overset{\text{CFG}}{\rightsquigarrow}$ com as regras de $\overset{\text{PEG}}{\rightsquigarrow}$ para ver as semelhanças e as diferenças entre CFGs e PEGs.

Podemos notar que somente $\overset{\text{PEG}}{\rightsquigarrow}$ possui regras onde o resultado de um casamento é `fail`. As regras de $\overset{\text{PEG}}{\rightsquigarrow}$ usam `fail` porque, como discutimos no capítulo 2, a noção de falha é importante em PEGs para definir a escolha ordenada e o predicado de negação. Sem a noção de falha, seria mais difícil expressar essas definições. Uma outra maneira de expressá-las seria através do uso de um quantificador existencial. Nesse caso, dada uma gramática G e uma cadeia x, expressaríamos a noção de falha como $\nexists x' \preceq x \cdot G\ x \overset{\text{CFG}}{\rightsquigarrow} x'$.

Embora o uso de `fail` seja essencial em $\overset{\text{PEG}}{\rightsquigarrow}$, ele não é apropriado nas regras de $\overset{\text{CFG}}{\rightsquigarrow}$, uma vez que em CFGs a escolha não é ordenada e não há predicados. Se tentássemos usar a noção de falha na definição das regras de $\overset{\text{CFG}}{\rightsquigarrow}$, o fato da escolha não ser ordenada resultaria em uma semântica onde um casamento poderia ser bem sucedido e falhar para uma mesma entrada. Nessa semântica, a noção de falha seria de pouca utilidade, pois quando o resultado do casamento de uma dada entrada é `fail`, isso não significa que não há um casamento bem sucedido para essa entrada.

Se analisarmos as regras de $\overset{\text{CFG}}{\rightsquigarrow}$ e de $\overset{\text{PEG}}{\rightsquigarrow}$ podemos ver que as regras *empty*.1, *char*.1 e *choice*.1 de $\overset{\text{CFG}}{\rightsquigarrow}$ são, respectivamente, idênticas às regras *empty*.1, *char*.1 e *ord*.1 de $\overset{\text{PEG}}{\rightsquigarrow}$, e que as regras *var*.1 e *con*.1 de $\overset{\text{CFG}}{\rightsquigarrow}$ diferem das regras *var*.1 e *con*.1 de $\overset{\text{PEG}}{\rightsquigarrow}$ somente pelo fato de que o resultado destas duas últimas regras também pode ser `fail`. Assim, podemos concluir que a principal diferença entre $\overset{\text{CFG}}{\rightsquigarrow}$ e $\overset{\text{PEG}}{\rightsquigarrow}$ está nas regras *choice*.2 e *ord*.2.

Como é possível usar a regra *choice*.2 em um casamento mesmo quando há um casamento bem sucedido através da regra *choice*.1, temos que o casamento usando a semântica de $\overset{\text{CFG}}{\rightsquigarrow}$ é não determinístico. Assim, dada uma gramática G e uma entrada x, o resultado do casamento de x em G usando $\overset{\text{CFG}}{\rightsquigarrow}$ pode não ser único.

Por outro lado, como a regra *ord*.2 só pode ser usada em um casamento quando não há um casamento bem sucedido através da regra *ord*.1, temos que o casamento usando a semântica de $\overset{\text{PEG}}{\rightsquigarrow}$ é determinístico. Assim, dada uma gramática G e uma entrada x, o resultado do casamento de x em G usando $\overset{\text{PEG}}{\rightsquigarrow}$ é sempre o mesmo.

4.3.1 Interpretação de uma Gramática como uma CFG e como uma PEG

Uma vez que a descrição de uma CFG é muito próxima da descrição de uma PEG, dada uma gramática G, podemos interpretá-la usando a semântica de $\overset{\text{CFG}}{\rightsquigarrow}$ ou a semântica de $\overset{\text{PEG}}{\rightsquigarrow}$.

A única diferença entre a descrição de uma CFG e de uma PEG está na notação usada para representar uma escolha. Como norma geral, daqui em diante usaremos a notação $p_1 \mid p_2$ para representar uma escolha nos dois formalismos. Contudo, quando desejarmos enfatizar que uma gramática está sendo interpretada como uma PEG, usaremos a notação $p_1 \,/ p_2$ para representar uma escolha.

Dada uma gramática G, vamos usar a notação $L^{CFG}(G)$ para representar a linguagem que G define quando interpretada como uma CFG. De modo análogo, usaremos a notação $L^{PEG}(G)$ para representar a linguagem que G define quando interpretada como uma PEG.

Dado que a semântica de $\overset{\text{CFG}}{\rightsquigarrow}$ é diferente da semântica de $\overset{\text{PEG}}{\rightsquigarrow}$, geralmente uma gramática G interpretada como uma CFG não define a mesma linguagem que ela define quando interpretada como uma PEG. Contudo, dada uma gramática G que é livre de repetição e livre de predicado, se G casa uma cadeia x quando interpretada como uma PEG, ela também casa x quando interpretada como uma CFG, conforme o lema a seguir:

Lema 4.3.1. *Dada uma gramática G, onde G é livre de repetição e livre de predicado, temos que se $G\ xy \overset{PEG}{\rightsquigarrow} y$ então $G\ xy \overset{CFG}{\rightsquigarrow} y$.*

Demonstração. A prova é por indução na altura da árvore de prova dada pela relação $\overset{\text{PEG}}{\rightsquigarrow}$. O caso interessante da prova é o da expressão de parsing $p_1 \,/\, p_2$. Há duas regras em $\overset{\text{PEG}}{\rightsquigarrow}$ relacionadas a esse caso: *ord*.1 e *ord*.2.

Se a regra *ord*.1 terminou a prova, então $G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$. Assim, pela hipótese de indução, $G[p_1]\ xy \overset{\text{CFG}}{\rightsquigarrow} y$, e pela regra *choice*.1 concluímos que $G[p_1 \mid p_2]\ xy \overset{\text{CFG}}{\rightsquigarrow} y$.

Se a regra *ord*.2 terminou a prova, então $G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$ e $G[p_2]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$. Pela hipótese de indução temos que $G[p_2]\ xy \overset{\text{CFG}}{\rightsquigarrow} y$, e pela regra *choice*.2 concluímos que $G[p_1 \mid p_2]\ xy \overset{\text{CFG}}{\rightsquigarrow} y$. □

Dada uma gramática G livre de repetição e livre de predicado, um corolário do lema 4.3.1 é que $L^{PEG}(G) \subseteq L^{CFG}(G)$.

Mais adiante, na seção 4.5, veremos que quando a gramática G é $LL(1)$, então temos que $L^{PEG}(G) = L^{CFG}(G)$.

4.4 Correspondência entre CFGs Lineares à Direita e PEGs

Na formalização usual de CFGs, gramáticas lineares à direita são uma classe de CFGs onde o lado direito de uma produção possui uma das seguintes formas [Hopcroft e Ullman, 1979]:

$$A \rightarrow w\,B \qquad\qquad A \rightarrow w$$

Como podemos ver, em gramáticas lineares à direita um não terminal sempre aparece na extremidade direita de uma concatenação, de modo que nunca temos uma concatenação onde um não terminal precede um símbolo da gramática.

A classe de linguagens descritas por CFGs lineares à direita é igual à classe de linguagens regulares, que são as linguagens descritas por expressões regulares e autômatos finitos.

Dado que todo autômato finito pode ser transformado em uma gramática linear à direita, ao discutirmos a correspondência entre CFGs lineares à direita e PEGs, estamos indiretamente discutindo como transformar autômatos finitos em PEGs, como sugerido em Ierusalimschy [2009].

Na nossa formalização de CFGs, uma gramática G é linear à direita se todas as concatenações da gramática são da forma $(w)\,B$[1] ou w, onde a expressão de parsing w representa a concatenação de vários terminais.

A partir de uma gramática linear à direita $G = (V, T, P, S)$ completa, podemos gerar uma PEG $G' = (V, T, P', S)$ equivalente da seguinte maneira, onde usamos a expressão `.` que é um açúcar sintático que casa qualquer terminal:

para toda produção $A \rightarrow p_1 \mid p_2 \mid \cdots \mid p_n \in P$:

$$P'(A) = matchEnd(p_1) \;/\; matchEnd(p_2) \;/\; \cdots \;/\; matchEnd(p_n)$$

A definição de *matchEnd* é dada a seguir:

$$matchEnd(p) = \begin{cases} (w)\,!. & \text{se } p = w \\ (w)\,B & \text{se } p = (w)\,B \end{cases}$$

Iremos nos referir ao processo anterior de obter a gramática G' a partir da gramática linear à direita G de *transformação ε-End*.

[1]Caso não usássemos parênteses para agrupar a expressão $w = a_1 a_2 \cdots a_n$, teríamos a concatenação $a_1\,(a_2 \cdots a_n B)$.

A gramática G' é completa, pois G não possui produções recursivas à esquerda e a transformação ε-End não introduz produções recursivas à esquerda.

Dado que o casamento da expressão de parsing $!\mathbf{.}$ é bem sucedido apenas quando a entrada é vazia, uma propriedade interessante da PEG G' obtida através da transformação ε-End é que um casamento em G' só é bem sucedido quando toda a entrada é consumida, como nos diz o lema a seguir:

Lema 4.4.1. *Dada uma gramática linear à direita G completa, e uma gramática G' obtida a partir de G usando a transformação ε-End, temos que se $G'[A]\ x \overset{PEG}{\rightsquigarrow} y$ então $y = \varepsilon$.*

Demonstração. A prova é por indução na altura da árvore de prova dada por $\overset{\text{PEG}}{\rightsquigarrow}$.

Dado que G' foi obtida a partir da transformação ε-End, então a forma geral de uma concatenação é $(w)!\mathbf{.}$ ou $(w)\,B$.

No caso de uma concatenação da forma $(w)\,!\mathbf{.}$, seja $x = wy$, temos que $G'[(w)\,!\mathbf{.}]\ wy \overset{\text{PEG}}{\rightsquigarrow} y$, e pela regra *con*.1 temos que $G'[!\mathbf{.}]\ y \overset{\text{PEG}}{\rightsquigarrow} y$. Dado que o casamento de $!\mathbf{.}$ só é bem sucedido para a entrada vazia, então concluímos que $y = \varepsilon$.

No caso de uma concatenação da forma $(w)\,B$, seja $x = wx'$, temos que $G'[(w)\,B]\ wx' \overset{\text{PEG}}{\rightsquigarrow} y$. Pela regra *con*.1 sabemos que $G'[B]\ x' \overset{\text{PEG}}{\rightsquigarrow} y$, e pela hipótese de indução concluímos que $y = \varepsilon$. □

Um corolário que segue do lema anterior é que em uma PEG G', obtida a partir de uma CFG linear à direita, o casamento de uma alternativa de uma escolha só é bem sucedido se toda a cadeia de entrada é consumida.

No caso de PEGs, onde temos uma escolha ordenada, devemos notar que quando as duas alternativas de uma escolha casam toda a entrada, o casamento dessa escolha ocorrerá somente através da primeira alternativa.

Agora, vamos usar o lema anterior para provar a correspondência entre CFGs lineares à direita e PEGs obtidas através da transformação ε-End:

Proposição 4.4.2. *Dada uma gramática linear à direita G completa, e uma gramática G' obtida a partir de G usando a transformação ε-End, temos que $G\ x \overset{CFG}{\rightsquigarrow} \varepsilon$ se e somente se $G'\ x \overset{PEG}{\rightsquigarrow} \varepsilon$.*

Demonstração. ($\Leftarrow$): Vamos provar esta parte por indução na altura da árvore de prova dada por $\overset{\text{PEG}}{\rightsquigarrow}$. O caso interessante é o da expressão de parsing da forma $p'_1 \,/\, p'_2$, cujas regras associadas são *ord*.1 e *ord*.2.

Se a regra *ord*.1 foi usada, então pela hipótese de indução temos que $G[p_1]\ x \overset{\text{CFG}}{\rightsquigarrow} \varepsilon$, e pela regra *choice*.1 concluímos que $G[p_1 \mid p_2]\ x \overset{\text{CFG}}{\rightsquigarrow} \varepsilon$.

Se a regra *ord*.2 foi usada, então pela hipótese de indução temos que $G[p_2]\ x \overset{\text{CFG}}{\rightsquigarrow} \varepsilon$, e pela regra *choice*.2 concluímos que $G[p_1 \mid p_2]\ x \overset{\text{CFG}}{\rightsquigarrow} \varepsilon$.

($\Rightarrow$): A prova desta parte é por indução na altura da árvore de prova dada por $\overset{\text{CFG}}{\rightsquigarrow}$. O caso interessante é o da expressão de parsing $p_1 \mid p_2$, cujas regras associadas são *choice*.1 e *choice*.2.

Se a regra *choice*.1 foi usada, pela hipótese de indução temos que $G'[p'_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \varepsilon$, e pela regra *ord*.1 concluímos que $G'[p'_1 \,/\, p'_2]\ x \overset{\text{PEG}}{\rightsquigarrow} \varepsilon$.

Se a regra *choice*.2 foi usada, pela hipótese de indução temos que $G'[p'_2]\ x \overset{\text{PEG}}{\rightsquigarrow} \varepsilon$. Vamos analisar dois casos: quando o casamento de p_1 em $\overset{\text{CFG}}{\rightsquigarrow}$ se relaciona com a cadeia vazia, e quando o casamento de p_1 em $\overset{\text{CFG}}{\rightsquigarrow}$ não se relaciona com a cadeia vazia.

Se p_1 se relaciona com a cadeia vazia em $\overset{\text{CFG}}{\rightsquigarrow}$, então pela hipótese de indução temos que $G'[p'_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \varepsilon$, e pela regra *ord*.1 concluímos que $G'[p'_1 \,/\, p'_2]\ x \overset{\text{PEG}}{\rightsquigarrow} \varepsilon$.

Se p_1 não se relaciona com a cadeia vazia em $\overset{\text{CFG}}{\rightsquigarrow}$, pela contrapositiva temos que p'_1 não se relaciona com a cadeia vazia em $\overset{\text{PEG}}{\rightsquigarrow}$. Como G' é completa, pela contrapositiva do lema 4.4.1 temos que $G'[p'_1]\ x \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$. Como sabemos que $G'[p'_2]\ x \overset{\text{PEG}}{\rightsquigarrow} \varepsilon$, pela regra *ord*.2 concluímos que $G'[p'_1 \,/\, p'_2]\ x \overset{\text{PEG}}{\rightsquigarrow} \varepsilon$. □

Se assumirmos que uma cadeia $w \in L(G)$ somente no caso em que $G\ w \overset{\text{CFG}}{\rightsquigarrow} \varepsilon$, então temos que $L^{CFG}(G) = L^{PEG}(G')$.

Como mostramos uma correspondência entre gramáticas lineares à direita e PEGs equivalentes, então podemos transformar um autômato finito em uma PEG equivalente.

Dado que podemos transformar expressões regulares em autômatos finitos, também temos uma outra correspondência entre expressões regulares e PEGs.

4.5 Correspondência entre CFGs $LL(1)$ e PEGs

As gramáticas $LL(1)$ são uma classe de CFGs em que um parser *top-down* correspondente pode decidir que produção da gramática deve ser usada olhando apenas o próximo terminal da entrada.

Como discutimos no capítulo 1, acredita-se que quando uma gramática é $LL(1)$, ela define a mesma linguagem quando interpretada como CFG e quando interpretada como PEG. Assim, para tentar comprovar essa conjectura, estudaremos nesta seção a correspondência entre CFGs $LL(1)$ e PEGs.

Na seção 4.5.1, mostramos que quando uma gramática G é $LL(1)$ e não possui expressões ε temos que $L^{CFG}(G) = L^{PEG}(G)$. Em seguida, na seção 4.5.2, provamos que, com uma pequena restrição na ordem das alternativas de

uma escolha, gramáticas $LL(1)$ com expressões ε também definem a mesma linguagem quando interpretadas como CFGs e quando interpretadas como PEGs.

Na discussão a seguir, usaremos o fato de que gramáticas $LL(1)$ não possuem regras recursivas à esquerda [Fischer e LeBlanc, 1991], e portanto são completas.

4.5.1 Gramáticas LL(1) sem Expressões ε

Vamos começar o nosso estudo da correspondência entre CFGs $LL(1)$ sem expressões ε e PEGs revendo a definição da função $FIRST^G$.

Dada uma gramática G, onde G não possui expressões ε, e uma cadeia α de símbolos da gramática, $FIRST^G(\alpha)$ é o conjunto de terminais que podem ser o primeiro terminal de uma cadeia gerada a partir de α na gramática G, como definido a seguir:

$$FIRST^G(\alpha) = \{\, a \in T \mid \alpha \Rightarrow_G^* a\beta \,\}$$

A definição correspondente de $FIRST^G$ na nossa formalização de CFGs, que usa expressões de parsing e a relação $\overset{\text{CFG}}{\rightsquigarrow}$, é dada a seguir:

$$FIRST^G(p) = \{\, a \in T \mid G[p]\ axy \overset{\text{CFG}}{\rightsquigarrow} y \,\}$$

Como a gramática G não possui nenhuma expressão de parsing da forma ε, então nenhuma expressão da gramática casa a cadeia vazia. Assim, podemos definir que uma gramática G é $LL(1)$ se todas as escolhas $p_1 | p_2$ de G satisfazem a seguinte condição:

$$FIRST^G(p_1) \cap FIRST^G(p_2) = \emptyset$$

De acordo com a definição acima, dada uma gramática $LL(1)$ G sem expressões ε, temos que para toda escolha $p_1 \,|\, p_2$ de G o casamento de uma alternativa nunca é bem sucedido para uma entrada quando o casamento da outra alternativa é bem sucedido para essa mesma entrada. Assim, o casamento de p_2 é bem sucedido somente quando o casamento de p_1 não é bem sucedido. Dessa forma, quando o casamento da escolha é bem sucedido, o resultado da interpretação da gramática G como uma CFG ou como uma PEG é o mesmo, como dito pela proposição a seguir:

Proposição 4.5.1. *Dada uma gramática $LL(1)$ G, onde G não possui expressões ε, temos que $G\ xy \overset{\text{CFG}}{\rightsquigarrow} y$ se e somente se $G\ xy \overset{\text{PEG}}{\rightsquigarrow} y$.*

Demonstração. Como o lema 4.3.1 prova a parte $\Leftarrow$ desta proposição, precisamos provar apenas a parte $\Rightarrow$. Vamos provar esta parte por indução na altura da árvore de prova dada por $\overset{\text{CFG}}{\rightsquigarrow}$. O caso interessante da prova é o da expressão de parsing $p_1 \mid p_2$. Há duas regras em $\overset{\text{CFG}}{\rightsquigarrow}$ associadas a esse caso: *choice*.1 e *choice*.2.

Se a regra *choice*.1 terminou a prova, então $G[p_1]\ xy \overset{\text{CFG}}{\rightsquigarrow} y$. Assim, pela hipótese de indução, $G[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$, e pela regra *ord*.1 concluímos que $G[p_1 \,/\, p_2]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$.

Se a regra *choice*.2 terminou a prova, sabemos que $G[p_2]\ xy \overset{\text{CFG}}{\rightsquigarrow} y$. Como G não possui expressões ε, uma expressão de parsing sempre casa uma cadeia não vazia, de modo que a cadeia x é da forma aw.

Dado que p_2 casa aw sabemos que $a \in FIRST^G(p_2)$, e como G é $LL(1)$ então $a \notin FIRST^G(p_1)$. Portanto, seja y' um sufixo de awy, temos que $\nexists y' \cdot G[p_1]\ awy \overset{\text{CFG}}{\rightsquigarrow} y'$. Como gramáticas $LL(1)$ são completas, sabemos que G é completa. Assim, pela contrapositiva do lema 4.3.1, temos que $G[p_1]\ awy \overset{\text{PEG}}{\rightsquigarrow}$ `fail`. Uma vez que $G[p_2]\ awy \overset{\text{CFG}}{\rightsquigarrow} y$, pela hipótese de indução temos que $G[p_2]\ awy \overset{\text{PEG}}{\rightsquigarrow} y$, e pela regra *ord*.2 concluímos que $G[p_1 \,/\, p_2]\ awy \overset{\text{PEG}}{\rightsquigarrow} y$. □

Um corolário da proposição 4.5.1 é que dada uma gramática $LL(1)$ G, onde G não possui expressões ε, temos que $G\ xy \overset{\text{CFG}}{\rightsquigarrow} y$ se e somente se $G\ xy \overset{\text{PEG}}{\rightsquigarrow} y$, ou seja, G define a mesma linguagem quando a interpretamos como uma CFG e quando a interpretamos como uma PEG.

4.5.2 Gramáticas LL(1) com Expressões ε

Se usarmos expressões ε para definir uma gramática, então alguma alternativa de uma escolha pode casar a cadeia vazia, o que torna a proposição 4.5.1 inválida.

Para ver a razão disso, vamos interpretar a expressão de parsing $p_1 \mid p_2$, onde p_1 casa a cadeia vazia. Se usarmos a semântica de $\overset{\text{PEG}}{\rightsquigarrow}$, então o casamento de p_1 sempre é bem sucedido e o casamento de $p_1 \,/\, p_2$ nunca acontece através da regra *ord*.2. Por outro lado, se interpretarmos $p_1 \mid p_2$ usando a semântica de $\overset{\text{CFG}}{\rightsquigarrow}$, continua sendo verdade que o casamento de p_1 sempre é bem sucedido, mas também pode acontecer um casamento bem sucedido da escolha através da regra *choice*.2. Em virtude disso, uma gramática $LL(1)$ com expressões ε pode definir linguagens diferentes quando interpretada como uma CFG e quando interpretada como uma PEG. A gramática $LL(1)$ G descrita a seguir ilustra esse fato:

$$S \to A \mid B \qquad A \to a\,A \mid \varepsilon \qquad B \to b \mid c$$

Se interpretarmos G como uma CFG, então $L^{CFG}(G) = \{a^n, b, c\}$. Por outro lado, se interpretarmos G como uma PEG, temos que $L^{PEG}(G) = \{a^n\}$, pois o casamento do não terminal A sempre é bem sucedido, e portanto o não terminal B nunca casa.

Como veremos mais adiante, se colocarmos uma restrição na ordem das alternativas de uma escolha podemos obter uma gramática que define a mesma linguagem quando interpretada como uma PEG e quando interpretada como uma CFG. Porém, antes de discutir essa restrição precisamos definir alguns conceitos relacionados a uma gramática $LL(1)$ com expressões ε.

Na discussão a seguir, assumiremos que a gramática G possui uma estrutura BNF. Como discutido na seção 4.2, se G é uma gramática com estrutura BNF, então uma escolha $p_1 \mid p_2$ nunca é uma subexpressão de uma concatenação, de modo que ela ou está associada a um não terminal A, ou ela é uma subexpressão de uma escolha que está associada a um não terminal A. Além disso, quando G possui estrutura BNF temos que $p_S = S$, ou seja, a expressão de parsing inicial de G é sempre um não terminal S.

Vamos começar atualizando a definição de $FIRST^G$:

$$FIRST^G(p) = \{\, a \in T \mid G[p]\ axy \overset{\text{CFG}}{\rightsquigarrow} y \,\} \cup matchEmpty(p)$$

$$matchEmpty(p) = \begin{cases} \{\varepsilon\} & \text{se } G[p]\ x \overset{\text{CFG}}{\rightsquigarrow} x \\ \emptyset & \text{caso contrário} \end{cases}$$

Também precisamos definir a função $FOLLOW^G$, que nos fornece o conjunto de terminais que podem seguir imediatamente um não terminal. Para definir $FOLLOW^G$ usaremos o símbolo \$ como um marcador de fim da entrada, onde $\$ \notin T$. A seguir, definimos quando um terminal a pertence ao conjunto $FOLLOW^G$ de um não terminal A:

> Dada uma gramática $G = (V, T, P, S)$ temos que $a \in FOLLOW^G(A)$ se há uma árvore de prova $G\ w\$ \overset{\text{CFG}}{\rightsquigarrow} \$$ com uma subárvore $G[A]\ xay \overset{\text{CFG}}{\rightsquigarrow} ay$.

Para computar $FOLLOW^G$, vamos analisar a forma geral da árvore de prova dada por $\overset{\text{CFG}}{\rightsquigarrow}$, a qual é mostrada na figura 4.2, e ver como a definição anterior de $FOLLOW^G$ funciona. Na figura 4.2, cada símbolo p_i representa uma concatenação e cada símbolo p_{ij} representa a cadeia vazia, um terminal, ou um não terminal. Nessa mesma figura, o símbolo b_i representa um terminal ou a cadeia vazia, e o símbolo x_i representa uma cadeia (possivelmente vazia).

Na parte de baixo da árvore de prova está o casamento da expressão de parsing inicial S. Pela definição de $FOLLOW^G$ sabemos que $\$ \in$

$$\dfrac{\dfrac{\dfrac{\dots \qquad \dots \qquad \dots \qquad \dots}{G[p_{i1}]\ b_1x_1\cdots b_nx_n\$ \overset{\text{CFG}}{\rightsquigarrow} b_2x_2\cdots b_nx_n\$ \quad G[p_{i2}]\ b_2x_2\cdots b_nx_n\$ \overset{\text{CFG}}{\rightsquigarrow} b_3x_3\cdots b_nx_n\$ \quad \dots \quad G[p_{in}]\ b_nx_n\$ \overset{\text{CFG}}{\rightsquigarrow} \$}}{G[p_{i1}\, p_{i2}\, \cdots\, p_{in}]\ b_1x_1\cdots b_nx_n\$ \overset{\text{CFG}}{\rightsquigarrow} \$}}{G[p_1 \mid p_2 \mid \cdots \mid p_m\ ]\ b_1x_1\cdots b_nx_n\$ \overset{\text{CFG}}{\rightsquigarrow} \$}}{G[S]\ b_1x_1\cdots b_nx_n\$ \overset{\text{CFG}}{\rightsquigarrow} \$}$$

Figura 4.2: Forma Geral da Árvore de Prova Quando a Gramática G Possui Estrutura BNF

$FOLLOW^G(S)$. No antecedente do casamento de S temos o casamento da expressão de parsing relacionada a S. Geralmente essa expressão de parsing é uma escolha, mas ela também pode ser uma concatenação, um símbolo da gramática, ou ε. Nestes três últimos casos, dizemos que temos uma *escolha unitária*.

No antecedente do casamento da escolha temos o casamento de uma das alternativas da escolha. Geralmente esse é o casamento de uma concatenação, mas também pode ser o casamento de um símbolo da gramática ou de ε.

Finalmente, temos as árvores de prova associadas com cada expressão de parsing da concatenação. A expressão de parsing p_{i1} casa a cadeia b_1x_1 (que pode ser vazia), a expressão de parsing p_{i2} casa a cadeia b_2x_2 (que pode ser vazia), e assim por diante.

Se considerarmos que p_{i1} é um não terminal A e que b_2 é um terminal, então pela definição de $FOLLOW^G$ sabemos que $b_2 \in FOLLOW^G(A)$. Como p_{i2} casa b_2x_2, também sabemos que $b_2 \in FIRST^G(p_{i2})$.

Se b_2x_2 é uma cadeia vazia e b_3 é um terminal, então $b_3 \in FOLLOW^G(A)$. Caso todas as cadeias b_jx_j, onde $1 < j \leq n$, sejam vazias, então $\$ \in FOLLOW^G(A)$.

Com base nessa descrição, podemos computar $FOLLOW^G$ para todos os não terminais $A \in V$ seguindo o algoritmo descrito na figura 4.3, onde X_{ij} pode ser ε, um terminal ou um não terminal, e a repetição converge quando nenhum terminal é adicionado a nenhum conjunto $FOLLOW^G(A)$.

No algoritmo para computar $FOLLOW^G$ usamos o operador $\otimes$, cuja definição é dada a seguir:

$$X \otimes Y = \begin{cases} X & \text{se } \varepsilon \notin X \\ (X - \{\varepsilon\}) \cup Y & \text{se } \varepsilon \in X \end{cases}$$

Vamos agora voltar a nossa atenção para a discussão sobre a correspondência entre CFGs $LL(1)$ e PEGs, onde usaremos $FIRST^G$ e $FOLLOW^G$ para definir o que é uma gramática $LL(1)$ com expressões ε.

para todo $A \in V$:
  $FOLLOW^G(A) \;=\; \emptyset$
$FOLLOW^G(S) = \{\,\$\,\}$
repita
  para toda produção $A \to X_{11} \cdots X_{1\,n_1} \mid \cdots \mid X_{m1} \cdots X_{m\,n_m}$:
    para toda concatenação $X_{i1} \cdots X_{i\,n_i}$:
      para $j = 1$ até n_i :
        se $X_{i\,j} \in V$ então :
          $FOLLOW^G(X_{i\,j}) \;=\; FOLLOW^G(X_{i\,j}) \;\cup$
                $(FIRST^G(X_{i\,j+1} \cdots X_{i\,n_i}) \;\otimes\; FOLLOW^G(A))$
até convergir

Figura 4.3: Algoritmo para Computar $FOLLOW^G$ dada uma Gramática $LL(1)$ G com Estrutura BNF

Em gramáticas $LL(1)$ as alternativas de uma escolha casam cadeias que começam com terminais diferentes, e no máximo uma das alternativas pode casar a cadeia vazia. Como vimos antes, dada uma escolha $p_1 \,|\, p_2$ onde p_1 casa a cadeia vazia, se interpretarmos essa escolha usando a semântica de $\overset{\text{PEG}}{\rightsquigarrow}$ o casamento de p_1 sempre é bem sucedido e portanto p_2 nunca casa. Para resolver esse problema, definimos na seção 4.2 que quando uma gramática possui estrutura BNF a alternativa de uma escolha que não casa a cadeia vazia sempre precede a alternativa que casa a cadeia vazia. Assim, em gramáticas $LL(1)$ que possuem estrutura BNF, dada uma escolha $p_1 \,|\, p_2$, sabemos que somente a alternativa p_2 pode casar a cadeia vazia. Em virtude disso, a alternativa p_1 deve casar pelo menos um terminal para que o seu casamento seja bem sucedido.

Essa restrição afeta o resultado do casamento de uma escolha somente quando a interpretamos como uma PEG, pois em CFGs a ordem das alternativas de uma escolha não é relevante.

Dada então uma gramática G, onde G possui estrutura BNF, dizemos que G é $LL(1)$ se para toda produção $A \to p$ de G, seja $p_1 \,|\, p_2$ uma subexpressão de p, temos que as seguintes condições são satisfeitas:

- $FIRST^G(p_1) \;\cap\; FIRST^G(p_2) \;=\; \emptyset$
- se $\varepsilon \in FIRST^G(p_2)$ então $FIRST^G(p_1) \;\cap\; FOLLOW^G(A) \;=\; \emptyset$

Quando G é $LL(1)$, queremos mostrar que $L^{CFG}(G) = L^{PEG}(G)$. Para alcançar esse objetivo, vamos definir a relação $\overset{\text{LL(1)}}{\rightsquigarrow}$ que, assim como $\overset{\text{CFG}}{\rightsquigarrow}$,

Cadeia Vazia $\dfrac{}{G[\varepsilon]\ x \overset{\text{LL(1)}}{\rightsquigarrow} x}\,(\textbf{empty.1})$ **Terminal** $\dfrac{}{G[a]\ ax \overset{\text{LL(1)}}{\rightsquigarrow} x}\,(\textbf{char.1})$

Variável $\dfrac{G[P(A)]\ xy \overset{\text{LL(1)}}{\rightsquigarrow} y}{G[A]\ xy \overset{\text{LL(1)}}{\rightsquigarrow} y}\,(\textbf{var.1})$ **Concatenação** $\dfrac{G[p_1]\ xyz \overset{\text{LL(1)}}{\rightsquigarrow} yz \quad G[p_2]\ yz \overset{\text{LL(1)}}{\rightsquigarrow} z}{G[p_1\,p_2]\ xyz \overset{\text{LL(1)}}{\rightsquigarrow} z}\,(\textbf{con.1})$

Escolha $\dfrac{G[p_1]\ xy \overset{\text{LL(1)}}{\rightsquigarrow} y}{G[p_1 \mid p_2]\ xy \overset{\text{LL(1)}}{\rightsquigarrow} y}\,(\textbf{choice.1})$

$$\frac{G[p_2]\ xy \overset{\text{LL(1)}}{\rightsquigarrow} y}{G[p_1 \mid p_2]\ xy \overset{\text{LL(1)}}{\rightsquigarrow} y},\ x = \varepsilon \Rightarrow \nexists\, y' \cdot G[p_1]\ xy \overset{\text{LL(1)}}{\rightsquigarrow} y'\,(\textbf{choice}_{\textbf{LL(1)}}\textbf{.2})$$

Figura 4.4: Definição da Relação $\overset{\text{LL(1)}}{\rightsquigarrow}$ Usando Semântica Natural

relaciona uma gramática G e uma entrada xy com um sufixo y da entrada. Na figura 4.4, apresentamos a definição de $\overset{\text{LL(1)}}{\rightsquigarrow}$ usando semântica natural.

Como podemos ver, as regras de $\overset{\text{LL(1)}}{\rightsquigarrow}$ são iguais às regras de $\overset{\text{CFG}}{\rightsquigarrow}$, com exceção da regra $choice_{LL(1)}.2$, que corresponde à regra $choice.2$ de $\overset{\text{CFG}}{\rightsquigarrow}$. Na semântica de $\overset{\text{LL(1)}}{\rightsquigarrow}$, a segunda alternativa de uma escolha só casa um prefixo vazio da entrada se o casamento da primeira alternativa da escolha não é bem sucedido para essa entrada.

Quando temos uma gramática $LL(1)$, podemos estabelecer a seguinte correspondência entre as relações $\overset{\text{CFG}}{\rightsquigarrow}$ e $\overset{\text{LL(1)}}{\rightsquigarrow}$:

Lema 4.5.2. *Dada uma gramática $LL(1)$ G, temos que se $G[A]\ xay \overset{CFG}{\rightsquigarrow} ay$, onde $a \in FOLLOW^G(A)$, então $G[A]\ xay \overset{LL(1)}{\rightsquigarrow} ay$.*

Demonstração. A prova é por indução na altura da árvore de prova dada por $\overset{\text{CFG}}{\rightsquigarrow}$. O caso interessante é quando a expressão de parsing associada ao não terminal A é uma escolha $p_1 \mid p_2$ e a regra $choice.2$ é usada. Nesse caso, temos a seguinte árvore de prova:

$$\frac{\dfrac{G[p_2]\ xay \overset{\text{CFG}}{\rightsquigarrow} ay}{G[p_1 \mid p_2]\ xay \overset{\text{CFG}}{\rightsquigarrow} ay}\,(\textbf{choice.2})}{G[A]\ xay \overset{\text{CFG}}{\rightsquigarrow} ay}\,(\textbf{var.1})$$

Dado o casamento $G[p_2]\ xay \overset{\text{CFG}}{\rightsquigarrow} ay$, temos que p_2 pode ter casado a cadeia vazia ou uma cadeia não vazia.

Se p_2 casou a cadeia vazia, então $x = \varepsilon$ e temos que $G[p_2]\ ay \overset{\text{CFG}}{\rightsquigarrow} ay$, onde $\varepsilon \in FIRST^G(p_2)$. Dado que $a \in FOLLOW^G(A)$ e que G é $LL(1)$, temos

que $FIRST^G(p_1) \cap FOLLOW^G(A) = \emptyset$, de modo que $a \notin FIRST^G(p_1)$ e o casamento de p_1 não é bem sucedido usando $\overset{\text{CFG}}{\rightsquigarrow}$. Logo, temos que o casamento de p_1 não é bem sucedido usando $\overset{\text{LL(1)}}{\rightsquigarrow}$. Pela hipótese de indução temos que $G[p_2]\ ay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, e pela regra $choice_{LL(1)}.2$ temos que $G[p_1 \mid p_2]\ ay \overset{\text{LL(1)}}{\rightsquigarrow} ay$. Portanto, pela regra $var.1$, concluímos que $G[A]\ ay \overset{\text{LL(1)}}{\rightsquigarrow} ay$.

Se p_2 casou uma cadeia não vazia, então pela hipótese de indução $G[p_2]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$. Pela regra $choice_{LL(1)}.2$ temos que $G[p_1 \mid p_2]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$ e pela regra $var.1$ concluímos que $G[A]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$. □

Poderíamos provar também a outra direção do lema anterior. Contudo, embora seja fácil provar que todo casamento bem sucedido através da relação $\overset{\text{LL(1)}}{\rightsquigarrow}$ também é bem sucedido através da relação $\overset{\text{CFG}}{\rightsquigarrow}$, não seria tão fácil assim mostrar que a condição $a \in FOLLOW^G(A)$ do lema anterior é satisfeita no casamento através de $\overset{\text{CFG}}{\rightsquigarrow}$ dado o casamento através de $\overset{\text{LL(1)}}{\rightsquigarrow}$. Como não precisamos da outra direção do lema 4.5.2 para provar a correspondência entre CFGs $LL(1)$ e PEGs, não iremos mostrá-la.

Após estabelecer a correspondência entre $\overset{\text{CFG}}{\rightsquigarrow}$ e $\overset{\text{LL(1)}}{\rightsquigarrow}$, vamos definir o seguinte lema a respeito do casamento em uma gramática $LL(1)$ usando a semântica de $\overset{\text{LL(1)}}{\rightsquigarrow}$ e a semântica de $\overset{\text{PEG}}{\rightsquigarrow}$:

Lema 4.5.3. *Dada uma gramática $LL(1)$ G, então $G[A]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, onde $a \in FOLLOW^G(A)$, se e somente se $G[A]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$.*

Demonstração. ($\Rightarrow$): Vamos provar esta parte por indução na altura da árvore de prova dada por $\overset{\text{LL(1)}}{\rightsquigarrow}$. O caso interessante é quando a expressão de parsing associada ao não terminal A é uma escolha $p_1 \mid p_2$. Existem duas regras em $\overset{\text{LL(1)}}{\rightsquigarrow}$ relacionadas a esse caso: $choice.1$ e $choice_{LL(1)}.2$.

Se a regra $choice.1$ foi usada, então $G[p_1]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, e pela hipótese de indução $G[p_1]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$. Assim, pela regra $ord.1$, temos que $G[p_1 \,/ p_2]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$ e pela regra $var.1$ concluímos que $G[A]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$.

Se a regra $choice_{LL(1)}.2$ foi usada, então $G[p_2]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, onde p_2 pode ter casado a cadeia vazia ou uma cadeia não vazia.

Se p_2 casou a cadeia vazia, então $x = \varepsilon$. Dado que a gramática G é $LL(1)$, temos que $FOLLOW^G(A) \cap FIRST^G(p_1) = \emptyset$, e como $a \in FOLLOW^G(A)$ então $a \notin FIRST^G(p_1)$. Portanto, o casamento de p_1 não é bem sucedido em $\overset{\text{CFG}}{\rightsquigarrow}$. Como G é completa, pela contrapositiva do lema 4.3.1 temos que $G[p_1]\ ay \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$. Dado que $G[p_2]\ ay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, pela hipótese de indução temos que $G[p_2]\ ay \overset{\text{PEG}}{\rightsquigarrow} ay$, e pela regra $ord.2$ temos que $G[p_1 \,/\, p_2]\ ay \overset{\text{PEG}}{\rightsquigarrow} ay$. Assim, pela regra $var.1$, concluímos que $G[A]\ ay \overset{\text{PEG}}{\rightsquigarrow} ay$.

Se p_2 casou uma cadeia não vazia bw, então $b \in FIRST^G(p_2)$. Pela definição de gramática $LL(1)$ sabemos que $FIRST^G(p_1) \cap FIRST^G(p_2) = \emptyset$,

de modo que $b \notin FIRST^G(p_1)$. Como p_1 não casa a cadeia vazia, temos que o casamento de p_1 não é bem sucedido em $\overset{\text{CFG}}{\rightsquigarrow}$. Uma vez que gramáticas $LL(1)$ são completas, pela contrapositiva do lema 4.3.1 temos que $G[p_1]\ xay \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$. Dado que $G[p_2]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, pela hipótese de indução temos que $G[p_2]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$, e pela regra $ord.2$ temos que $G[p_1 \,/\, p_2]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$. Assim, pela regra $var.1$ concluímos que $G[A]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$.

($\Leftarrow$): Vamos provar esta parte por indução na altura da árvore de prova dada por $\overset{\text{PEG}}{\rightsquigarrow}$. O caso interessante é quando a expressão de parsing associada ao não terminal A é da forma $p_1 \,/\, p_2$. As duas regras associadas a esse caso são $ord.1$ e $ord.2$.

Se a regra $ord.1$ foi usada, então $G[p_1]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$. Pela hipótese de indução $G[p_1]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, e pela regra $choice.1$ temos que $G[p_1 \,|\, p_2]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$. Portanto, pela regra $var.1$ concluímos que $G[A]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$, onde $a \in FOLLOW^G(A)$.

Se a regra $ord.2$ foi usada, então $G[p_1]\ xay \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$ e $G[p_2]\ xay \overset{\text{PEG}}{\rightsquigarrow} ay$, onde p_2 pode ter casado a cadeia vazia ou uma cadeia não vazia.

Se p_2 casou a cadeia vazia, pela hipótese de indução temos que $G[p_2]\ ay \overset{\text{LL(1)}}{\rightsquigarrow} ay$. Pela regra $ord.2$ sabemos que $G[p_1]\ ay \overset{\text{PEG}}{\rightsquigarrow} \texttt{fail}$, e pela contrapositiva temos que o casamento de p_1 usando $\overset{\text{LL(1)}}{\rightsquigarrow}$ não é bem sucedido. Assim, pela regra $choice_{LL(1)}.2$, temos que $G[p_1 \,|\, p_2]\ ay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, e pela regra $var.1$ concluímos que $G[A]\ ay \overset{\text{LL(1)}}{\rightsquigarrow} ay$ onde $a \in FOLLOW^G(A)$.

Se p_2 casou uma cadeia não vazia, pela hipótese de indução temos que $G[p_2]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$, e pela regra $choice_{LL(1)}.2$ temos que $G[p_1 \,|\, p_2]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$. Assim, pela regra $var.1$, concluímos que $G[A]\ xay \overset{\text{LL(1)}}{\rightsquigarrow} ay$ onde $a \in FOLLOW^G(A)$. □

Dada uma gramática $LL(1)$ G, a seguinte proposição afirma que $L^{CFG}(G) = L^{PEG}(G)$:

Proposição 4.5.4. *Dada uma gramática $LL(1)$ $G = (V, T, P, S)$, temos que $G\ x\$ \overset{\text{CFG}}{\rightsquigarrow} \$$ se e somente se $G\ x\$ \overset{\text{PEG}}{\rightsquigarrow} \$$.*

Demonstração. ($\Rightarrow$): Dado que $G\ x\$ \overset{\text{CFG}}{\rightsquigarrow} \$$ e que $\$ \in FOLLOW^G(S)$, pelo lema 4.5.2 temos que $G\ x\$ \overset{\text{LL(1)}}{\rightsquigarrow} \$$, e pelo lema 4.5.3 concluímos que $G\ x\$ \overset{\text{PEG}}{\rightsquigarrow} \$$.

($\Leftarrow$): Dado que $G\ x\$ \overset{\text{PEG}}{\rightsquigarrow} \$$, pelo lema 4.3.1 concluímos que $G\ x\$ \overset{\text{CFG}}{\rightsquigarrow} \$$. □

4.6 Correspondência entre CFGs $LL(k)$-Forte e PEGs

As gramáticas $LL(k)$-forte são uma classe de CFGs onde um parser top-down correspondente pode decidir que produção da gramática deve ser usada olhando somente os próximos k terminais da entrada.

A classe de linguagens descrita pelas gramáticas $LL(k)$-forte é uma subclasse da classe de linguagens descrita pelas gramáticas $LL(k)$. Uma gramática G é $LL(k)$ se um parser top-down correspondente pode decidir que produção da gramática usar com base nos próximos k terminais da entrada e também em quais produções da gramáticas foram usadas anteriormente, ou seja, um parser $LL(k)$ precisa de um contexto para decidir corretamente que produção da gramática deve ser usada [Fischer e LeBlanc, 1991]. No caso específico em que $k = 1$ temos que a classe de linguagens descrita por gramáticas $LL(1)$-forte é igual à classe de linguagens descrita por gramáticas $LL(1)$ [Fischer e LeBlanc, 1991, Grune e Jacobs, 2006].

Nesta seção iremos mostrar que a partir de uma CFG $LL(k)$-forte podemos gerar uma PEG equivalente que possui a mesma estrutura da CFG.

De modo similar ao que fizemos na seção 4.5.2, iremos assumir que a gramática G possui uma estrutura BNF, pois assim cada escolha é uma subexpressão de uma escolha que está associada a algum não terminal A, e a expressão de parsing inicial da gramática é um não terminal S.

Antes de discutir a correspondência entre CFGs $LL(k)$-forte e PEGs, precisamos definir as funções $FIRST_k^G$ e $FOLLOW_k^G$, que estendem, respectivamente, as funções $FIRST^G$ e $FOLLOW^G$.

Vamos começar definindo a função $FIRST_k^G$ como a seguir:

$$FIRST_k^G(p) \;=\; \{\, take_k(x) \mid G[p] \;\; xy \overset{\text{CFG}}{\rightsquigarrow} y \,\}$$

Na definição anterior usamos a função $take_k$, que recebe uma cadeia e retorna um prefixo dessa cadeia com comprimento menor ou igual a k, como descrito a seguir:

$$take_k(xy) = \begin{cases} xy & \text{se } |xy| \leq k \\ x & \text{se } |xy| > k, \text{ onde } |x| = k \end{cases}$$

Agora, iremos definir a função $FOLLOW_k^G$, que fornece o conjunto de cadeias de comprimento k que podem seguir um não terminal.

Vamos usar novamente o símbolo \$ para indicar o fim da entrada, onde $\$ \notin T$. Quando discutimos gramáticas $LL(1)$, colocamos um único símbolo \$ no final da entrada. Porém, no caso de gramáticas $LL(k)$-forte colocaremos k

para todo $A \in V$:
  $FOLLOW_k^G(A) \;=\; \emptyset$
$FOLLOW_k^G(S) = \{\$^k\}$
repita
  para toda produção $A \to X_{11} \cdots X_{1\,n_1} \mid \cdots \mid X_{m1} \cdots X_{m\,n_m}$:
    para toda concatenação $X_{i1} \cdots X_{i\,n_i}$:
      para $j = 1$ até n_i :
        se $X_{i\,j} \in V$ então :
          $FOLLOW_k^G(X_{i\,j}) \;=\; FOLLOW_k^G(X_{i\,j}) \;\cup$
                $(FIRST_k^G(X_{i\,j+1} \cdots X_{i\,n_i}) \;\otimes_k\; FOLLOW_k^G(A))$
até convergir

Figura 4.5: Algoritmo para Computar $FOLLOW_k^G$ dada uma Gramática $LL(k)$-Forte G com Estrutura BNF

símbolos \$ no final da entrada, de modo que cada não terminal é seguido por pelo menos k terminais. Usaremos a notação $\$^k$ para indicar uma cadeia de k símbolos \$. A seguir, definimos quando uma cadeia de terminais pertence ao conjunto $FOLLOW_k^G$ de um não terminal A:

> Dada uma gramática $G = (V, T, P, S)$, temos que $take_k(y) \in FOLLOW_k^G(A)$ se há uma árvore de prova $G\ w\$^k \overset{\text{CFG}}{\rightsquigarrow} \$^k$ com uma subárvore $G[A]\ xy \overset{\text{CFG}}{\rightsquigarrow} y$.

Dada a definição anterior, o lema 4.2.1 nos diz que em toda subárvore $G[A]\ xy \overset{\text{CFG}}{\rightsquigarrow} y$ temos que $|y| \geq k$, o que implica que todas as cadeias em $FOLLOW_k^G(A)$ possuem comprimento k. Assim, nenhuma cadeia de $FOLLOW_k^G(A)$ é um prefixo de outra cadeia desse conjunto. Esse fato será útil mais adiante, ao gerarmos uma expressão de parsing para casar os elementos de $FOLLOW_k^G(A)$.

Podemos computar o conjunto $FOLLOW_k^G$ para todos os não terminais $A \in V$ usando o algoritmo descrito na figura 4.5, que é similar ao algoritmo da figura 4.3, onde $X_{i\,j}$ pode ser ε, um terminal ou um não terminal, e a repetição converge quando não é possível adicionar nenhuma cadeia a nenhum conjunto $FOLLOW_k^G(A)$.

O operador $\otimes_k$ usado na figura 4.5 é definido a seguir:

$$X \otimes_k Y \;=\; \{\, take_k(w) \mid w \in XY \,\}$$

Após definirmos as funções $FIRST_k^G$ e $FOLLOW_k^G$, vamos definir o que é uma gramática $LL(k)$-forte. Dada uma gramática G, onde G possui estrutura BNF, dizemos que G é $LL(k)$-forte se todas as escolhas $p_1 \mid p_2$ associadas a um não terminal A de G satisfazem a seguinte condição:

$$(FIRST_k^G(p_1) \otimes_k FOLLOW_k^G(A)) \cap (FIRST_k^G(p_2) \otimes_k FOLLOW_k^G(A)) = \emptyset$$

A seguir, temos o exemplo de uma gramática G que é $LL(2)$-forte:

$$S \rightarrow A \mid B \qquad A \rightarrow a\,b \mid C \qquad B \rightarrow a \mid C\,d \qquad C \rightarrow c$$

Os conjuntos $FIRST_2^G$ e $FOLLOW_2^G$ associados aos não terminais da gramática G são os seguintes:

- $FIRST_2^G(S) = \{a,\, ab,\, c,\, cd\}$
- $FIRST_2^G(A) = \{ab,\, c\}$
- $FIRST_2^G(B) = \{a,\, cd\}$
- $FIRST_2^G(C) = \{c\}$
- $FOLLOW_2^G(S) = FOLLOW_2^G(A) = FOLLOW_2^G(B) = \{\$\$\}$
- $FOLLOW_2^G(C) = \{d\$,\, \$\$\}$

Se interpretarmos G como uma CFG, então $L^{CFG}(G) = \{\, a,\, ab,\, c,\, cd \,\}$. Por outro lado, se interpretarmos G como uma PEG temos que $L^{PEG}(G) = \{\, a,\, ab,\, c \,\}$. Podemos ver que $cd \notin L^{PEG}(G)$, uma vez que o casamento de A sempre é bem sucedido quando a entrada possui prefixo c. Portanto, a gramática G descreve linguagens diferentes quando interpretada como uma CFG e quando interpretada como uma PEG.

Ao contrário de gramáticas $LL(1)$ com expressões ε, uma simples restrição na ordem das alternativas de uma escolha não torna a linguagem da CFG igual à linguagem da PEG. Se mudarmos a escolha $A \mid B$ associada a S para $B \mid A$, o resultado seria que $L^{PEG}(G) = \{\, a,\, c,\, cd \,\}$. Nesse caso, temos que $ab \notin L^{PEG}(G)$, uma vez que o casamento de B sempre é bem sucedido quando a entrada possui prefixo a.

Dada uma gramática $LL(k)$-forte G, como $L^{CFG}(G) \neq L^{PEG}(G)$, para estabelecer a correspondência entre CFGs $LL(k)$-forte e PEGs iremos gerar uma nova gramática G' a partir da gramática G, onde $L^{CFG}(G) = L^{PEG}(G')$.

4.6.1 Transformação de uma CFG $LL(k)$-forte em uma PEG Equivalente

Para transformar uma CFG $LL(k)$-forte em uma PEG equivalente iremos usar, como mencionado na seção 2.3, a expressão de parsing $\&p$ como um açúcar sintático para a expressão $!!p$. Na discussão a seguir, também usaremos o fato de que gramáticas $LL(k)$-forte não possuem regras recursivas à esquerda, e portanto são completas.

Dada uma cadeia de terminais w, usaremos p_w para representar a expressão de parsing correspondente, onde p_w casa w apenas. Como p_w não possui não terminais, o casamento de p é independente da gramática G.

Para transformar uma CFG $LL(k)$-forte em uma PEG equivalente vamos precisar definir a função auxiliar *set2choice*, que recebe um conjunto Z, cujos elementos são cadeias de terminais, e nos dá uma expressão de parsing que casa os elementos de Z:

$$set2choice\,(Z) \;=\; p_{z_1} \;\mid\; p_{z_2} \;\mid\; \cdots \;\mid\; p_{z_n}\text{ , onde } z_i \in Z$$

A função *set2choice* nos dá uma expressão de parsing que não possui não terminais, de modo que o casamento dessa expressão é independente da gramática. No caso da gramática $LL(2)$-forte G que apresentamos anteriormente, a função *set2choice* dos daria as seguintes expressões de parsing:

$$set2choice(S) = set2choice(A) = set2choice(B) = \$\,\$ \qquad set2choice(C) = d\,\$\,/\$\,\$$$

Dada um não terminal A, como todos os elementos de $FOLLOW_k^G(A)$ possuem o mesmo comprimento, nenhum elemento pode ser prefixo de outro. Em virtude disso, a escolha resultante de $set2choice(FOLLOW_k^G(A))$ é uma escolha disjunta. Dado um não terminal A, usaremos $\phi(A)$ para representar $set2choice(FOLLOW_k^G(A))$.

A partir de uma gramática $LL(k)$-forte $G = (V,\, T,\, P,\, S)$, podemos gerar uma PEG $G' = (V,\, T,\, P',\, S)$ equivalente da seguinte maneira:

$$\begin{array}{c}\text{para toda produção } A \to p_1 \mid p_2 \mid \cdots \mid p_n \in P : \\ P'(A) \;=\; p_1\,\&\phi(A) \;/\; p_2\,\&\phi(A) \;/\; \cdots \;/\; p_n\,\&\phi(A)\end{array}$$

Acima, dado um não terminal A, usamos $\&\phi(A)$ para testar se é possível casar alguma cadeia $x \in FOLLOW_k^G(A)$ depois de casar uma alternativa da escolha associada a A. Iremos nos referir ao método anterior de obter a gramática G' a partir da gramática G de *transformação $LL(k)$-PEG*. A gramática G' obtida dessa forma é completa, pois G é completa a transformação $LL(k)$-PEG

Cadeia Vazia $\dfrac{}{G[\varepsilon]\ x \overset{\text{LL(k)}}{\rightsquigarrow} x}\ (\mathbf{empty.1})$ **Terminal** $\dfrac{}{G[a]\ ax \overset{\text{LL(k)}}{\rightsquigarrow} x}\ (\mathbf{char.1})$

Variável $\dfrac{G[P(A)]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y}{G[A]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y}\ ,\ take_k(y) \in FOLLOW_k^G(A)\ (\mathbf{var_{LL(k)}.1})$

Concatenação $\dfrac{G[p_1]\ xyz \overset{\text{LL(k)}}{\rightsquigarrow} yz \quad G[p_2]\ yz \overset{\text{LL(k)}}{\rightsquigarrow} z}{G[p_1\, p_2]\ xyz \overset{\text{LL(k)}}{\rightsquigarrow} z}\ (\mathbf{con.1})$

Escolha $\dfrac{G[p_1]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y}{G[p_1 \mid p_2]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y}\ (\mathbf{choice.1})$ $\dfrac{G[p_2]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y}{G[p_1 \mid p_2]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y}\ (\mathbf{choice.2})$

Figura 4.6: Definição da Relação $\overset{\text{LL(k)}}{\rightsquigarrow}$ Usando Semântica Natural

não introduz produções recursivas à esquerda, nem expressões de parsing da forma p^* onde p cada a cadeia vazia.

No caso da gramática $LL(2)$-forte G que apresentamos anteriormente, a transformação $LL(k)$-PEG nos daria a seguinte gramática G':

$$S \rightarrow A\,\&(\$\,\$) \,/\, B\,\&(\$\,\$) \qquad A \rightarrow a\,b\,\&(\$\,\$) \,/\, C\,\&(\$\,\$)$$
$$B \rightarrow a\,\&(\$\,\$) \,/\, C\,d\,\&(\$\,\$) \qquad C \rightarrow c\,\&(d\,\$ \,/\, \$\,\$)$$

Para provar a equivalência entre uma CFG $LL(k)$-forte G e a PEG G' obtida através da transformação $LL(k)$-PEG, vamos definir uma nova relação $\overset{\text{LL(k)}}{\rightsquigarrow}$ que, assim como $\overset{\text{CFG}}{\rightsquigarrow}$, relaciona uma gramática G e uma entrada xy com um sufixo y da entrada. Na figura 4.6, apresentamos a definição de $\overset{\text{LL(k)}}{\rightsquigarrow}$ usando semântica natural.

Assim como as regras de $\overset{\text{LL(1)}}{\rightsquigarrow}$, quase todas as regras de $\overset{\text{LL(k)}}{\rightsquigarrow}$ são iguais às regras de $\overset{\text{CFG}}{\rightsquigarrow}$. No caso de $\overset{\text{LL(k)}}{\rightsquigarrow}$, a única regra diferente é a regra $var_{LL(k)}.1$, que trata do casamento de um não terminal. Na semântica de $\overset{\text{LL(k)}}{\rightsquigarrow}$, o casamento de um não terminal A é bem sucedido para uma entrada xy somente quando A casa um prefixo x da entrada e $take_k(y) \in FOLLOW_k^G(A)$.

No caso da relação $\overset{\text{LL(1)}}{\rightsquigarrow}$, poderíamos ter definido uma regra $var_{LL(1)}.1$, análoga à regra $var_{LL(k)}.1$, ao invés de ter definido a regra $choice_{LL(1)}.2$. Contudo, se definíssemos $var_{LL(1)}.1$, para provar a equivalência entre CFGs $LL(1)$ e PEGs teríamos que gerar uma PEG a partir de uma gramática $LL(1)$, de modo similar ao que estamos fazendo para gramáticas $LL(k)$-forte. Embora a definição da regra $choice_{LL(1)}.2$ não seja tão simples quanto o que seria a definição da regra $var_{LL(1)}.1$, o uso de $choice_{LL(1)}.2$ nos possibilitou interpretar

gramáticas $LL(1)$ G como CFGs e PEGs e mostrar que $L^{CFG}(G) = L^{PEG}(G)$.

Dadas as definições de $\overset{\text{CFG}}{\rightsquigarrow}$ e de $\overset{\text{LL(k)}}{\rightsquigarrow}$, podemos estabelecer a seguinte correspondência entre essas relações quando a gramática G é $LL(k)$-forte:

Lema 4.6.1. *Dada uma gramática $LL(k)$-forte G, temos que $G[A]\ xy \overset{CFG}{\rightsquigarrow} y$, onde $take_k(y) \in FOLLOW_k^G(A)$, se e somente se $G[A]\ xy \overset{LL(k)}{\rightsquigarrow} y$.*

Demonstração. Trivial, pois quando $G[A]\ \ xy \overset{\text{CFG}}{\rightsquigarrow} y$ com $take_k(y) \in FOLLOW_k^G(A)$, a semântica de $\overset{\text{CFG}}{\rightsquigarrow}$ torna-se idêntica à semântica de $\overset{\text{LL(k)}}{\rightsquigarrow}$. □

Após estabelecer a correspondência entre $\overset{\text{CFG}}{\rightsquigarrow}$ e $\overset{\text{LL(k)}}{\rightsquigarrow}$, vamos definir o seguinte lema sobre o casamento de expressões de parsing que possuem a forma restrita das expressões resultantes da função *set2choice*:

Lema 4.6.2. *Dada uma expressão de parsing p, onde p só possui subexpressões da forma ε, a, $p_1\,p_2$ e $p_1 \mid p_2$, e onde todas as subexpressões da forma $p_1 \mid p_2$ possuem alternativas disjuntas, temos que $p\ xy \overset{LL(k)}{\rightsquigarrow} y$ se e somente se $p\ xy \overset{PEG}{\rightsquigarrow} y$, para quaisquer gramáticas G e G'.*

Demonstração. A parte $\Rightarrow$ usa indução na altura da árvore de prova dada por $\overset{\text{LL(k)}}{\rightsquigarrow}$, enquanto que a parte $\Leftarrow$ usa indução na altura da árvore de prova dada por $\overset{\text{PEG}}{\rightsquigarrow}$.

Dado que as alternativas de uma escolha são disjuntas e não há subexpressões da forma A, é trivial mostrar que o resultado de um casamento é o mesmo nas duas semânticas. Dado que p não possui subexpressões da forma A, o seu casamento não depende da gramática. □

Agora, vamos usar o lema 4.6.2 para provar o seguinte lema a respeito do casamento em G usando $\overset{\text{LL(k)}}{\rightsquigarrow}$ e do casamento em G' usando $\overset{\text{PEG}}{\rightsquigarrow}$:

Lema 4.6.3. *Dada uma gramática $LL(k)$-forte G e uma gramática G' obtida a partir de G usando a transformação $LL(k)$-PEG, temos que $G\ xy \overset{LL(k)}{\rightsquigarrow} y$ se e somente se $G'\ xy \overset{PEG}{\rightsquigarrow} y$.*

Demonstração. ($\Leftarrow$): A prova desta parte usa indução na altura da árvore de prova dada por $\overset{\text{PEG}}{\rightsquigarrow}$. O caso interessante é quando a expressão de parsing associada a um não terminal A é uma escolha $p_1 \,/\, p_2$. Há duas regras em $\overset{\text{PEG}}{\rightsquigarrow}$ relacionadas ao casamento de uma escolha: *ord*.1 e *ord*.2.

Se a regra *ord*.1 foi usada, então $G'[p_1 \,\&\phi(A)]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$. Pela regra *con*.1 temos que $G'[p_1]\ xy \overset{\text{PEG}}{\rightsquigarrow} y$, e por *con*.1 e *not*.1 temos que $G'[\phi(A)]\ y \overset{\text{PEG}}{\rightsquigarrow} y'$. Pela hipótese de indução temos que $G[p_1]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$, e pelo lema 4.6.2 temos que $G[\phi(A)]\ y \overset{\text{LL(k)}}{\rightsquigarrow} y'$. Assim, pela regra *choice*.1, temos que $G[p_1 \,|p_2]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$. Como o casamento de $\phi(A)$ é bem sucedido, então $take_k(y) \in FOLLOW_k^G(A)$, e pela regra $var_{LL(k)}$.1 podemos concluir que $G[A]\ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$.

Se a regra *ord*.2 foi usada, então $G'[p_2 \,\&\phi(A)]\ \ xy \overset{\text{PEG}}{\rightsquigarrow} y$. Pela regra *con*.1 temos que $G'[p_2]\ \ xy \overset{\text{PEG}}{\rightsquigarrow} y$ e por *con*.1 e *not*.1 temos que $G'[\phi(A)]\ \ y \overset{\text{PEG}}{\rightsquigarrow} y'$. Pela hipótese de indução $G[p_2]\ \ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$, e pelo lema 4.6.2 sabemos que $G[\phi(A)]\ \ y \overset{\text{LL(k)}}{\rightsquigarrow} y'$. Assim, pela regra *choice*.2, temos que $G[p_1 \mid p_2]\ \ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$. Dado que o casamento de $\phi(A)$ é bem sucedido, então $take_k(y) \in FOLLOW_k^G(A)$, e pela regra $var_{LL(k)}$.1 concluímos que $G[A]\ \ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$.

($\Rightarrow$): Esta parte da prova usa indução na altura da árvore de prova dada por $\overset{\text{LL(k)}}{\rightsquigarrow}$. O caso interessante é o da expressão de parsing $p_1 \mid p_2$, cujas regras associadas são *choice*.1 e *choice*.2. Nesta prova usaremos o fato de que G possui estrutura BNF, e portanto toda escolha $G[p_1 \mid p_2]\ \ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$ possui um consequente $G[A]\ \ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$, onde por $var_{LL(k)}$.1 sabemos que $take_k(y) \in FOLLOW_k^G(A)$.

Se a regra *choice*.1 foi usada, temos que $G[p_1]\ \ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$, e pela hipótese de indução $G'[p_1]\ \ xy \overset{\text{PEG}}{\rightsquigarrow} y$. Dado que $take_k(y) \in FOLLOW_k^G(A)$, como $\phi(A)$ casa $take_k(y)$ então $G[\phi(A)]\ \ y \overset{\text{LL(k)}}{\rightsquigarrow} y'$. Pelo lema 4.6.2 sabemos que $G'[\phi(A)]\ \ y \overset{\text{PEG}}{\rightsquigarrow} y'$ e pela regra *not*.1 temos que $G'[\&\phi(A)]\ \ y \overset{\text{PEG}}{\rightsquigarrow} y$. A regra *con*.1 nos dá que $G'[p_1 \,\&\phi(A)]\ \ xy \overset{\text{PEG}}{\rightsquigarrow} y$, e pela regra *ord*.1 concluímos que $G'[p_1 \,/\, p_2]\ \ xy \overset{\text{PEG}}{\rightsquigarrow} y$.

Se *choice*.2 foi usada, então $G[p_2]\ \ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$. Como p_2 casa x, temos que $x \in FIRST_k^G(p_2)$, e dado que $take_k(y) \in FOLLOW_k^G(A)$ sabemos que $G[\phi(A)]\ \ y \overset{\text{LL(k)}}{\rightsquigarrow} y'$. Uma vez que G é $LL(k)$-forte temos que $take_k(xy) \notin (FIRST_k^G(p_1) \otimes_k FOLLOW_k^G(A))$, e portanto o casamento de $p_1\,\phi(A)$ não é bem sucedido quando a entrada possui prefixo $take_k(xy)$.

Como G' é completa, pela contrapositiva temos que $G'[p_1\,\phi(A)]\ \ xy \overset{\text{PEG}}{\rightsquigarrow}$ `fail`, logo também temos que $G'[p_1 \,\&\phi(A)]\ \ xy \overset{\text{PEG}}{\rightsquigarrow}$ `fail`. Dado que $G[p_2]\ \ xy \overset{\text{LL(k)}}{\rightsquigarrow} y$, pela hipótese de indução temos que $G'[p_2]\ \ xy \overset{\text{PEG}}{\rightsquigarrow} y$, e dado que $G[\phi(A)]\ \ y \overset{\text{LL(k)}}{\rightsquigarrow} y'$, pelo lema 4.6.2 e pela regra *not*.1 temos que $G'[\&\phi(A)]\ \ y \overset{\text{PEG}}{\rightsquigarrow} y$. Finalmente, pela regra *con*.1 temos que $G'[p_2 \,\&\phi(A)]\ \ xy \overset{\text{PEG}}{\rightsquigarrow} y$, e pela regra *ord*.2 podemos concluir que $G'[p_1 \,/\, p_2]\ \ xy \overset{\text{PEG}}{\rightsquigarrow} y$. □

Dada uma CFG $LL(k)$-forte G e uma PEG G' obtida a partir de G através da transformação $LL(k)$-PEG, a proposição a seguir nos diz que $L^{CFG}(G) = L^{PEG}(G')$:

Proposição 4.6.4. *Dada uma gramática $LL(k)$-forte $G = (V, T, P, S)$ e uma gramática G' obtida a partir de G usando a transformação $LL(k)$-PEG, temos que $G\ w\$^k \overset{CFG}{\rightsquigarrow} \$^k$ se e somente se $G'\ w\$^k \overset{PEG}{\rightsquigarrow} \$^k$.*

Classe da CFG G	PEG Equivalente	Equivalência
Linear à Direita	G' (transformação ε-End)	$G\ w \overset{\text{CFG}}{\rightsquigarrow} \varepsilon \Leftrightarrow G'\ w \overset{\text{PEG}}{\rightsquigarrow} \varepsilon$
LL(1)	G (com estrutura BNF)	$G\ w\$ \overset{\text{CFG}}{\rightsquigarrow} \$ \Leftrightarrow G\ w\$ \overset{\text{PEG}}{\rightsquigarrow} \$$
$LL(k)$-forte	G' (transformação $LL(k)$-PEG)	$G\ w\$^k \overset{\text{CFG}}{\rightsquigarrow} \$^k \Leftrightarrow G'\ w\$^k \overset{\text{PEG}}{\rightsquigarrow} \$^k$

Tabela 4.1: Equivalência Entre Classes de CFGs e PEGs

Demonstração. ($\Rightarrow$): Dado que $G\ w\$^k \overset{\text{CFG}}{\rightsquigarrow} \$^k$ e que $\$^k \in FOLLOW_k^G(S)$, pelo lema 4.6.1 temos que $G\ w\$^k \overset{\text{LL(k)}}{\rightsquigarrow} \$^k$, e pelo lema 4.6.3 concluímos que $G'\ w\$^k \overset{\text{PEG}}{\rightsquigarrow} \$^k$.

($\Leftarrow$): Dado que $G'\ w\$^k \overset{\text{PEG}}{\rightsquigarrow} \$^k$, pelo lema 4.6.3 temos que $G\ w\$^k \overset{\text{LL(k)}}{\rightsquigarrow} \$^k$, e pelo lema 4.6.1 concluímos que $G\ w\$^k \overset{\text{CFG}}{\rightsquigarrow} \$^k$. □

Na tabela 4.1, podemos ver um resumo da correspondência que apresentamos entre algumas classes de CFGs e PEGs equivalentes.