5
Conclusão

Neste capítulo, vamos discutir alguns trabalhos relacionados e elencar as contribuições deste trabalho.

Embora o surgimento de PEGs tenha gerado um grande interesse prático e muitos geradores de parsers baseados em PEGs estejam disponíveis [Grimm, 2006, Ierusalimschy, 2009, Piumarta, 2007], o interesse acadêmico por PEGs mostrou-se reduzido, de modo que a literatura sobre PEGs é escassa.

Na próxima seção discutimos trabalhos relacionados e na seção 5.2 revemos as contribuições deste trabalho.

5.1
Trabalhos Relacionados

A formalização de PEGs apresentada por Ford foi bastante influenciada pelo trabalho anterior de Birman [Birman, 1970, Birman e Ullman, 1973], que definiu TDPL e GTDPL. Várias das propriedades de TDPL e GTDPL foram depois adaptadas por Ford para PEGs. Se analisarmos as definições desses formalismos, podemos ver que as características fundamentais de PEGs são a noção de falha e o uso de uma escolha ordenada, pois são características que também estão presentes em TDPL e GTDPL. Ao contrário de PEGs, o uso prático de TDPL e GTDPL foi bastante limitado. Segundo Ford [Ford, 2004], isso se deveu em grande parte ao fato de TDPL e GTDPL terem sido apresentados como modelos formais para parsers top-down, e não como formalismos sintáticos que poderiam ser usados para a descrição de parsers top-down.

Ford [Ford, 2002, 2004] apresenta PEGs como um formalismo para descrever linguagens que, ao contrário de CFGs, não permite expressar ambiguidade. Como o próprio Ford nota, embora o uso de PEGs para descrever algumas linguagens seja mais conveniente, nem sempre PEGs são a ferramenta mais adequada, e o uso de um operador de escolha ordenada introduz o problema de determinar corretamente a ordem das alternativas. Em seu trabalho, Ford ressalta a importância de estudar o relacionamento entre CFGs e PEGs, mas não realiza esse estudo nem sugere uma abordagem específica para estabelecer

essa correspondência.

A transformação Π apresentada aqui pode ser facilmente estendida de modo a converter extensões usadas por bibliotecas de casamento de padrões para PEGs. O trabalho de Oikawa et al. [2010] apresenta mais informalmente a transformação Π, sem mostrar uma prova de corretude da transformação, e discute como adaptá-la para converter extensões usadas por bibliotecas de casamento de padrões para PEGs.

Um outro método para converter expressões regulares para PEGs é descrito por Ierusalimschy [Ierusalimschy, 2009]. No método proposto por Ierusalimschy, uma expressão regular é primeiro convertida para um autômato, e em seguida o autômato é convertido para uma PEG. Uma desvantagem dessa abordagem é que ela não permite acomodar extensões usadas por bibliotecas de casamento de padrões, uma vez que não poderíamos representar essas extensões através de autômatos.

Outra tentativa de usar conceitos clássicos de parsers top-down preditivos para definir a linguagem de PEGs foi feita por Redziejowski [Redziejowski, 2009], que definiu as funções $FIRST^G$ e $FOLLOW^G$ para PEGs. Contudo, as funções $FIRST^G$ e $FOLLOW^G$ usadas por Redziejowski permitem estabelecer apenas de forma aproximada qual é a linguagem definida por uma PEG.

Alguns geradores de parsers baseados em PEGs, como Rats! [Grimm, 2006], implementam regras específicas quando uma das alternativas de uma escolha ordenada pode casar a cadeia vazia. No caso de Rats!, não é possível gerar um parser se a gramática possui uma escolha ordenada onde a alternativa que casa a cadeia vazia não é a última.

ANTLR [Parr e Quong, 1995, Parr, 2007] é um gerador de parsers top-down para linguagens $LL(k)$-forte e foi uma das primeiras ferramentas a permitir o uso de predicados sintáticos. Quando predicados sintáticos são usados, ANTLR gera um parser que pode fazer backtracking. Uma outra maneira em ANTLR de gerar um parser que faz backtracking é através do modo de backtracking. Quando esse modo está habilitado, caso não seja possível gerar um parser $LL(k)$-forte a partir de uma dada gramática, é gerado um parser que tenta casar as alternativas de uma escolha em ordem. Nesse caso, os parsers gerados por ANTLR implementam a mesma semântica de PEGs [Parr, 2007, Ford, 2004].

A correspondência entre CFGs lineares à direita e PEGs foi apontada anteriormente por Ierusalimschy [Ierusalimschy, 2009]. Contudo, uma prova mais detalhada dessa correspondência não foi apresentada.

5.2 Contribuições

Apresentamos um estudo sobre PEGs e estabelecemos a sua correspondência com expressões regulares e CFGs lineares à direita e $LL(k)$-forte.

Revimos a formalização original de PEGs dada por Ford, apresentamos uma nova formalização de PEGs baseada em semântica natural, e mostramos a correspondência entre a nossa formalização e a formalização usada por Ford.

Também apresentamos uma formalização de expressões regulares baseada em semântica natural, e definimos a equivalência entre expressões regulares e PEGs. Com base nessa definição de equivalência, discutimos a função Π que transforma uma expressão regular em uma PEG equivalente e provamos a sua corretude. Além disso, mostramos como podemos obter expressões regulares bem formadas.

No estudo da correspondência entre CFGs e PEGs, apresentamos uma nova formalização de CFGs baseada em semântica natural. Essa nova formalização facilitou o estudo da correspondência de CFGs lineares à direita e $LL(k)$-forte com PEGs, pois nos permitiu ver claramente quais são os pontos em comum e quais são as diferenças entre as semânticas de CFGs e de PEGs. Mostramos que a diferença principal entre CFGs e PEGs está na definição das regras que tratam de uma escolha, uma vez que a definição dada para CFGs faz com que o casamento usando a semântica de $\overset{\text{CFG}}{\rightsquigarrow}$ seja não determinístico, enquanto que a definição dada para PEGs faz com que o casamento usando a semântica de $\overset{\text{PEG}}{\rightsquigarrow}$ seja determinístico.

Mostramos como transformar uma CFG linear à direita em uma PEG quase idêntica e que descreve a mesma linguagem que a CFG quando consideramos apenas os casamentos onde toda a entrada é consumida.

Discutimos a interpretação de uma gramática como uma CFG e como uma PEG, e provamos que uma gramática $LL(1)$ sem expressões ε descreve a mesma linguagem quando interpretada como uma CFG e quando interpretada como uma PEG.

Em seguida, provamos que uma gramática $LL(1)$ com uma pequena restrição, a de que somente a última alternativa de uma escolha pode casar a cadeia vazia, descreve a mesma linguagem se a interpretarmos como uma CFG ou como uma PEG.

Mostramos também como converter uma CFG $LL(k)$-forte em uma PEG equivalente que é bastante similar, onde para cada produção $A \rightarrow p_1 \mid p_2$ na CFG, há uma produção $A \rightarrow p_1 \,\&\phi(A) \,/\, p_2 \,\phi(A)$ correspondente na PEG.

Além de provar a equivalência entre algumas classes de CFGs e PEGs, outra contribuição deste trabalho é a abordagem formal que apresentamos para

estudar o relacionamento entre CFGs e PEGs. Esperamos que essa abordagem seja usada para estabelecer a correspondência entre outras classes de CFGs e PEGs equivalentes.