**Compiladores**

**Aula Prática 2**

***Objetivos***

* Se familiarizar com regras e restrições de um *parser* LL(1),
* Compreender o conceito de ambiguidade e
* Experimentar mais com a escrita de gramáticas.

**Tarefa 1 – Criar um diretório de trabalho e copiar kit.**

Descompacte o arquivo lab2.zip. Nele, você tem os arquivos necessários para criar compiladores em COCO/R: Coco.jar, scanner.frame, parser.frame e Compile.java. Este kit tem seis sub-diretórios, cada um para uma das tarefas práticas a seguir (tarefas 2 a 6).

**Tarefa 2 - Eliminar meta-símbolos EBNF**

Abaixo temos a gramática para uma calculadora de quatro funções. Modifique-a (apenas a seção PRODUCTIONS) de forma a fornecer uma gramática equivalente sem usar os símbolos EBNF {} e []. A gramática gerada deve ser LL(1).

COMPILER Calc

CHARACTERS

digit = "0123456789" .

hexdigit = digit + "ABCDEF" .

TOKENS

decNumber = digit { digit } [ "." { digit } ]

| "." digit { digit } .

hexNumber = "$" hexdigit { hexdigit } .

IGNORE '\n' + '\t' + '\r'

PRODUCTIONS

Calc = { Expression "=" } EOF .

Expression = [ "+" | "-" ] Term { "+" Term | "-" Term } .

Term = Factor { "\*" Factor | "/" Factor } .

Factor = Primary { "!" } .

Primary = decNumber | hexNumber | "(" Expression ")" | "abs" "(" Expression ")" .

END Calc.

*Dicas*: para indicar lambda em COCO, você pode escrever regras como ModificadorDeAcesso = "public" | "private" | "protected" | . Mude aos poucos, garantindo LL(1). Use $F se necessário. Note que uma iteração pode ser transformada em uma recursão à direita: Soma = T {+ T} é equivalente a Soma = T RestoSoma. RestoSoma = + T RestoSoma | .

**Tarefa 3 – Rádio!**

Como um exemplo de checagem de restrições LL(1) para uma gramática EBNF, vamos descrever a programação de uma rádio. A cada hora, vários apresentadores conduzem seus programas que incluem músicas e publicidade. Informações são transmitidas por meio de boletins de notícias, descrição do clima e entrevistas. Boletins de notícias consistem de séries homogêneas de estórias enquanto entrevistas consistem de frases intercaladas de apresentadores e entrevistados, com o apresentador introduzindo e concluindo a conversa.

Itens como "propaganda", "chuva", "música" estão na categoria de terminais léxicos que um scanner (o ouvinte :)) vai reconhecer como símbolos chave enquanto ouve a transmissão. Assim, uma gramática para descrever a programação diária seria:

Radio = { Entrevista | BoletimNoticias | "musica" | "propaganda" } EOF .

BoletimNoticias = "propaganda" Noticia { Noticia } [ Clima ] Preenchimento .

Noticia = "Republicanos" [ "Trump" ] | "racismo" | [ "Trump" ] "Republicanos" | "xenofobia"

| "corrupção" | "comentario" "musica" | Acidente .

Entrevista = "apresentador" { "entrevistado" " apresentador" } [ Preenchimento ] .­

Acidente = "colisão" "com" numero "de" "vitimas".

Preenchimento = "musica" | "propaganda" .

Clima = { "chuva" | "nublado" | "ensolarado" | "seco" } .

Analise a gramática em detalhe. Se necessário, remova [ ] e { } para facilitar a análise. Se verificar que ela não é LL(1), modifique-a para fornecer uma gramática LL(1) ou argumente porque isso não é possível.

**Tarefa 4 – Palíndromos**

Palíndromos são strings de caracteres que não se modificam se invertidas como, por exemplo, "Ame a ema" ou "a b b a b b a". A seguir temos exemplos de várias gramáticas que descrevem palíndromos apenas com símbolos *a* e *b*:

(1) Pal = "a" Pal "a" | "b" Pal "b" .

(2) Pal = "a" Pal "a" | "b" Pal "b" | "a" | "b" .

(3) Pal = "a" [ Pal ] "a" | "b" [ Pal ] "b" .

(4) Pal = [ "a" Pal "a" | "b" Pal "b" | "a" | "b" ] .

Quais gramáticas realmente descrevem palíndromos? Se não, porque não? Quais são LL(1)? Você consegue sugerir uma gramática LL(1) pra essa tarefa?

**Tarefa 5 – Pensando um pouco**

Quais destas afirmações são verdadeiras:

* Uma gramática LL(1) não pode ser ambígua.
* Uma gramática não LL(1) deve ser ambígua.
* Uma linguagem ambígua não pode ser descrita por uma gramática LL(1).
* É possível encontrar uma gramática LL(1) para qualquer linguagem não ambígua;

**Task 6 – Mais uma linguagem**

Escreva uma gramática para uma linguagem com as seguintes regras de sintaxe:

0. Os caracteres da linguagem são as 26 letras do alfabeto e há distinção de maiúsculas e minúsculas.

1. Toda letra minúscula é uma sentença correta.

2. Se σ é uma sentence correta, também é correta uma sentença iniciada por uma vogal maiuscula seguida por σ, como Aσ Eσ Iσ Oσ ou Uσ

3. Se σ e τ são sentenças corretas, também são corretas as sentenças começadas por uma consoante maiuscula seguida por στ, como Bστ Cστ Dστ ... Zστ.

4. Um caractere especial “.” é usado para terminar uma sequencia de sentenças.

**Algumas consideração práticas sobre Coco/R**

*Convenções de nomes com JAVA e COCO*

* Não use nomes de diretórios com espaços.
* Use nomes curtos para símbolos (ex, CIf em lugar de ComandoIf);
* Lembre que o nome da gramática é o nome do arquivo e deve aparecer após COMPILER, como a primeira regras e depois de END;
* Gramáticas devem ser salvas em arquivos com extensão atg.
* Armazene todos os arquivos associados a uma gramática em um mesmo subdiretório

*Checagem de Erros*

Checagem de erros em COCO/R se dá em dois estágios. O primeiro está relacionado com erros de sintaxe – como esquecer o ponto depois da produção. Esses são fáceis de corrigir. O segundo corresponde a não terminais que não são definidos, ou não podem ser alcançados, ou são cíclicos, inúteis ou que não satisfazem as restrições LL(1). Nós só vamos nos concentrar em restrições LL1 mais tarde e, pra esse laboratório, o jeito é torcer pra elas não serem muito chatas. A forma mais comum de violar LL1 é ter alternativas para um não terminal que inicia com a mesma string. Assim, um parser LL1 não é capaz de decidir que alternativa seguir, como no exemplo:

assignment = variableName ":=" expression

| variableName index ":=" expression.

index = "[" subscript "]".

Ambas alternativas para assignment começam com variableName. Contudo, é fácil escrever produções que não tem esse problema:

assignment = variableName [ index ] ":=" expression.

index = "[" subscript "]".

Note também que qualquer expressão escrita usando recursão pela esqueda, como abaixo:

expression = term | expression "," term.

violam LL(1) e deveriam ser re-escritas como:

expression = term {"," term}.

Por enquanto, se você encontrar restrições deste tipo, as mostre ao professor que pode tentar ajuda-los em resolvê-las.

**Respostas**

**Tarefa 2**

COMPILER Calc1 $F

CHARACTERS

digit = "0123456789" .

hexdigit = digit + "ABCDEF" .

TOKENS

decNumber = digit { digit } [ "." { digit } ]

| "." digit { digit } .

hexNumber = "$" hexdigit { hexdigit } .

IGNORE '\n' + '\t' + '\r'

PRODUCTIONS

Calc1 = Expression "=" Calc1 | .

Expression = ( AddOp | ) Term Soma .

Soma = AddOp Term Soma | .

Term = Factor Mult .

Mult = MulOp Factor Mult | .

Factor = Primary FactSym .

FactSym = "!" FactSym | .

Primary = decNumber | hexNumber | "(" Expression ")" | "abs" "(" Expression ")" .

AddOp = "+" | "-".

MulOp = "\*" | "/".

END Calc1.

**Tarefa 3**

COMPILER Radio $F

CHARACTERS

digito = "0123456789".

TOKENS

numero = digito {digito}.

IGNORE '\n' + '\t' + '\r'

PRODUCTIONS

Radio = { Entrevista | "musica" | "propaganda" [ RestoBNot ] } EOF .

RestoBNot = Not { Not } { Clima } Preenchimento .

Not = "Republicanos" [ "Trump" ] | "racismo"

| [ "Trump" ] "Republicanos" | "xenofobia"

| "corrupcao" | "comentario" "musica" | Acidente .

Entrevista = "apresentador" { "entrevistado" "apresentador" } .

Acidente = "colisao" "com" numero "de" "vitimas".

Preenchimento = "musica" | "propaganda" .

Clima = "chuva" | "nublado" | "ensolarado" | "seco" .

END Radio.

Outra solução para o Clima poderia ser:

RestoBNot = Not { Not } [ Clima ] Preenchimento .

Clima = EventoClima { EventoClima } .

EventoClima = "chuva" | "nublado" | "ensolarado" | "seco" .

Neste caso, estamos dizendo que Clima é opcional no resto do boletim de notícias, mas então a definição de Clima requer que ao menos um evento climático seja descrito.

**Tarefa 4**

(1) Pal = "a" Pal "a" | "b" Pal "b" .

(2) Pal = "a" Pal "a" | "b" Pal "b" | "a" | "b" .

(3) Pal = "a" [ Pal ] "a" | "b" [ Pal ] "b" .

(4) Pal = [ "a" Pal "a" | "b" Pal "b" | "a" | "b" ] .

- As gramáticas são: (1) incompleta, pois não define caso base; (2) palíndromo ímpar que viola R1; (3) palíndromo par, que viola R2; (4) palíndromo qualquer, que viola R1 e R2;

- Apenas (1) é LL1, porém incompleta;

- Para reconhecer um palíndromo de tamanho arbitrário N é necessário saber onde é o meio da sentença (N/2). Logo, qualquer desambiguação necessita resolver N/2 símbolos. Uma gramática LL-k, com k fixado não pode ser usada para isso. Na prática, as linguagens LL-k incluem todas as regulares, mas são apenas um sub-conjunto das livres de contexto. De um certo modo, se pensarmos que lambda é um palíndromo e um palíndromo, por definição, deve começar com um símbolo que o segue (por causa da simetria), o palíndromo por definição deve violar R2.

**Tarefa 5**

Quais destas afirmações são verdadeiras:

* Uma gramática LL(1) não pode ser ambígua. VERDADE
* Uma gramática não LL(1) deve ser ambígua. FALSO (vimos o palíndromo)
* Uma linguagem ambígua *não* pode ser descrita por uma gramática LL(1). VERDADE
* É possível encontrar uma gramática LL(1) para qualquer linguagem não ambígua; FALSO (vimos o palíndromo)

O palíndromo é não ambíguo. Ideia de prova: o palíndromo requer uma árvore de derivação simétrica e não há duas árvores de derivação simétrica que não sejam a mesma árvore.

**Tarefa 6**

COMPILER Mul

CHARACTERS

minuscula = 'a'..'z'.

vogal = "AEIOU".

consoante = "BCDFGHJKLMNPQRSTVWXYZ".

TOKENS

letraMin = minuscula.

vogalM = vogal.

consoanteM = consoante.

IGNORE '\n' + '\t' + '\r'

PRODUCTIONS

Mul = { { Sentenca } "." } EOF .

Sentenca = letraMin | vogalM Sentenca | consoanteM Sentenca Sentenca.

END Mul.