Segunda Prova de MAB 471 2013.2 — Compiladores I

Fabio Mascarenhas

11 de Dezembro de 2013

A prova é individual e sem consulta. Responda as questões na folha de respostas, a lápis ou a caneta. Se tiver qualquer dúvida consulte o professor.

Nome: _____

DRE: _____

Questão:	1	2	Total
Pontos:	4	6	10
Nota:			

1. Considere a gramática a seguir:

```
S -> E
E -> E + E | E - E | E * E | ( E ) | num
```

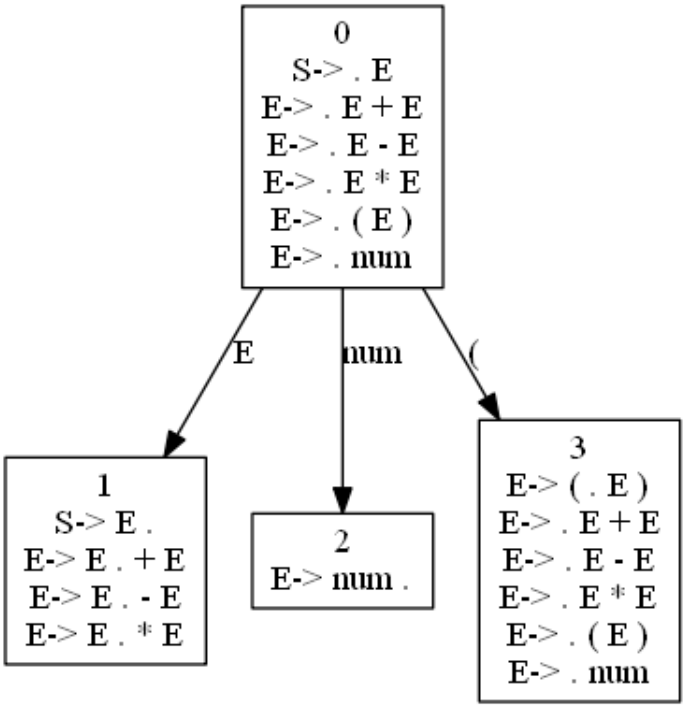
   (a) (2 pontos) Dê **duas** sequência de ações (shift, reduce, accept) do analisador LR para o termo `num + num * num`. Não é preciso dar o número dos estados nas ações de shift. Mostre como essas duas sequências de ações provam que a gramática é ambígua.

   Resposta: Primeira sequência: shift, reduce `E -> num`, shift, shift, reduce `E->num`, reduce `E -> E + E`, shift, shift, reduce `E -> num`, reduce `E -> E * E`, reduce `S -> E`, accept.

   Segunda sequência: shift, reduce `E -> num`, shift, shift, reduce `E->num`, shift, shift, `E->num`, reduce `E -> E * E`, `E -> E + E`, reduce `S -> E`, accept.

   As reduções de cada sequência, quando lidas de trás para frente, dão duas derivações mais à direita diferentes para o termo dado, logo a gramática é ambígua.

   (b) (2 pontos) Dê o estado inicial do autômato LR(0) dessa gramática, as transições que saem desse estado, e os estados alvo dessas transições.



Resposta:

2. Vetores são um tipo de dado bastante comum em linguagems de programação. Eles são um tipo *derivado*: se τ é um tipo qualquer da linguagem, $\tau[]$ é um vetor em que cada elemento tem tipo τ. A operação principal em um vetor é a *indexação*, que pode ser para leitura ou escrita.

```
class TipoVetor implements Tipo {
  Tipo tipoElem;
}
```

Na indexação de leitura, uma expressão que deve ter o tipo $\tau[]$ é indexada por uma expressão de tipo inteiro, e o resultado é um valor do tipo τ (assumindo que o índice está dentro dos limites do vetor), o elemento correspondente ao índice.

```
interface Exp {
  Tipo tipo(TabSimb<Tipo> vars);
  void codigoVal(Contexto c, TabSimb<Tipo> vars);
}

class IndexaLe implements Exp {
  Exp vetor;
  Exp indice;
}
```

Na indexação de escrita, uma expressão que deve ter tipo $\tau[]$ e indexada por uma expressão de tipo inteiro está do lado esquerdo de uma atribuição, e uma expressão do tipo τ deve estar do lado direito. O resultado é substituir o elemento na posição dada pelo índice pelo resultado do lado direito da atribuição.

```
interface Cmd {
  void tipos(TabSimb<Tipo> vars);
```

```
  void codigo(Contetxo c, TabSimb<Tipo> vars);
}

class IndexaEscreve implements Cmd {
  Exp vetor;
  Exp indice;
  Exp ldir;
}
```

(a) (3 pontos) Implemente a verificação de tipos para as operações de indexação de vetores, dados os fragmentos da AST acima. Assuma que um tipo inteiro é i

Resposta:

```
// IndexaLe
Tipo tipo(TabSimb<Tipo> vars) {
  Tipo tv = vetor.tipo(vars);
  if(!tv instanceof TipoVetor)
    throw new RuntimeException("expressão não é vetor");
  Tipo ti = indice.tipo(vars);
  if(!ti instanceof TipoInteiro)
    throw new RuntimeException("índice não é inteiro");
  return ((TipoVetor)tv).tipoElem;
}

// IndexaEscreve
void tipos(TabSimb<Tipo> vars) {
  Tipo tv = vetor.tipo(vars);
  if(!tv instanceof TipoVetor)
    throw new RuntimeException("expressão não é vetor");
  Tipo ti = indice.tipo(vars);
  if(!ti instanceof TipoInteiro)
    throw new RuntimeException("índice não é inteiro");
  if(!((TipoVetor)tv).tipoElem.equals(ldir.tipo(vars))
    throw new RuntimeException("tipos não batem");
}
```

(b) (3 pontos) Implemente a geração de código para as operações de indexação de vetores, dados os fragmentos de AST acima. Assuma que o objeto `Contexto` tem duas operações com vetores: `iloadv()` retira da pilha um vetor e um índice, e empilha o elemento que está naquele índice, enquanto `istorev()` retira da pilha um vetor, um índice e outro valor e guarda esse valor no vetor, na posição dada pelo índice.

Resposta:

```
// IndexaLe
void codigoVal(Contexto c, TabSimb<Tipo> vars) {
  // Considerei quem trocou a ordem
  vetor.codigoVal(c, vars);
  indice.codigoVal(c, vars);
  c.iloadv();
}

// IndexaEscreve
```

```
void codigo(Contetxo c, TabSimb<Tipo> vars) {
  // Considerei quem trocou a ordem
  vetor.codigoVal(c, vars);
  indice.codigoVal(c, vars);
  ldir.codigoVal(c, vars);
  c.istorev();
}
```

BOA SORTE!