Exame EAPLI 2014

Humberto Carvalho

July 17, 2014

Part I

1. O ANTLR gera um analisador parsico top down LL(*)em java ou em C#. O bison gera um analisador parsico bottom up LALR(1).

   O Bison gera um parser cuja logica de processamento esta contido na informacao do programa e nao no codigo do parser, tendo uma footprint de memoria inferior.

   O ANTLR gera parsers descendentes recursivos, cuja logica de processamento esta no codigo do parser, sao geralmente mais rapidos do que os parsers de tabela.

2. (a) Existem dois tipos de derivacao, a derivacao canonica mais a esquerda substitui o nao terminal mais a esquerda enquanto a derivacao canonica a direita substitui para reescrita o nao terminal mais a direita.

   (b) $S \longrightarrow aSA \longrightarrow aaSA \longrightarrow aa\varepsilon A \longrightarrow aabA \longleftrightarrow aab\varepsilon \longrightarrow aab$
   $S \longrightarrow aSA \longrightarrow aaSba \longrightarrow aaSb\varepsilon \longrightarrow aa\varepsilon b\varepsilon \longrightarrow aab$

3. Uma gramatica de atributos define como o conjunto de atributos e regras semanticas que expressam como a computacao dos atributos se relacionam com as regras gramaticais da linguagem.

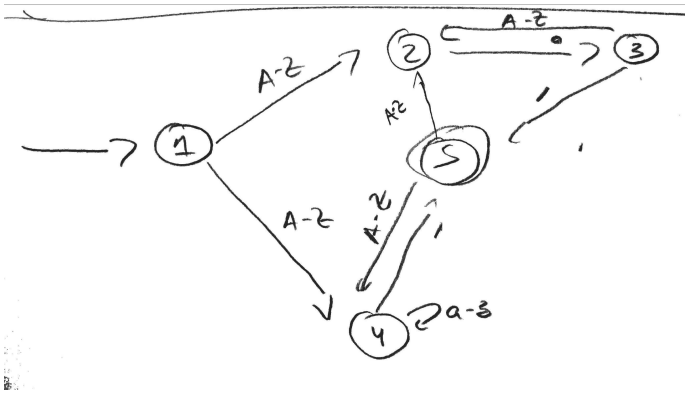
   A gramatica de atributos e utilizada na analise semantica, e definem dependencias entre atributos, se na totalidade do conjunto de acoes de uma gramatica apenas houver atributos que dependam dos filhos, essa gramatica e S atribuida e os atributos sao sintetizados. Se houver dependencias entre atributos de simbolos do lado direito das regras(filhos) ou estes dependerem de de atributos do nao terminal do lado esquerdo(pai), estes sao herdados. Se, para os atributos herdados, estes dependerem apenas de atributos a sua esquerda, ou do pai, a gramatica e L Atributed.

4. O backend e responsavel pela parte de sintese, onde e construido o programa-alvo a partir do codigo intermedio. O facto de existir uma representacao intermedia simplifica a producao de compiladores pois podemos tero utra linguagem, ao trocarmos o front end podemos compilar para outra arquitetura.O front end e tambem responsavel por optimizar o codigo final, nomeadamente a optimizacao peephole onde sao trocadas sequencias por outras mais eficientes.

Part II

1. (a) $([A-Z][a-z]+(/\ [A-Z][a-z]+)*(:\ [A-Z][a-z]+(/\ [A-Z][a-z]+)*)?(,vol.[0-9]+)?,)$
   |
   $([A-Z][a-z]+(/\ [A-Z][a-z]+)*(,[0-9]+ndedition)?,)$

   (b) Diagram



   (c) Automato finito nao determistisco, pois para certos nos existem dois caminhos possiveis para o mesmo input, que e o caso do estado 1.

2.

	x	y	z
(1) $\longrightarrow S0$	S0(1)	S1(1)	S0(1)
(1) S1	S0(1)	S1(1)	S2(1)
(1) S2	S3(2)	S1(1)	S0(1)
(2) *S3	S3(2)	S1(1)	S0(1)

	x	y	z
(1) $\longrightarrow S0$	S0(1)	S1(1)	S0(1)
(1) S1	S0(1)	S1(1)	S2(2)
(2) S2	S3(3)	S1(1)	S0(1)
(3) *S3	S3	S1	S0

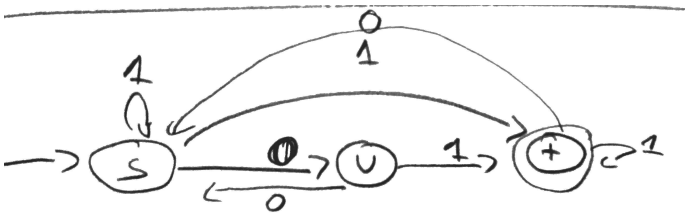
	x	y	z
(1) $\longrightarrow S0$	S0(1)	S1(2)	S0(1)
(2) S1	S0(1)	S1(2)	S2(3)
(3) S2	S3(4)	S1(2)	S0(1)
(4) *S3	S3	S1	S0

Part III

1. (a) $G = (\{S, A, B\}, \{x, y, z\}, F, S)$
   onde F,
   $S \longrightarrow xAx \mid yAy \mid zAz,$
   $A \longrightarrow xB \mid yB \mid zB \mid \varepsilon,$
   $B \longrightarrow xA \mid yA \mid zA$

(b) Gramatica do tipo 2 pois as suas producoes sao do tipo $x \longrightarrow y$, onde x e um unico nao terminal e y uma sequencia de terminais e nao terminais.

2. A gramatica e do tipo 3, regular pois as suas producoes sao do tipo $A \longrightarrow aB$ ou $A \longrightarrow Ba$.



Part IV

1.

```
<xsd:complexType name="TMarca">
      <xsd:sequence>
          <xsd:element name="Modelo" maxOccurs="unbounded"
  type="TModelo"/>
      </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="TModelo">
      <xsd:sequence>
          <xsd:element name="Descricao" type="TDescricao"/>
          <xsd:element name="Mecanica" type="TMecanica"/>
          <xsd:element name="Chassis" type="TChassis"/>
          <xsd:element name="Prestacoes" type="TPrestacoes"/
  >
      </xsd:sequence>
      <xsd:attribute name="TNome" use="required" type="
  xsd:string"/>
      <xsd:attribute name="codigo" use="required" type="
  TCodigo"/>
      <xsd:attribute name="stock" use="required" type="
  xsd:nonNegativeInteger"/>
      <xsd:attribute name="preco" use="required" type="
  TPreco"/>
  </xsd:complexType>

  <xsd:simpleType name="TDescricao">
      <xsd:restriction base="xsd:string">
          <xsd:maxLength value="50"/>
      </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="TNome">
      <xsd:restriction base="xsd:string">
          <xsd:pattern value="[A-Za-z]{1,15}"/>
      </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="TCodigo">
      <xsd:restriction base="xsd:string">
          <xsd:pattern value="[A-Za-z]{2}[0-9]{2,3}"/>
      </xsd:restriction>
```

```
    </xsd:simpleType>

    <xsd:simpleType name="TPreco">
        <xsd:restriction xml:base="xsd:integer">
            <xsd:minInclusive value="5000"/>
        </xsd:restriction>
    </xsd:simpleType>

```

2.

```
/Stand/Marca/Modelo[Mecanica/@combustivel=gasoleo and
  Prestacoes/Consumos/@combinado<5.5]

```

3.

```
<xsl:template match="/">
        <html>
            <body>
                <h1>Stand Automovel</h1>

                Total de automoveis em stock: <xsl:value-of
    select="sum(Stand/Marca/Modelo/@stock)"/>
                <xsl:apply-templates select="Stand/Marca[sum(
    Modelo/@stock)>2]">
                    <xsl:sort select="@nome" data-type="text"/>
                </xsl:apply-templates>
            </body>
        </html>
    </xsl:template>

    <xsl:template match="Marca">
        <h3>Marca: Honda</h3>
        <table border="1">
            <tr>
                <td></td>
                <td><b>Modelo</b></td>
                <td><b>Descricao</b></td>
                <td><b>Preco</b></td>
                <td><b>#Unidades</b></td>
                <td><b>Total</b></td>
            </tr>
        <xsl:apply-templates select="Modelo[@stock]">
            <xsl:sort select="@nome" data-type="text"/>
        </xsl:apply-templates>
        </table>

    </xsl:template>

    <xsl:template match="Modelo">
         <tr>
            <td>
                <xsl:value-of select="position()"/>
            </td>
             <td>
                 <xsl:value-of select="@nome"/>
            </td>
            <td>
                 <xsl:value-of select="Descricao"/>
            </td>
```

```
43              <td>
                    <xsl:value-of select="@preco"/>
45              </td>
                <td>
47                  <xsl:value-of select="@stock"/>
                </td>
49              <td>
                    <xsl:value-of select="@preco*@stock"/>
51              </td>
            </tr>
53      </xsl:template>
```