Processamento de Linguagens (3º ano de MIEI)

Rede Semântica do Museu do Artista

Relatório de Desenvolvimento (TP3)

Luís Alves (a80165)

Rafaela Rodrigues (a80516)

10 de Junho de 2019

Resumo

Este relatório inicia-se com uma breve contextualização, seguindo-se a declaração dos objetivos deste trabalho e em que é que este consiste. De seguida é feita uma descrição informal do problema cuja resolução é apresentada no capítulo seguinte, sendo especificada a linguagem desenvolvida, bem como as estruturas de dados auxiliares ao processamento de um texto fonte dessa linguagem. Termina-se esse capítulo com o método utilizado para produzir o grafo de navegação conceptual do repositório. Por fim, são apresentados alguns testes realizados com base em textos fonte produzidos por nós, bem como os respetivos resultados, sendo feita uma análise e tiradas conclusões sobre o desenvolvimento do programa após esta secção.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

O presente relatório foi elaborado no âmbito do terceiro Trabalho Prático da Unidade Curricular de Processamento de Linguagens, que se insere no 2º semestre do 3º ano do primeiro ciclo de estudos do Mestrado Integrado em Engenharia Informática.

1.2 Objetivos e Trabalho Proposto

Para este trabalho foi proposta a especificação de uma gramática concreta, acompanhada de um reconhecedor léxico e sintático com recurso ao par de ferramentas geradoras `Flex/Yacc`. Para além disso, deverá ser desenvolvido um gerador de código associando ações semânticas de tradução às produções da gramática.

1.3 Resumo do trabalho a desenvolver

O programa a desenvolver deverá ser o resultado da definição de uma linguagem específica de domínio e da especificação de uma gramática independente do contexto. Assim, o reconhecedor léxico e sintático que acompanham essas especificações deverão ser capazes de gerar um grafo com a ferramenta `GraphViz` e produzir páginas `HTML` que apresentem a informação presente no ficheiro de entrada.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Pretende-se obter um grafo que permita uma navegação conceptual sobre um repositório de conhecimento, que se encontra definido numa linguagem a especificar. Esse repositório será relativo a **Artistas de Cinema**. Assim, é necessário cumprir os seguintes requisitos:

1. Definir uma gramática independente de contexto (em `Yacc`);
2. Desenvolver um analisador léxico (em `Flex`);
3. Associar ações semânticas na gramática que produzam um grafo (usando `GraphViz`);
4. Validar os relacionamentos e nodos do repositório;
5. Especificar diversos textos-fonte válidos e inválidos.

Capítulo 3

Conceção/desenho da Resolução

3.1 Nodos e relacionamentos

O repositório de conhecimento poderá conter 3 tipos de nodos distintos e 4 relacionamentos, sendo que estes últimos se podem classificar em 3 tipos: artista-obra, artista-evento e artista-artista.

Na tabela 3.1 são apresentados os atributos de cada um dos nodos.

De seguida, é apresentado o dicionário de dados de cada um dos atributos.

1. Id - identificativo único do nodo. Deve iniciar-se pela primeira letra do tipo do nodo, seguida de um número (Ex: a802 - ID de um artista).
2. Data - ano de realização de um evento ou obra. Deve ser um valor inteiro não negativo (Ex: 1962).
3. Idade - idade atual de um artista. Deve ser um valor inteiro não negativo (Ex: 72).
4. Nome - nome de um evento, obra ou artista. É uma *string* (Ex: Marlon Brando).
5. Edição - número da edição de um evento. Deve ser um valor inteiro (Ex: 5).
6. Duração - número de minutos que uma determinada obra demora. Deve ser um valor inteiro não negativo (Ex: 120).
7. Formação - informação sobre o percurso educativo de um artista. É uma *string* (Ex: Universidade de Medellin).
8. Tipo - representa a categoria do evento. É uma *string* (Ex: Festival).
9. Género - representa a categoria em que se insere um filme. É uma *string* (Ex: Documentário)

Evento	**Obra**	**Artista**
Id	Id	Id
Data	Data	Idade
Nome	Nome	Nome
Edição	Duração	Formação
Tipo	Género	Produtoras

Tabela 3.1: Atributos por nodo

10. Produtoras - informação acerca das produtoras com as quais um artista trabalhou. É uma *string* (Ex: Disney).

Os relacionamentos previstos para o repositório são os seguintes:

1. Artista-obra
   (a) Produziu - indica obra em que o artista participou
2. Artista-evento
   (a) Participou em - indica evento a que o artista marcou presença
3. Artista-artista
   (a) Aprendeu com - indica artista que foi mentor de outro artista
   (b) Colaborou com - indica artista que trabalhou com outro artista

3.2 Estruturas de Dados

De forma a validarmos os dados, foi necessário definir:

- 3 *Arrays* - (Atributo) -> (Bool);
- 2 *Strings*;
- 1 *Hash Table* - (Id) -> (Bool).

Estas estruturas são usadas para as funcionalidades que a seguir se apresentam.

Arrays

Para cada tipo de nodo, existe um *array* de controlo que verifica a existência de um determinado atributo. Assim, quando é processado um novo atributo, é possível verificar se este já havia sido definido para o nodo em questão. Cada um dos *arrays* tem tamanho igual ao número máximo de atributos diferentes por nodo.

Strings

Para ser possível colocar uma *label* em cada nodo no grafo final, é necessário armazenar o valor do seu Nome e do seu Id enquanto os seus atributos são processados.

Hash Table

De forma a impedir a inserção de nodos com Ids repetidos, sempre que um novo nodo é processado o seu Id é armazenado numa tabela de *Hash*, para consultas futuras relativas à sua existência no repositório de conhecimento.

3.3 Gramática Independente de Contexto

Tendo em conta o que foi apresentado anteriormente, foi possível definir uma gramática que permite expressar o conhecimento pretendido. É a que se apresenta de seguida.

```
MvA: LConteudos

LConteudos: Conteudo
          | LConteudos Conteudo

Conteudo: Nodo
        | Relacao '.'

Nodo: ARTISTA '{' LTagArtista '}'
    | OBRA '{' LTagObra '}'
    | EVENTO '{' LTagEvento '}'

Nome: NOME ':' STRING

LTagArtista: ID_TAG ':' ID_ARTISTA
           | LTagArtista ',' TagArtista

TagArtista: IDADE ':' INT
          | FORMACAO ':' STRING
          | PRODUTORAS ':' STRING
          | Nome

LTagObra: ID_TAG ':' ID_OBRA
        | LTagObra ',' TagObra

TagObra: Data
       | GENERO ':' STRING
       | DURACAO ':' INT
       | Nome

Data: DATA ':' INT

LTagEvento: ID_TAG ':' ID_EVENTO
          | LTagEvento ',' TagEvento

TagEvento: Data
         | EDICAO ':' INT
         | TIPO ':' STRING
         | Nome

Relacao: ID_ARTISTA RArtistaObra ID_OBRA
       | ID_ARTISTA RArtistaEvento ID_EVENTO
       | ID_ARTISTA RArtistaArtista ID_ARTISTA
```

```
RArtistaObra: PRODUZIU

RArtistaEvento: PARTICIPOU

RArtistaArtista: APRENDEU
               | COLABOROU
```

Para esta gramática, o símbolo inicial é MvA, os símbolos terminais são escritos só em maiúsculas ou entre apóstrofes e os restantes (começados por maiúsculas) são os símbolos não-terminais.

3.3.1 Símbolos Terminais

Existem 17 palavras reservadas na linguagem, sendo elas: `Artista`, `Obra`, `Evento`, `Nome`, `Idade`, `Formacao`, `Produtoras`, `Data`, `Genero`, `Duracao`, `Edicao`, `Tipo`, `produziu`, `aprendeuCom`, `colaborouCom`, `participouEm` e `Id`. Para além destas, existem símbolos entre apóstrofes que delimitam atributos, *strings*, nodos e relacionamentos, sendo os que se apresentam de seguida:

1. `.` - indica o fim de um relacionamento
2. `,` - indica o fim de um atributo, quando um nodo possui mais que um atributo
3. `:` - indica que o que precedeu foi o tipo de atributo, e o que se segue é o seu valor
4. {} - são os delimitadores dos atributos de um nodo

Para além destes símbolos terminais fixos, existem 5 que são variáveis. São os que se apresentam de seguida:

1. `INT`
   (a) `[0-9]+`
   (b) Número inteiro não negativo
2. `ID_ARTISTA`
   (a) `a[0-9]+`
   (b) Id de um artista
3. `ID_OBRA`
   (a) `o[0-9]+`
   (b) Id de uma obra
4. `ID_EVENTO`
   (a) `e[0-9]+`
   (b) Id de um evento
5. `STRING`
   (a) `"[^"]+"`
   (b) Conteúdo de um atributo do tipo *string*, que se encontra entre aspas

3.4 Produção do Grafo

De forma a ser possível gerar o grafo que permite a navegação conceptual sobre o repositório de conhecimento, recorreu-se à ferramenta `GraphViz`. Assim, para a produção do grafo, o algoritmo é o seguinte:

```
enquanto (encontra conteudo)
    se (nodo)
        linha = id [URL="id.html", label="nome"];
    se (relacionamento)
        linha = id_esquerda -- id_direita [label="tipo de relacionamento"];
```

Assim, sempre que um nodo é processado, é gerada uma página HTML que contém todos os atributos do nodo, e cujo nome é o Id desse nodo. Para além disso, é adicionado ao ficheiro em formato `DOT` uma linha que identifica o nodo pelo seu Id, e que adiciona um URL para a página HTML gerada, de forma a ser possível clicar no nodo e ser redirecionado para essa página.
Relativamente ao relacionamento, é adicionada uma linha ao ficheiro `DOT` que permite estabelecer a ligação binária entre os nodos envolvidos.

Capítulo 4

Codificação e Testes

4.1 Geração do grafo

De seguida apresenta-se o resultado de executar o programa passando como *input* o anexo A.0.3, que possui 5 artistas, 4 obras, 4 eventos e 20 relacionamentos.

As primeiras 3 imagens mostram as informações de cada tipo de nodo e a quarta imagem o grafo criado tendo como base os nodos e relacionamentos do texto-fonte.

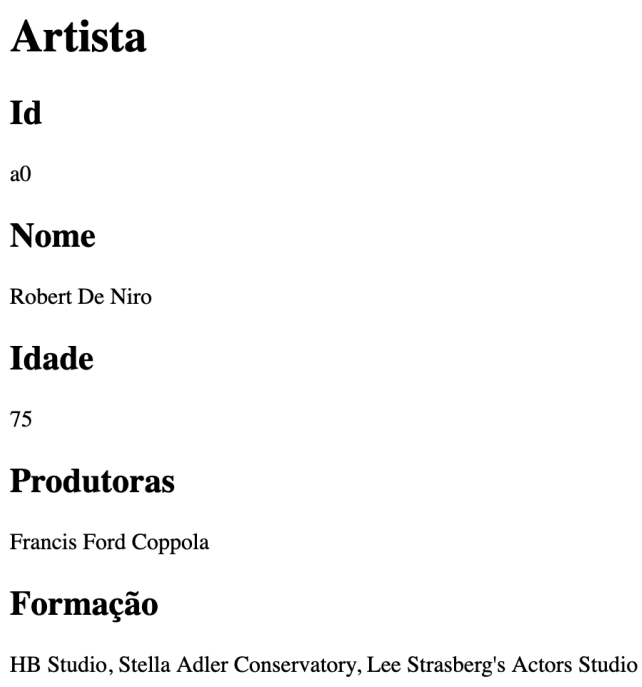

Figura 4.1: Informação sobre um artista

Obra

Id

o2

Nome

The Godfather Part II

Data

1974

Género

Crime

Duração

200

Figura 4.2: Informação sobre uma obra

Evento

Id

e0

Nome

The 47th Academy Awards

Data

1975

Tipo

Festival

Edição

47

Figura 4.3: Informação sobre um evento

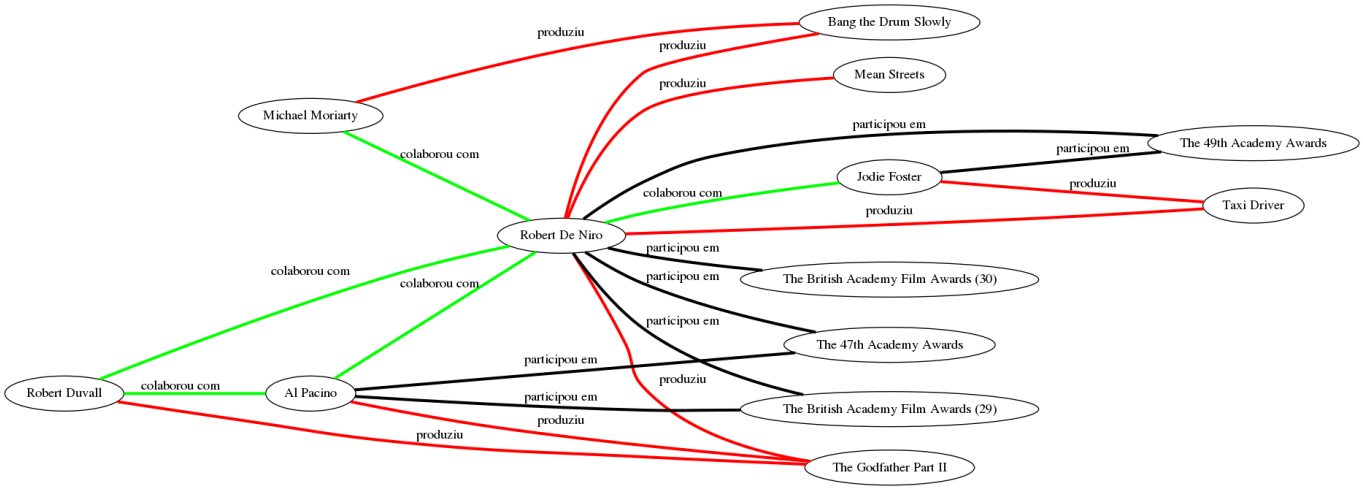

Figura 4.4: Grafo gerado

4.2 Exemplos de erros sintáticos

Com o intuito de testar o bom funcionamento do programa e a deteção correta de erros sintáticos, foram criados textos-fonte contendo esses erros, sendo alguns exemplos apresentados de seguida.

Ids repetidos

Um dos erros possíveis é dois artistas possuírem o mesmo Id.

```
artista {
    id: a23,
    nome: "Luis",
    idade: 22,
    formacao: "ESCA"
}
artista {
    id: a23,
    nome: "Rafaela",
    idade: 21,
    formacao: "ESAS"
}
```

Atributo com tipo de conteúdo errado

Este erro consiste em representar a duração com uma *string*, quando seria esperado um inteiro não negativo.

```
obra {
    id: o0,
    nome: "Aladdin",
    data: 1992,
    genero: "Animação",
```

```
    duracao: "uma hora e trinta"
}
```

Atributo Inexistente

Neste exemplo, é apresentado o atributo `apresentador`, sendo que um evento não prevê a existência deste.

```
evento {
    id: e0,
    tipo: "Entrega de Prémios",
    data: 2012,
    edicao: 24,
    apresentador: "Barack Obama"
}
```

Atributo Repetida

Outro erro possível é dentro do mesmo nodo haver dois ou mais atributos repetidos.

```
artista {
    id: a201,
    nome: "Margot",
    nome: "Robbie",
    idade: 28,
    formacao: "New Zealand Art College"
}
```

Relacionamento incompleto

Neste exemplo temos um relacionamento em que não é explicitado o segundo Id.

```
artista {
    id: a245,
    nome: "Chris Evans",
    idade: 38,
    formacao: "Hollywood"
}

obra {
    id: o83,
    nome: "Avengers: Endgame ",
    data: 2019,
    genero: "Action",
    duracao: 181
}

a245 participouEm
```

Relacionamento com Id inexistente

Neste caso, é referida uma obra que não existe.

```
artista {
    id: a78,
    nome: "Christopher Nolan",
    idade: 48,
    formacao: "Vida"
}

a78 produziu o345.
```

Apesar de aqui se listar um grande número de erros sintáticos possíveis, existem muitos mais que não são aqui apresentados, e que estão relacionados com a forma como o conteúdo é escrito, e não com o conteúdo em si.

Capítulo 5

Conclusão

Ao longo do desenvolvimento da solução para o problema apresentado, fomos tomando várias decisões sobre a sua implementação, o que resultou num produto final mais modular e capaz de ser expandido, quer seja através de novos tipos de relacionamentos ou de atributos e nodos. Isto deveu-se a uma especificação aprimorada da gramática da linguagem definida, o que nos permite concluir que esta definição é a peça central do trabalho.

Relativamente às ações semânticas associadas, não foi necessária a elaboração de estratégias muito rebuscadas para produzir os resultados pretendidos, uma vez que estes eram simples e facilmente atingíveis com a gramática e ferramentas disponíveis.

Assim sendo, concluímos que o programa final cumpre com todos os requisitos enunciados.

Apêndice A

Código do Programa

A.0.1 gramatica.y

```
%{
#define _GNU_SOURCE
#include <stdio.h>
#include <string.h>
#include "lex.yy.c"
#include <glib.h>

int yyerror(char * s);

/* Nº de tags de cada nodo */
#define N_ARTISTA_TAGS 5
#define N_EVENTO_TAGS 5
#define N_OBRA_TAGS 5

#define UNUSED 0
#define USED 1

/* Posições da Tags de Artista */
#define cID 0
#define cNOME 1
#define cIDADE 2
#define cFORMACAO 3
#define cPRODUTORAS 4

/* Posições das Tags de Evento */
#define cDATA 2
#define cTIPO 3
#define cEDICAO 4

/* Posições das Tags de Obra */
#define cGENERO 3
#define cDURACAO 4
```

```
/* Tipos de Nodos */
#define cARTISTA 0
#define cEVENTO 1
#define cOBRA 2

int artista[N_ARTISTA_TAGS];
int evento[N_EVENTO_TAGS];
int obra[N_OBRA_TAGS];

/* used to store the current ID */
char * id = NULL;
/* used to store the current name */
char * nome = NULL;

/* Table of IDs */
GHashTable * idHash;

/**
  * Clears the array of used tags in artista
 */
void clearArtista() {
  for(int i = 0; i < N_ARTISTA_TAGS; i++)
    artista[i] = UNUSED;
}

/**
  * Clears the array of used tags in evento
 */
void clearEvento() {
  for(int i = 0; i < N_EVENTO_TAGS; i++)
    evento[i] = UNUSED;
}

/**
  * Clears the array of used tags in obra
*/
void clearObra() {
  for(int i = 0; i < N_OBRA_TAGS; i++)
    obra[i] = UNUSED;
}

/**
  * Marks the tag of a given a node as used
  * @return 1 if was already used, 0 if not
*/
int useTagNode(int tag, int node) {
  int * usedArray;
  int maxSize = 0;
```

```
  switch (node) {
    case cARTISTA: usedArray = artista;
                   maxSize = N_ARTISTA_TAGS;
                   break;
    case cEVENTO:  usedArray = evento;
                   maxSize = N_EVENTO_TAGS;
                   break;
    case cOBRA:    usedArray = obra;
                   maxSize = N_OBRA_TAGS;
                   break;
  }

  if (tag >= maxSize)
    return 1;

  if (usedArray[tag] == UNUSED) {
    usedArray[tag] = USED;
    return UNUSED;
  }
  return USED;
}

/**
  * Inserts a new id into the hash, returns whether it already
  * had that ID
 */
int insertId(char * id) {
    char * idCopy = strdup(id);
    int hadKey = g_hash_table_insert(idHash, idCopy, strdup(idCopy));
    return !hadKey;
}

/**
  * Returns 1 if does not have one (or both) of the ids
  * 0 if it does have them
*/
int hasBothIds(char * idL, char * idR) {
    void * hasFirst = g_hash_table_lookup(idHash,idL);
    void * hasSecond = g_hash_table_lookup(idHash,idR);
    if (!hasFirst || !hasSecond)
        return 1;
    return 0;
}

char * formatFieldNode(char * title, char * content, int header) {
  char * result;
  asprintf(&result,"\t<h%d>%s</h%d>\n\t\t<p>%s</p>\n", header, title, header, content);
```

```
  return result;
}

char * intToString(int num) {
  char result[12];
  sprintf(result,"%d",num);
  return strdup(result);
}

void addFile(char * fileContent) {
  if (id) {
    char fileName[20];
    sprintf(fileName,"%s.html",id);
    FILE * file = fopen(fileName,"w");
    if (file) {
      fwrite(fileContent, 1, strlen(fileContent), file);
      fclose(file);
    }
  }
}

void printGraph(char * content) {
  char * fileName = "graph.dot";
  FILE * file = fopen(fileName,"w");
  if (file) {
      char * header = "graph G {\n\trankdir=\"LR\";\n";
      fwrite(header, 1, strlen(header), file);
      fwrite(content, 1, strlen(content), file);
      char * footer = "}\n";
      fwrite(footer, 1, strlen(footer), file);
      fclose(file);
  }
  else
    puts("Could not open graph.dot!");
}


%}
%union { char * string; int integer; }

/* Tipos de nodos */
%token ARTISTA OBRA EVENTO

/* Tipos de Tags */
%token NOME IDADE FORMACAO PRODUTORAS DATA GENERO DURACAO EDICAO TIPO ID_TAG

/* STRING das Tags e do ID */
%token <string> STRING ID_ARTISTA ID_OBRA ID_EVENTO
```

```
/* Valores das Tags*/
%token <integer> INT

/* Tipos de Relações */
%token PRODUZIU APRENDEU COLABOROU PARTICIPOU

/* Tipos das regras da gramática */
%type <string> Nome TagArtista LTagArtista Nodo LTagObra LTagEvento LConteudos Conteudo
Relacao TagObra TagEvento Data RArtistaObra RArtistaEvento RArtistaArtista

%%
MvA: LConteudos                         { printGraph($1); }
   ;

LConteudos: Conteudo                    { $$ = $1; }
          | LConteudos Conteudo         { asprintf(&$$,"%s%s",$1,$2); }
          ;

Conteudo: Nodo                           { addFile($1);
                                           int err = insertId(id);
                                           if (err)
                                              yyerror("Id já existente!");
                                           if (nome)
                                              asprintf(&$$, "\t%s [URL=\"%s.html\",label=\"%s\"];
                                              \n", id, id, nome);
                                           else
                                              asprintf(&$$, "\t%s [URL=\"%s.html\"];\n", id, id);
                                           id = NULL;
                                           nome = NULL;
                                         }
        | Relacao '.'                    { $$ = $1; }
        ;

Nodo: ARTISTA '{' LTagArtista '}'        { asprintf(&$$,"<h1>Artista</h1>\n%s",$3);
                                           clearArtista();}
    | OBRA '{' LTagObra '}'              { asprintf(&$$,"<h1>Obra</h1>\n%s",$3); clearObra(); }
    | EVENTO '{' LTagEvento '}'          { asprintf(&$$,"<h1>Evento</h1>\n%s",$3);
                                           clearEvento();}
    ;

Nome: NOME ':' STRING                    { nome = $3;
                                           $$ = $3; }
    ;

LTagArtista: ID_TAG ':' ID_ARTISTA       { int err = useTagNode(cID,cARTISTA);
                                           if (err)
                                             yyerror("Id já havia sido definido!");
```

```
                                        id = $3;
                                        asprintf(&$$,"%s",formatFieldNode("Id",$3,2));
                                      }
          | LTagArtista ',' TagArtista { asprintf(&$$,"%s\n%s",$1,$3); }
          ;

TagArtista: IDADE ':' INT             { int err = useTagNode(cIDADE,cARTISTA);
                                        if (err)
                                           yyerror("Idade já havia sido definida!");
                                        char * idade = intToString($3);
                                        asprintf(&$$,"%s",formatFieldNode("Idade",idade,2));
                                        free(idade);
                                      }
          | FORMACAO ':' STRING       { int err = useTagNode(cFORMACAO,cARTISTA);
                                        if (err)
                                           yyerror("Formação já havia sido definida!");
                                        asprintf(&$$,"%s",formatFieldNode("Formação",$3,2));
                                      }
          | PRODUTORAS ':' STRING     { int err = useTagNode(cPRODUTORAS,cARTISTA);
                                        if (err)
                                           yyerror("Produtoras já haviam sido definidas!");
                                        asprintf(&$$,"%s",formatFieldNode("Produtoras",$3,2));
                                      }
          | Nome                      { int err = useTagNode(cNOME, cARTISTA);
                                        if (err)
                                           yyerror("Nome já havia sido definido!");
                                        asprintf(&$$,"%s",formatFieldNode("Nome",$1,2));
                                      }
          ;

LTagObra: ID_TAG ':' ID_OBRA          { int err = useTagNode(cID,cOBRA);
                                        if (err)
                                           yyerror("Id já havia sido definido!");
                                        id = $3;
                                        asprintf(&$$,"%s",formatFieldNode("Id",$3,2));
                                      }
        | LTagObra ',' TagObra        { asprintf(&$$,"%s\n%s",$1,$3); }
        ;

TagObra: Data                         { int err = useTagNode(cDATA, cOBRA);
                                        if (err)
                                           yyerror("Data já havia sido definida!");
                                        asprintf(&$$,"%s",formatFieldNode("Data",$1,2));
                                      }
       | GENERO ':' STRING            { int err = useTagNode(cGENERO,cOBRA);
                                        if (err)
                                           yyerror("Género já havia sido definido!");
                                        asprintf(&$$,"%s",formatFieldNode("Género",$3,2));
```

```
                                    }
        | DURACAO ':' INT           { int err = useTagNode(cDURACAO,cOBRA);
                                      if (err)
                                         yyerror("Duração já havia sido definida!");
                                      char * duracao = intToString($3);
                                      asprintf(&$$,"%s",formatFieldNode("Duração",duracao,2))
                                      ;
                                      free(duracao);
                                    }
        | Nome                      { int err = useTagNode(cNOME,cOBRA);
                                      if (err)
                                         yyerror("Nome já havia sido definido!");
                                      asprintf(&$$,"%s",formatFieldNode("Nome",$1,2));
                                    }
        ;

Data: DATA ':' INT                  { $$ = intToString($3); }
    ;

LTagEvento: ID_TAG ':' ID_EVENTO    { int err = useTagNode(cID,cEVENTO);
                                      if (err)
                                         yyerror("Id já havia sido definido!");
                                      id = $3;
                                      asprintf(&$$,"%s",formatFieldNode("Id",$3,2));
                                    }
          | LTagEvento ',' TagEvento { asprintf(&$$,"%s\n%s",$1,$3); }
          ;

TagEvento: Data                     { int err = useTagNode(cDATA, cEVENTO);
                                      if (err)
                                         yyerror("Data já havia sido definida!");
                                      asprintf(&$$,"%s",formatFieldNode("Data",$1,2));
                                    }
         | EDICAO ':' INT           { int err = useTagNode(cEDICAO,cEVENTO);
                                      if (err)
                                         yyerror("Edição já havia sido definida!");
                                      char * edicao = intToString($3);
                                      asprintf(&$$,"%s",formatFieldNode("Edição",edicao,2));
                                      free(edicao);
                                     }
         | TIPO ':' STRING           { int err = useTagNode(cTIPO,cEVENTO);
                                       if (err)
                                         yyerror("Tipo já havia sido definido!");
                                       asprintf(&$$,"%s",formatFieldNode("Tipo",$3,2));
                                      }
         | Nome                       { int err = useTagNode(cNOME,cEVENTO);
                                        if (err)
                                          yyerror("Nome já havia sido definido!");
```

```
                                        asprintf(&$$,"%s",formatFieldNode("Nome",$1,2));
                                      }
        ;

Relacao: ID_ARTISTA RArtistaObra ID_OBRA       { int err = hasBothIds($1,$3);
                                                 if (err)
                                                    yyerror("Impossível estabelecer relação
                                                     entre IDs inexistentes!");
                                                 asprintf(&$$,"\t%s -- %s %s;\n",$1, $3, $2);
                                               }
       | ID_ARTISTA RArtistaEvento ID_EVENTO   { int err = hasBothIds($1,$3);
                                                 if (err)
                                                    yyerror("Impossível estabelecer relação
                                                     entre IDs inexistentes!");
                                                 asprintf(&$$,"\t%s -- %s %s;\n",$1, $3, $2); }
       | ID_ARTISTA RArtistaArtista ID_ARTISTA { int err = hasBothIds($1,$3);
                                                 if (err)
                                                   yyerror("Impossível estabelecer relação
                                                   entre IDs inexistentes!");
                                                 asprintf(&$$,"\t%s -- %s %s;\n",$1, $3, $2); }
       ;

RArtistaObra: PRODUZIU                 { $$ = "[label=\"produziu\",color=red,penwidth=3.0]"; }
            ;

RArtistaEvento: PARTICIPOU             { $$ = "[label=\"participou em\",color=black,
                                       penwidth=3.0]"; }
              ;

RArtistaArtista: APRENDEU              { $$ = "[label=\"aprendeu com\",color=blue,
                                       penwidth=3.0]"; }
               | COLABOROU             { $$ = "[label=\"colaborou com\",color=green,
                                       penwidth=3.0]"; }
               ;
%%

int yyerror(char *s){
  printf("Erro sintático: %s\n",s);
  return 0;
}

int main(){
    idHash = g_hash_table_new_full(g_str_hash,g_str_equal,free,free);
    yyparse();
    g_hash_table_remove_all(idHash);
    g_hash_table_destroy(idHash);
    return 0;
}
```

A.0.2 lexico.l

```
%{
#include "y.tab.h"
#include <stdio.h>
%}

numero                  [0-9]
INT                     {numero}+

%%
(?i:Artista)            { return ARTISTA; }
(?i:Obra)               { return OBRA; }
(?i:Evento)             { return EVENTO; }
(?i:Nome)               { return NOME; }
(?i:Idade)              { return IDADE; }
(?i:Formacao)           { return FORMACAO; }
(?i:Produtoras)         { return PRODUTORAS; }
(?i:Data)               { return DATA; }
(?i:Genero)             { return GENERO; }
(?i:Duracao)            { return DURACAO; }
(?i:Edicao)             { return EDICAO; }
(?i:Tipo)               { return TIPO; }
(?i:produziu)           { return PRODUZIU; }
(?i:aprendeuCom)        { return APRENDEU; }
(?i:colaborouCom)       { return COLABOROU; }
(?i:participouEm)       { return PARTICIPOU; }
(?i:id)                 { return ID_TAG; }
{INT}                   { yylval.integer = atoi(yytext); return INT; }
a[0-9]+                 { yylval.string = strdup(yytext); return ID_ARTISTA; }
o[0-9]+                 { yylval.string = strdup(yytext); return ID_OBRA; }
e[0-9]+                 { yylval.string = strdup(yytext); return ID_EVENTO; }
\"[^"]+\"               {
                          yytext[yyleng-1] = '\0';
                          yylval.string = strdup(yytext + 1);
                          return STRING;
                        }
[,:{}.]                 { return yytext[0];}
.|\n                    { ; }

%%

int yywrap(){
  return 1;
```

```
}
```

A.0.3 Texto-fonte exemplo

```
artista {
    id: a0,
    nome: "Robert De Niro",
    idade: 75,
    produtoras: "Francis Ford Coppola",
    formacao: "HB Studio, Stella Adler Conservatory, Lee Strasberg's Actors Studio"
}
artista {
    id: a1,
    nome: "Jodie Foster",
    idade: 56,
    formacao: "Yale University"
}
artista {
    id: a2,
    nome: "Al Pacino",
    idade: 79,
    formacao: "High School of Performing Arts"
}
artista {
    id: a3,
    nome: "Michael Moriarty",
    idade: 78,
    formacao: "London Academy of Music and Dramatic Art"
}
artista {
    id: a4,
    nome: "Robert Duvall",
    idade: 88,
    formacao: "Neighborhood Playhouse School of the Theatre"
}
obra {
  id: o0,
  nome: "Bang the Drum Slowly",
  data: 1973,
  genero: "Drama",
  duracao: 96
}
obra {
  id: o1,
  nome: "Mean Streets",
  data: 1973,
  genero: "Crime",
  duracao: 112
```

```
}
obra {
  id: o2,
  nome: "The Godfather Part II",
  data: 1974,
  genero: "Crime",
  duracao: 200
}
obra {
  id: o3,
  nome: "Taxi Driver",
  data: 1976,
  genero: "Thriller",
  duracao: 113
}
evento {
  id: e0,
  nome: "The 47th Academy Awards",
  data: 1975,
  tipo: "Festival",
  edicao: 47
}
evento {
  id: e1,
  nome: "The 49th Academy Awards",
  data: 1977,
  tipo: "Festival",
  edicao: 49
}
evento {
  id: e2,
  nome: "The British Academy Film Awards (29)",
  data: 1976,
  tipo: "Festival",
  edicao: 29
}
evento {
  id: e3,
  nome: "The British Academy Film Awards (30)",
  data: 1977,
  tipo: "Festival",
  edicao: 30
}
a0 produziu o0.
a0 produziu o1.
a0 produziu o2.
a0 produziu o3.
a0 participouEm e0.
```

```
a0 participouEm e1.
a0 participouEm e2.
a0 participouEm e3.
a1 participouEm e1.
a1 produziu o3.
a0 colaborouCom a1.
a2 produziu o2.
a2 colaborouCom a0.
a2 participouEm e0.
a2 participouEm e2.
a3 produziu o0.
a3 colaborouCom a0.
a4 produziu o2.
a4 colaborouCom a0.
a4 colaborouCom a2.
```