Universidade do Minho
Mestrado Integrado em Engenharia Informática
Departamento de Informática

Gramáticas nas Compreensão de Software

Simulação de Q&A Systems

Grupo 2

Eduardo Gil Ribeiro da Rocha - **A77048**
Manuel Gouveia Carneiro de Sousa - **A78869**

5 de Fevereiro de 2019

Resumo

Este relatório pretende explicar quais os passos tomados para o desenvolvimento de um trabalho sobre sistemas *Q&A*, abordando também as propriedades das tecnologias utilizadas.

Conteúdo

1 Introdução

No âmbito da Unidade Curricular de *Gramáticas na Compreensão de Software*, foi-nos proposto o desenvolvimento de uma *DSL* com o intuito de simular um sistema de Pergunta-Resposta. Este sistema, iria conter uma Base de Conhecimento formada por pares (Questão, Resposta), em que a questão é modelada por um triplo, o qual continha o tipo da questão, a ação tomada e uma lista de objetos envolventes. Através desta modelação, era possível fornecer a respetiva resposta para uma determinada pergunta.

Posto isto, iremos detalhar as diversas etapas que constituíam o desenvolvimento da *DSL* proposta.

2 Background

Um sistema *Q&A (Question and Answer)* é um tipo de *software* desenhado para tentar responder a perguntas colocadas pelos utilizadores.

Existem diferentes tipos de sistemas *Q&A*, entre os quais sistemas que necessitam o envolvimento de outra pessoa para além do utilizador, para que possa responder às perguntas, e sistemas automáticos em que o *software* tem acesso a toda a informação que precisa para responder.

3 Trabalho Desenvolvido

Inicialmente, foi feita a escolha de uma área de conhecimento na qual iríamos focar a nossa *DSL*. Esta área de conhecimento foi, então, Cinema/-Cultura Cinematográfica. Posto isto, de forma a dar resposta aos desafios propostos no enunciado, resolvemos então dividir o desenvolvimento da *DSL* em duas partes. A primeira, era a estruturação da base do conhecimento, e a forma de como esta iria ser escrita. Numa segunda fase, passaríamos então ao desenho da *DSL*, e posteriormente à forma de processamento da informação que iria sendo reconhecida. Passaremos então à explicação de cada uma das partes.

3.1 Estruturação da Base de Conhecimento

Para o desenho da base de conhecimento, resolvemos tomar como inspiração as estruturas das linguagens mais famosas como *C* ou *Java*. Para dividir a Base de Conhecimento com as Questões propriamente ditas, identificamos cada um desses componentes com *keywords* explícitas. Depois, logo a seguir a cada *keyword*, e delimitado por chavetas ({ ... }) colocamos então os triplos correspondentes a cada pergunta na secção da base de conhecimento, e as questões completas na secção das mesmas.

```
BASE_CONHECIMENTO {
    (Tipo-Q, Ação, [Objeto]) = "RESPOSTA" ;
    (...)
}

QUESTOES {
    " Exemplo de uma questão ? " ;
    (...)
}
```

Vista a estrutura do ficheiro a ser analisado, passaremos então à gramática para reconhecer toda a informação nele contida.

3.2 Reconhecimento da Base de Conhecimento

Com o intuito de reconhecer a estrutura anteriormente desenhado, foi então criada uma *DSL* para reconhecer a Base de Conhecimento e posteriormente dar resposta aos diversos pedidos nela incluídos. Para isso, foi então escrita a gramática dessa *DSL*, com o auxílio da ferramenta *ANTLR*, o qual nos produzia o *parser* para a nossa linguagem.

Numa primeira fase, foi criada a "GIC" (Gramática Independente de Contexto) para dessa forma, estruturarmos a nossa gramática final, e testar o reconhecimento dos símbolos. Após o correto reconhecimento, partimos para a criação da "GA" (Gramática de Atributos), fazendo o uso de atributos para comunicar informação entre as diversas produções.

Posto isto, iremos então explicar como é que a informação reconhecida foi estruturada e posteriormente processada.

3.2.1 Estruturação da Informação

Para estruturar a informação que iria ser reconhecida na Base de Conhecimento, resolvemos fazer o uso da primitiva *"@members"* fornecida pelo *ANTLR*, e nesta foi declarada uma classe Triplo, a qual iria guardar a informação relativa a um triplo, contendo então duas *strings* relativas ao tipo de questão e à ação, e uma lista de *strings* que corresponde à lista dos objetos/*keywords* da questão. Para além disso, foi ainda declarada a estrutura de dados principal do programa, a qual iria associar a resposta (*String*) ao seu respetivo Triplo. Aqui, foi usada a estrutura de dados *HashMap* fornecida pelo *Java*.

```
@members {
    public class Triplo {
        String tipoQ;
        String acao;
        ArrayList<String> keywords;

        public Triplo() {
            this.tipoQ = "";
            this.acao = "";
            this.keywords = new ArrayList<String>();
        }

        public void setKeywords(ArrayList<String> keywords) {
            for (String s : keywords) {
                this.keywords.add(s);
            }
        }
    }

    // Estrutura de dados principal.
    Map<String, Triplo> triplos;
}
```

É importante realçar o facto da escolha de guardar esta estrutura de dados principal como uma variável global, visto que o número de acessos era elevado, e o transporte desta ao longo da gramática como um atributo iria produzir uma solução bastante caótica em termos de código escrito.

3.2.2 Processamento de Informação

Depois de reconhecida a Base de Conhecimento e processada a informação correspondente, eram então reconhecidos os símbolos correspondentes à estrutura das questões, e processada a informação respetiva. Uma questão era processada palavra a palavra, analisando assim a sua classe. Estas classes, são previamente definidas no *Lexer* da gramática. Aqui, estão contidos os diversos tipos de questões (Onde, Quem, ...), as ações tomadas numa questão (verbos) e, por fim, a identificação de determinantes ou preposições na frase.

```
questao returns[String tipo_Q, String verbo,
                ArrayList<String> keywords, StringBuilder pergunta]
@init {
    $questao.keywords = new ArrayList<String>();
    $questao.pergunta = new StringBuilder();
}
    : (  t=TIPO_Q
         { $questao.tipo_Q = $t.text;
           $questao.pergunta.append($t.text + " "); }
       | a=ACAO
         { $questao.verbo = $a.text;
           $questao.pergunta.append($a.text + " "); }
       | c=(DETERMINANTE | PREPOSICAO)
         { $questao.pergunta.append($c.text + " "); }
       | k=TEXTO
         { $questao.keywords.add($k.text);
           $questao.pergunta.append($k.text + " "); } )+
;
```

As questões eram então reconhecidas consoante o *Lexer* definido.

```
/* ANALISADOR LÉXICO */

TIPO_Q       : 'Qual' | 'Quem' | 'Quando' | 'Onde' ;
ACAO          : 'é' | 'foi' ;
DETERMINANTE  : 'o' | 'a' | 'os' | 'as' ;
PREPOSICAO    : 'de' | 'da' | 'do' | 'em' | 'para' | 'sem' ;
(...)
```

Ao ser reconhecida e processada uma questão, era verificada a existência de uma resposta consoante a informação reconhecida (o tipo da questão, a ação e as *keywords*), e posteriormente escolhida a resposta mais adequada. Esta resposta, é baseada no número de *keywords* encontradas. Se esse número for igual ao número de *keywords* no triplo inicialmente reconhecido, então foi encontrada a resposta. Se isto nunca acontecer, a resposta irá ser aquela que tenha mais *keywords* iguais.

Esta foi então a produção correspondente a esse reconhecimento, bem como o algoritmo feito para encontrar a resposta correta, ou a mais próxima da verdadeira, consoante a informação na Base de Conhecimento.

```
string returns[int maxKeywords, int maxKeyFound,
               ArrayList<String> pergunta, String resp]
    : "\"" q=questao "?" "\"" ";"
    {
        (...)

        for (String r : triplos.keySet()) {
            Triplo t = triplos.get(r);

            if ( tipoQ.equals(t.tipoQ) && acao.equals(t.acao) ) {
                $string.maxKeywords = t.keywords.size();
                // Guarda resposta atual na variável auxiliar.
                $string.resp = r;
                for (String k : keywords) {
                    if (t.keywords.contains(k)) {
                        maxTmp++;
                    }
                }
            }

            if (maxTmp == t.keywords.size()) {
                $string.maxKeyFound = maxTmp;
                break; // Foi encontrada a resposta!
            } else if (maxTmp > $string.maxKeyFound) {
                $string.maxKeyFound = maxTmp;
            }
```

```
            maxTmp = 0;
        }
    }
;
```

Este processamento é feito à medida em que é são reconhecidas as questões. Com isto, são então imprimidas para a consola as respetivas respostas escolhidas para a questão em causa.

```
questoes
@init {
    System.out.println("Respostas às questões processadas:");
}
    : "QUESTOES" "{"
      ( s=string
        {
            System.out.print("\n" + $s.pergunta.toString());
            System.out.println("?\nR: " + $s.resp +
                               " :: Keywords: " + $s.maxKeyFound + "/"
                                                + $s.maxKeywords);
        } )+
      "}"
;
```

4 Conclusão

De acordo com os principais objetivos propostos no enunciado do trabalho, concluímos que estes foram atingidos com sucesso. De facto, fomos capazes de criar uma *DSL* a qual dá respostas a diversas perguntas com alguma estrutura.

Através deste trabalho, foi-nos possível testar um pouco de toda a potencialidade que a ferramenta *ANTLR* nos traz. Com o auxílio da linguagem *Java* e funcionalidades fornecidas pelo *ANTLR*, algum do processamento feito tornou-se bastante intuitivo. Para além disso, o facto de termos produzido uma Gramática de Atributos, e através dos atributos herdados e atributos sintetizados, facilitou a forma de como todas as produções comunicavam entre elas, permitindo assim a troca de informação.

Em suma, o trabalho realizado deixou-nos aprofundar os conhecimentos adquiridos na Unidade Curricular assim como entender as utilidades de Gramáticas de Atributos em diversas áreas de desenvolvimento de *software*.

5 Anexos

5.1 Gramática Independente de Contexto

```
grammar BC_QA_GIC;

insts : base questoes
;


/* Processamento Base de Conhecimento */

base : 'BASE_CONHECIMENTO' '{' triplos '}'
;

triplos : (triplo)+
;

triplo : '(' tipoQ ',' acao ',' '[' objetos ']' ')'
         '=' '\"' resposta '\"' ';'
;

tipoQ : TIPO_Q
;

acao : ACAO
;

objetos : TEXTO (',' TEXTO)*
;

resposta : NUMERO
         | TEXTO (TEXTO)*
;


/* Processamento Questões */
```

```
questoes : 'QUESTOES' '{' (string)+ '}'
;

string : '\"' questao '?' '\"' ';'
;

questao : ( TIPO_Q | ACAO | (DETERMINANTE | PREPOSICAO) | TEXTO )+
;


/* ANALISADOR LÉXICO */

TIPO_Q        : 'Qual' | 'Quem' | 'Quando' | 'Onde' ;
ACAO          : 'é' | 'foi' ;
DETERMINANTE  : 'o' | 'a' | 'os' | 'as' ;
PREPOSICAO    : 'de' | 'da' | 'do' | 'em' | 'para' | 'sem' ;

NUMERO   : [0-9]+ ;
TEXTO    : (LETRA)+ ;

Separador: ('\r'? '\n' | ' ' | '\t')+  -> skip ;

/* LETRA não é um terminal.
   Simplesmente foi definido para simplificar
   outras expressões regulares. */
fragment LETRA : [a-zA-ZáéíóúÁÉÍÓÚâêîôûÂÊÎÔÛãõÃÕàèìòùÀÈÌÒÙçÇ_] ;
```

5.2 Gramática de Atributos

```
grammar BC_QA_GA;

@header {
    import java.util.Map;
    import java.util.HashMap;
    import java.util.List;
    import java.util.ArrayList;
}
```

```
@members {
    public class Triplo {
        String tipoQ;
        String acao;
        ArrayList<String> keywords;

        public Triplo() {
            this.tipoQ = "";
            this.acao = "";
            this.keywords = new ArrayList<String>();
        }

        public void setKeywords(ArrayList<String> keywords) {
            for (String s : keywords) {
                this.keywords.add(s);
            }
        }
    }

    // Estrutura de dados principal.
    Map<String, Triplo> triplos;
}

insts
@init {
    // Iniciar o mapeamento de triplos.
    triplos = new HashMap<String, Triplo>();
}
     : base questoes
;


/* Processamento Base de Conhecimento */

base    : 'BASE_CONHECIMENTO' '{' triplos '}'
;
```

```
triplos : (triplo)+
;

triplo  : '(' t=tipoQ ',' a=acao ',' '[' o=objetos ']' ')'
          '=' '\"' r=resposta '\"' ';'
        {
            Triplo t = new Triplo();
            t.tipoQ = $t.text;
            t.acao = $a.text;
            t.setKeywords($o.keywords);

            triplos.put($r.resp, t);
        }
;

tipoQ    : TIPO_Q
;

acao     : ACAO
;

objetos returns[ArrayList<String> keywords]
         : t1=TEXTO { $objetos.keywords = new ArrayList<>();
                      $objetos.keywords.add($t1.text); }
           (',' t2=TEXTO { $objetos.keywords.add($t2.text); } )*
;

resposta returns[String resp]
          : n=NUMERO { $resposta.resp = $n.text;}
          | t1=TEXTO { $resposta.resp = $t1.text; }
           (t2=TEXTO { $resposta.resp += (" " + $t2.text); } )*
;


/* Processamento Questões */

questoes
@init {
```

```
    System.out.println("Respostas às questões processadas:");
}
    : 'QUESTOES' '{'
        ( s=string
          {
              System.out.print("\n" + $s.pergunta.toString());
              System.out.println("?\nR: " + $s.resp +
                                 " :: Keywords: " + $s.maxKeyFound + "/"
                                                  + $s.maxKeywords);
          }
        )+
      '}'
;

string returns[int maxKeywords, int maxKeyFound,
               StringBuilder pergunta, String resp]
        : '\"' q=questao '?' '\"' ';'
        {
            $string.maxKeywords = 0;
            $string.maxKeyFound = 0;
            $string.pergunta = $q.pergunta;
            $string.resp = "ERRO! Não foi encontrada resposta.";

            String tipoQ = $q.tipo_Q;
            String acao = $q.verbo;
            ArrayList<String> keywords = $q.keywords;

            int maxTmp = 0;

            for (String r : triplos.keySet()) {
                Triplo t = triplos.get(r);

                if ( tipoQ.equals(t.tipoQ) && acao.equals(t.acao) ) {
                    $string.maxKeywords = t.keywords.size();
                    // Guarda resposta atual na variável auxiliar.
                    $string.resp = r;
                    for (String k : keywords) {
                        if (t.keywords.contains(k)) {
```

```
                        maxTmp++;
                    }
                }
            }

            if (maxTmp == t.keywords.size()) {
                $string.maxKeyFound = maxTmp;
                break; // Foi encontrada a resposta!
            } else if (maxTmp > $string.maxKeyFound) {
                $string.maxKeyFound = maxTmp;
            }

            maxTmp = 0;
        }
    }
;

questao returns[String tipo_Q, String verbo,
            ArrayList<String> keywords, StringBuilder pergunta]
@init {
    $questao.keywords = new ArrayList<String>();
    $questao.pergunta = new StringBuilder();
}
    : ( t=TIPO_Q
        { $questao.tipo_Q = $t.text;
          $questao.pergunta.append($t.text + " "); }
      | a=ACAO
        { $questao.verbo = $a.text;
          $questao.pergunta.append($a.text + " "); }
      | c=(DETERMINANTE | PREPOSICAO)
        { $questao.pergunta.append($c.text + " "); }
      | k=TEXTO
        { $questao.keywords.add($k.text);
          $questao.pergunta.append($k.text + " "); } )+
;


/* ANALISADOR LÉXICO */
```

```
TIPO_Q        : 'Qual' | 'Quem' | 'Quando' | 'Onde' ;
ACAO          : 'é' | 'foi' ;
DETERMINANTE  : 'o' | 'a' | 'os' | 'as' ;
PREPOSICAO    : 'de' | 'da' | 'do' | 'em' | 'para' | 'sem' ;

NUMERO   : [0-9]+ ;
TEXTO    : (LETRA)+ ;

Separador: ('\r'? '\n' | ' ' | '\t')+  -> skip ;

/* LETRA não é um terminal.
   Simplesmente foi definido para simplificar
   outras expressões regulares. */
fragment LETRA : [a-zA-ZáéíóúÁÉÍÓÚâêîôûÂÊÎÔÛãõÃÕàèìòùÀÈÌÒÙçÇ_] ;
```

Referências

[1] https://en.wikipedia.org/wiki/Q%26A_software

[2] https://www.antlr3.org/works/