Università degli studi di Bergamo

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

CORSO DI

Linguaggi formali e compilatori

Formal languages and compilers

(COD.CORSO:38070)

Anno accademico:	**2017/2018**
Settore scientifico-disciplinare:	**ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI**
Dipartimento:	**Ingegneria gestionale, dell'informazione e della produzione**

Titolo: Specifiche dello strumento

Autori:

Davide Capelli Matr# 1025005
Marco Vitetta Matr# 1048711

1 La grammatica

La grammatica del linguaggio definito è disponibile integralmente al seguente indirizzo[1] ed è riportata, nel seguito, nelle sue parti essenziali.

```
/*
 * Lexer aims to recognise single elements (sequence of related
 * terminal symbols, namely characters, called tokens).
 * Rules must start with uppercase  letter and can be recursive
 * (but NOT left-recursive).
 */

/* Reserved words of the language */
ARTICLE:    'article';
AUTHOR:     'author';
CV:         'cv';
END:        'end';
LETTER:     'letter';
RULE:       'rule';
TEMPLATE:   'template';
THESIS:     'thesis';
TITLE:      'title';

/* Generic words and special characters */
EOL:        ('\r'? '\n');
PERCENT:    '%';
WORD:       ([!-$] | [&-Z] | [a-z] | [¡-ÿ])+;
WS:         (' ' | '\t')+;

/*
 * Parser aims to  recognize  some  structured sequences (which
 * ones are syntactically and, hopefully, semantically correct)
 * of symbols (terminal and  non-terminal, namely other parsing
 * rules).
 * Rules must start with lowercase letter and can't be recursive.
 */
axiom:      template preamble? content END;
preamble:   (author? title) | (title? author);
title:      TITLE line;
author:     AUTHOR line;
template:   TEMPLATE WS (article | custom | cv | letter | thesis) EOL;
article:    ARTICLE;
custom:     WORD;
cv:         CV;
letter:     LETTER;
thesis:     THESIS;

/* Auxiliary parsing rules */
comment:    (WS? PERCENT line)+;
content:    (comment | customRule | text)+;
line:       (WORD | WS)+ EOL;
customRule: WS? RULE line;
text:       line+;
```

Listato 1 - Parti essenziali della grammatica del linguaggio

La grammatica può essere divisa logicamente in tre parti:

1. la prima è detta intestazione (o header e, per semplicità, non è stata riportata nel **Listato 1**) e specifica il pacchetto Java (il linguaggio target utilizzato nell'ambito del progetto, per maggiori informazioni si faccia riferimento alla relazione introduttiva allo strumento) all'interno del quale ANTLR[2] genererà il parser;

2. la seconda parte specifica le regole del lexer (ossia il riconoscitore delle sequenze di simboli terminali correlati dette token) che devono iniziare con una lettera maiuscola affinché ANTLR[2] le distingua dalle regole del parser;

3. la terza parte specifica le regole del parser (ossia il riconoscitore delle sequenze di token) che devono iniziare con una lettera minuscola affinché ANTLR[2] le distingua dalle regole del lexer.

La convenzione adottata all'interno del progetto utilizza solo lettere maiuscole per distinguere le regole del lexer (ossia i token) da quelle del parser.

Le regole del lexer permettono di riconoscere alcuni simboli speciali (il percento, *%*, ed il terminatore di linea di entrambi gli standard Unix, *\n*, e Windows *\r\n*), gli spazi e le parole, comprese quelle riservate del linguaggio (*article*, *author*, *cv*, *end*, *letter*, *rule*, *template*, *thesis*, *title*).

La regola principale del parser è chiamata *axiom* e permette di riconoscere il template scelto dall'utente (attraverso la regola *template*), l'autore ed il titolo (attraverso la regola *preamble* che rende opzionali entrambi gli elementi) ed il contenuto del documento (attraverso la regola *content*).

```
axiom:      template preamble? content END;
preamble:   (author? title) | (title? author);
title:      TITLE line;
author:     AUTHOR line;
template:   TEMPLATE WS (article | custom | cv | letter | thesis) EOL;
```

Listato 2 - Alcune parti essenziali delle regole del parser

La regola *template* riconosce la parola riservata *template* seguita da uno spazio e dal nome del template selezionato dall'utente (che può essere personalizzato, riconosciuto dalla regola *custom*, o uno di quelli predefiniti, riconosciuti dalle regole *article*, *cv*, *letter* e *thesis*).

Il contenuto del documento è costituito da una sequenza qualsiasi di testo (riconosciuto dalla regola *text*), commenti (preceduti dal simbolo percento, %, e riconosciuti dalla regola *comment*) ed eventuali regole personalizzate definite all'interno del template scelto (precedute dalla parola riservata *rule* e riconosciute dalla regola *customRule*)

Le regole *comment*, *customRule* e *text* si basano a loro volta sulla regola *line* che riconosce le sequenze di parole sulla medesima riga.

```
comment:    (WS? PERCENT line)+;
content:    (comment | customRule | text)+;
line:       (WORD | WS)+ EOL;
customRule: WS? RULE line;
text:       line+;
```

Listato 3 - Regole ausiliarie del parser

2 La traduzione

ANTLR[2] genera automaticamente il parser per il linguaggio definito ed il rispettivo traduttore che implementa i metodi entra ed esci di ciascuna regola con il corpo vuoto.

La traduzione del testo riconosciuto da ciascuna regola può essere svolta direttamente all'interno dei metodi entra ed esci generati da ANTLR[2] (all'interno del traduttore) ma si consiglia vivamente di estendere la classe del traduttore affinché lo strumento non sovrascriva la propria implementazione ogni volta che ricompila la grammatica.

Il **Listato 4** riporta, a titolo di esempio, l'implementazione di alcune regole realizzate all'interno del traduttore *SimplifiedLatexTranslator*[3] che estende la classe *SimplifiedLatexBaseListener* generata automaticamente da ANTLR[2].

```
@Override
public void enterTitle(TitleContext ctx) {
    this.trim = true;
    System.out.print("\\title{");
}

@Override
public void exitTitle(TitleContext ctx) {
    System.out.println("}");
    System.out.println("\\maketitle");
}

@Override
public void enterArticle(ArticleContext ctx) {
    System.out.print("article");
}
```

Listato 4 - Alcune traduzioni realizzate

2.1 Traduzioni personalizzate

I template personalizzati e le rispettive regole sono stati integrati all'interno del costruttore del traduttore.

```
public SimplifiedLatexTranslator() {
    Template ladox = new LadoxTemplate(false);

    this.availTemplates = new HashMap<>();
    this.availTemplates.put("ladox", ladox);
}
```

Listato 5 - Integrazione dei template personalizzati

Il riconoscimento del template personalizzato avviene all'interno del metodo *enterCustom()* che verifica che il template indicato dall'utente sia disponibile.

```
@Override
public void enterCustom(CustomContext ctx) {
    String templateName = ctx.getText();

    if (this.availTemplates.containsKey(templateName)) {
           this.customTemplate = this.availTemplates.get(templateName);
           System.out.print(templateName);
    }
}
```

Listato 6 - Metodo di verifica del template personalizzato

La regole personalizzate vengono applicate in due fasi:

1. quando lo strumento riconosce una regola personalizzata (perché preceduta dalla parola riservata *rule*) allora verifica che l'utente abbia selezionato un template personalizzato e, in tal caso, abilita l'applicazione delle regole personalizzate (operazioni effettuate all'interno del metodo *enterCustomRule()*);

```
@Override
public void enterCustomRule(CustomRuleContext ctx) {
    if (this.customTemplate != null) {
           this.applyCustomRules = true;
           this.trim             = true;
    }
}
```

Listato 7 - Metodo di abilitazione delle regole personalizzate

2. le regole vengono eseguite successivamente all'interno del metodo di riconoscimento delle righe *enterLine()* (a tal proposito si osservi la struttura della regola *customRule* definita nella grammatica).

```
@Override
public void enterLine(LineContext ctx) {
    String line = ctx.getText();

    if (this.trim)
           line = line.trim();
    if (this.applyCustomRules) {
           line = this.customTemplate.translate(line);
           this.applyCustomRules = false;
    }
    System.out.print(line);
}
```

Listato 8 - Metodo di applicazione delle regole personalizzate

3 Sitografia

[1] Grammatica - https://github.com/ErVito/FLC2017-18/blob/master/SimplifiedLatex.g4

[2] ANTLR - http://www.antlr.org/

[3] Traduttore - https://github.com/ErVito/FLC2017-18/blob/master/FLC - Tester proj/src/SimplifiedLatexTranslator.java