# Predictive parsing and LL(1) grammars

**Contact info:**
Address *ELIS (floor -3), Technicum (building I), Sint-Pietersnieuwstraat 41*
Email *compilers@lists.ugent.be*

**IMPORTANT**: You should always hand in your solutions using the Dropbox on Minerva. Generate an archive using the `make_tarball.sh` script, and make sure you send it to the teaching assistants.

**DEADLINE**: March 15, 23:59.

## 1 Introduction

### 1.1 BibTex format

BibTex is a format widely used to keep track of a bibliography list, and is closely related to the LaTeX typesetting system. It's a rather old format, but still extensively used in the research world. More information can be found at `http://bibliographic.openoffice.org/bibtex-defs.html`.

**Make sure** you only use this website as source of extra information on the BibTex format. Always follow the guidelines in this assignment, because we deviate slightly from the strict definitions.

Some examples, both with correct and faulty syntax, can be found among the files for this assignment. You can use those for testing your solution.

### 1.2 ANTLR - ANother Tool for Language Recognition

During this assignment ANTLR v4 will be used (`http://www.antlr.org/`). ANTLR is a tool that provides a framework for generating automata from grammatical descriptions in the EBNF syntax (Extended Backus Naur Form). Note that the same syntax is used to generate token recognizers, parsers and tree parsers.

Additional documentation over ANTLR v4 can be found at `https://github.com/antlr/antlr4/blob/master/doc/index.md`. **Make sure** you use the documentation of version 4.

# 2 Assignment

## 2.1 Grammar

You will have to construct a grammar to parse BibTex entries. Because BibTex is fairly extensive, we will limit the grammar requirements to recognize the following types of entries:

- **`book`**: describes a book with one or more authors
  mandatory fields: `label`, `author`, `title`, `year`, `publisher`
- **`article`**: describes an article in a scientific journal
  mandatory fields: `label`, `author`, `title`, `journal`, `pages`
  optional fields: (`volume`, `number`) and/or `year` (at least one of these is required!)
- **`inproceedings`**: describes a publication on an international conference
  mandatory fields: `label`, `author`, `title`, `booktitle`, `year`
  optional fields: `month`, `pages` (layout: `<number> - - <number>`)

Complete the following grammar:

```
S  -> B$
B  -> B B' <newline>
B  ->
B' -> BOOK
B' -> ARTICLE
B' -> CONFERENCE
```

**Remarks**:

- Assume that every field is on a separate line, and that fields are separated by a comma (`,`) and a newline character.
- Only the `label` field has a mandatory location (e.g. `@book { book1, ...`), the other fields may appear in a random order. Make sure all possible orders are considered valid.
- A list of authors is constructed by separating authors using `AND` or `and`:
  `Jan Spier AND Suske Van Antigoon and Robbe Does`
  Make sure your grammar assumes that there is always at least one author.
- While constructing the grammar, don't take into account whether fields are mandatory or optional. Consider each field to be optional, but make sure there is at least one field (not counting the `label` field which is always mandatory). Fields appearing multiple times should also be ignored.

The obtained grammar can be made LL(1) by applying **left recursion elimination** and **left factoring** where needed.

## 2.2 Implementation

ANTLR supports a number of target languages such as C, C++, Java, ... For this assignment you should use Java as target language, since it is well supported in the *ANTLRWorks* environment. Note that you need `javac` for generating the grammar. Install it within the virtual machine with the following commands:

```
sudo apt-get update
sudo apt-get install openjdk-7-jdk
```

Starting from the `Pract3.g4` file, construct in ANTLR an **LL(1)** parser that implements your grammar. The parser needs to spit out correctly parsed BibTex entries in an XML format based on the EndNote XML format. An example of a valid EndNote XML file is `endnote_correct.xml`. This file should be generated by your parser after processing `bibtex_correct.xml`.

All uninteresting input lines are omitted from the EndNote output. You can check the output by importing this XML-file as text into EndNote (accessible via Athena, choose `Import Option` → `EndNote generated XML`). A small example of such XML output can be found in the assignment files.

Both grammars (lexer and parser) must be implemented in one ANTLR grammar file. You can use the `Run > Run in TestRig` functionality from ANTLRWorks to generate and compile your parser. All necessary files can be found in the `pract3.tar.gz` tarball.

Some more remarks:

- There are multiple possibilities for constructing a BibTex entry. For a single entry both '`{`' and '`}`' or '`(`' and '`)`' can be used, and both braces (`{`, `}`) as well as quotes (`"`) can be used for strings.
- As mentioned before, a list of authors in BibTex is constructed by separating authors using `AND` or `and`. In EndNote XML, on the contrary, authors should be listed separately. For this purpose, use `<author> ... </author>`.
- The `month`-field offers choices: one can either use a textual abbreviation (`Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, `Dec`), the entire description (`January`, ...) or a number (`1`, `2`, ..., `12`). Implement a suitable error message in case of a faulty value.
- While reading page numbers, make sure the first number is smaller than the second one or, in other words, make sure the page numbers make sense. This can be done in plain Java. There is no need to write parser rules to enforce this.