CS F 363 Compiler Construction Assignments

Prof. Santonu Sarkar

January 23, 2017

3 Creating a BibTeX Parser

```
# Lexical grammar, mode 1: top-level
 AT                    \@
 NEWLINE               \n
 COMMENT               \%~[\n]*\n
 WHITESPACE            [\ \r\t]+
 JUNK                  ~[\@\n\ \r\t]+

 # Lexical grammar, mode 2: in-entry
 NEWLINE               \n
 COMMENT               \%~[\n]*\n
 WHITESPACE            [\ \r\t]+
 NUMBER                [0-9]+
 NAME                  [a-z0-9\!\$\&\*\+\-\.\/\:\;\<\>\?\[\]\^\_\‘\|]+
 LBRACE                \{
 RBRACE                \}
 LPAREN                \(
 RPAREN                \)
 EQUALS                =
 HASH                  \#
 COMMA                 ,
 QUOTE                 \"
 STRING                #Can be anything within {}

 # Syntactic grammar:
 bibfile : entries
         |
 entries : entry NEWLINE entries
         | entry

 entry : AT NAME LBRACE key COMMA fields RBRACE
 key   : NAME
       | NUMBER

fields : field COMMA fields
       | field
```

```
   field : NAME EQUALS LBRACE value RBRACE

   value : STRING
         | NUMBER
         | NAME
================================================
```

To understand this assignment, you should already be familiar with the BibTeX language—more specifically, the BibTeX data description language. (BibTeX being the complex beast that it is, one can conceive of the term applying to the program, the data language, the particular database structure described in the original BibTeX documentation, the ".bst" formatting language, and the set of conventions embodied in the standard styles included with the BibTeX distribution.

In particular, you should have a good idea what's going on in the following:

@article{key1,
author = {Sarkar, Santonu},
title = "John Smith"}

@book{ourbook,
author = {joe smith and john Kurt},
title = {Our Little Book},
year = {1997}
}

Before going much further, though, you're going to have to learn some of the terminology I use for describing BibTeX data. Most of it's the same as you'll find in any BibTeX documentation, but it's important to be sure that we're talking about the same things here. So, some definitions:

- top-level
  All text in a BibTeX file from the start of the file to the start of the first entry, and between entries thereafter.

- name
  A string of letters, digits, and the following characters:

  ```
  ! $ & * + - . / : ; < > ? [ ] ^ _ ‘ |
  ```

  A "name" is a catch-all used for entry types, entry keys, and field and macro names. For BibTeX compatibility, there are slightly different rules for these four entities; currently, the only such rule actually implemented is that field and macro names may not begin with a digit. Some names in the above example: string, and.

- entry
  A chunk of text starting with an "at" sign (@) at top-level, followed by a name (the entry type), an entry delimiter ({ or (), and proceeding to

the matching closing delimiter. Also, the data structure that results from parsing this chunk of text. There are two entries in the above example.

- entry type
  The name that comes right after an @ at top-level. Examples from above: string, book. With the standard BibTeX database structure, article, book, inbook, etc. all fall under the "regular entry" metatype. You should check if the entry is a valid one as per Bibtex.
  Standard entry types: article, book, booklet, conference, inbook, incollection, inproceedings, manual, masterthesis, misc, phdthesis, proceedings, techreport, unpublished

- entry delimiters
  { and }, or (and): the pair of characters that (almost) mark the boundaries of an entry. "Almost" because the start of an entry is marked by an @, not by the "entry open" delimiter.

- entry key
  (Or just key when it's clear what we're speaking of.) The name immediately following the entry open delimiter in a regular entry, which uniquely identifies the entry. Example from above: ourbook. Only regular entries have keys.

- field
  A name to the left of an equals sign in a regular or macro-definition entry. In the latter context, might also be called a macro name. Examples: address, author, title, volume, year.

- field list
  In a regular entry, everything between the entry delimiters except for the entry key. In a macro definition entry, everything between the entry delimiters (possibly also called a macro list).

- value
  The text that follows an equals sign (=) in a regular entry. Can be a string, name, or number.

- string
  (Or, sometimes, "quoted string.") A chunk of text between quotes (") or braces ({ and }). Braces must balance: {this is a {string} is not a BibTeX string, but {this is a {string}} is. ("this is a {string" is also illegal, mainly to avoid the possibility of generating bogus TeX code–which BibTeX will do in certain cases.)

- number
  An unquoted string of digits.

Version 1:

Create a mysql table with fields as shown in the following table where you populate the entries for Bibtex file. Do a check if the entry types are valid with respect to Bibtex language.

```
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| bibkey    | varchar(50)  | NO   | PRI | NULL    |       |
| bibtype   | varchar(50)  | NO   |     | NULL    |       |
| address   | varchar(255) | YES  |     | NULL    |       |
| author    | varchar(255) | NO   |     | NULL    |       |
| booktitle | varchar(255) | YES  |     | NULL    |       |
| chapter   | varchar(50)  | YES  |     | NULL    |       |
| edition   | varchar(25)  | YES  |     | NULL    |       |
| journal   | varchar(100) | YES  |     | NULL    |       |
| number    | varchar(25)  | YES  |     | NULL    |       |
| pages     | varchar(25)  | YES  |     | NULL    |       |
| publisher | varchar(100) | YES  |     | NULL    |       |
| school    | varchar(100) | YES  |     | NULL    |       |
| title     | varchar(255) | YES  |     | NULL    |       |
| volume    | varchar(50)  | YES  |     | NULL    |       |
| year      | varchar(25)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

Reference :

- Java: http://www.javaworld.com/article/2077315/java-app-dev/looking-for-lex-and-yacc-for-java–you-don-t-know-jack.html
- C++ : Lex/Yacc
- Python: http://www.dabeaz.com/ply/
- BibTex : https://www.cs.arizona.edu/ collberg/Teaching/07.231/BibTeX/bibtex.html