编译原理 Project-1

王思伦 2014/5/12

1. 实验内容

给定一个 BibTex 文件(.bib 后缀的文件)作为输入，同学们需要实现一个有限自动机，对该文件进行解析，分析出其中源自会议论文集的参考文献的引用，并将这部分引用以正确的论文中参考文献的格式输出。

2. 实验分析

BibTex 的文件格式如下：

```
@INPROCEEDINGS{pwalk:www02,
   AUTHOR = "Taher H. Haveliwala",
   TITLE = "Topic-sensitive pagerank",
   BOOKTITLE = "Proceedings of the 11th international conference on World Wide Web",
   PAGES = "517-526",
   YEAR = {2002} }
```

有关 BibTex 文件格式的详细说明请参考 http://en.wikipedia.org/wiki/BibTeX

实验中要求我们实现一个有限自动机，经过和同学、以及后来和老师、助教的探讨，我的理解是有两种实现思路：

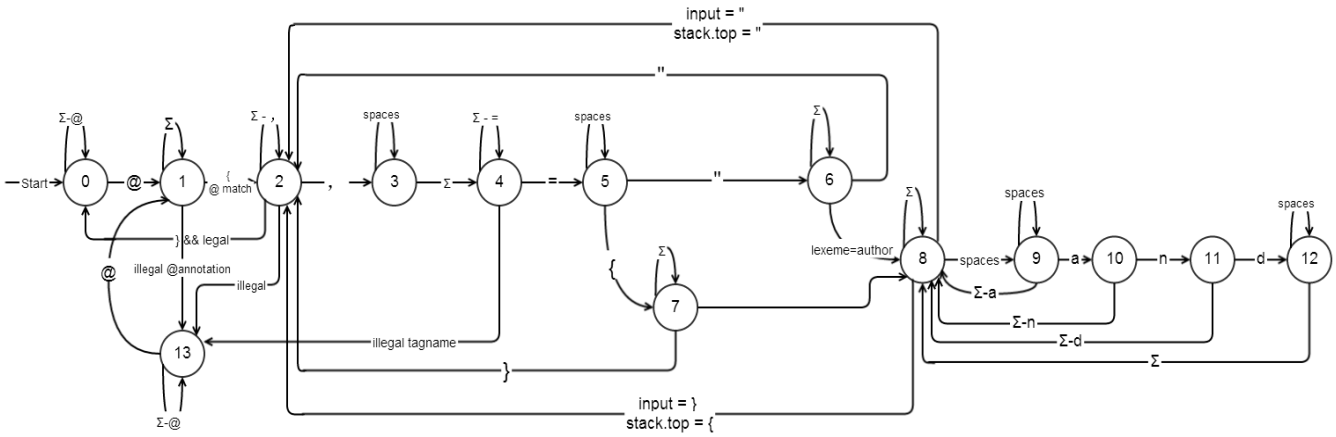
1. 一种方法是实现一个“微观上”类似于字符串模式匹配的自动机，这个自动机会随着字符串的改变而改变，并且状态较少，因为它将仅仅作为 BibTex 编译器中的一小部分，负责实现 indexOf()函数功能。
2. 另一种方法则是实现一个“宏观上”的下推自动机，因为如果这种全局设计仍然固执地使用 DFA 会导致状态过多（预计超过 100 个）。使用下推自动机比较方便，需要一个栈作为辅助，因为我们需要匹配{}””等字符串对。

我选择第二种“宏观上”的方法。

3. 自动机设计

下图是我设计的自动机，它一共有 13 个状态。具有如下功能：

1. 扫描文件，按照正确的格式输出参考文献引用；
2. 识别出姓名中不符合标准的字符；
3. 从“Author1 and Author2 and…”中分离出多个作者 1，作者 2 的姓名，按照格式输出；
4. BibTex 的类型(article/book/inproceeding)、格式(输出顺序)、规范(可选域如 pages、必要域如 title)可以重定义，易扩展；
5. 参考文献标签内容的输出顺序可以选择；



4. 程序实现

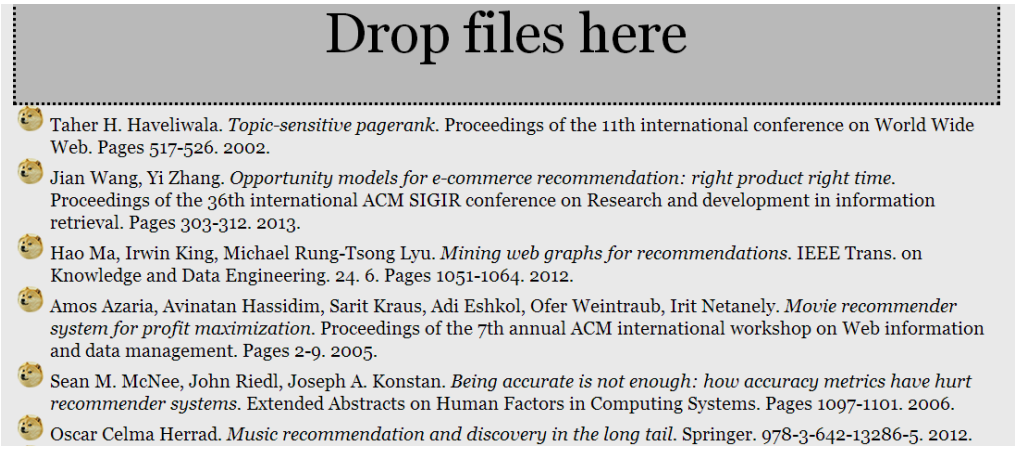
编程语言：javascript

浏览器：Chrome、Safari、Opera

输入方式：将文件直接拖拽到下图所示区域，释放即可

输出方式：文件载入成功后自动编译，直接在拖拽区域下方输出。

输出结果示例——



5. 算法设计

主体算法在 BibTex.js 文件中，流程化结构设计

5.1 几个全局变量

Output：等待输出到网页上的字符串
Sigma：字符集合
State：当前自动机的状态
BibTex_Type：当前文献的类型？INPROCEEDINGS ARTICLE BOOK 等
Tag_name：当前标签名？author title pages 等
BibTex：定义 BibTex 格式规范，标签输出顺序
Stack：下推自动机的栈

5.2 定义字符集 Σ

Alphabets: 英文字母等可能组成作者姓名的字符（包括了-._）
Numbers: 数字字符(0-9)
Symbols: 特殊字符，标点符号等
Spaces: 空格符、制表符、换行符等

```
var Sigma = {

    "alphabets": ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '_', '.', '-'],

    "numbers": ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0'],

    "symbols": ['?', '!', '<', '>', '$', '#', '@', ':', ';', '\'', '+', '=', '~', '\\', '|', '&', '*', '%',
 '(', ')', '[', ']', '/', '^'],

    "spaces": [' ', \t', '\n', '\r']
};
```

5.3 定义 BibTex 格式规范

JSON 格式，操作方便，可扩展性强

tags: 形如 A={XX}中可能出现的 A 标签

necessity: 不可缺少的域如 author

option: 可选域如 pages

BibTex 默认按照自上而下的顺序输出 tags 域中标签值，如果需要更改顺序，直接修改顺序即可。

```
var BibTex = {

    "inproceedings": {
        "tags": {
            "author": "",
            "title": "",
            "booktitle": "",
            "pages": "",
            "year": ""
        },
        "necessity": ["author", "title", "booktitle", "year"],
        "option": ["pages"]
    },
```

5.4 主程序设计

按照一个字符一个字符地输入，根据输入的字符确定状态的转换。

```
function BibAnalysis(str) {

    while (i < str.length) {

        var ch = str.charAt(i);
        switch (state) {
            case 0:
                Bib_State_0(ch);
                break;
            case 1:
                Bib_State_1(ch);
                break;
            case 2:
```

下图以 state=1 为例，我们在这个状态中希望能够匹配并筛选出形如

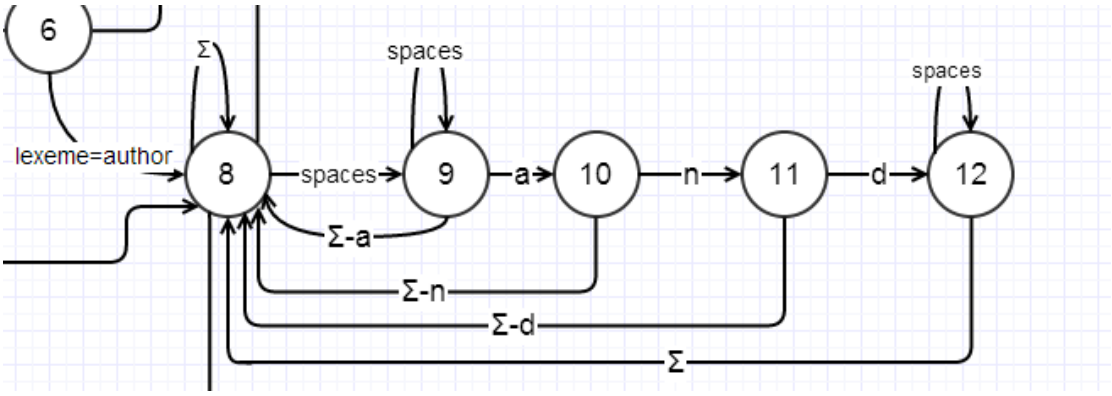
@article{...}中的“article”串。如果输入的字符属于 alphabets，则将字符拼接到 lexeme 后；如果输入的是‘{’并且分离出的 lexeme 属于“inproceedings”/“article”/“book”这样 BibTex 接受的类型，我们认为匹配成功，进入状态 3；否则，如不能成功匹配，则进入 error 状态 13。

```
//match @XXXXXX
function Bib_State_1(ch) {

    //letters
    if (contains(Sigma.alphabets, ch)) {
        lexeme = lexeme + ch;
    }
    //lexeme found in BibTex format: article book inproceedings.etc
    else if (ch == '{' && lexeme.toLowerCase() in BibTex) {
        stack.push(ch); // push '{'
        BibTex_Type = lexeme.toLowerCase();
        state = 2;
        lexeme = "";
    }
    //error
    else {
        state = 13;
```

5.5 分离出多个 Author

如下是分析并分离 AUTHOR 的自动机部分，进入状态 8 的前提：tag_name=lexeme=“author”



6. 待改进不足

字符集的定义还不够标准，后期需要仔细查阅 BibTex 格式文档。

程序健壮性需要提升。