Predictive parsing en LL(1) grammatica's

Contactinformatie:
Adres *ELIS (verdieping -3), Technicum (blok I), Sint-Pietersnieuwstraat 41*
E-mail *compilers@lists.ugent.be*

BELANGRIJK: Het indienen van de oplossingen dient steeds te gebeuren via het Dropbox-menu op Minerva. Gebruik het `make_tarball.sh` script om een archief te genereren, en zorg dat het naar de begeleiders verzonden wordt.

DEADLINE: 15 maart, 23:59.

1 Inleiding

1.1 BibTex formaat

BibTex is een formaat dat gebruikt wordt om een bibliografielijst bij te houden en hangt nauw samen met het LaTeX typesetting systeem. Het is een vrij oud formaat, maar wordt nog steeds heel veel gebruikt in de onderzoekswereld. Meer informatie kan je o.a. terugvinden op `http://bibliographic.openoffice.org/bibtex-defs.html`.

Let op dat je deze webpagina enkel gebruikt als bron van extra informatie omtrent het BibTex formaat. Volg dus steeds de richtlijnen in deze opgave, waarbij we soms lichtjes zullen afwijken van de strikte definities.

Bij de opgavebestanden, die je kan terugvinden op de website, zitten enkele voorbeeldjes van correcte en foute BibTex entries. Die kan je gebruiken bij het testen van jullie oplossing.

1.2 ANTLR - ANother Tool for Language Recognition

In dit practicum zal ANTLR v4 gebruikt worden (`http://www.antlr.org/`). ANTLR is een tool om automaten te genereren uitgaande van een grammatica-beschrijving in EBNF syntax (Extended Backus Naur Form). Merk op dat dezelfde syntax gebruikt wordt voor het genereren van token-herkenners, parsers en boom-parsers.

Er is voldoende documentatie over ANTLR v4 te vinden op `https://github.com/antlr/antlr4/blob/master/doc/index.md`. **Let op** dat je de documentatie van versie 4 gebruikt.

2 Opgave

2.1 Grammatica

Stel een grammatica op om BibTex entries te parsen. Omdat BibTex vrij uitgebreid is, zullen we ons beperken tot de volgende types entries:

- **`book`**: omschrijft een boek met een of meerdere auteurs
  verplichte velden: `label`, `author`, `title`, `year`, `publisher`
- **`article`**: omschrijft een artikel in een wetenschappelijk tijdschrift
  verplichte velden: `label`, `author`, `title`, `journal`, `pages`
  optionele velden: (`volume`, `number`) en/of `year` (één van beide is verplicht!)
- **`inproceedings`**: omschrijft een publicatie op een internationale conferentie
  verplichte velden: `label`, `author`, `title`, `booktitle`, `year`
  optionele velden: `month`, `pages` (v.d. vorm `<number> - - <number>`)

Vertrek hierbij van de volgende grammatica:

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

Opmerkingen:

- Ga ervan uit dat elk veld op een aparte lijn staat en dat er op het einde van elke lijn een komma (`,`) en een newline karakter staan.
- Enkel het `label` veld heeft een vaste plaats (b.v. `@book { book1, ...`), de andere velden mogen in een willekeurige volgorde voorkomen. Zorg er dan ook voor dat alle mogelijke volgordes aanvaard worden.
- Een lijst van auteurs wordt aangegeven door elke auteur te scheiden m.b.v. `AND` of `and`, b.v.:

  ```
  Jan Spier AND Suske Van Antigoon and Robbe Does
  ```

  Zorg ervoor dat de grammatica aanneemt dat er steeds minstens één auteur is.
- Bij het opstellen van de grammatica hoef je nog geen rekening te houden met verplichte en optionele velden. Beschouw ieder veld als optioneel, maar zorg er wel voor dat er naast het `label` veld minstens één ander veld voorkomt. Je hoeft ook geen rekening te houden met het al dan niet dubbel voorkomen van velden.

De bekomen grammatica kan vervolgens LL(1) gemaakt worden aan de hand van **left recursion elimination** en **left factoring**.

2.2 Implementatie

ANTLR ondersteunt tal van talen waaronder C, C++, Java, ... In dit practicum is het de bedoeling om Java als doeltaal te gebruiken. Java heeft als voordeel goed ondersteund te worden in de *ANTLRWorks* omgeving. Merk op dat je `javac` nodig hebt om de grammatica te genereren. Installeer het in de virtual machine met de volgende commando's:

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

Maak met behulp van ANTLR een **LL(1)** parser die jouw grammatica implementeert (`Pract3.g4`). De parser dient als uitvoer de geparsete BibTex entries te produceren in het EndNote XML formaat. Als voorbeeld van een geldig EndNote XML document werd `endnote_correct.xml` meegeleverd met deze opgave. Dit is tevens de output die je parser moet genereren na het parsen van `bibtex_correct.xml`.

De overbodige input (lijnen die ons niet interesseren) mag je weggooien, m.a.w. hiervoor genereer je geen EndNote velden. Je kan de output controleren door die via een tekstbestand te importeren in EndNote (toegankelijk via Athena, kies `EndNote generated XML` bij `Import Option`).

Beide grammatica's (lexer en parser) moeten in hetzelfde ANTLR grammaticabestand geïmplementeerd worden. M.b.v. de `Run > Run in TestRig` functie in ANTLRWorks genereer je de Java broncode die vervolgens door ANTLRWorks gecompileerd wordt naar Java bytecode. Alle nodige files zijn te vinden in de `pract3.tar.gz` file.

Nog enkele opmerkingen:

- Bij een BibTex-entry zijn er meerdere syntax mogelijkheden. Zo kunnen zowel '`{`' en '`}`' als '`(`' en '`)`' gebruikt worden voor de volledige entry, en kunnen ofwel accolades (`{, }`) ofwel aanhalingstekens (`"`) gebruikt worden om strings te beschrijven.

- Zoals eerder vermeld, wordt een lijst van auteurs in BibTex aangegeven door elke auteur te scheiden m.b.v. `AND` of `and`. In EndNote XML, daarentegen, moet elke auteur apart vermeld staan. Gebruik hiervoor `<author> ... </author>`.

- Bij het `month`-veld zijn er meerdere mogelijkheden: ofwel d.m.v. een tekstuele afkorting (`Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, `Dec`), ofwel door de volledige maand te vermelden (`January`, ...), ofwel d.m.v. een getal (`1`, `2`, ..., `12`). Zorg bij de verschillende mogelijkheden telkens voor een gepaste foutboodschap indien er een foute waarde wordt meegegeven.

- Ga na bij het inlezen van de pagina-nummers of het eerste getal kleiner is dan het tweede getal, m.a.w. of de opgegeven waarde zinnig is. Dit mag gebeuren aan de hand van gewone Java code en hoeft dus niet te gebeuren via parser-regels.