ADQL PEG Grammar

Grégory Mantelet[1], Jon Juaristi Campillo[2], Markus Demleitner[2]

[1]CDS (Centre de Données astronomiques de Strasbourg)
[2]GAVO (ARI, Heidelberg)

21st May 2024

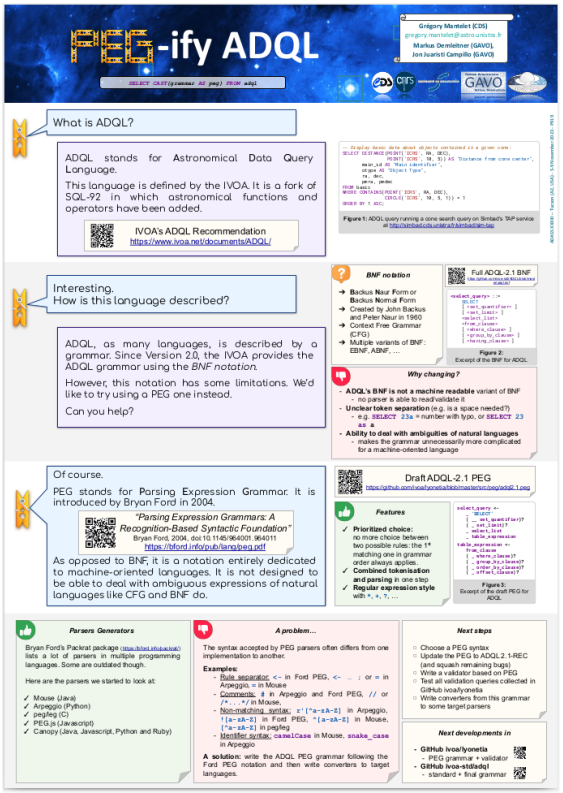
Université de Strasbourg

Observatoire astronomique de Strasbourg | ObAS

CDS Centre de Données astronomiques de Strasbourg

ADQL grammar

- In ADQL-2.1, defined with a **BNF** (Backus Naur Form)
  - Multiple known variants: EBNF, ABNF, ...
  - ADQL's BNF notation is kind of unique
  - ....so, not machine readable
  - ...so, impossible to generate a reliable parser/validator
- Solution: **PEG** notation.
- *Poster at ADASS XXXIII, Tucson, Grégory Mantelet, Jon Juaristi Campillo and Markus Demleitner*



Why PEG instead?

Why changing?

- **ADQL's BNF is not a machine readable** variant of BNF
  - no parser is able to read/validate it
- **Unclear token separation** (e.g. is a space needed?)
  - e.g. `SELECT 23a` = number with typo, or `SELECT 23 as a`
- **Ability to deal with ambiguities of natural languages**
  - makes the grammar unnecessarily more complicated for a machine-oriented language

BNF vs PEG

ADQL in BNF

```
<select_query> ::=
    SELECT
    [ <set_quantifier> ]
    [ <set_limit> ]
    <select_list>
    <from_clause>
    [ <where_clause> ]
    [ <group_by_clause> ]
    [ <having_clause> ]
```

ADQL in PEG

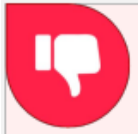
```
select_query <-
    _ 'SELECT'
    ( __ set_quantifier)?
    ( _ set_limit)?
    _ select_list
    _ table_expression

table_expression <-
    from_clause
    ( _ where_clause)?
    ( _ group_by_clause)?
    ( _ order_by_clause)?
    ( _ offset_clause)?
```

PEG parser generators

Parser generator	Target language
Mouse	Java
Arpeggio	Python
peg/leg	C
PEG.js	Javascript
Canopy	Java, Javascript, Python, Ruby
...	...

But. . . .



A problem…

The syntax accepted by PEG parsers often differs from one implementation to another.

Examples:

- Rule separator: `<-` in Ford PEG, `<- … ;` or `=` in Arpeggio, `=` in Mouse
- Comments: `#` in Arpeggio and Ford PEG, `//` or `/*...*/` in Mouse,
- Non-matching syntax: `r'[^a-zA-Z]` in Arpeggio, `![a-zA-Z]` in Ford PEG, `^[a-zA-Z]` in Mouse, `[^a-zA-Z]` in peg/leg
- Identifier syntax: `camelCase` in Mouse, `snake_case` in Arpeggio

What can we done, then?

*[**Disclaimer:** This is still experimental and may change in function of encountered difficulties.]*

1. Write the ADQL grammar in a single PEG notation
   - **Chosen notation:** Ford PEG *(the first and original PEG notation)*
2. Provide a converter from this notation into any desired variant
   - **How?** *Makefile* applying substitutions with regular expressions.
3. Validate the generated grammars
   - **How?** Use GitHub CI to generate all parsers and to run all tests listed in the GitHub repository Lyonetia

Current status

- Makefile prototype started
- Target parser generator: Canopy
- Conversion into Canopy's PEG grammar

```
${PEG_ORIGINAL}: ${DIR_GRAMMARS}
    wget -O "${PEG_ORIGINAL}" '${URL_PEG_GRAMMAR}'

${PEG_CANOPY_JAVA}: #${PEG_ORIGINAL}
    @echo "grammar ADQL\n" > ${PEG_CANOPY_JAVA}
    @sed -e 's/\(^\|[ \t\n\r]\)_\([ \t\n\r]\|$$\)/\1MAYBE_SPACE\2/g' \
         -e 's/\(^\|[ \t\n\r]\)__\([ \t\n\r]\|$$\)/\1SPACES\2/g' \
         "${PEG_ORIGINAL}" >> ${PEG_CANOPY_JAVA}
```

- Generation into a Java parser successful
- Parsing test with a very simple Java program successful

What next?

1. Generate and test with Canopy for other possibles languages (Python and Ruby)
2. Do the same with the other parser generators
3. Try to validate test suite of Lyonetia with one generated parser
4. Do the same with the other generated parsers
5. Create CI to perform all these tests whenever something is pushed on Lyonetia

Appendix A - History

- **Apr. 2017**: First detailed mention of PEG in DAL Email *(by G. Mantelet)*
- **May 2017**: Walter Landry slides *(Interop May 17, Shanghai)*
- **Oct. 2018**: First version of ADQL grammar in PEG notation *(by J. Juaristi Campillo on Lyonetia GitHub repo)*
- **May 2019**: Jon Juaristi slides *(Interop May 19, Paris)*
- **Nov. 2013**: ADASS poster *(by G. Mantelet, M. Demleitner, J. Juaristi Campillo)*

Appendix B - Useful links

- **About PEG:**
  - Bryan Ford paper about PEG
  - PEG parser generators suggested by Bryan Ford
  - PEG.js *(online PEG parser)*
- **About ADQL:**
  - adql2.1.peg *(in-development ADQL grammar in PEG notation)*
  - Lyonetia *(GitHub repository for ADQL validation)*