POPL Mid Project Report

dlql (Digital Library Query Language)

Jaffrey Joy
2022201006

Amit Marathe
2022201013

IIIT Hyderabad

1 Objective:

Build an `S-expression` based query language to search the ACM Digital Library with granular filters. Queries can include both **selection** (filter data based on a fixed set of attributes) and **projection** (choose what you want to see about the filtered data). A query result (logical)[1] can be reused by applying more selection or projection operations on the **variables** containing these query results.

2 Concrete Syntax:

The concrete syntax for `dlql` is defined as follows:

2.1 Program expression

```
;;  <exp> ::= <query-result>
;;          | (<list-of-define-query> <run-stmt>)
;;
;;  <list-of-define-query> := ()
;;                         | ((define-query <symbol> <query>) <list-of-define-query>)
;;
;;  <run-stmt> ::= (run-query <query>)
;;               | (run-query (project <list-of-project-attr> <query>))
```

2.2 Query expression

```
;;  <query> ::= <symbol>
;;            | (conj <list-of-query>)
;;            | (disj <list-of-query>)
;;            | (<select-attr> (conj <list-of-attr>))
;;            | (<select-attr> (disj <list-of-attr>))
;;
;;  <list-of-query> ::= ()
;;                    | (<query> <list-of-query>)
;;
;;  <list-of-attr> ::= ()
;;                    | (<attr> <list-of-attr>)
;;
;;  <attr> ::= <string>
;;
;;  <list-of-project-attr> ::= ()
;;                         | (<project-attr> <list-of-project-attr>)
```

[1] *logical* since queries mapped to the variables are re-evaluated to produce the *query result*

2.3 Select and Project attributes

```
;;  <select-attr> ::= pub-date
;;                  | paper-title
;;                  | pub-title
;;                  | author
;;                  | abstract
;;                  | full-text
;;                  | conf-location
;;                  | conf-sponsor
;;                  | isbn
;;                  | doi
;;
;;  <project-attr> ::= paper-title
;;                   | authors
;;                   | issued-in
;;                   | page-count
;;                   | pub-date
;;                   | doi
;;                   | abstract
;;                   | citation-count
;;                   | references
;;                   | citations
```

NOTE: Some aspects of the grammar are subject to change based on implementation.

3 Implementation

We will first create a parser for the grammar we just defined. We use the `define-datatype` provided by `eopl` to define our language where the major datatypes are:

- `exp`
- `query`
- `query-result`
- `select-attr`
- `project-attr`

The *AST* generated by the parser will then be first fed into a *query-plan* generator, which might flatten some of the expressions to optimize the query.

This *query-plan* will then be fed to an evaluation engine. This is where the actual execution part of (getting data that the query expects) will be offloaded to a `python` module using an API call. The results returned by the said `python` module with then be further used, as the *query-plan* tree is walked and evaluated till the root is reached which might involved a project operation.