POPL Final 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 and nested filters. Queries can be filtered base on a fixed amount of available **selection** attributes (mentioned in the grammar section). A query result (logical)[1] can be reused by applying more selection operations on the **variables** containing these (logical) query results.

2 Syntax:

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

2.1 Program expression

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

[1] *logical* since queries mapped to the variables are evaluated only in the run statement to produce the *query result*

2.3 Select attributes

```
;;  <select-attr> ::= paper-title
;;                  | pub-title
;;                  | author
;;                  | abstract
;;                  | full-text
;;                  | conf-location
;;                  | conf-sponsor
;;                  | isbn
;;                  | doi
```

NOTE: The **project** clause and its attributes were omitted in the final version of the grammar.

3 Implementation

The interpreter for the DSL uses **lazy evaluation** for the query variables declared in the `define-query` statements. To enable this, a hashmap is maintained which acts as an environment for the query variables. Initially every key in the map i.e. the symbol/id of the query variable is assigned a value i.e. the query AST it corresponds to. If this variable then appears in the run statement or in a query corresponding to a variable encountered during the evaluation of the run statement, only then will the query be evaluated and its corresponding entry in the hashmap will be updated with its evaluated string. The next time the variable is referred this evaluated value will be used. In this way we avoid evaluating any unnecessary variables.

After the query string is generated we append it to the base url and open it in a browser (as opposed to the original plan to scrape data using python and present in in the command line itself). This is because the result interface provied by ACM has far too many options that cant be provided through the current implementation through the command line. So, for now, we left what is already done well by the ACM website to them. It would be possible to make this better, if ACM provides a REST API to its digital library, as this could be exploited to add back our omitted **project** clause which can then be used within the **select** clause where the data is filtered based on results of queries rather than manual string literals written by the user which makes it more dynamic in nature.