Architecture of the QSemanticDB implementation

Rehno Lindeque

March 28, 2010

Abstract

The architecture an OpenSemanticDB reference implementation is described.

1 Introduction

1.1 Terminology

- *Semantic symbol / symbol* – The conceptual representation of a semantic value. For example a could be a symbol and so could $a.b$
- *Semantic id / id* – The internal representation of some semantic symbol.
- *Fully qualified id* – An id that represents some semantic symbol inside some context.
- *Unqualified id* – An id that represents some semantic symbol disregarding the context it is found in. (I.e. $a.(b.c)$ is qualified, but c or even $b.c$ is unqualified with reference to a).
- *Relation* – A declaration in the form $a \rightarrow b$ which is represented by the existence of the query $a.b$.
- *Unqualified relation* – A pair of id's $(domainId, codomainId)$ such that $codomainId$ is unqualified with respect to $domainId$.

1.2 Data structures

The following basic indexes are needed.

- SplitRelations: $qualifiedCodomain \rightarrow (domain, unqualifiedCodomain)$
- UnifyRelations: $(domain, unqualifiedCodomain) \rightarrow qualifiedCodomain$
- DomainQCodomains: $domain \rightarrow qualifiedCodomain$
- DomainUCodomains: $domain \rightarrow unqualifiedCodomain$

1.3 Special relations

In order to represent certain operations and syntactic sugar, certain special structures must be reserved.

1.3.1 Anonymous symbols

An anonymous symbol is syntactic sugar that allows a user to select from a set without binding it to its own symbol. Hence the database must assign automatically generate an anonymous symbol for the set. Furthermore, anonymous sets can be nested inside other queries and sets, so the evaluator needs to ignore these symbols when they are not used in a query.

An anonymous set is structured as follows:

$anonymous \rightarrow HIDDEN \rightarrow \{x\ y\ z\}$

where *anonymous* is an automatically generated id and $HIDDEN$ is a reserved id used to prevent the evaluator from evaluating *anonymous* or any of its codomains (aside from the query that uses the symbol).

Hence, a query such as `result = { { x y z }.z }` , will be represented as:

$result \rightarrow \{anonymous \rightarrow HIDDEN \rightarrow \{x\ y\ z\}\} \xrightarrow[speculative]{} x$

1.4 Scheduler

In order to perform queries with undetermined outcomes a scheduler is needed. This allows the interpreter to perform queries ahead of the evaluation steps that returns the string of symbols to a client program.

The scheduler works in the following manner:

- First, when the interpreter recieves its first symbol it is pushed on to the schedule.
- The evaluator pops the first available symbol from the schedule. If this is the last symbol in the schedule, then the codomain of the symbol is pushed onto the schedule.
- If the symbol is a query then all of its speculative codomains are pushed onto the schedule.
-