Simple Aggregations in Algebraic Databases

Patrick Schultz, David I. Spivak, Ryan Wisnesky

Abstract

This document describes an extension to the Algebraic Databases formalism (Schultz & Wisnesky, 2017) that allows for simple aggregations in uber-flower queries.

1 Extending Multi-sorted Equational Logic

In this paper we define a syntax, semantics, and proof system that extends multi-sorted equational logic. This system is used in the AQL tool to implement simple aggregations in uber-flower queries.

1.1 Syntax

A *signature Sig* consists of:

1. A set *Sorts* whose elements are called *sorts*,
2. A set *Symbols* of pairs $(f, s_1 \times \ldots \times s_k \to s)$ with $s_1, \ldots, s_k, s \in \mathit{Sorts}$ and no f occurring in two distinct pairs. We write $f : X$ instead of $(f, X) \in \mathit{Symbols}$. When $k = 0$, we may call f a *constant symbol* and write $f : s$ instead of $f : \to s$. Otherwise, we may call f a *function symbol*.

We assume we have some countably infinite set $\{v_1, v_2, \ldots\}$, whose elements we call *variables* and which are assumed to be distinct from any sort or symbol we ever consider. A *context* Γ is defined as a finite set of variable-sort pairs, with no variable given more than one sort:

$$\Gamma := \{v_1 : s_1, \ldots, v_k : s_k\}$$

We inductively define the set $\mathit{Terms}^s(\mathit{Sig}, \Gamma)$ of *terms* of sort s over signature *Sig* and context Γ as:

1. $x \in \mathit{Terms}^s(\mathit{Sig}, \Gamma)$, if $x : s \in \Gamma$,
2. $f(t_1, \ldots, t_k) \in \mathit{Terms}^s(\mathit{Sig}, \Gamma)$, if $f : s_1 \times \ldots \times s_k \to s$ and $t_i \in \mathit{Terms}^{s_i}(\mathit{Sig}, \Gamma)$ for $i = 1, \ldots, k$. When $k = 0$, we may write f for $f()$.
3. $(f, o)\{\texttt{for } \Gamma' \texttt{ where } t_1 = t'_1, \ldots, t_k = t'_k \texttt{ return } e\} \in \mathit{Terms}^s(\mathit{Sig}, \Gamma)$ when Γ' is a context such that $\Gamma \cap \Gamma' = \emptyset$ and $f : s \times s \to s$ and $o : s$ and $e \in \mathit{Terms}^s(\mathit{Sig}, \Gamma \cup \Gamma')$ and for every $1 \leq i \leq k$ there exists a sort s_i such that $t_i, t'_i \in \mathit{Terms}^{s_i}(\mathit{Sig}, \Gamma \cup \Gamma')$.

Note that the monoid comprehensions (`for` Γ'-terms) are binding constructs: the variables in Γ' are considered bound. (Capture-avoiding) substitution of a term t for a variable v in a term e is written as $e[v \mapsto t]$ and is defined as usual.

An *equation* over *Sig* is a formula $\forall\Gamma.\ t_1 = t_2 : s$ with $t_1, t_2 \in \mathit{Terms}^s(\mathit{Sig},\Gamma)$; we will omit the $: s$ when doing so will not lead to confusion. A *theory* is a pair of a signature and a set of equations over that signature. Associated with a theory Th is a binary relation between terms, called *provable equality*. We write $Th \vdash \forall\Gamma.\ t = t' : s$ to indicate that the theory Th proves that terms $t, t' \in \mathit{Terms}^s(\mathit{Sig},\Gamma)$ are equal according to the usual rules of multi-sorted equational logic extended with five monoid comprehension laws. The equational logic rules are

$$\frac{t \in \mathit{Terms}^s(\mathit{Sig},\Gamma)}{Th \vdash \forall\Gamma.\ t = t : s} \qquad \frac{Th \vdash \forall\Gamma.\ t = t' : s}{Th \vdash \forall\Gamma.\ t' = t : s} \qquad \frac{Th \vdash \forall\Gamma.\ t = t' : s \qquad Th \vdash \forall\Gamma.\ t' = t'' : s}{Th \vdash \forall\Gamma.\ t = t'' : s}$$

$$\frac{Th \vdash \forall\Gamma.\ t = t' : s \qquad v \notin \Gamma}{Th \vdash \forall\Gamma, v : s'.\ t = t' : s} \qquad \frac{Th \vdash \forall\Gamma, v : s.\ t = t' : s' \qquad Th \vdash \forall\Gamma.\ e = e' : s}{Th \vdash \forall\Gamma.\ t[v \mapsto e] = t'[v \mapsto e'] : s'}$$

and the monoid rules are

$$\frac{}{Th \vdash \forall\Gamma.\ (f,o)\{\texttt{for}\ -\ \texttt{where}\ -\ \texttt{return}\ e\} = e}$$

$$\frac{}{Th \vdash \forall\Gamma.\ (f,o)\{\texttt{for}\ \Gamma'\ \texttt{where}\ \phi\ \texttt{return}\ o\} = o}$$

$$\frac{}{\begin{array}{c}Th \vdash \forall\Gamma.\ (f,o)\{\texttt{for}\ \Gamma'\ \texttt{where}\ \phi\ \texttt{return}\ f(e,e')\} = f\big(\\ (f,o)\{\texttt{for}\ \Gamma'\ \texttt{where}\ \phi\ \texttt{return}\ e\}, \qquad (f,o)\{\texttt{for}\ \Gamma'\ \texttt{where}\ \phi\ \texttt{return}\ e'\}\ \big)\end{array}}$$

$$\frac{}{\begin{array}{c}Th \vdash \forall\Gamma.\ (f,o)\{\texttt{for}\ \Gamma'\ \texttt{where}\ \phi\ \texttt{return}\ (f,o)\{\texttt{for}\ \Gamma''\ \texttt{where}\ \phi'\ \texttt{return}\ e\} = \\ (f,o)\{\texttt{for}\ \Gamma' \cup \Gamma''\ \texttt{where}\ \phi \cup \phi\ \texttt{return}\ e\}\end{array}}$$

$$\frac{Th \vdash h(o) = o' \qquad Th \vdash \forall xy.h(f(x,y)) = f'(h(x),h(y))}{\begin{array}{c}Th \vdash \forall\Gamma.h\big(\ (f,o)\{\texttt{for}\ \Gamma'\ \texttt{where}\ \phi\ \texttt{return}\ e\}\ \big) = \\ (f',o')\{\texttt{for}\ \Gamma'\ \texttt{where}\ \phi\ \texttt{return}\ h(e)\}\end{array}}$$

We say that a theory Th is *OK for aggregation* when, for every term in Th of the form

$$(f,o)\{\texttt{for}\ \Gamma\ \texttt{where}\ t\ \texttt{return}\ e\}$$

we have

$$Th \vdash' \forall xyz.\ f(x, f(y,z)) = f(f(x,y),z) \quad Th \vdash' \forall x.\ f(x,o) = x = f(o,x) \quad Th \vdash' \forall xy.f(x,y) = f(y,x)$$

where $Th \vdash' \phi$ indicates that Th entails ϕ under the rules of multi-sorted equational logic without use of the monoid laws. We only consider theories that are OK for aggregation.

1.2 Semantics

An *algebra* A over a signature *Sig* consists of:

- a set $A(s)$ for each sort s; the elements of $A(s)$ are called *carriers*, and

- a function $A(f) : A(s_1) \times \ldots \times A(s_k) \to A(s)$ for each symbol $f : s_1 \times \ldots s_k \to s$.

Let $\Gamma := \{x_1 : s_1, \ldots, x_n : s_n\}$ be a context. An *A-environment* η for Γ associates each variable x_i with an element of $A(s_i)$. Write $[\![\Gamma]\!]$ to indicate the set of all A-environments for Γ.

The meaning of a term t in $Terms(Sig, \Gamma)$ relative to A-environment η for Γ is written $[\![t]\!]\eta$ and recursively defined as:

$$[\![x]\!]\eta = \eta(x) \qquad [\![f(t_1, \ldots, t_n)]\!]\eta = A(f)([\![t_1]\!]\eta, \ldots, [\![t_i]\!]\eta)$$

To extend the above definition to aggregations

$$[\![(f, o)\{\texttt{for}\ \Gamma'\ \texttt{where}\ t_1 = t'_1, \ldots\ \texttt{return}\ e\}]\!]\eta$$

first define the set

$$\zeta := \{\eta' \mid \eta' \in [\![\Gamma']\!]\ , [\![t_1]\!]\eta \cup \eta' = [\![t'_1]\!]\eta \cup \eta', \ldots\}$$

and the meaning of the agg term is

- $A(o)$ if ζ is empty,
- $[\![e]\!]\eta'_1 \cup \eta$ if $\zeta = \{\eta'_1\}$,
- $A(f)([\![e]\!]\eta'_1 \cup \eta, [\![e]\!]\eta'_2 \cup \eta)$ if $\zeta = \{\eta'_1, \eta'_2\}$,
- etc

References

Patrick Shultz & Ryan Wisnesky *Algebraic Data Integration*.