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 $\text{Sig}$ consists of:
1. A set $\text{Sorts}$ whose elements are called $\text{sorts}$,
2. A set $\text{Symbols}$ of pairs $(f, s_1 \times \ldots \times s_k \rightarrow s)$ with $s_1, \ldots, s_k, s \in \text{Sorts}$ and no $f$ occurring in two distinct pairs. We write $f : X$ instead of $(f, X) \in \text{Symbols}$. When $k = 0$, we may call $f$ a constant symbol and write $f : s$ instead of $f : \rightarrow s$. 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 $\text{variables}$ and which are assumed to be distinct from any sort or symbol we ever consider.

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

Note that the monoid comprehensions ($\text{for } \Gamma'$-terms) are binding constructs: the variables in $\Gamma'$ 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 $\text{Sig}$ is a formula $\forall \Gamma. t_1 = t_2 : s$ with $t_1, t_2 \in \text{Terms}^e(\text{Sig}, \Gamma)$; 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 $t, t' \in \text{Terms}^e(\text{Sig}, \Gamma)$ are equal according to the usual rules of multi-sorted equational logic extended with five monoid comprehension laws. The equational logic rules are

$$
\begin{align*}
t \in \text{Terms}^e(\text{Sig}, \Gamma) & \quad Th \vdash \forall \Gamma. t = t' : s & & Th \vdash \forall \Gamma. t = t' : s & & Th \vdash \forall \Gamma. t = t'' : s \\
Th \vdash \forall \Gamma. t = t' : s & \quad v \notin \Gamma & & Th \vdash \forall \Gamma. v : s. t = t' : s & & Th \vdash \forall \Gamma. e = e' : s \\
Th \vdash \forall \Gamma. v : s'. t = t' : s & & Th \vdash \forall \Gamma. t[v \mapsto e] = t'[v \mapsto e'] : s'
\end{align*}
$$

and the monoid rules are

$$
Th \vdash \forall \Gamma. (f,o)\{\text{for } \phi \text{ where } o \text{ return } e]\} = e
$$

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

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

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

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

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

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

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 $\phi$ 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 $\text{Sig}$ consists of:

- a set $A(s)$ for each sort $s$; the elements of $A(s)$ are called carriers, and
• a function $A(f) : A(s_1) × \ldots × A(s_k) → A(s)$ for each symbol $f : s_1 × \ldots s_k → s$.

Let $Γ := \{x_1 : s_1, \ldots x_n : s_n\}$ 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 $Terms(Sig, Γ)$ relative to $A$-environment $η$ for $Γ$ is written $[t] η$ and recursively defined as:

\[
\begin{align*}
[x] η &= η(x) \\
[f(t_1, \ldots t_n)] η &= A(f)([t_1] η, \ldots, [t_n] η)
\end{align*}
\]

To extend the above definition to aggregations

\[
[(f, o)\{\text{for} \ Γ'\ \text{where} \ t_1 = t'_1, \ldots \ \text{return} \ e}\}] η
\]

first define the set

$$ζ := \{η' | η' ∈ [Γ'] , [t_1] η ∪ η' = [t'_1] η ∪ η', \ldots\}$$

and the meaning of the agg term is

• $A(o)$ if $ζ$ is empty,
• $[e] η_1' ∪ η$ if $ζ = \{η_1'\}$,
• $A(f)([e] η_1' ∪ η, [e] η_2' ∪ η)$ if $ζ = \{η_1', η_2'\}$.
• etc

References

Patrick Shultz & Ryan Wisnesky *Algebraic Data Integration*. 