IVC

July 27, 2011

1 Definitions

In this section we explain the domain which is represented here by dom. The dom of a variable is a set of values that the variable can take. The dom of a map is defined by all the values of both the input and output variables(which are represented as vectors).

$$\mathsf{dom}_i = \left\{ x \in map_i[\vec{x}][\vec{y}] | (x \in \vec{x}) \vee (x \in \vec{y}) \right\}$$

We need to compute the dependences between the domains of maps.The domains of input variables will always depend on the domains of output variables from maps that are on the left side of the map with input variables. Without loose of generality we always need to work with two maps when we want to compute their doms. For this we define function F recursively as follow:

$$F = \mathsf{dom}_i | F \cap F | F \cup F$$

Function F takes two sets(two doms) and returns their unions or intersections. We can apply this function recursively to compute any set of unions or intersections between the doms. Also we can achieve this by giving a set of sets to F to compute the desirable set. So we can compute the dependencies between the doms like this:

$$\mathsf{dom}_i(x \in m_i[\vec{x}][\vec{y}]) \leftarrow F(\mathsf{dom}_i(x' \leftarrow m_j[\vec{x'}][\vec{y'}]))$$

Where j is all the maps which occurs on the left hand of phrase for computing m_i as we have in **??** since the information can only be passed from left to right.

We call the arity of a schema as the number of its input variables. For example we will have the relation:

$$q = R(a, b) \cdot S(b, c)$$

the schema for q will have three variables (a, b, c) The arity of a tuple will be the number of occurrences in the relation, in other words the order of multiplicity of that tuple. The arity of a tuple will increase or decrease if insertions or respectively deletions will be made to a relation.

For example:

Relation R:

R	a	b	arity
	<1	2>	1
	<1	3>	2
	<3	4>	1

Relation S:

S	b	c	arity
	<1	1>	1
	<2	2>	2
	<3	5>	2

Relation $R \cdot S$: The arity of a tuple is the number of its multiplicity. In the

$R \cdot S$	a	b	c	arity
	<1	2	2>	2
	<1	3	5>	4
	<3	4	$*>$	0
	$<*$	1	1>	0

other words with insertion and deletion of tuples we increase and decrease the arity of the tuple respectively.

We need to maintain the maps for certain values as long as they have a tuple with non-zero arity. If the arity of the tuple drops to zero it means that this tuple does not present in the map anymore.

We need to store the arities inside each map. We need a way to compute the arity of an AGCA expression. Since we substitute the subexpressions with maps, we can easily consider the maps without input variables as some relations. Also a map with input variables can be seen as a relation with a group-by clause. Input variable of a map bind some variables. Thus if we compute the map values by all different combinations of these variables, we will look up into these values and return the appropriate value according to the input variables.

2 Computing the arities of the AGCA expression

We can compute the arities recursively. Suppose we have an expression which is consist of two parts joining together as:

$$e = e_1 + e_2 | e_1 \cdot e_2 | v \leftarrow e_1 | e_1 \Theta e_2 | R(.)$$

Having the arities of e_1, e_2, we can compute the arity of e easily. For example the arity of comparison operator Θ is either 0 or 1. The arity of $e_1 + e_2$ is the sum of the arity of e_1, e_2. The arity of product is also computable from the join operator. For assignment the arity is always 1. Thus, with the above argument we can compute the arity of all AGCA expression easily.