Addendum To Technical Report on "Exploring Parameterized Relational Consistency" (TR-UNL-CSE-2009-0009)

Shant K. Karakashian*, Robert J. Woodward*,
Berthe Y. Choueiry*, and Christian Bessiere**

*Constraint Systems Laboratory
Department of Computer Science & Engineering
University of Nebraska-Lincoln
Email: {shantk|rwoodwar|choueiry}@cse.unl.edu

**LIRMM-CNRS
University Montpellier, France
Email: bessiere@lirmm.fr

November 6, 2009

In this document, we revise the pseudo-code of all three algorithms in the technical report, improving on their performance.

Initializing the constraints queue

The initialization phase Algorithm 1 builds a queue of all combination-relation pairs.

Algorithm 1: INITIALIZE-$\mathcal{Q}$ initializes the queue.

```
Input: $\zeta$: set of all possible combinations
Output: $\mathcal{Q}$: a queue of all combination-constraint pairs
foreach $\varphi \in \zeta$ do
    foreach $R \in \varphi$ do
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\langle \varphi, R \rangle\}$
    end
end
$revision\text{-}time \leftarrow 0$
```

Processing the constraint queue

The procedure PROCESSQUEUE, described in Algorithm 2, revises every relation-combination pair in the queue to ensure that all their tuples are supported in each combination of m constraints where the relation appears.

We modified the queue of relations (as described in the technical report), into a queue of combination-relation pairs for the following reason. Originally, when a relation R_i is popped from the queue for revision,

- It was revised in *every combination* where it appears, and
- When the revision modified R_i, every other relation in every other combination where the relation R_i appears was inserted in the queue.

According to the new queue management strategy, when a pair of combination-relation $\langle \varphi, R_i \rangle$ is popped from the queue for revision,

- It is revised in *only* the paired combinationϕ, and
- When the revision modified R_i, every other relation in every other combination where the relation R_i appears is inserted in the queue paired with the corresponding combination.

This mechanics saves in computational effort, while maintaining soundness and completeness.

Algorithm 2: ProcessQueue deletes tuples that have lost their support.

```
   Input: $\mathcal{Q}$, $\zeta$,$revisiouTime$
   Output: $true$ is the problem is R($*$,$m$)C, $false$ otherwise
1  $consistent \leftarrow true$
2  while $(\mathcal{Q} \neq \emptyset) \wedge (consistent = true)$ do
3      $\langle \varphi, R \rangle \leftarrow$ Top($\mathcal{Q}$)
4      $revision\text{-}time \leftarrow revision\text{-}time$ +1
5      foreach $\langle \varphi, R' \rangle \in \mathcal{Q}$ do
6          Remove($\langle \varphi, R' \rangle, \mathcal{Q}$)
7          $deleted \leftarrow false$
8          foreach $\tau \in R'$ do
9              if RevisionTime($\tau$) $= revision\text{-}time$ then
10                 GoTo 8
11             end
12             $support \leftarrow$FindSupport($(\tau, R'), \varphi$)
13             if $support = false$ then
14                 Delete($\tau$)
15                 if $R' = \emptyset$ then
16                     $consistent \leftarrow false$
17                     GoTo 29
18                     $deleted \leftarrow true$
19                 end
20             end
21         end
22         if $deleted$ then foreach $\varphi' \in \zeta$ do
23             if $R' \in \varphi'$ then foreach $R'' \in (\varphi' \setminus \{R'\})$ do
24                 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\langle \varphi', R'' \rangle\}$
25             end
26         end
27     end
28 end
29 return $consistent$
```

To access all the combination-relation pairs in the queue pertaining to the same combination, we implement a hash-table on the queue whose indices are combinations and the values are the relations in the combinations.

Further, when we find the tuples $\{\tau'\}$ that support the tuple τ in a given combination ϕ, all those tuples are guaranteed 'support' and need not be rechecked for support in the combination ϕ. We use a 'time stamp' mechanism to record this situation and save redundant checks, see Line 10.

revision-time is a global variable throughout the execution so that the time stamp uniquely marks a revision of a combination. The time stamp remains the same during the revision of all the relations in a given combination. For that purpose, we need to revise, for a given same combination, all combination-relation pairs in the queue sequentially.

Finding a support

The marking of the tuples with the time stamp is performed in the FindSupport algorithm. Every time a support is found (either by search or simply retrieved from the data structure *Last*), all the tuples in the support are marked with the time stamp in Line 10 of Algorithm 3.

Algorithm 3: FindSupport finds a support for a tuple in a combination.

```
   Input: $(\tau, R_i)$, $\varphi$, revision-time
1  support ← true
2  if Last($(\tau, R_i), \varphi$) = $\emptyset$ then
3      Last($(\tau, R_i), \varphi$) ← Search($\varphi$, $R_i \leftarrow \tau$)
4      if Last($(\tau, R_i), \varphi$) = $\emptyset$ then
5          support ← false
6          Goto 12
7      end
8  end
9  foreach $\tau' \in$ Last($(\tau, R_i), \varphi$) do
10     RevisionTime($\tau'$) ← revision-time
11 end
12 return support
```