Report July 23th

While discussing with Umberto, I told him my doubts, and ideas. Based on that, he explained and clarified to me important issues about the algorithm.

Dependency between variables The trick point here is: if you have a variable which is not in the head of the service definition, and this variable is used in different services, it is impossible to guarantee that both of them are the same variable with the same associated value. In the case of services that have those variables they have to cover the entire query. In our example:

Q(disease, patients, dnas, birth, gen, addr, wheatherstatistic) :- A1(disease, patients), A2(patients,dnas), A3(patients, birth, gen, addr), A4(addr, wheatherstatistic)

S2(disease, patientsssn, dna) :- A1(disease, patientsssn), A2(patientsssn, dna)
S3(patientsssn, dna) :- A2(patientsssn, dna)
S4(patientsssn, dateofbirth, gender, address) :- A3(patientsssn, dateofbirth, gender, address)
S5(address, wheatherstaticalinformation) :- A4(address, wheatherstaticalinformation)
S6(patientsssn, diseases) :- A5(patientsssn, diseases)
S7(patientsssn, vaccinateddiseases) :- A6(patientsssn, vaccinateddiseases)

In this case, no PCD is created for S2 because I can not guarantee that the 'patientsssn' which is not in the head of the service definition is the same used by S4. In that case, S2 should cover the entire query in order to have to be part of the rewriting solution.

Input and output decorations Apparently, Umberto has a version of the algorithm that can parse the query using the input (?) and output (!) decorations, but they are not took into account inside the algorithm (while creating PCDs and dependencies, and rewriting).

Rewriting problem In the current implementation, there are some rewriting results that call the same concrete service more than one time, such as:

Q(...) :- S2(disease,patients,_), S2(_,patients,dnas), S4(...), S5(...)

Such kind of execution does not exists while we are calling services. We can not call just one single functionality in a service that we agreed to use. For example, consider that a service **S** performs verification and withdraw in a bank account. I can not call S just to verify the account or just to withdraw money. When we call this service both operations are executed, and this can originate big problems.

The solution here is to change the rewriting in order to merge service invocations in rewriting solutions that replicates services when the mappings between variables are possible. If the mappings are not compatible, the rewriting should be discarded.

Inclusion of SLAs I presented our idea regarding the SLA, and in which points I am planning to change the algorithm, for example: checking SLA constraints while creating the PCDs to avoid the burden of creating and processing the rewriting for a service that already has a SLA constraint that violates the user requirement, and while combining PCDs checking if the requirements are not violated.

I also propose to use the original version of the algorithm to avoid the bugs we found while testing. However, if Umberto finds the other implementation that considers input and output parameters, we can move to it.