Meeting September 11th

Describing the input, processing and output date in the algorithm

The **input data** for the algorithm is a query (Q) and a list of concrete services (C1, C2, ...). The query and concrete services are defined as follows:

$$Q(\bar{t}) := A_1(\bar{t}), A_2(\bar{t}), .., A_n(\bar{t})$$
$$C(\bar{t}) := A_1(\bar{t}), A_2(\bar{t}), .., A_n(\bar{t})$$

The query and concrete services are defined as a set of abstract services $(A_1, A_2, ..., A_n)$. $\bar{t}$ is q set of input or output variables. The first **processing step** of the algorithm creates the data structures that represent the query and concrete services. The query and concrete service data structure will be formed by its *name*, a set of head variables (V_h) and a set of abstract services (A_s).

$$Q = \langle name, V_h, A_s \rangle$$
$$C = \langle name, V_h, A_s \rangle$$

For each abstract service A in the set A_s, a data structure containing its *name* and a set of variables (V) is build. Notice that the set of variables (V) can contain head variables or local variables.

$$A = \langle name, V \rangle$$

Once these structures were created, the **second step in the algorithm is to form the PCDs**. PCDs are created only for those concrete services which have abstract services that can answer that query or part of it. PCDs are defined as follows:

$$\langle S, h, \varphi, G, Def, has\_opt \rangle$$

S is a concrete service. h are mapping between terms in the head of the service definition to terms in the right side of the service definition. φ are mapping between terms from the abstract composition to terms in the concrete service definition. ***G*** is a set of abstract services names and quality constraints covered by ***S***. ***Def*** is a set conditions that are not covered by ***S*** alone. ***has_opt*** is a boolean flag to indicate that some abstract service in the definition of ***S*** has been used in ***G*** and has an optional parameter.

After that, all the **PCDs formed in the previous step are combined**. Once we have all the possible combinations of PCDs, **each combination is verified** to confirm if the combination is a valid rewriting of the former query or not. A valid rewriting is a list of PCDs that answers the complete query without redundancies, and without more than one mapping to the same variable with distinct values.

Describing an example

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

Concrete services:
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)

The **rewriting results** were:
1) Q(disease,patients,dnas,birth,gen,addr,wheatherstatistic) :-
S2(disease,patients,_),S2(_,patients,dnas),S4(patients,birth,gen,addr),S5(addr,wheatherstatistic)
2) Q(disease,patients,dnas,birth,gen,addr,wheatherstatistic) :-
S2(disease,patients,_),S3(patients,dnas),S4(patients,birth,gen,addr),S5(addr,wheatherstatistic)

In the result (1) the service S2 is called two times, and in (2) S1 and S3 are called together. "**Why do we have these answers if we have the dependency between A1 and A2 in S2?**"

The rewriting answers are correct. Note that we have all variables in the head of the service definition, because of that we do not have dependency between A1 and A2, and two different PCDs are created for this service.

2. New Example

Considering Umberto's comments I tried to produce a new example expressing the dependencies between abstract services.

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

Concrete services:
S1(disease, patientsssn) :- A1(disease, patientsssn)
S2(disease, 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)

Note that I removed *patientsssn* from the head of the service S2 and I supposed that now this variable creates a dependency between the abstract services A1 and A2. However the rewriting answer was:

Q(disease,patients,dnas,birth,gen,addr,wheatherstatistic) :-
S1(disease,patients),S3(patients,dnas),S4(patients,birth,gen,addr),S5(addr,wheatherstatistic)

And the PCDs created were:

S1 [A1(disease,patients)]
S3 [A2(patients,dnas)]
S4 [A3(patients,birth,gen,addr)]
S5 [A4(addr,wheatherstatistic)]

No answer using S2 was created and also no PCD for it. I also tried to remove S1, but I got an error running the algorithm.

3. Example with SLA

Query:
Q(disease, patients, dnas, birth, gen, addr, wheatherstatistic) :- A1(disease, patients), A2(patients,dnas), A3(patients, birth, gen, addr), A4(addr, wheatherstatistic){ cost $\leq$ 1$, location = close, response time < 2s, data provenance = any}

I added in the end of the query, between braces, the user's requirements that should be after matched in the SLAs. I put them between braces to differentiate from the constraints that could also be part of the query. I am supposing that I have access to the services SLAs. Considering the concrete services below, the PCDs created are:

Concrete services:
S1(disease, patientsssn) :- A1(disease, patientsssn)
S2(disease, patientsssn) :- A1(disease, patientsssn)
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)

PCD 1:

S S1

SLA { cost = 0.2, location = close, response time < 2s, data provenance = fresh }

h { disease $\longrightarrow$ disease, patientsssn $\longrightarrow$ patientsssn }

φ { disease $\longrightarrow$ disease, patientsssn $\longrightarrow$ patients }

G A1(disease, patientsssn)

Def $\emptyset$

has_opt false

PCD 2:

S S2

SLA { cost = 0.4, location = close, response time < 3s, data provenance = fresh }

h { disease $\longrightarrow$ disease, patientsssn $\longrightarrow$ patientsssn }

φ { disease $\longrightarrow$ disease, patientsssn $\longrightarrow$ patients }

G A1(disease, patientsssn)

Def $\emptyset$

has_opt false

PCD 3:

S S3

SLA { cost = 0.1, location = close, response time < 1s, data provenance = fresh }

h { patientsssn $\longrightarrow$ patientsssn, dna$\longrightarrow$ dna }

φ { patientsssn $\longrightarrow$ patients, dna $\longrightarrow$ dnas }

G A2(patientsssn, dna)

Def $\emptyset$

has_opt false

PCD 4:

S S4

SLA { cost = 0.1, location = close, response time < 1s, data provenance = fresh }

h { patientsssn $\longrightarrow$ patientsssn, dateofbirth $\longrightarrow$ dateofbirth, gender $\longrightarrow$ gender, address $\longrightarrow$ address }

φ { patientsssn $\longrightarrow$ patients, dateofbirth $\longrightarrow$ birth, gender $\longrightarrow$ gen, address $\longrightarrow$ addr }

G A3(patientsssn, dateofbirth, gender, address)

Def $\emptyset$

has_opt false

PCD 5:

S S5

SLA { cost = 0.1, location = close, response time < 1s, data provenance = fresh }

h { address $\longrightarrow$ address, wheatherstaticalinformation $\longrightarrow$ wheatherstaticalinformation }

φ { address $\longrightarrow$ addr, wheatherstaticalinformation $\longrightarrow$ wheatherstatistic }

G A4(address, wheatherstaticalinformation)

Def $\emptyset$

has_opt false