Interactive Mapping Specification with Exemplar Tuples

Angela Bonifati — angela.bonifati@univ-lyon1.fr
Ugo Comignani — ugo.comignani@univ-lyon1.fr
Emmanuel Coquery — emmanuel.coquery@univ-lyon1.fr
Romuald Thion — romuald.thion@univ-lyon1.fr

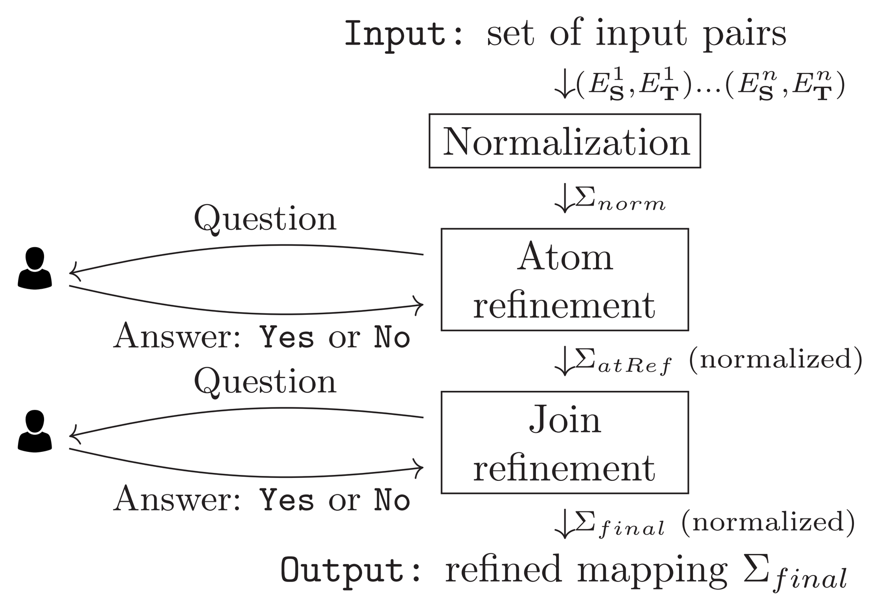
CNRS LIRIS and Université Claude Bernard Lyon 1

1. Motivations and goals

We propose an Interactive Mapping Specification process targeting non-expert users that:

- bootstraps with a set of exemplar tuples $(E_{\mathbf{S}}, E_{\mathbf{T}})$, corresponding to a limited number of tuples provided as input;
- challenges the non-expert users with simple boolean questions, which are affordable for such users;
- is guaranteed to always produce a GLAV mapping $\mathcal{M}'$ that generalizes a mapping $\mathcal{M}$ in the user's mind, which is unknown beforehand.

2. Proposed framework



Prior to the interactive process, a normalization step is undertaken in order to simplify the mappings and make them self-explanatory for non-expert users. Then, an atom refinement step and a join refinement step are respectively devoted to get the normalized mapping as closer as possible to the mapping that the user has in mind. This interactive process leverages simple boolean questions on even smaller refinement-driven tuples, derived from the initial exemplar tuples provided as input. We guarantee that (i) the obtained refined mappings are in normal form and that (ii) they are more general than the canonical mapping.

3. Specifying mappings using user exemplar tuples

User exemplar tuples:

$E_{\mathbf{S}}$:

Company

IdCompany	*Name*	*Town*
'C1'	'AA'	'Paris'
'C2'	'Ev'	'Lyon'

Flight

Departure	*Arrival*	*IdCompany*
'Lyon'	'Paris'	'C1'
'Paris'	'Lyon'	'C2'

Travel Agency

IdAgency	*Name*	*Town*
'A1'	'TC'	'L.A.'

$E_{\mathbf{T}}$:

Firm

Id	*Name*	*Town*
'Id1'	'AA'	'Paris'
'Id2'	'Ev'	'Lyon'
'Id3'	'TC'	'L.A.'

Departure

Town	*IdFirm*
'Lyon'	'Id1'
'Paris'	'Id2'

Arrival

Town	*IdFirm*
'Paris'	'Id1'
'Lyon'	'Id2'

Canonical mapping:

$$\begin{aligned}\mathbf{m}: &\ Company(\mathrm{c1}, \mathrm{aa}, \mathrm{paris}) \wedge Company(\mathrm{c2}, \mathrm{ev}, \mathrm{lyon}) \wedge TravelAgency(\mathrm{a1}, \mathrm{tc}, \mathrm{la})\\ &\wedge Flight(\mathrm{lyon}, \mathrm{paris}, \mathrm{c1}) \wedge Flight(\mathrm{paris}, \mathrm{lyon}, \mathrm{c2})\\ &\rightarrow \exists \mathrm{id1}, \mathrm{id2}, \mathrm{id3}, Firm(\mathrm{id1}, \mathrm{aa}, \mathrm{paris}) \wedge Departure(\mathrm{lyon}, \mathrm{id1}) \wedge Arrival(\mathrm{paris}, \mathrm{id1})\\ &\wedge Firm(\mathrm{id2}, \mathrm{ev}, \mathrm{lyon}) \wedge Departure(\mathrm{paris}, \mathrm{id2}) \wedge Arrival(\mathrm{lyon}, \mathrm{id2}) \wedge Firm(\mathrm{id3}, \mathrm{tc}, \mathrm{la})\end{aligned}$$

Normalized mapping:

Before initiating the refinement steps, the canonical mapping is normalized [1] in order to separate unrelated connected components in the right-hand sides. This leads to the following split-reduced mapping:

$$\begin{aligned}\mathbf{m_a}: &\ Company(\mathrm{c1}, \mathrm{aa}, \mathrm{paris}) \wedge Company(\mathrm{c2}, \mathrm{ev}, \mathrm{lyon}) \wedge TravelAgency(\mathrm{a1}, \mathrm{tc}, \mathrm{la})\\ &\wedge Flight(\mathrm{lyon}, \mathrm{paris}, \mathrm{c1}) \wedge Flight(\mathrm{paris}, \mathrm{lyon}, \mathrm{c2})\\ &\rightarrow \exists \mathrm{id1}, Firm(\mathrm{id1}, \mathrm{aa}, \mathrm{paris}) \wedge Departure(\mathrm{lyon}, \mathrm{id1}) \wedge Arrival(\mathrm{paris}, \mathrm{id1})\end{aligned}$$

$$\begin{aligned}\mathbf{m_b}: &\ Company(\mathrm{c1}, \mathrm{aa}, \mathrm{paris}) \wedge Company(\mathrm{c2}, \mathrm{ev}, \mathrm{lyon}) \wedge TravelAgency(\mathrm{a1}, \mathrm{tc}, \mathrm{la})\\ &\wedge Flight(\mathrm{lyon}, \mathrm{paris}, \mathrm{c1}) \wedge Flight(\mathrm{paris}, \mathrm{lyon}, \mathrm{c2})\\ &\rightarrow \exists \mathrm{id2}, Firm(\mathrm{id2}, \mathrm{ev}, \mathrm{lyon}) \wedge Departure(\mathrm{paris}, \mathrm{id2}) \wedge Arrival(\mathrm{lyon}, \mathrm{id2})\end{aligned}$$

$$\begin{aligned}\mathbf{m_c}: &\ Company(\mathrm{c1}, \mathrm{aa}, \mathrm{paris}) \wedge Company(\mathrm{c2}, \mathrm{ev}, \mathrm{lyon}) \wedge TravelAgency(\mathrm{a1}, \mathrm{tc}, \mathrm{la})\\ &\wedge Flight(\mathrm{lyon}, \mathrm{paris}, \mathrm{c1}) \wedge Flight(\mathrm{paris}, \mathrm{lyon}, \mathrm{c2})\\ &\rightarrow \exists \mathrm{id3}, Firm(\mathrm{id3}, \mathrm{tc}, \mathrm{la})\end{aligned}$$

During normalization, split reduction is followed by the deletion of logically equivalent tgds. In the above example, this leads to the removal of the tgd $\mathbf{m_a}$ (or, equivalently, $\mathbf{m_b}$) in order to obtain Σ_{norm}.

Final mapping after interactive specification:

$$\begin{aligned}\mathbf{m_1}: &\ Company(\mathrm{c1}, \mathrm{aa}, \mathrm{paris_1}) \wedge Flight(\mathrm{lyon}, \mathrm{paris_2}, \mathrm{c1})\\ &\rightarrow \exists \mathrm{id1}, Firm(\mathrm{id1}, \mathrm{aa}, \mathrm{paris_1}) \wedge Departure(\mathrm{lyon}, \mathrm{id1}) \wedge Arrival(\mathrm{paris_2}, \mathrm{id1})\end{aligned}$$

$$\mathbf{m_2}: TravelAgency(\mathrm{a1}, \mathrm{tc}, \mathrm{la}) \rightarrow \exists \mathrm{id3}, Firm(\mathrm{id3}, \mathrm{tc}, \mathrm{la})$$

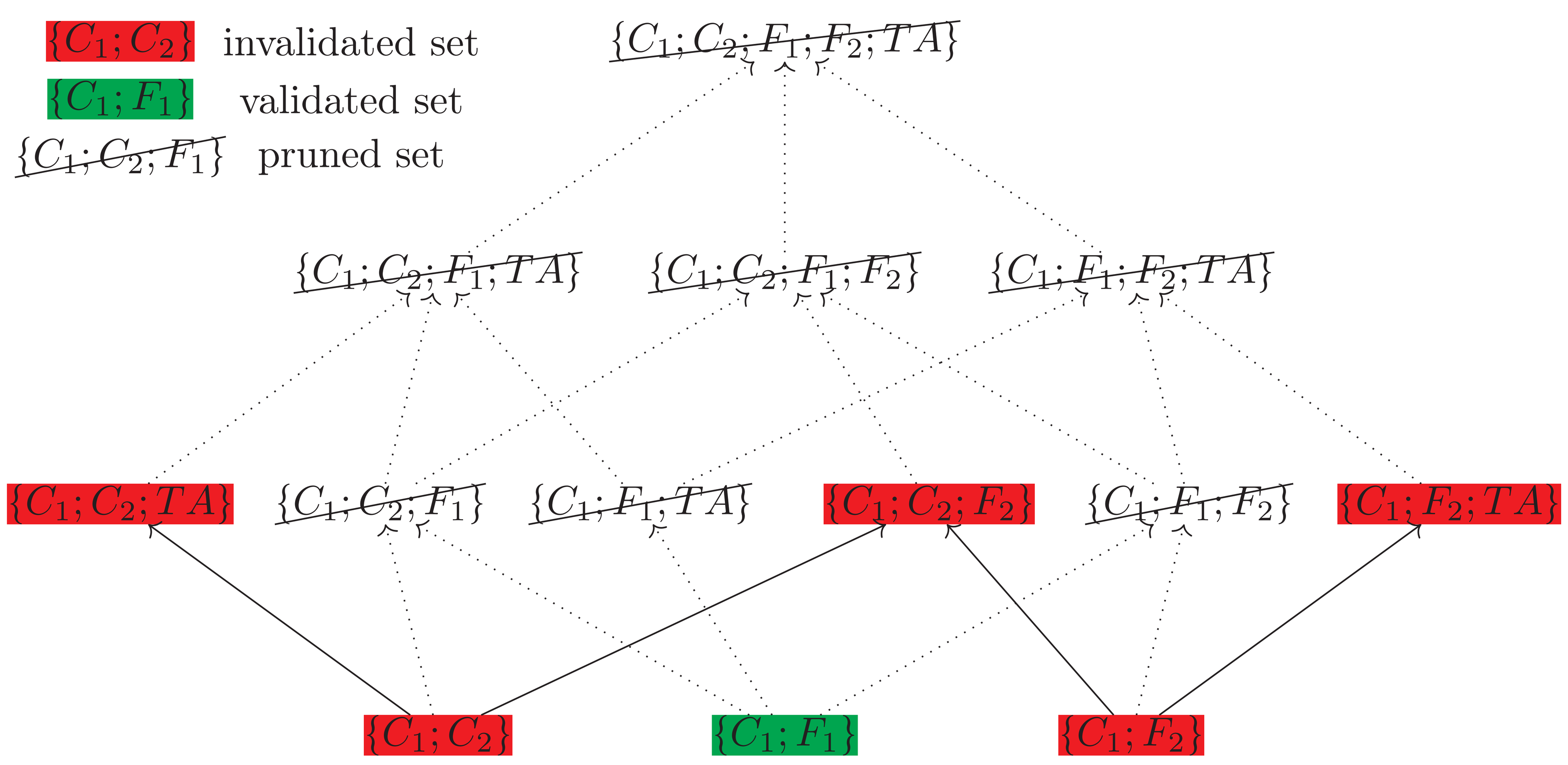
4. Atom refinement

Principle:
Remove superfluous atoms in the left-hand sides of normalized tgds.

Exploration space:
Given a tgd, the possible sets of left-hand side atoms can be represented by a semi-lattice where the lower levels are the smallest valid sets of left-hand side atoms.

User feedback:
During semi-lattice exploration, the user is asked about the target tuples generated using only the subset of source tuples corresponding to the atoms in one node at a time.

Example with a bottom-up breadth-first approach:



As the sole set validated is $\{C_1; F_1\}$, the following tgd is generated:

$$Company(\mathrm{c1}, \mathrm{aa}, \mathrm{paris}) \wedge Flight(\mathrm{lyon}, \mathrm{paris}, \mathrm{c1}) \rightarrow \exists \mathrm{id1}, Firm(\mathrm{id1}, \mathrm{aa}, \mathrm{paris}) \wedge Departure(\mathrm{lyon}, \mathrm{id1}) \wedge Arrival(\mathrm{paris}, \mathrm{id1})$$

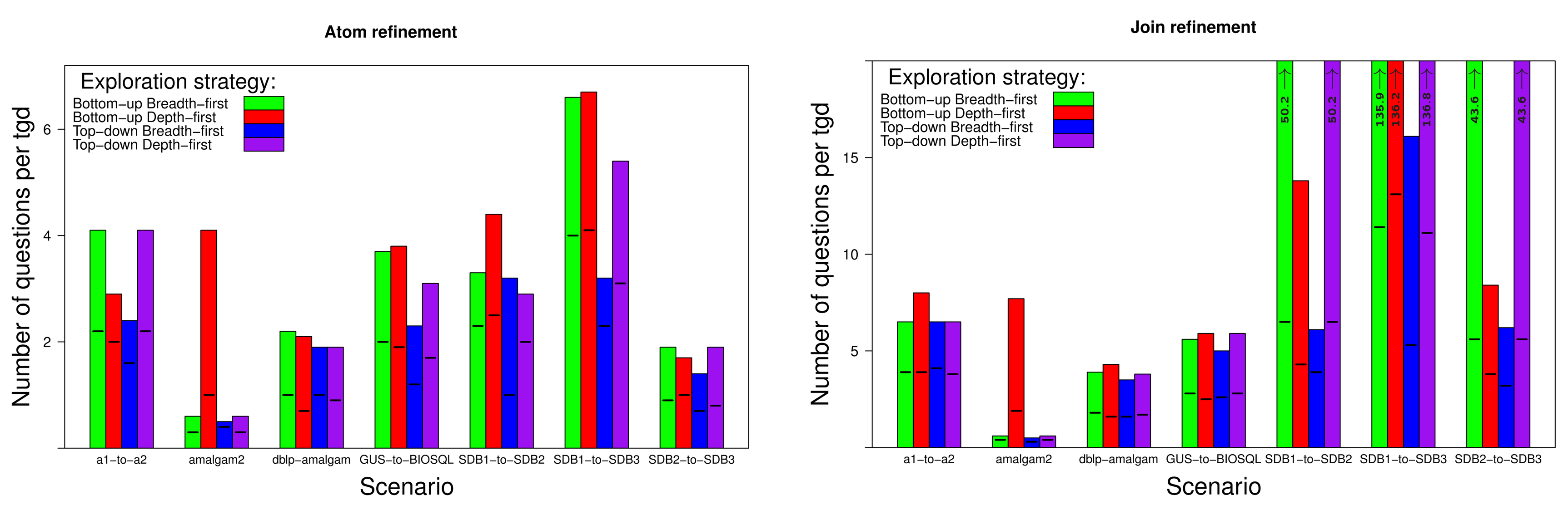
5. Join refinement

Principle:
For each tgd generated by atom refinement, identify redundant joins entailed by multiple occurrences of a given variable.

Exploration space:
As with atom refinement, we can build a semi-lattice of partitions representing possible joins between variable occurrences.

User feedback:
Similarly to atom refinement, the user is asked about the validity of small sets of tuples.

6. Experimental study



- We have used seven real data integration scenarios of the iBench benchmark [2].
- We have simulated the user's ambiguities by adding up to ten superfluous atoms or redundant joins to the initial exemplar tuples, derived from the above scenarios.
- We have tested four exploration strategies : bottom-up and top-down, each one with depth-first and breadth-first variations.

References

[1] G. Gottlob, R. Pichler, and V. Savenkov: *Normalization and optimization of schema mappings.* VLDB J., 20(2):277–302, 2011.

[2] P.C. Arocena, B. Glavic, R. Ciucanu, and R.J. Miller: *The ibench integration metadata generator.* Proceedings of VLDB, 9(3):108–119, 2015.