Propagating Dependencies under Schema Mappings – A Graph-based Approach

Qing Wang

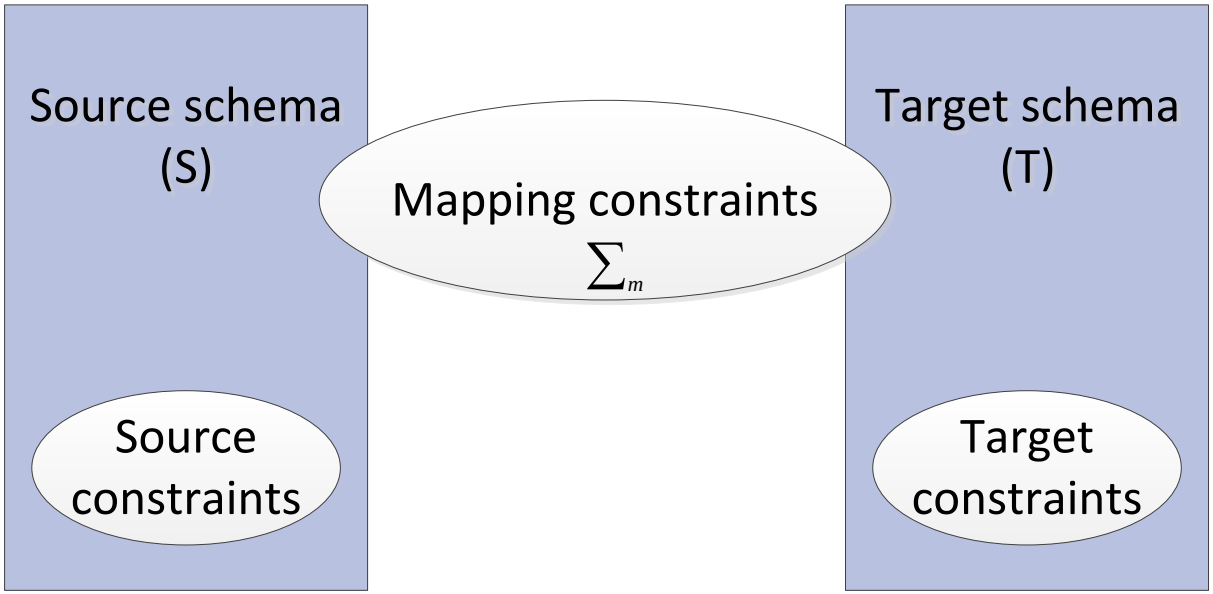
Research School of Computer Science
Australian National University
Canberra ACT 0200, Australia
qing.wang@anu.edu.au

Xi Wen

Department of Computer Science
Nanchang University
Nanchang City, China
wenxi@ncu.edu.cn

Schema Mappings – Definition

- Schema mapping plays an important role in many database-related transformation tasks, such as data exchange, data integration and data migration.
- A *schema mapping* is a triple $M = (S, T, \Sigma_m)$ consisting of
  - a *source schema* S,
  - a *target schema* T, and
  - a set Σ_m of *mapping constraints* over S and T.



Schema Mappings – Background

- However, designing a schema mapping is not an easy task. Generally, two lines of research exist:

  (1) *Generate schema mappings from a visual specification* provided by users (traditionally studied);

  However. a visual specification is often ambiguous.

  (2) *Derive schema mappings based on data examples* (attracted more interest in recent years).

  However, data examples may not be available, or could be biased.

- Existing approaches either require a manual process of tuning schema mappings or demand more data examples for improving accuracy.

- Even though, the design quality of schema mappings is still often not satisfactory.

Schema Mappings – Questions

- Some common questions in practice:

(1) Can we ensure certain properties of a source database to be preserved in the desired target database through the design of a schema mapping?

(2) Can we determine whether or not a target constraint can be enforced on a target database before the target database is transformed from a given source database?

(3) If some target constraints cannot be enforced, can we efficiently identify which data in the source database need to be cleansed, or determine whether the schema mapping and target constraints need to be re-designed?

(4) ...

Our Research

- **General goal** of our research:

  To develop methods/tools that help check whether a schema mapping is designed meaningfully and effectively in advance, before an implementation takes place.

- **Specific task** in this paper:

  Given a schema mapping, a question is:

  > How can we discover logical consequences among its source, target and mapping constraints?

Schema Mappings – Motivating Example

- Source schema S
  - $\textsc{RentClient}(id, name, address)$, $\textsc{RentProperty}(no, address, rent)$, and $\textsc{AllClient}(name, dob, gender, cid)$
  - $\Sigma_s = \emptyset$
- Target schema T
  - $\textsc{Property}(no, address)$, $\textsc{Client}(id, name)$, and $\textsc{Rental}(id, no, rent)$
  - $\Sigma_t = \{\textsc{Rental} : no \rightarrow rent, \textsc{Rental}[id] \subseteq \textsc{Client}[id], \textsc{Rental}[no] \subseteq \textsc{Property}[no]\}$
- Mapping constraints Σ_m

(C1) $\forall x, y, z.(\textsc{RentClient}(x, y, z) \Rightarrow \textsc{Client}(x, y))$;
(C2) $\forall x, y, z, x', z'.(\textsc{RentClient}(x, y, z) \wedge \textsc{RentProperty}(x', z, z') \Rightarrow \textsc{Rental}(x, x', z'))$;
(C3) $\forall x, y, z.(\textsc{RentProperty}(x, y, z) \Rightarrow \exists x'.\textsc{Property}(x, y) \wedge \textsc{Rental}(x', x, z))$.
(C4) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists z'.\textsc{RentProperty}(y, z', z))$;
(C5) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists y', z'.\textsc{RentClient}(x, y', z'))$;
(C6) $\textsc{Rental}(x, y, z) \Rightarrow \exists x', y', z'.\textsc{AllClient}(x', y', z', x)$.

Example – Schema Mappings

- Source schema S
  - $\textsc{RentClient}(id, name, address)$, $\textsc{RentProperty}(no, address, rent)$, and $\textsc{AllClient}(name, dob, gender, cid)$
  - $\Sigma_s = \emptyset$
- Target schema T
  - $\textsc{Property}(no, address)$, $\textsc{Client}(id, name)$, and $\textsc{Rental}(id, no, rent)$
  - $\Sigma_t = \{\textsc{Rental} : no \rightarrow rent, \textsc{Rental}[id] \subseteq \textsc{Client}[id], \textsc{Rental}[no] \subseteq \textsc{Property}[no]\}$
- Mapping constraints Σ_m

(C1) $\forall x, y, z.(\textsc{RentClient}(x, y, z) \Rightarrow \textsc{Client}(x, y))$;
(C2) $\forall x, y, z, x', z'.(\textsc{RentClient}(x, y, z) \wedge \textsc{RentProperty}(x', z, z') \Rightarrow \textsc{Rental}(x, x', z'))$;
(C3) $\forall x, y, z.(\textsc{RentProperty}(x, y, z) \Rightarrow \exists x'.\textsc{Property}(x, y) \wedge \textsc{Rental}(x', x, z))$.
(C4) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists z'.\textsc{RentProperty}(y, z', z))$;
(C5) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists y', z'.\textsc{RentClient}(x, y', z'))$;
(C6) $\textsc{Rental}(x, y, z) \Rightarrow \exists x', y', z'.\textsc{AllClient}(x', y', z', x)$.

How much semantics specified by Σ_t and Σ_m is captured by Σ_s?

Example – Schema Mappings

- Source schema S
  - $\textsc{RentClient}(id, name, address)$, $\textsc{RentProperty}(no, address, rent)$, and $\textsc{AllClient}(name, dob, gender, cid)$
  - $\Sigma_s = \emptyset$
- Target schema T
  - $\textsc{Property}(no, address)$, $\textsc{Client}(id, name)$, and $\textsc{Rental}(id, no, rent)$
  - $\Sigma_t = \{\textsc{Rental} : no \rightarrow rent, \textsc{Rental}[id] \subseteq \textsc{Client}[id], \textsc{Rental}[no] \subseteq \textsc{Property}[no]\}$
- Mapping constraints Σ_m

(C1) $\forall x, y, z.(\textsc{RentClient}(x, y, z) \Rightarrow \textsc{Client}(x, y))$;
(C2) $\forall x, y, z, x', z'.(\textsc{RentClient}(x, y, z) \wedge \textsc{RentProperty}(x', z, z') \Rightarrow \textsc{Rental}(x, x', z'))$;
(C3) $\forall x, y, z.(\textsc{RentProperty}(x, y, z) \Rightarrow \exists x'.\textsc{Property}(x, y) \wedge \textsc{Rental}(x', x, z))$.
(C4) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists z'.\textsc{RentProperty}(y, z', z))$;
(C5) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists y', z'.\textsc{RentClient}(x, y', z'))$;
(C6) $\textsc{Rental}(x, y, z) \Rightarrow \exists x', y', z'.\textsc{AllClient}(x', y', z', x)$.

How much semantics specified by Σ_t and Σ_m is captured by Σ_s?

– $\Sigma_s^\dagger = \{\textsc{RentProperty} : no \rightarrow rent\}$

Example – Schema Mappings

- By comparing Σ_s and $\Sigma_s^\dagger$,
  - $\Sigma_s = \emptyset$
  - $\Sigma_s^\dagger = \{\textsc{RentProperty} : no \rightarrow rent\}$

  we know that every source instance that violates $RentProperty : no \rightarrow rent$ must violate Σ_t after being transformed into a target instance.

RentClient		
id	*name*	*address*
c1	Tim Jenkin	5 Jicket St, Dunedin
c2	Linda Lee	36 Novar St, Dunedin
c3	Mike Carl	2 Manor St, Dunedin

RentProperty		
no	*address*	*rent*
1	5 Jicket St, Dunedin	350
1	5 Jicket St, Dunedin	500
2	2 Manor St, Dunedin	450

- It means that either RentClient needs to be cleansed, or target constraints need to be reconsidered.

Example – Schema Mappings

- Source schema S
  - $\textsc{RentClient}(id, name, address)$, $\textsc{RentProperty}(no, address, rent)$, and $\textsc{AllClient}(name, dob, gender, cid)$
  - $\Sigma_s = \emptyset$
- Target schema T
  - $\textsc{Property}(no, address)$, $\textsc{Client}(id, name)$, and $\textsc{Rental}(id, no, rent)$
  - $\Sigma_t = \{\textsc{Rental} : no \rightarrow rent, \textsc{Rental}[id] \subseteq \textsc{Client}[id], \textsc{Rental}[no] \subseteq \textsc{Property}[no]\}$
- Mapping constraints Σ_m

(C1) $\forall x, y, z.(\textsc{RentClient}(x, y, z) \Rightarrow \textsc{Client}(x, y))$;
(C2) $\forall x, y, z, x', z'.(\textsc{RentClient}(x, y, z) \wedge \textsc{RentProperty}(x', z, z') \Rightarrow \textsc{Rental}(x, x', z'))$;
(C3) $\forall x, y, z.(\textsc{RentProperty}(x, y, z) \Rightarrow \exists x'.\textsc{Property}(x, y) \wedge \textsc{Rental}(x', x, z))$.
(C4) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists z'.\textsc{RentProperty}(y, z', z))$;
(C5) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists y', z'.\textsc{RentClient}(x, y', z'))$;
(C6) $\textsc{Rental}(x, y, z) \Rightarrow \exists x', y', z'.\textsc{AllClient}(x', y', z', x)$.

How much semantics specified by Σ_s and Σ_m is preserved by Σ_t?

Example – Schema Mappings

- Source schema S
  - $\textsc{RentClient}(id, name, address)$, $\textsc{RentProperty}(no, address, rent)$, and $\textsc{AllClient}(name, dob, gender, cid)$
  - $\Sigma_s = \emptyset$
- Target schema T
  - $\textsc{Property}(no, address)$, $\textsc{Client}(id, name)$, and $\textsc{Rental}(id, no, rent)$
  - $\Sigma_t = \{\textsc{Rental} : no \rightarrow rent, \textsc{Rental}[id] \subseteq \textsc{Client}[id], \textsc{Rental}[no] \subseteq \textsc{Property}[no]\}$
- Mapping constraints Σ_m

(C1) $\forall x, y, z.(\textsc{RentClient}(x, y, z) \Rightarrow \textsc{Client}(x, y))$;
(C2) $\forall x, y, z, x', z'.(\textsc{RentClient}(x, y, z) \wedge \textsc{RentProperty}(x', z, z') \Rightarrow \textsc{Rental}(x, x', z'))$;
(C3) $\forall x, y, z.(\textsc{RentProperty}(x, y, z) \Rightarrow \exists x'.\textsc{Property}(x, y) \wedge \textsc{Rental}(x', x, z))$.
(C4) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists z'.\textsc{RentProperty}(y, z', z))$;
(C5) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists y', z'.\textsc{RentClient}(x, y', z'))$;
(C6) $\textsc{Rental}(x, y, z) \Rightarrow \exists x', y', z'.\textsc{AllClient}(x', y', z', x)$.

How much semantics specified by Σ_s and Σ_m is preserved by Σ_t?

– $\Sigma_t^{\dagger} = \{\textsc{Rental}[id] \subseteq \textsc{Client}[id], \textsc{Rental}[no] \subseteq \textsc{Property}[no]\}$

Example – Schema Mappings

- By comparing Σ_t and $\Sigma_t^\dagger$,
  - $\Sigma_t = \{\textsc{Rental} : no \to rent, \textsc{Rental}[id] \subseteq \textsc{Client}[id], \textsc{Rental}[no] \subseteq \textsc{Property}[no]\}$
  - $\Sigma_t^\dagger = \{\textsc{Rental}[id] \subseteq \textsc{Client}[id], \textsc{Rental}[no] \subseteq \textsc{Property}[no]\}$

  we know that $\textsc{Rental}[id] \subseteq \textsc{Client}[id]$ and $\textsc{Rental}[no] \subseteq \textsc{Property}[no]$ can hold on every target instance under this schema mapping.

Client	
id	*name*
c1	Tim Jenkin
c2	Linda Lee
c3	Mike Carl

Property	
no	*address*
1	5 Jicket St, Dunedin
2	2 Manor St, Dunedin

Rental		
id	*no*	*rent*
c1	1	500
c3	2	450

- Hence we only need to check whether or not $\textsc{Rental} : no \to rent$ holds on the target instance.

Specifying Mapping Constraints

- A **tuple-generating dependency (tgd)** is an expression of the form

$$\forall \bar{x}, \bar{y}.(\varphi(\bar{x}, \bar{y}) \Rightarrow \exists \bar{z}.\psi(\bar{x}, \bar{z})),$$

  where φ and ψ are conjunctions of atoms, but only relational atoms occur in ψ.

  - φ is called the **premise** of σ, and
  - ψ is called the **conclusion** of σ.

- Mapping constraints Σ_m are **bipartite tgds** over S and T, i.e., tgds with φ and ψ over S and T, respectively, or vice versa.

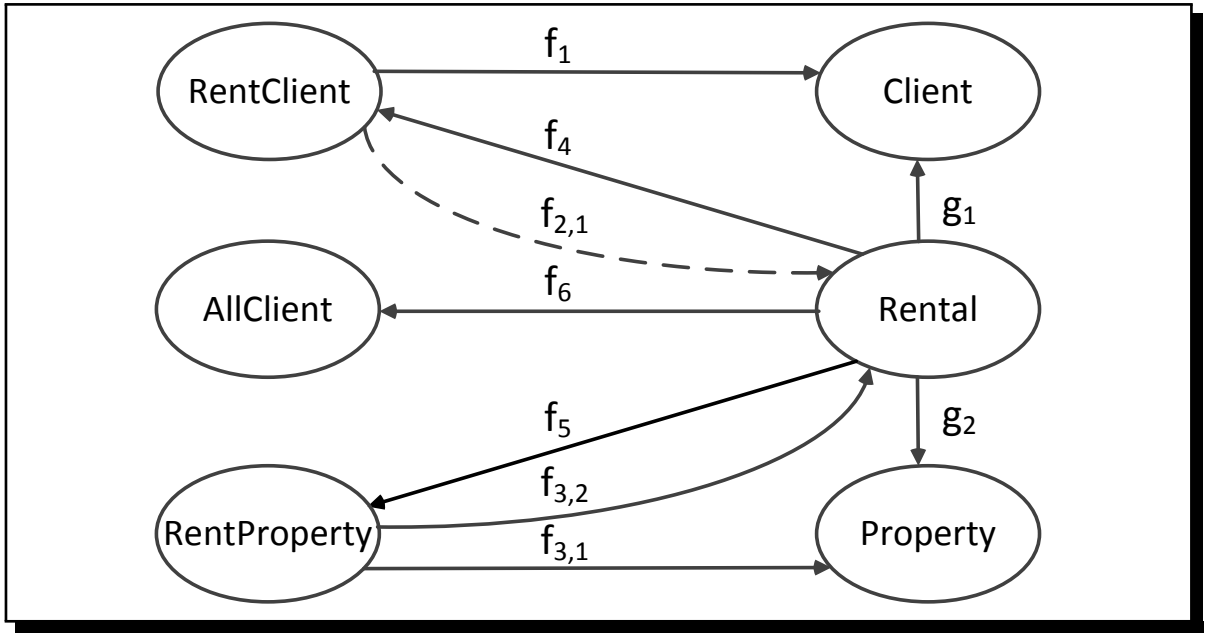
- The co-existence of target-to-source and source-to-target tgds in Σ_m enables us to precisely build a high-level specification for the relationship between source and target instances.

Propagation Graphs

- A **propagation graph** $G = (V, E)$ consists of
  - A set V of vertices, where each $v \in V$ is a relation schema name R, and
  - A set E of edges, where each $R \hookrightarrow_f R' \in E$ is labelled by a function
$$f : attr(R) \mapsto attr(R').$$
- There are two types of edges: **approximate** and **exact**.
- A **propagation path** with a label f is a sequence of edges
$$\langle R_1 \hookrightarrow_{f_1} R_2, \ldots, R_{n-1} \hookrightarrow_{f_{n-1}} R_n \rangle$$
such that $f = f_{n-1} \circ \cdots \circ f_1$ is a function that maps a *non-empty subset of attributes* of R_1 into attributes of R_n.

Propagation Graphs – Example

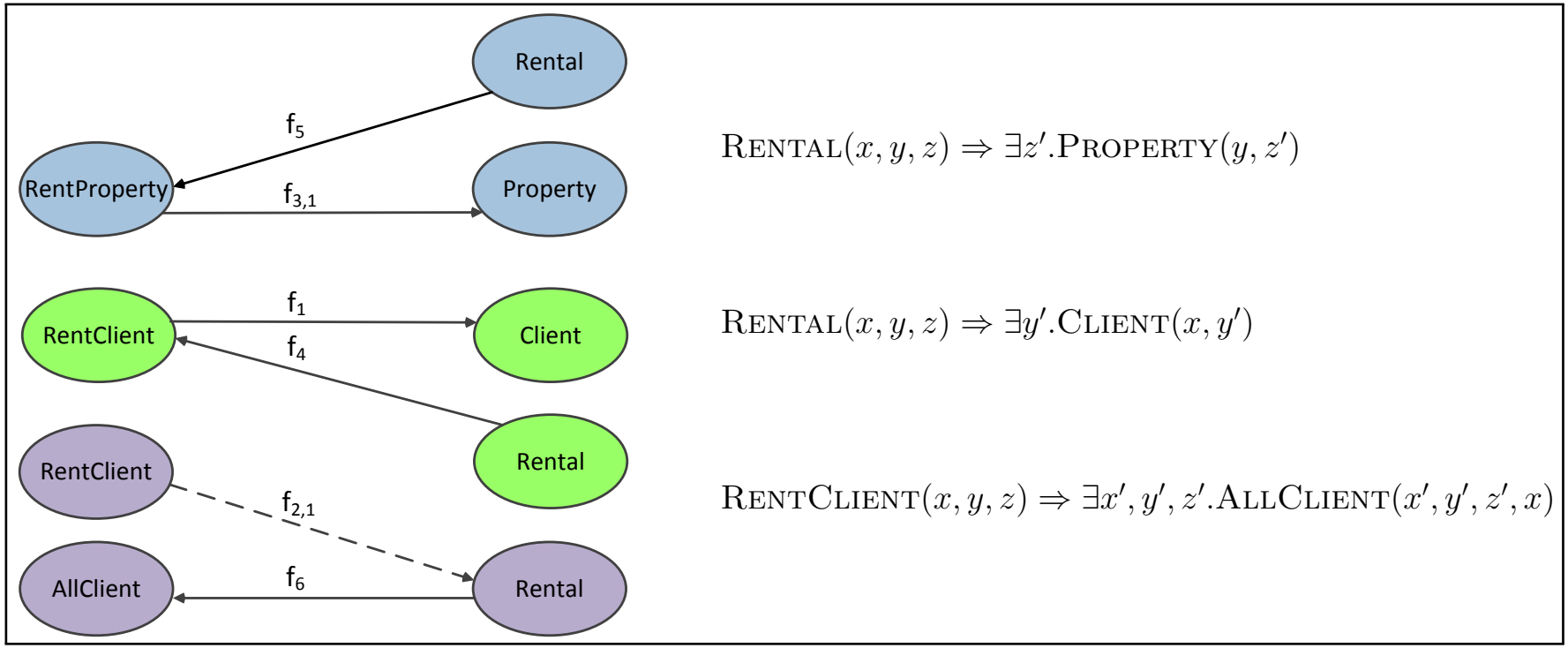
(C1) $\forall x, y, z.(\textsc{RentClient}(x, y, z) \Rightarrow \textsc{Client}(x, y))$;
(C2) $\forall x, y, z, x', z'.(\textsc{RentClient}(x, y, z) \wedge \textsc{RentProperty}(x', z, z') \Rightarrow \textsc{Rental}(x, x', z'))$;
(C3) $\forall x, y, z.(\textsc{RentProperty}(x, y, z) \Rightarrow \exists x'.\textsc{Property}(x, y) \wedge \textsc{Rental}(x', x, z))$.
(C4) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists z'.\textsc{RentProperty}(y, z', z))$;
(C5) $\forall x, y, z.(\textsc{Rental}(x, y, z) \Rightarrow \exists y', z'.\textsc{RentClient}(x, y', z'))$;
(C6) $\textsc{Rental}(x, y, z) \Rightarrow \exists x', y', z'.\textsc{AllClient}(x', y', z', x)$.
Σ_t: $\textsc{Rental}[id] \subseteq \textsc{Client}[id]$ and $\textsc{Rental}[no] \subseteq \textsc{Property}[no]$.



Labels of Edges
$f_1 : 1 \mapsto 1, 2 \mapsto 2.$
$f_{2,1} : 1 \mapsto 1.$
$f_{2,2} : 1 \mapsto 2, 3 \mapsto 3.$
$f_{3,1} : 1 \mapsto 1, 2 \mapsto 2.$
$f_{3,2} : 1 \mapsto 2, 3 \mapsto 3.$
$f_4 : 1 \mapsto 1.$
$f_5 : 2 \mapsto 1, 3 \mapsto 3.$
$g_1 : 1 \mapsto 1.$
$g_2 : 2 \mapsto 1.$

Propagating INDs

- Each propagation path from R_1 to R_2 over T corresponds to an IND over T.
- Depending on the types of the edges, an IND is either exact or approximate.
- Approximate INDs are prompted to users for fine-tuning the semantics of a schema mapping.

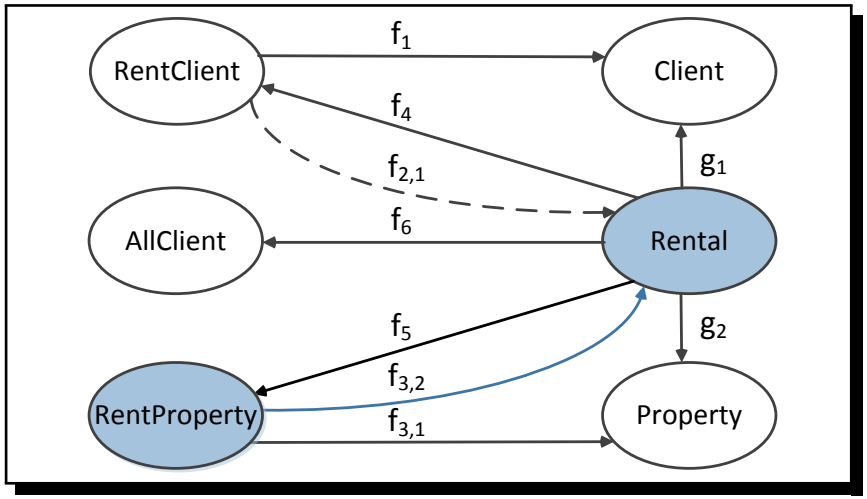


Propagating FDs

- Let $R : X \rightarrow Y$ be a FD over T, and $R' \in S$
  - **Push backward**

    If there is a propagation path $\langle R', \ldots, R \rangle$ labelled by f, where $X' = \{f^{-1}(x) | x \in X\}$ and $Y' = \{f^{-1}(y) | y \in Y\}$, then $R' : X' \rightarrow Y'$ over S can be propagated.
  - **Push forward**

    If there is a propagation path $\langle R, \ldots, R' \rangle$ labelled by f, where $X' = \{f(x) | x \in X\}$ and $Y' = \{f(y) | y \in Y\}$, then $R' : X' \rightarrow Y'$ over S can be propagated *with the condition on a subset of tuples in a relation of* R'.



$$\textsc{Rental} : no \rightarrow rent \in \Sigma_t$$

$$\Downarrow$$

$$\Sigma_s^{\dagger} = \{\textsc{RentProperty} : no \rightarrow rent\}$$

Experiments

- We have developed a bipartite schema mapping (BSM) tool to help schema mapping designers in several aspects:

(1) Visualizing propagation graphs for any given schema mappings;
(2) Assessing the design quality of schema mappings;
(3) Facilitating the data cleaning tasks of source instances.

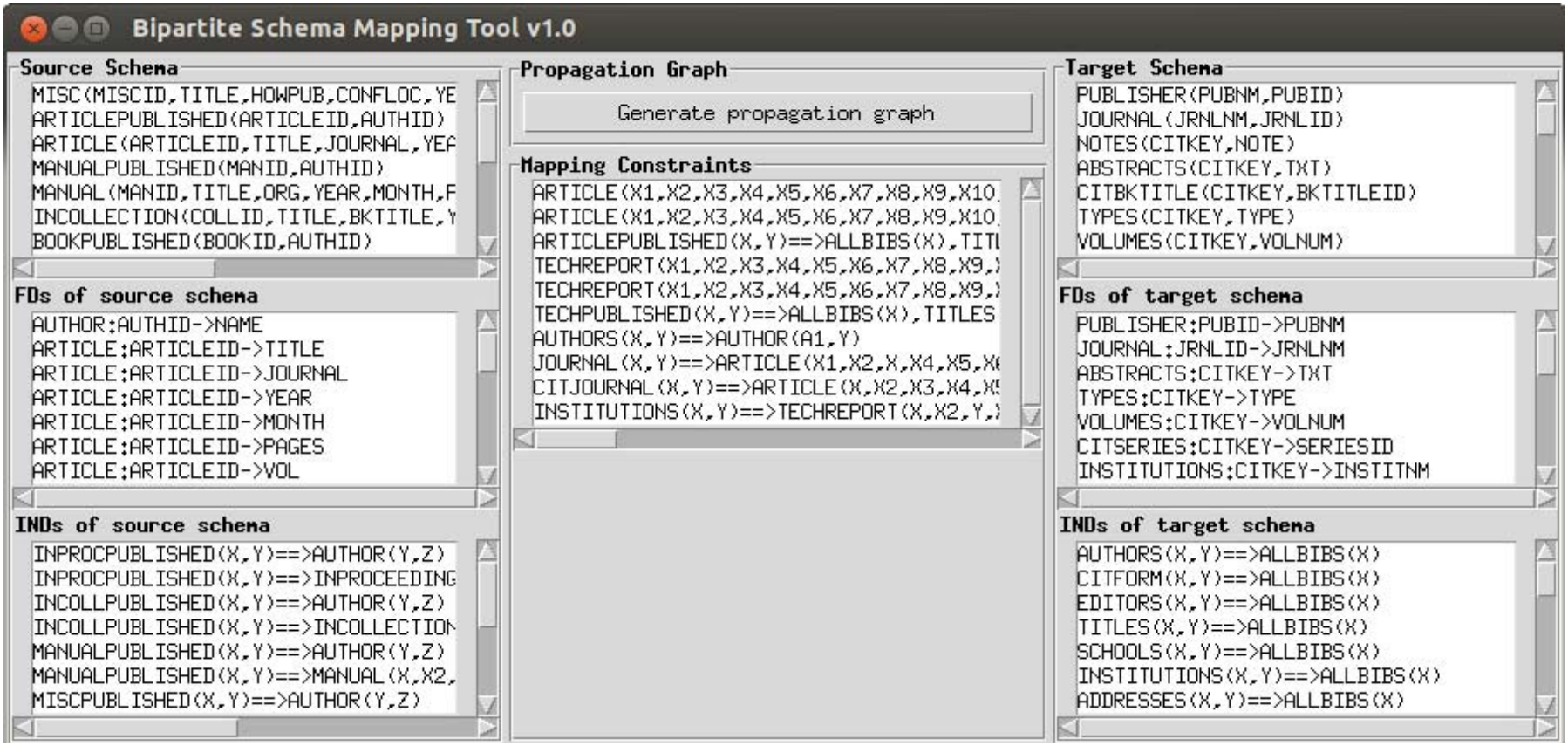
- We have conducted our experiments over two schema mapping data sets:
  - RentalApp: as previously described;
  - Amalgam: taken from the Clio Project[1].

	Source schema (S_1)	Target schema (S_2)
No of relations	15	27
No of INDs	14	26
No of FDs	23	21
No of MCs	10	

[1] http://dblab.cs.toronto.edu/ miller/amalgam/

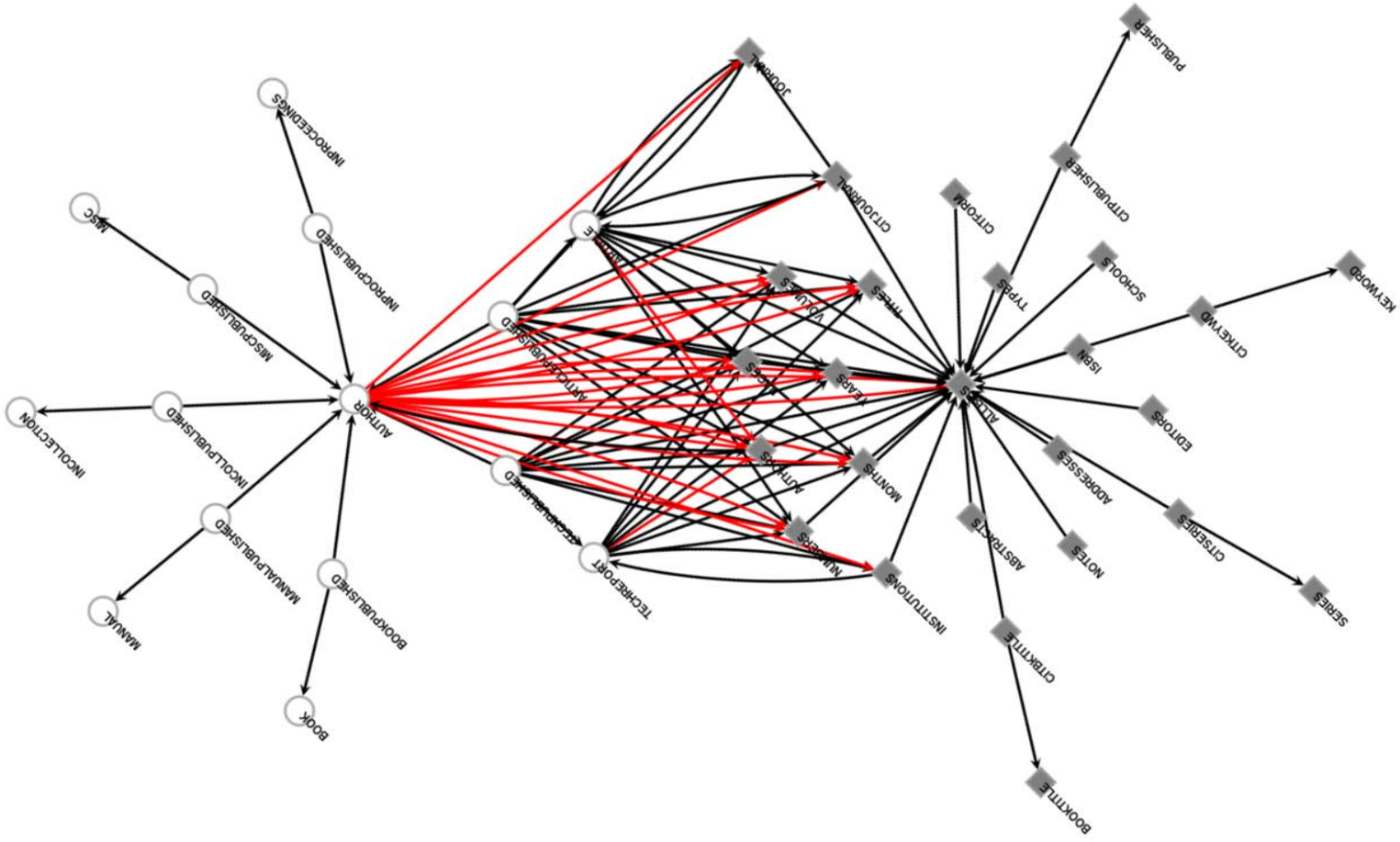
Experiments – User Interface

- A schema mapping over AMALGAM: S_1 is the source schema and S_2 is the target schema.



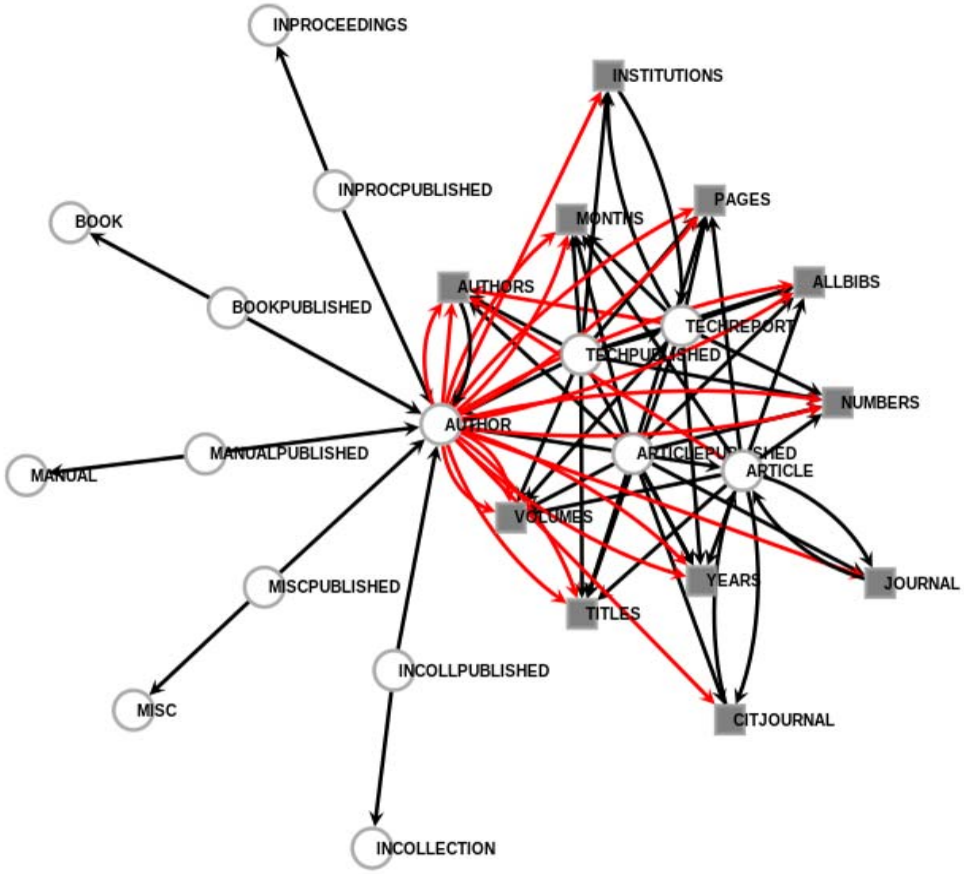
Experiments – Propagation Graphs

- The propagation graph for visualizing the schema mapping over AMALGAM.



Experiments – Propagation Graphs

- The propagation graph for deriving target constraints over S_2 in AMALGAM.



Conclusions

- We have developed a graphical model to represent the inter-relationships among the attributes of relation schemas.
- On that basis, we have studied the dependency propagation problem in the context of schema mappings.
- Mapping constraints of a schema mapping are permitted to be bipartite TGDs, which enables us to precisely specify the relationship between source and target databases.
- Our work can lead to a conceptual analysis tool that exploits the semantics of a schema mapping through propagation paths in propagation graphs.
- In doing so, the design quality of schema mappings can be assessed before actually implementing them.

Thanks and Questions