Foundations of Data Exchange and Metadata Management

Marcelo Arenas

Ron Fagin Special Event - SIGMOD/PODS 2016

The need for a formal definition

- We had a paper with Ron in PODS 2004
- Back then I was a Ph.D. student, and asked Ron whether I could do an internship in IBM Almaden
- He was very positive about the idea, but there were some funding issues

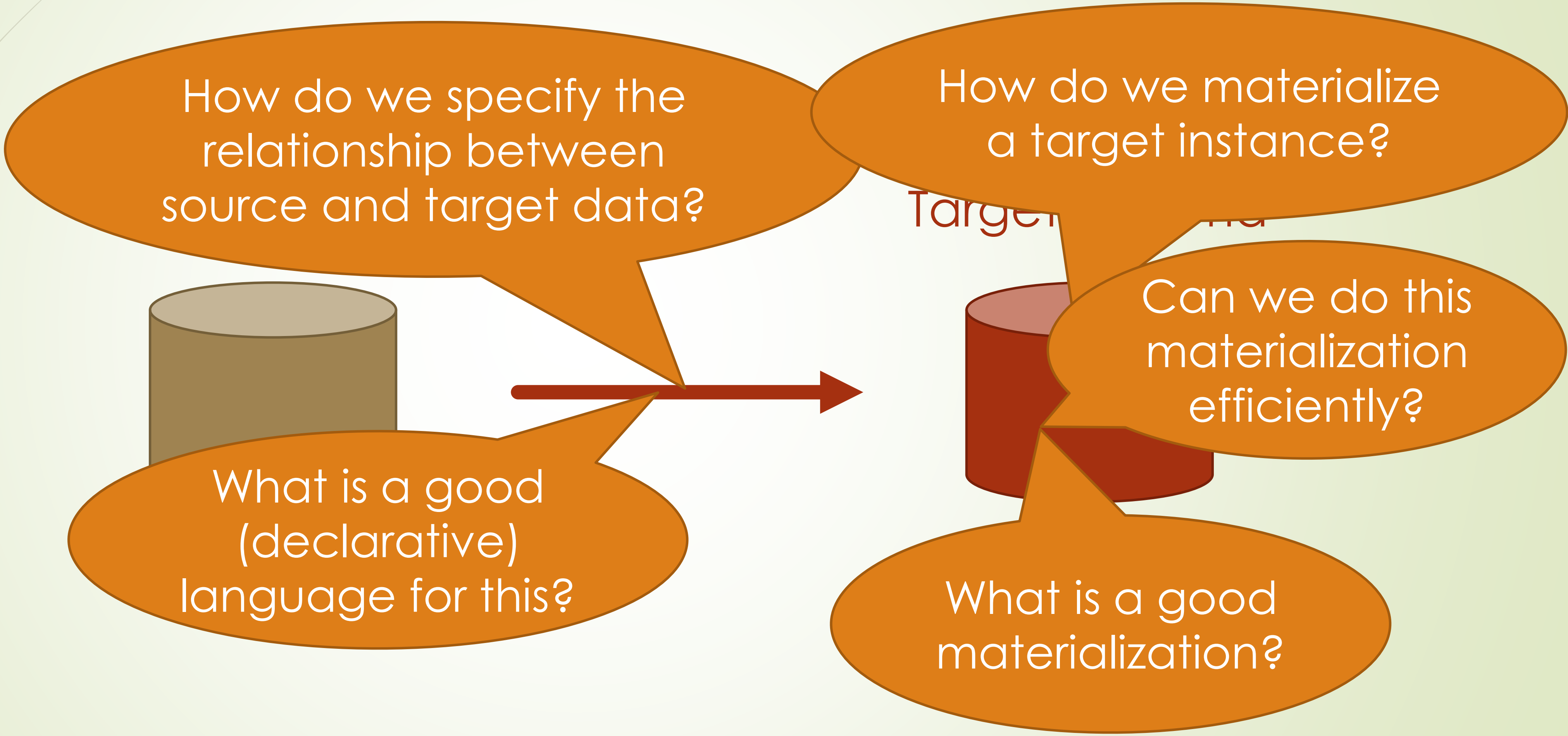
The need for a formal definition

- The solution: applied as Hispanic
- The issue: How Hispanic I am?
  - Is there a precise definition of the notion of being Hispanic?
- The final solution: The IBM Ph.D. fellowship

The old data exchange problem

- The first systems for restructuring and translating data were built several decades ago
  - EXPRESS (1977): A data extraction, processing, and restructuring system
- This problem is particularly relevant today
  - There is a need for a simple, yet general, solution to it

The data exchange problem



The data exchange problem

Worker(name)

name
Ron
John

Worker(x) → Emp(x)

What do we allow in this rule language?

What is a good materialization?

Emp(name)

name
Ron
John
Paul
Ringo

The data exchange problem

Worker(name, salary)

name	salary
Ron	100K
John	90K
Paul	70K

$\xrightarrow{?}$

Emp(name, dept)

name	dept
Ron	?
John	
Paul	

A solution to the problem

Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, Lucian Popa. *Data Exchange: Semantics and Query Answering*. ICDT 2003

This article proposed a simple, elegant and general solution

- It has a big impact (1000+ citations in Google scholar)

A mapping language

Given: source schema **S** and a target schema **T** with no relation names in common

- A source-to-target tuple-generating dependency (st-tgd) is a formula of the form:

$$\forall \mathbf{x} \forall \mathbf{y}\ \varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z})$$

where $\varphi(\mathbf{x},\mathbf{y})$ and $\psi(\mathbf{x},\mathbf{z})$ are conjunctions of atoms over **S** and **T**, respectively

A mapping language

- A mapping from **S** to **T** is specified by a set $\sum_{ST}$ of st-tgds

$$
\begin{aligned}
\mathbf{S} &= \{\ \mathrm{Worker}(\cdot)\ \} \\
\mathbf{T} &= \{\ \mathrm{Emp}(\cdot)\ \} \\
\sum_{ST} &= \{\forall x\ \mathrm{Worker}(x) \rightarrow \mathrm{Emp}(x)\ \}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{S} &= \{\ \mathrm{Worker}(\cdot,\cdot)\ \} \\
\mathbf{T} &= \{\ \mathrm{Emp}(\cdot,\cdot)\ \} \\
\sum_{ST} &= \{\forall x \forall y\ \mathrm{Worker}(x,y) \rightarrow \exists z\ \mathrm{Emp}(x, z)\ \}
\end{aligned}
$$

A definition of a mapping

- A mapping **M** is just a tuple (**S**, **T**, $\sum_{ST}$)
  - An instance of **S** is called a source instance, while an instance of **T** is called a target instance
  - $\sum_{ST}$ specifies the relationship between source and target data
- What is the semantics of a mapping?
  - When is a target instance considered to be a valid materialization for a source instance under **M**?

A semantics for mappings

A target instance J is a solution for a source instance I under a mapping $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{ST})$ if:

(I,J) satisfies Σ_{ST} under the usual semantics of first-order logic

A semantics for mappings

Assume we have a mapping specified by Worker(x) → Emp(x) and instances:

I = { Worker(Ron), Worker(John), Worker(Paul) }

J_1 = { Emp(Ron), Emp(John), Emp(Paul) }

J_2 = { Emp(Ron), Emp(John) }

J_1 is a solution for I: $(I, J_1) \models \forall x\, \text{Worker}(x) \rightarrow \text{Emp}(x)$

J_2 is not a solution for I: and $(I, J_2) \not\models \forall x\, \text{Worker}(x) \rightarrow \text{Emp}(x)$

A semantics for mappings

Consider a mapping specified by Worker(x,y) $\rightarrow \exists z$ Emp(x, z)

Worker	
Ron	100K
John	90K
Paul	70K

Emp	
Ron	D1
John	D2
Paul	D3
Ringo	D1

What is a good solution?

The classical notions of null value and homomorphism are used to solve this issue

- Target instances are allowed to contain constants and nulls
- Homomorphisms are used to define a notion of most general solution

Solutions with null values

Consider a mapping specified by Worker(x,y) $\rightarrow$ $\exists z$ Emp(x, z)

Worker	
Ron	100K
John	90K
Paul	70K

Emp	
Ron	$\perp_1$
John	$\perp_2$
Paul	$\perp_3$

The notion of homomorphism

Consider two instances J_1 and J_2 of the same schema

Consider a function h from the set of constants and nulls to the set of constants and nulls

h is a homomorphism from J_1 to J_2 if

- $h(c) = c$ for every constant c
- if $R(a_1, \ldots, a_n)$ is a fact in J_1, then $R(h(a_1), \ldots, h(a_n))$ is a fact in J_2

The notion of homomorphism

h(Ron)	=	Ron	$h(\perp_1)$	=	D1
h(John)	=	John	$h(\perp_2)$	=	$\perp_4$
h(Paul)	=	Paul	$h(\perp_3)$	=	D1

Emp	
Ron	$\perp_1$
John	$\perp_2$
Paul	$\perp_3$

Emp	
Ron	D1
John	$\perp_4$
Ringo	D2
Paul	D1

The notion of universal solution

Given a mapping **M** and a source instance I

A solution J for I under **M** is a universal solution if:

for every solution K for I under **M**,

there exists a homomorphism from J to K

The notion of universal solution

Consider a mapping specified by Worker(x,y) → ∃z Emp(x, z)

Worker	
Ron	100K
John	90K
Paul	70K

Emp	
Ron	$\perp_1$
John	$\perp_2$
Paul	$\perp_3$

Emp	
Ron	D1
John	D2
Paul	D3

Emp	
Ron	$\perp$
John	$\perp$
Paul	$\perp$

Emp	
Ron	$\perp_1$
John	$\perp_2$
Paul	$\perp_3$
Ringo	$\perp_4$

Computing universal solutions efficiently

The last ingredient: a polynomial-time algorithm for computing universal solutions

The well-known notion of chase can be used to compute universal solutions

- Existential variables in st-tgds are replaced by fresh nulls

Computing universal solutions efficiently

Consider a mapping specified by Worker(x,y) $\rightarrow \exists z$ Emp(x, z)

Worker	
Ron	100K
John	90K
Paul	70K

Emp	
Ron	$\perp_1$
John	$\perp_2$
Paul	$\perp_3$

Lessons learned

- Framework has to be simple
  - Syntax and semantics of mappings are simple
- Framework has to be general enough to be of practical interest
  - Based on realistic assumptions

Lessons learned

- Main notions have to be well formalized
  - It is important to have a precise definition of what a valid translation of a source instance is
- Do not reinvent the wheel: use well-known and widely-studied concepts, bring tools from other areas
  - Syntax and semantics of mappings are based on first-order logic
  - Universal solutions are defined in terms of homomorphisms

And this is not all …

The fundament … t queries was also considered

How do we answer a target query?

Worker	
Ron	100K
John	90K
Paul	70K

Emp	
Ron	$\perp$
John	$\perp$
Paul	$\perp$

A semantics for target queries

- How to evaluate a query Q over an instance I is well understood
  - Q(I) is used to denote the answer to Q over I
- This notion is used to define the answer to a target query with respect to a source instance given a mapping

A semantics for target queries

Given a mapping **M**, a source instance I and a query Q over the target schema

The set of certain answers of Q with respect to I given **M** is defined as:

$$\text{certain}_{\mathbf{M}}(Q,I) = \bigcap_{J \text{ is a solution for } I \text{ under } \mathbf{M}} Q(J)$$

A semantics for target queries

Consider a mapping specified by Worker(x,y) → ∃z Emp(x, z)
and the target query Q(u) = ∃v Emp(u,v)

Answer to Q =
{ Ron, John, Paul, Ringo }

Answer to Q =
{ Ron, John, Paul }

Worker	
Ron	100K
John	90K
Paul	70K

Em	
Ringo	

certain$_M$(Q,I) =
{ Ron, Jonh, Paul }

Answer to Q =
{ Ron, John, Paul }

Computing certain answers efficiently

Given a mapping **M,** a source instance I and a universal solution J for I under **M**

For every union of conjunctive queries Q:

$$\mathrm{certain}_{\mathbf{M}}(Q,I) = \{\ \mathbf{a} \mid \mathbf{a} \in Q(J) \text{ and } \mathbf{a} \text{ only mentions constants}\ \}$$

Why this approach was so influential?

The simple and well-defined framework for data exchange open many directions for further research

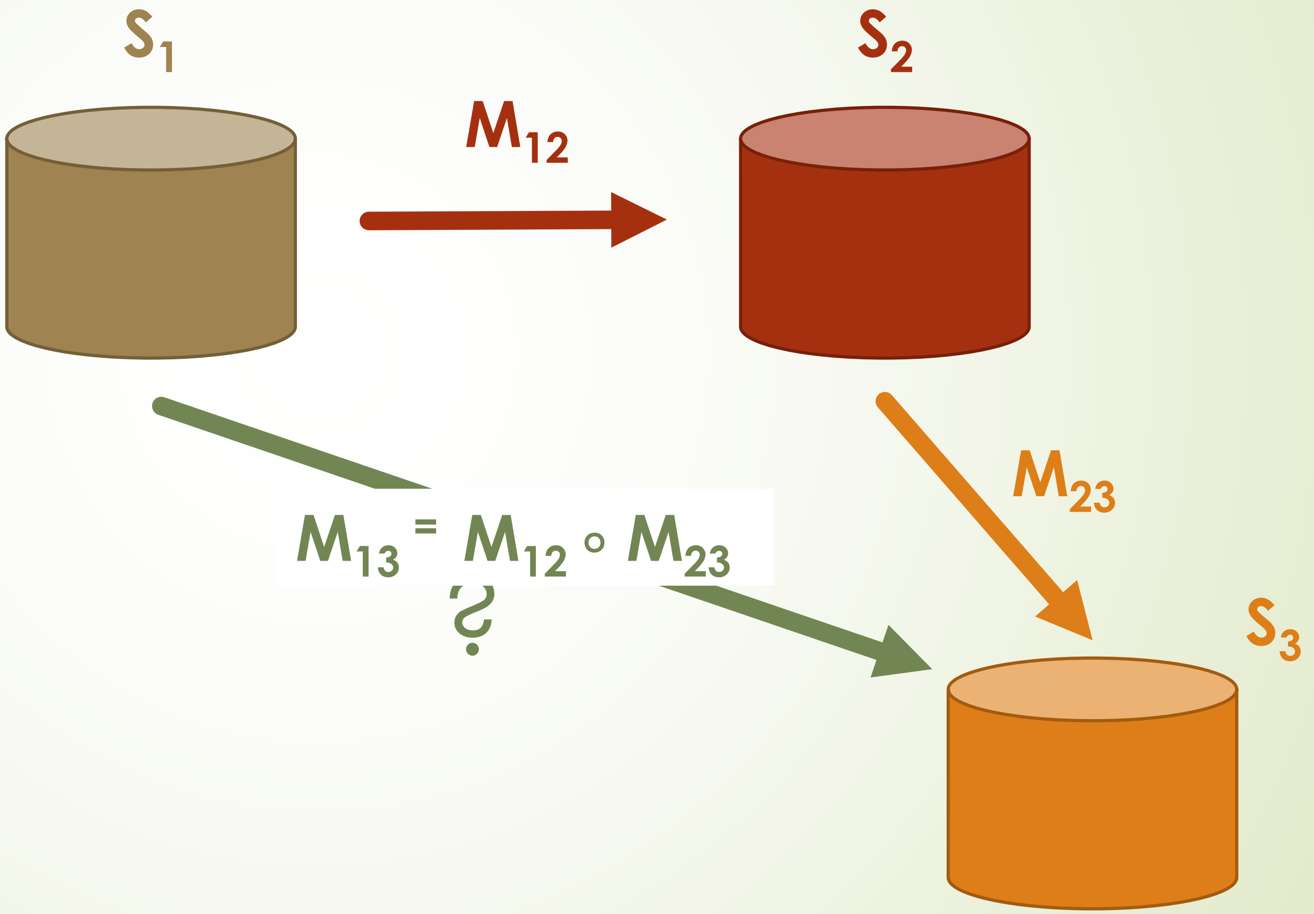
- Design of efficient algorithms for computing universal solutions (minimal ones)
- Design of efficient query answering algorithms for target positive queries
- Identification of more expressive query languages (inequalities, negation and aggregation)
- Use of source and target integrity constraints
- Optimization of mappings

Why this approach was so influential?

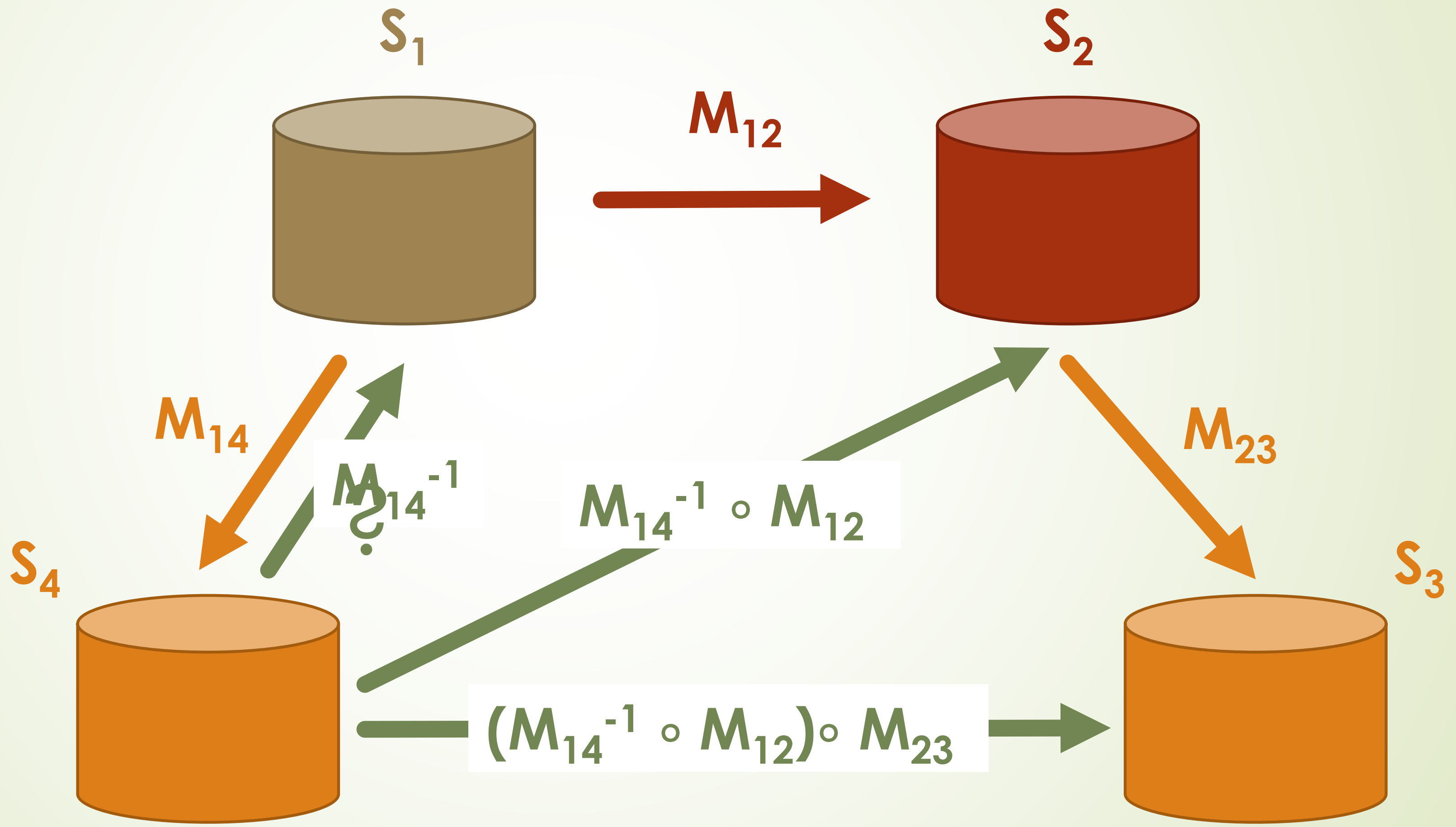
The simple and well-defined framework for data exchange open many directions for further research:

- Use of more expressive mapping languages
- Study of different notions of solutions and semantics for query answering (OWA versus CWA)
- Development of data exchange settings in other data models: XML, graph databases, probabilistic databases
- **Development of mapping operators**

Metadata management



Metadata management



The need for mapping operators

Philip A. Bernstein. *Applying Model Management to Classical Meta Data Problems*. CIDR 2003

- Once a formal definition of mapping is given, these operators can be formally defined and studied

The composition operator

- $\mathbf{S_1}$, $\mathbf{S_2}$ and $\mathbf{S_3}$ denote pairwise disjoint schemas
  - Instances of $\mathbf{S_k}$ are denoted as I_k ($k = 1, 2, 3$)
- $\mathbf{M_{12}}$ and $\mathbf{M_{23}}$ denote mappings from $\mathbf{S_1}$ to $\mathbf{S_2}$ and $\mathbf{S_2}$ to $\mathbf{S_3}$, respectively

The composition operator

The composition of $\mathbf{M}_{12}$ with $\mathbf{M}_{23}$ is defined as a mapping $\mathbf{M}_{12} \circ \mathbf{M}_{23}$ such that:

I_3 is a solution for I_1 under $\mathbf{M}_{12} \circ \mathbf{M}_{23}$

if and only if

there exists I_2 such that

I_2 is a solution for I_1 under $\mathbf{M}_{12}$ and

I_3 is a solution for I_2 under $\mathbf{M}_{23}$

A mapping language for the composition operator

- What is the right language to express the composition operator?

- Are st-tgds closed under composition?
  - If $\mathbf{M}_{12}$ and $\mathbf{M}_{23}$ are specified by sets of st-tgds, can also $\mathbf{M}_{12} \circ \mathbf{M}_{23}$ be specified by a set of st-tgds?

A mapping language for the composition operator

Having a formal definition of mappings these questions can be answered

- st-tgds are not closed under composition
  - There exist mappings $\mathbf{M}_{12}$ and $\mathbf{M}_{23}$ specified by sets of st-tgds, such that $\mathbf{M}_{12} \circ \mathbf{M}_{23}$ cannot be specified by a set of st-tgds

- There exists a mapping language that is appropriate for composition

The power of composition

Consider a mapping $\mathbf{M}_{12}$ specified by the following st-tgds:

$$\mathrm{Node}(x) \rightarrow \exists u\ \mathrm{Paint}(x, u)$$
$$\mathrm{Edge}(x, y) \rightarrow \mathrm{Arc}(x, y)$$

and a mapping $\mathbf{M}_{23}$ specified by the following st-tgds:

$$\mathrm{Paint}(x,u) \rightarrow \mathrm{Color}(u)$$
$$\mathrm{Arc}(x,y) \wedge \mathrm{Paint}(x,u) \wedge \mathrm{Paint}(y,u) \rightarrow \mathrm{Error}(x) \wedge \mathrm{Error}(y)$$

Adding second-order quantification

Unless P = NP, the previous mapping $\mathbf{M}_{12} \circ \mathbf{M}_{23}$ cannot be defined in first-order logic

What does it need to be added to st-tgds to capture the composition of two mappings?

- Fagin's theorem gives us a good idea of what needs to be added: NP = ∃SO

The language of second-order tgds

A simple extension of st-tgds gives rise to a mapping language that is appropriate to define the composition:

$$\exists f \left(\begin{array}{l} \forall x\ [\ \mathrm{Node}(x) \rightarrow \mathrm{Color}(f(x))\ ]\ \wedge \\ \forall x \forall x\ [\ \mathrm{Edge}(x,y) \wedge f(x)=f(y) \rightarrow \mathrm{Error}(x) \wedge \mathrm{Error}(y)\ ] \end{array} \right)$$

These dependencies are called second-order st-tgds (SO tgds)

The language of second-order tgds

It is the right language for specifying the composition of mappings defined by st-tgds

- The composition of a sequence of mappings specified by sets of st-tgds can be specified by an SO tgd
- SO tgds are closed under composition
- For every SO tgd φ, there exists a sequence of mappings specified by sets of st-tgds such that its composition is specified by φ

The language of second-order tgds

Besides, it has (almost) the same good properties as st-tgds for data exchange

- Universal solutions are defined in the same way
- There is a polynomial-time algorithm (based on the chase) for computing universal solutions
- Certain answers to union of conjunctive queries can be computed by using universal solutions

Lessons learned

- Having a simple and formal definition of mappings is a key ingredient to study mapping operators
- Use well-known and widely-studied concepts
  - A simple form of second-order quantification gives rise to a simple yet powerful mapping language that is appropriate to define the composition operator

Thanks Ron for many well-defined and inspiring concepts!