Master 2 TIW
UE TIW2 Intéroperabilité

Univ. Lyon 1, A.Y. 2021-2022

Angela Bonifati

Course Syllabus, Introduction to Data Integration and Data Exchange

Course Syllabus

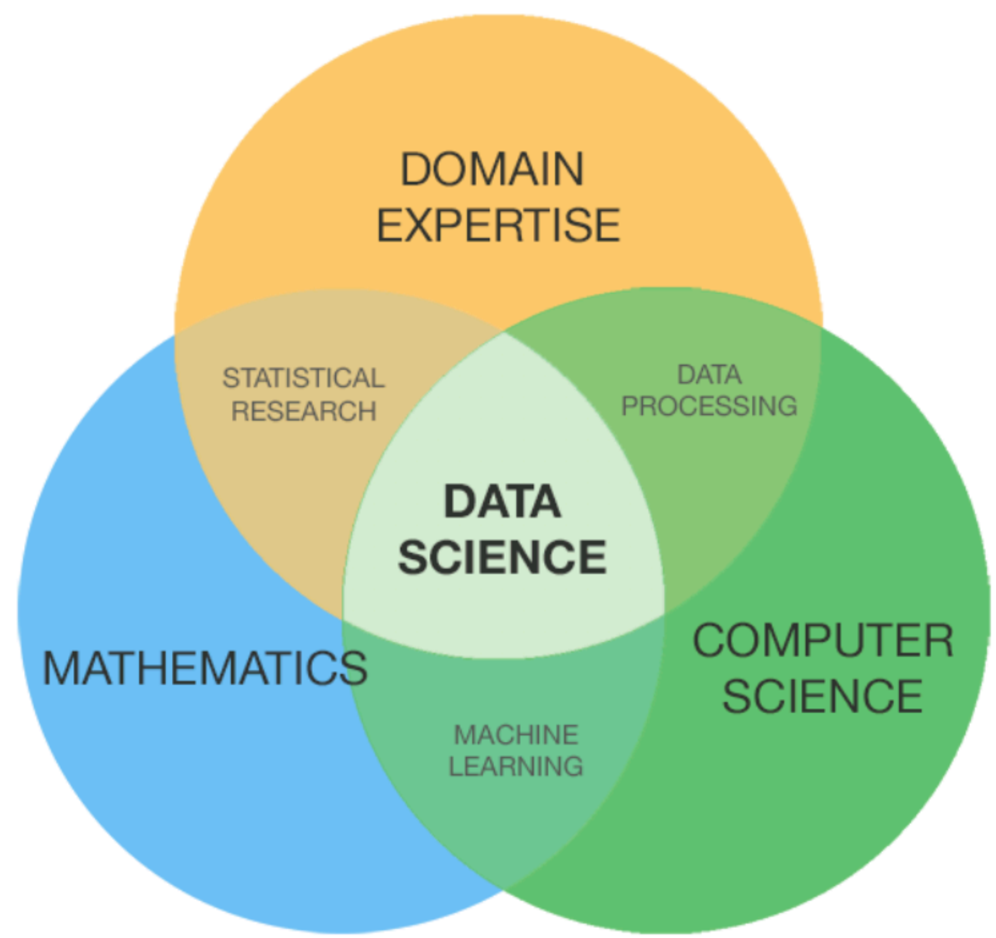
- Interoperability is the problem of making machines and algorithms interoperable, i.e. capable of understanding the semantics of data no matter what the format and data model are
  - A pathway to “Data Intelligence”, a mandatory requirement for ML and DS pipelines
- The course will touch upon the following topics:
  - Part I: Data integration and data exchange for heterogeneous data sources
  - Part II: Data curation and sanitization; repairing and quantifying inconsistencies
  - Part III: Knowledge graphs and property graphs to enforce interoperability across data formats; geographic information systems

Practical work

- TP on Talend Open Studio (Data Integration Tool)
- TP on data cleaning
- TP on Neo4j graph database

Data Integration in DS pipelines

- Data Science is evolving into a set of principles to make sense of data:
  - domain expertise must intervene in data processing
  - datasets become larger, with more complex structure
  - need of integrating heterogeneous data sources



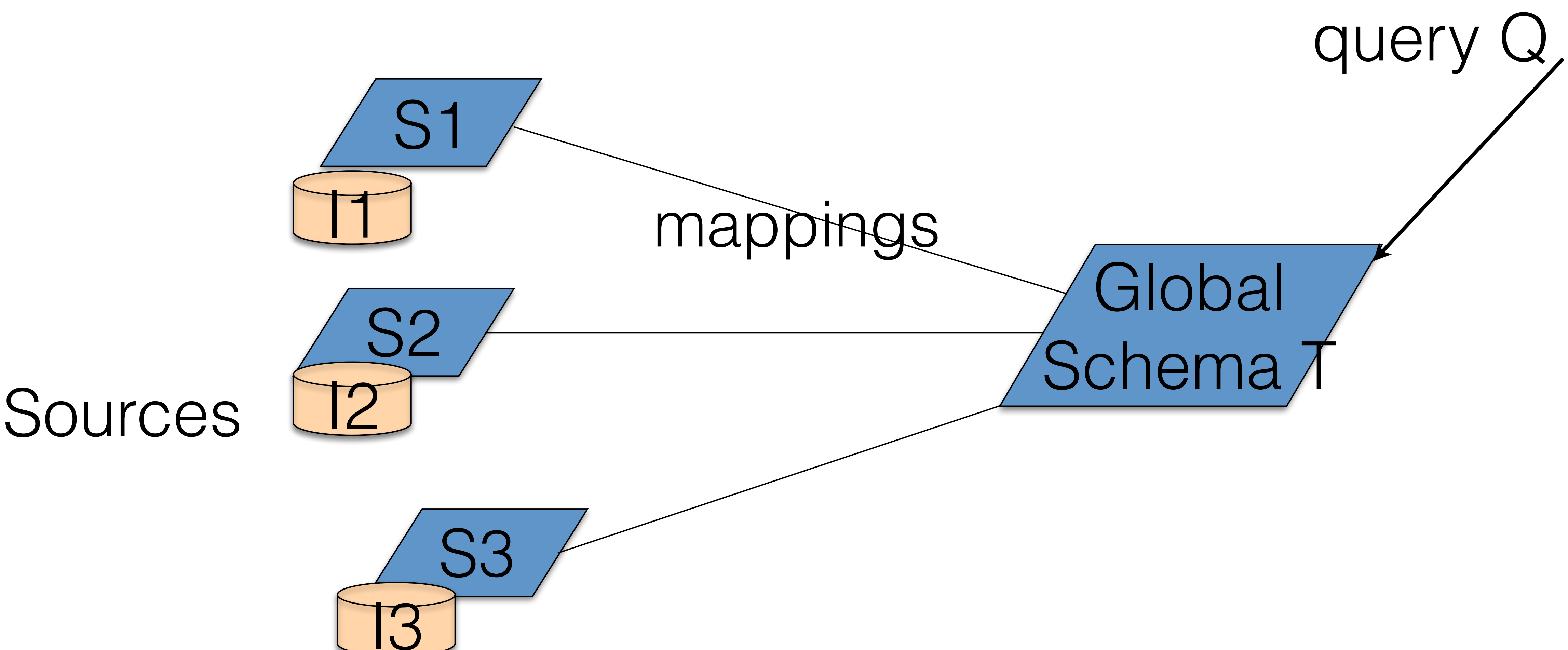
Source: Palmer, Shelly. Data Science for the C-Suite. New York: Digital Living Press, 2015. Print.

Introduction

- Data is inherently heterogeneous
  - Due to the explosion of online data repositories
  - Due to the variety of users, who develop a wealth of applications
    - At different time
    - With disparate requirements in their mind
- A fundamental requirement is to translate data across different formats and to ensure data interoperability
  - Data Integration, Data Exchange are two facets of the same problem
  - Schema Integration and Schema Evolution are also important

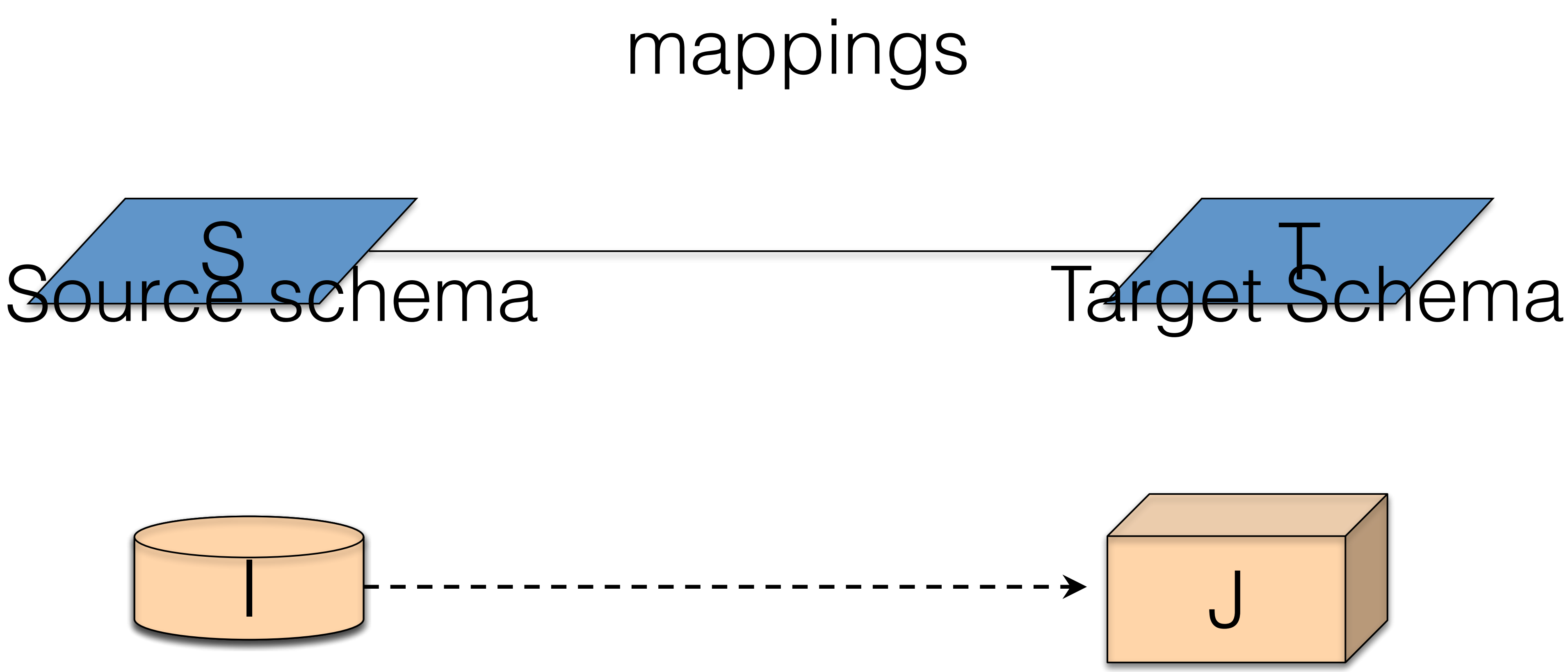
Data Integration

- Data integration [Lenzerini 2002]
  - Query heterogeneous data in different sources via a virtual global schema



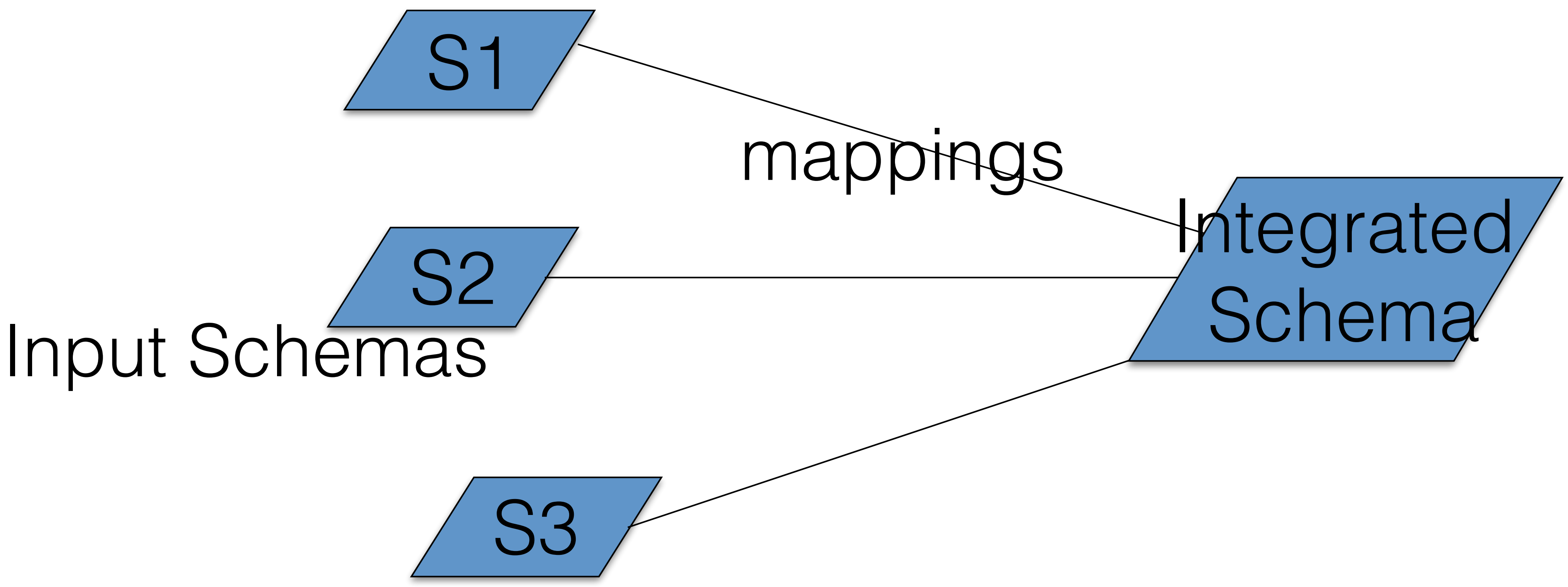
Data exchange

- Data exchange [Fagin et al. 2005]
    - Transform data structured under a source schema into data structured under a different target schema.



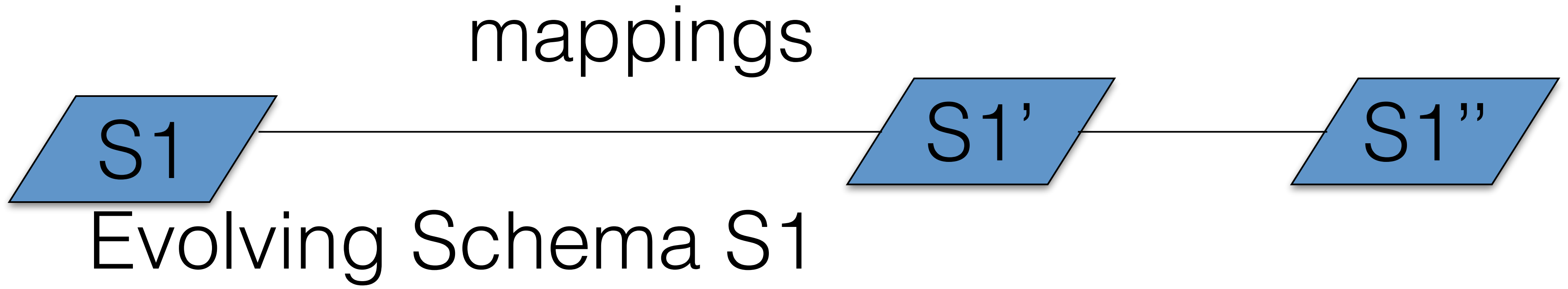
Schema Integration

- Schema integration [Batini et al. 1986]
  - A set of source schemas need to be integrated into one mediated schema



Schema Evolution

- Schema evolution [Lerner 2000]
  - An original schema S1 evolves into subsequent versions S1', S1'' etc.



Data Exchange

- Data Exchange is an old, but recurrent, database problem
- Phil Bernstein, Microsoft – 2003 "Data exchange is the oldest database problem"
- EXPRESS: IBM San Jose Research Lab – 1977 EXtraction, Processing, and REStructuring System for transforming data between hierarchical databases.
- Data Exchange underlies: Data Warehousing, ETL (Extract-Transform-Load) tasks; XML Publishing, XML Storage; more recently, exporting relational data to RDF.

Schema mappings

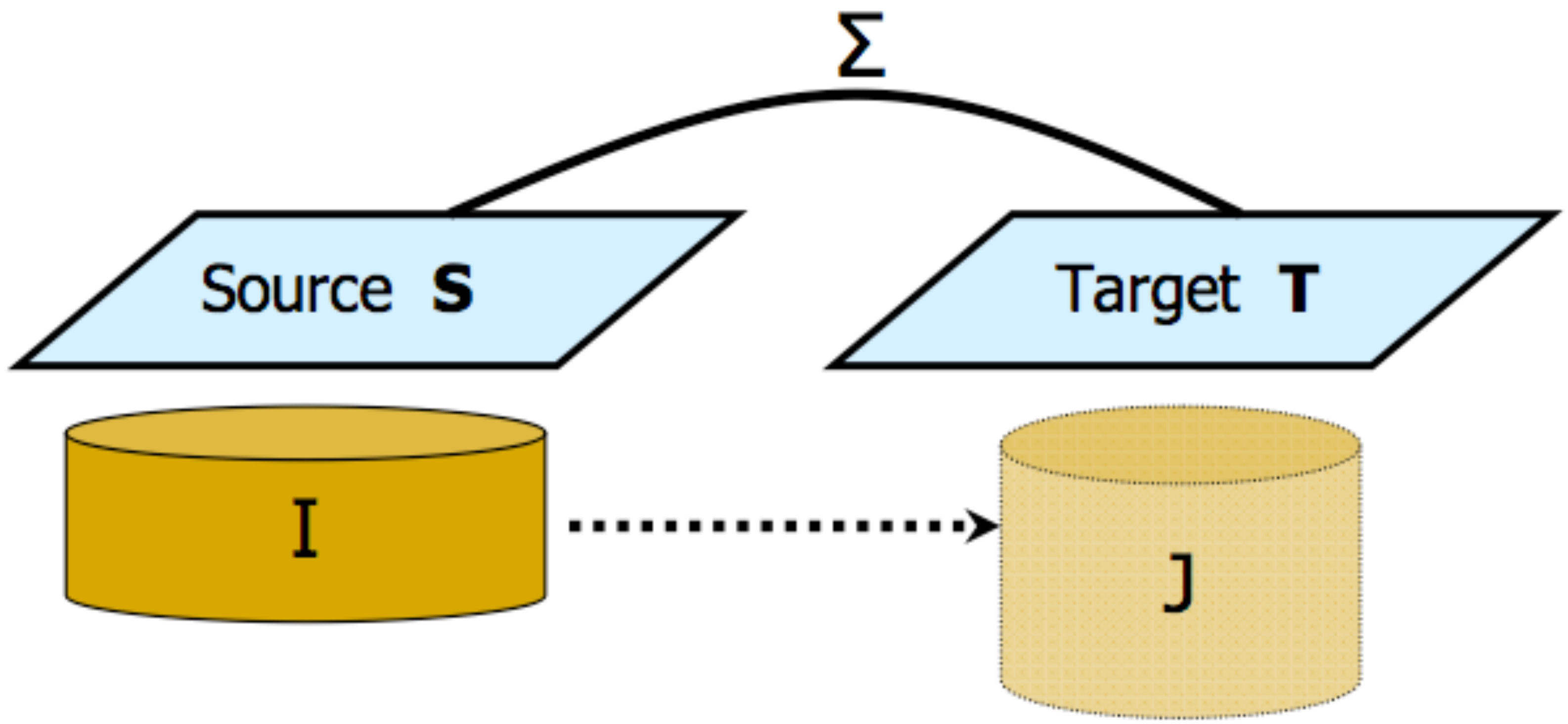
- Schema mappings: high-level, declarative assertions that specify the relationship between two schemas.

- Ideally, schema mappings should be

    - expressive enough to specify data interoperability tasks; simple enough to be efficiently manipulated by tools.

- Schema mappings constitute the essential building blocks in formalizing data integration and data exchange.

- Schema mappings help with the development of tools: are easier to generate and manage (semi)-automatically; can be compiled into SQL/XSLT scripts automatically.



- Schema Mapping M = (S, T, Σ)
    - Source schema S, Target schema T
    - High-level, declarative assertions Σ that specify the relationship between S and T.
- Data Exchange via the schema mapping M = (S, T, Σ)
    - Transform a given source instance I to a target instance J, so that <I, J> satisfy the specifications Σ of M.

Solutions in schema mappings

- Definition: Schema Mapping M = (S, T, Σ) If I is a source instance, then a solution for I is a target instance J such that (I, J) satisfy Σ.

- Fact: In general, for a given source instance I,

- No solution for I may exist or

- Multiple solutions for I may exist; in fact, infinitely many solutions for I may exist.

Schema mapping specification languages

- Question: How are schema mappings specified?
- Answer: Use logic. In particular, it is natural to try to use first-order logic as a specification language for schema mappings.
- Fact: There is a fixed first-order sentence specifying a schema mapping M* such that Sol(M*) is undecidable.
  - Reason: undecidability of validity in FOL
- Hence, we need to restrict ourselves to well-behaved fragments of first-order logic.

Embedded dependencies

- Dependency Theory: extensive study of constraints in relational databases in the 1970s and 1980s.

- Embedded Implicational Dependencies: R. Fagin, C. Beeri and M. Vardi, ...

  - Class of constraints with a balance between high expressive power and good algorithmic properties:

  - Tuple-generating dependencies   (tgds) Inclusion and multi-valued dependencies are a special case.

  - Equality-generating dependencies (egds) Functional dependencies are a special case.

Schema mapping specification language

- The relationship between source and target is given by formulas of first-order logic, called Source-to-Target Tuple Generating Dependencies (s-t tgds)
- $\phi(x) \rightarrow \exists y\ \psi(x, y)$, where
- $\phi(x)$ is a conjunction of atoms over the source; $\psi(x, y)$ is a conjunction of atoms over the target.
- Example:
- $(\text{Student}(s) \land \text{Enrolls}(s,c)) \rightarrow \exists t\ \exists g\ (\text{Teaches}(t,c) \land \text{Grade}(s,c,g))$

Schema mapping specification language

- s-t tgds assert that: some SPJ source query is contained in some other SPJ target query
- (Student (s) $\wedge$ Enrolls(s,c)) $\rightarrow$ $\exists$t $\exists$g (Teaches(t,c) $\wedge$ Grade(s,c,g))
- s-t tgds generalize the main specifications used in data integration:
  - They generalize LAV (local-as-view) specifications:
    - P(x) $\rightarrow$ $\exists$y ψ(x, y), where P is a source schema.
  - They generalize GAV (global-as-view) specifications:
    - ϕ(x) $\rightarrow$ R(x), where R is a target schema
    - They are equivalent full tgds: ϕ(x) $\rightarrow$ ψ(x), where ϕ(x) and ψ(x) are conjunctions of atoms

Examples of simple mapping tasks

- Let us consider some simple tasks that a schema mapping specification language should support:
  - Copy (Nicknaming): Copy each source table to a target table and rename it.
  - Projection: Form a target table by projecting on one or more columns of a source table.
  - Decomposition: Decompose a source table into two or more target tables.
  - Column Augmentation: Form a target table by adding one or more columns to a source table.
  - Join: Form a target table by joining two or more source tables.
  - Combinations of the above (e.g., “join + column augmentation”)

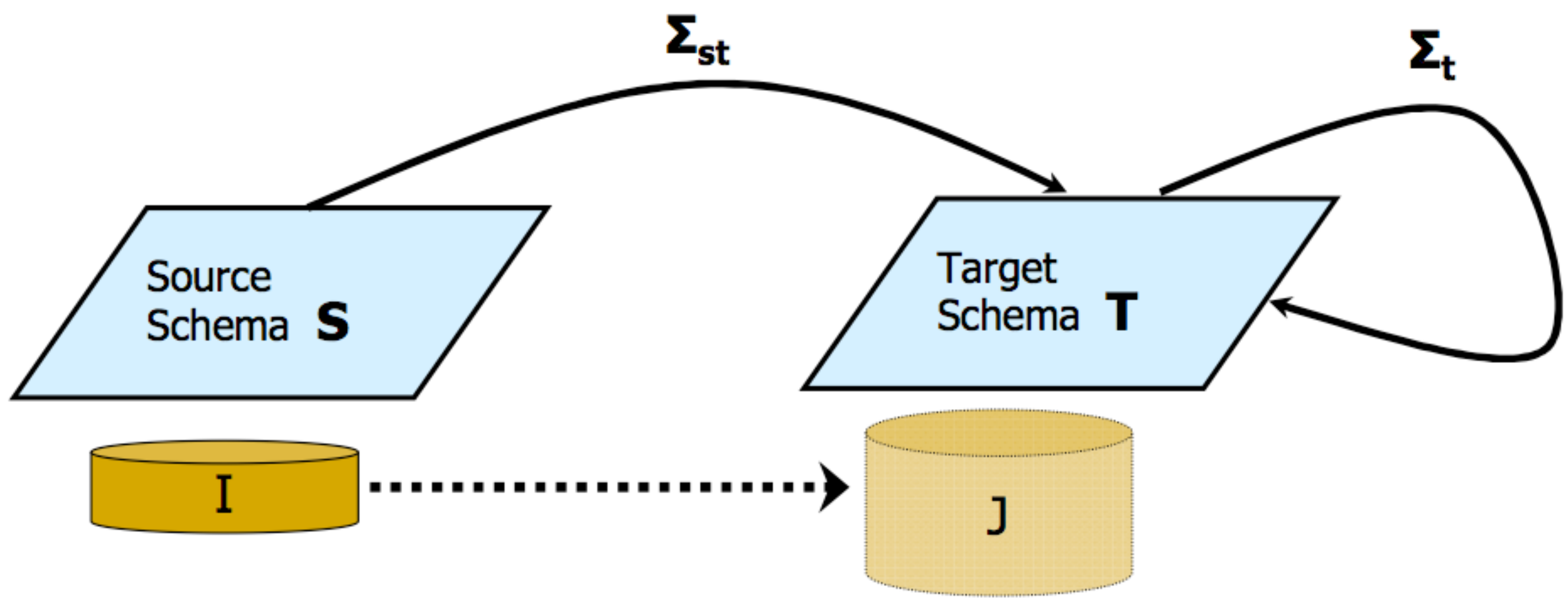
Examples of simple mapping tasks

- Copy (Nicknaming):
  - $\forall x_1, \ldots, x_n(P(x_1,\ldots,x_n) \rightarrow R(x_1,\ldots,x_n))$
- Projection:
  - $\forall x,y,z(P(x,y,z) \rightarrow R(x,y))$
- Decomposition:
  - $\forall x,y,z\ (P(x,y,z) \rightarrow R(x,y) \wedge T(y,z))$
- Column Augmentation:
  - $\forall x,y\ (P(x,y) \rightarrow \exists\ z\ R(x,y,z))$
- Join:
  - $\forall x,y,z(E(x,z) \wedge F(z,y) \rightarrow R(x,y,z))$
- Combinations of the above (e.g., "join + column augmentation")
  - $\forall x,y,z(E(x,z) \wedge F(z,y) \rightarrow \exists\ w\ T(x,y,z,w)))$

Target dependencies

- In addition to source-to-target dependencies, we also consider target dependencies:
  - Target Tgds : $\phi_T(x) \rightarrow \exists y\, \psi_T(x, y)$
  - Dept (did, dname, mgr_id, mgr_name) → Mgr (mgr_id, did) (a target inclusion dependency constraint)
  - Target Egds (Equality Generating Dependencies): $\phi_T(x) \rightarrow (x1 = x2)$
  - $(Mgr(e,d1) \wedge Mgr(e,d2)) \rightarrow (d1 = d2)$ (a target key constraint)

Data exchange framework



- Schema Mapping M = (S, T, Σst , Σt), where
  - Σst is a set of source-to-target tgds
  - Σt is a set of target tgds and target egds

Multiple solutions

- Fact: Given a source instance, multiple solutions may exist.
- Example: Source relation E(A,B), target relation H(A,B)
- Σ: $E(x,y) \rightarrow \exists z\, (H(x,z) \wedge H(z,y))$
    - Source instance I = {E(a,b)}
    - Solutions: Infinitely many solutions exist

Example

- Consider a set of source-to-target dependencies Σst :
  - (d1) $EmpCity(e, c) \rightarrow \exists H\ Home(e, H)$,
  - (d2) $EmpCity(e, c) \rightarrow \exists D\ (EmpDept(e, D) \wedge DeptCity(D, c))$,
  - (d3) $LivesIn(e, h) \rightarrow Home(e, h)$,
  - (d4) $LivesIn(e, h) \rightarrow \exists D \exists C (EmpDept(e, D) \wedge DeptCity(D, C))$,
- and a source instance I such that:
  - I = {EmpCity(Alice, SJ), EmpCity(Bob, SD), LivesIn(Alice, SF), LivesIn(Bob, LA)}.
- **Which possible solutions J do exist?**

Possible solutions J0,J,J0’

- J0 = {Home(Alice, SF), Home(Bob, LA), EmpDept(Alice, D1), EmpDept(Bob, D2), DeptCity(D1, SJ), DeptCity(D2, SD)},

- J = {Home(Alice, SF), Home(Bob, LA), Home(Alice, H1), Home(Bob, H2), EmpDept(Alice, D1), EmpDept(Bob, D2), DeptCity(D1, SJ), DeptCity(D2, SD)},

- J0′= {Home(Alice, SF), Home(Bob, LA), EmpDept(Alice, D), EmpDept(Bob, D), DeptCity(D, SJ), DeptCity(D, SD)}.

Exercise1

- Say whether/why the next J is a solution for the schema mapping Σ: $E(x,y) \rightarrow \exists z\ (H(x,z) \wedge H(z,y))$ with source instance $I = \{E(a,b)\}$
  - Q1.1: $J1 = \{H(a,b), H(b,b)\}$
  - Q1.2: $J2 = \{H(a,a), H(a,b)\}$
  - Q1.3: $J3 = \{H(a,X), H(X,b)\}$
  - Q1.4: $J4 = \{H(a,X), H(X,b), H(a,Y), H(Y,b)\}$
  - Q1.5: $J5 = \{H(a,X), H(X,b), H(Y,Y)\}$

Main issues in data exchange

- For a given source instance, there may be multiple target instances satisfying the specifications of the schema mapping.

- When more than one solution exist, which solutions are “better” than others?

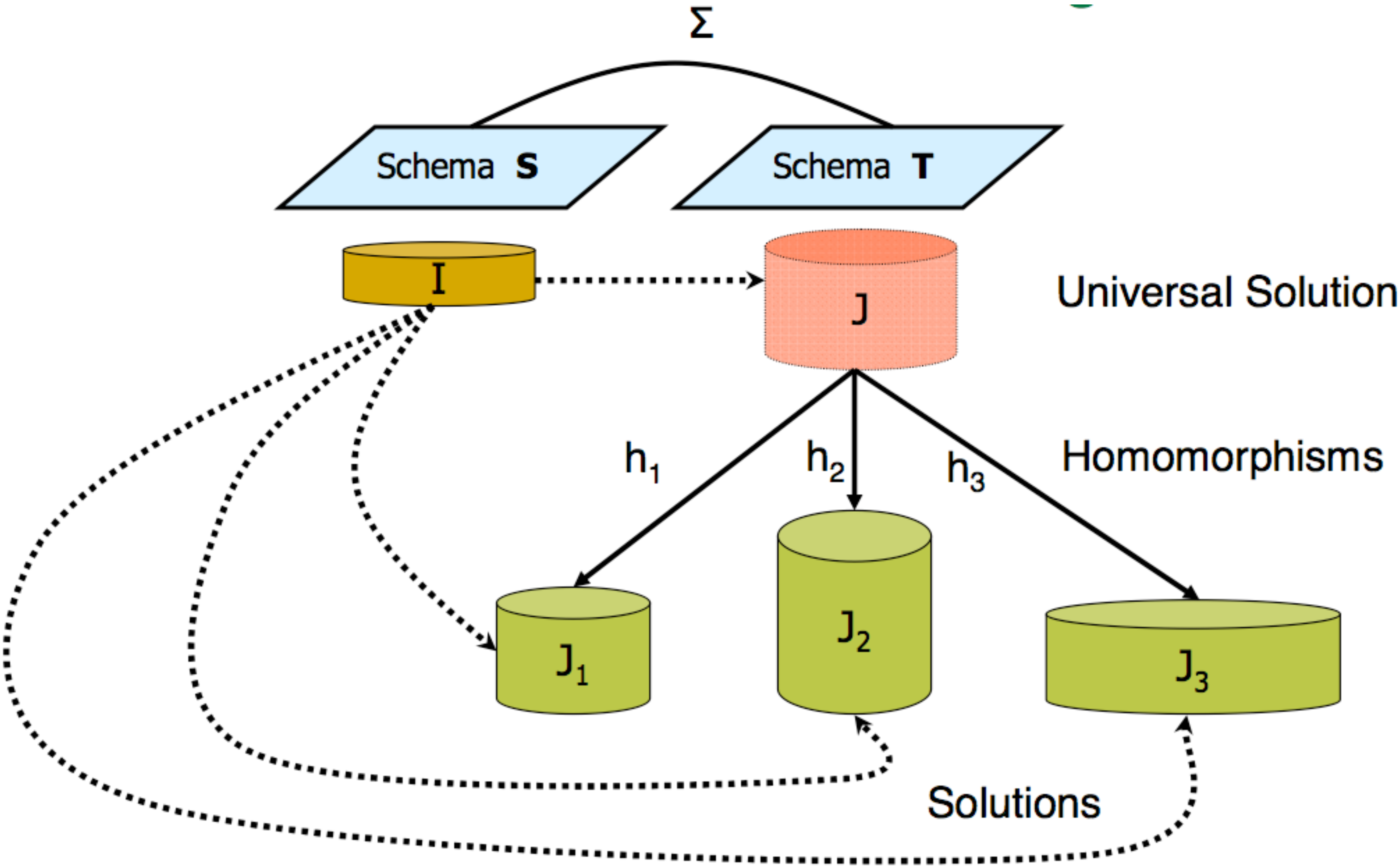
- How do we compute a “best” solution?

- In other words, what is the “right” semantics of data exchange?

Universal solutions in data exchange

- We introduce the notion of universal solutions as the "best" solutions in data exchange.
- By definition, a solution is universal if it has homomorphisms to all other solutions (thus, it is a "most general" solution).
- Constants: entries in source instances;
- Variables (labeled nulls): invented entries in target instances
- Homomorphism h: J1 → J2 between target instances:
  - h(c) = c, for every constant c in J1
  - For every fact P(a1,...,am) in J1, then we have that P(h(a1),...,h(am)) is a fact in J2

Universal solutions in data exchange



Exercise1 (cont'd)

- Say whether/why the next J is a 'universal' solution
  - Q1.6: J1 = {H(a,b), H(b,b)}
  - Q1.7: J2 = {H(a,a), H(a,b)}
  - Q1.8: J3 = {H(a,X), H(X,b)}
  - Q1.9: J4 = {H(a,X), H(X,b), H(a,Y), H(Y,b)}
  - Q1.10: J5 = {H(a,X), H(X,b), H(Y,Y)}

Structural Properties of Universal Solutions

- Universal solutions are analogous to most general unifiers in logic programming.

- Uniqueness up to homomorphic equivalence: If J and J' are universal for I, then they are homomorphically equivalent.

- Representation of the entire space of solutions: Assume that J is universal for I, and J' is universal for I'. Then the following are equivalent:

    - I and I' have the same space of solutions.

    - J and J' are homomorphically equivalent.

Existence-of-solutions problem

- Question: What can we say about the existence-of-solutions problem Sol(M) for a fixed schema mapping M = (S, T, Σst,Σt) specified by s-t tgds and target tgds and egds?
- Answer: Depending on the target constraints in Σt:
    - Sol(M) can be trivial (solutions always exist).
    - Sol(M) can be in PTIME
    - Sol(M) can be undecidable.

Existence-of-solutions problem

- Proposition: If M = (S, T, Σst, Σt) is a schema mapping such that Σt is a set of full target tgds, then:
  - Solutions always exist; hence, Sol(M) is trivial.
  - There is a Datalog program π over the target T that can be used to compute universal solutions as follows: Given a source instance I,
    - Compute a universal solution J^* for I w.r.t. the schema mapping M^* = (S, T, Σst) using the naïve chase algorithm.
    - Run the Datalog program π on J^* to obtain a universal solution J for I w.r.t. M.
  - Consequently, universal solutions can be computed in polynomial time.

The Chase Algorithm

- Naïve Chase Algorithm for $M^* = (S, T, \Sigma st)$: given a source instance I, build a target instance J^* that satisfies each s-t tgd in Σst by introducing new facts in J as dictated by the RHS of the s-t tgd and by introducing new values (variables) in J each time existential quantifiers need witnesses.
- Example: $M = (S, T, \Sigma st, \Sigma t)$
  - Σst: $E(x,y) \rightarrow \exists\, z(F(x,z) \wedge F(z,y))$
  - Σt: $F(u,w) \wedge F(w,v) \rightarrow F(u,v)$
- The naïve chase returns a relation F^* obtained from E by adding a new node between every edge of E.
- The Datalog program π computes the transitive closure of F^*.

Algorithmic properties of universal solutions

- Theorem (FKMP 2003): Schema mapping M= (S, T, Σst, Σt) such that:
    - Σst is a set of source-to-target tgds;
    - Σt is the union of a weakly acyclic set of target tgds with a set of target egds.
- Then: Universal solutions exist if and only if solutions exist.
- Sol(M) is in PTIME.
- If a solution exists, then a universal solution can be produced in polynomial time using the chase procedure.

Weakly acyclic sets of tgds

- Position graph of a set Σ of target tgds:
- Nodes: R.A, with R relation symbol, A attribute of R
- Edges: for every $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x},\mathbf{y})$ in Σ, for every x in $\mathbf{x}$ occurring in ψ, for every occurrence of x in ϕ in position R.A:
  - For every occurrence of x in ψ in position S.B, add an edge $\quad$ R.A $\longrightarrow$ S.B
  - In addition, for every existentially quantified y that occurs in ψ in position T.C, add a special edge $\quad$ R.A $\Rightarrow$ T.C
- Σ is weakly acyclic if the position graph has no cycle containing a special edge.
- A tgd θ is weakly acyclic if so is the singleton set $\{\theta\}$.

Weakly acyclic sets of tgds: examples

- **Example 1:** $\{ D(e,m) \rightarrow M(m),\ M(m) \rightarrow \exists\, e\, D(e,m) \}$
  is weakly acyclic, but cyclic.

$$\begin{tikzcd} D.1 & M.1 \arrow[l, Rightarrow] \arrow[r, shift right] & D.2 \arrow[l, shift right] \end{tikzcd}$$

- **Example 2:** $\{ E(x,y) \rightarrow \exists\, z\, E(y,z) \}$
  is not weakly acyclic.

$$\begin{tikzcd} E.1 & E.2 \arrow[l] \arrow[loop right, Rightarrow] \end{tikzcd}$$

- A tgd θ is weakly acyclic if so is the singleton set {θ} .

Exercise 2

$$\Sigma_{st} = \{ \texttt{DeptEmp}(d, n, e) \rightarrow \exists M (\texttt{Dept}(d, M, n) \wedge \texttt{Emp}(e, d)) \},$$

$$\begin{aligned} \Sigma_t = \{ & \texttt{Dept}(d, m, n) \rightarrow \exists D\, \texttt{Emp}(m, D), \\ & \texttt{Emp}(e, d) \rightarrow \exists M \exists N\, \texttt{Dept}(d, M, N) \}. \end{aligned}$$

- (Q2.1) Build the dependency graph for the following schema mapping and say whether/why the set of tgds is weakly acyclic.

Exercise 2

$$\Sigma_{st} = \{\ \texttt{DeptEmp}(d, n, e) \rightarrow \exists M(\texttt{Dept}(d, M, n) \wedge \texttt{Emp}(e, d))\ \},$$

$$\begin{aligned} \Sigma_t' = \{\ & \texttt{Dept}(d, m, n) \rightarrow \texttt{Emp}(m, d), \\ & \texttt{Emp}(e, d) \rightarrow \exists M \exists N\ \texttt{Dept}(d, M, N)\ \}. \end{aligned}$$

- (Q2.2) Build the dependency graph for the following schema mapping and say whether/why the set of tgds is weakly acyclic

Data Exchange with weakly acyclic sets of tgds

- Theorem: Schema mapping M= (S, T, Σst, Σt) such that:
  - Σst is a set of source-to-target tgds;
  - Σt is the union of a weakly acyclic set of target tgds with a set of target egds.
- There is an algorithm, based on the chase procedure, so that:
  - Given a source instance I, the algorithm determines if a solution for I exists; if so, it produces a universal solution for I.
  - The running time of the algorithm is polynomial in the size of I.
  - Hence, the existence-of-solutions problem Sol(M) for M, is in PTIME.

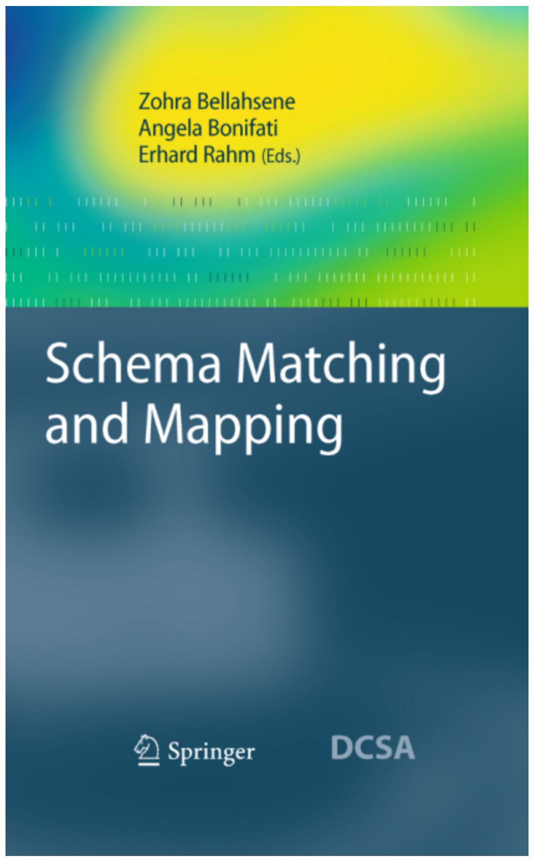
Chase Procedure for Tgds and Egds

- Given a source instance I,
- 1. Use the naïve chase to chase I with Σst and obtain a target instance J*.
- 2. Chase J * with the target tgds and the target egds in Σt to obtain a target instance J as follows:
    - 2.1. For target tgds introduce new facts in J as dictated by the RHS of the s-t tgd and introduce new values (variables) in J each time existential quantifiers need witnesses.
    - 2.2. For target egds $\phi(x) \rightarrow x1 = x2$
        - 2.2.1. If a variable is equated to a constant, replace the variable by that constant;
        - 2.2.2. If one variable is equated to another variable, replace one variable by the other variable.
        - 2.2.3 If one constant is equated to a different constant, stop and report “failure”.

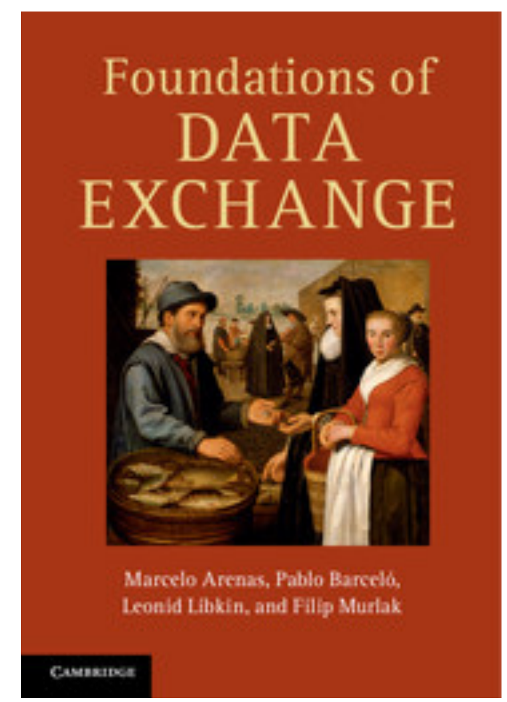
The existence-of-solutions problem

- Summary: The existence-of-solutions problem
  - is undecidable for schema mappings in which the target dependencies are arbitrary tgds and egds;
  - is in PTIME for schema mappings in which the set of the target dependencies is the union of a weakly acyclic set of tgds and a set of egds.

Books



[Bellahsene et al. 2011] Z. Bellahsene, A. Bonifati and E. Rahm, "Schema Matching and Mapping", Springer-Verlag, 2011



[Arenas et al. 2014] M. Arenas et al., "Foundations of Data Exchange", Cambridge University Press, 2014

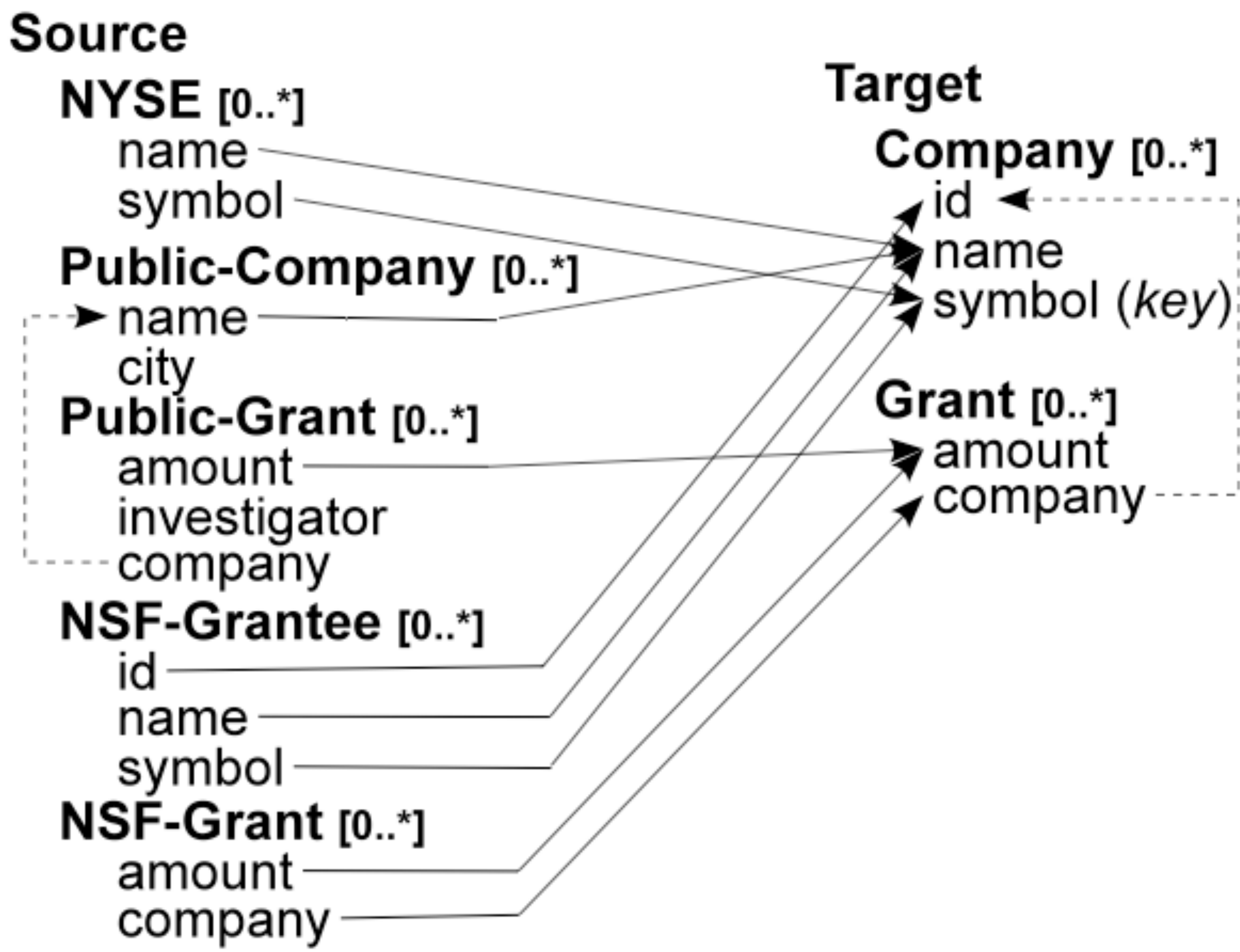
List of References

- [Batini et al. 1986] Batini C, Lenzerini M, Navathe SB (1986) A Comparative Analysis of Methodologies for Database Schema Integration. ACM Comp. Surv. 18(4):323-364
- [Lenzerini 2002] Lenzerini Maurizio "Data Integration: A Theoretical Perspective". In: PODS, pp 233-246
- [Fagin et al. 2005] Fagin R, Kolaitis PG, Miller RJ, Popa L (2005) Data exchange: semantics and query answering. Theoretical Computer Science 336(1):89-124
- [Lerner 2000] Lerner BS (2000) A Model for Compound Type Changes Encountered in Schema Evolution. TPCTC 25(1):83–127
- Slides:
  - P. Kolaitis talk at PODS 2005 http://users.soe.ucsc.edu/~kolaitis/talks/datexch-pods05.pdf
  - A. Bonifati and Y. Velegrakis tutorial at EDBT 2011 http://www.lifl.fr/~bonifati/pubs/EDBT11MappingBenchmarkTutorial.pdf

List of References

- [Batini et al. 1986] Batini C, Lenzerini M, Navathe SB (1986) A Comparative Analysis of Methodologies for Database Schema Integration. ACM Comp. Surv. 18(4):323-364
- [Lenzerini 2002] Lenzerini Maurizio "Data Integration: A Theoretical Perspective". In: PODS, pp 233-246
- [Fagin et al. 2005] Fagin R, Kolaitis PG, Miller RJ, Popa L (2005) Data exchange: semantics and query answering. Theoretical Computer Science 336(1):89-124
- [Lerner 2000] Lerner BS (2000) A Model for Compound Type Changes Encountered in Schema Evolution. TPCTC 25(1):83–127
- Extra slides:
  - A. Bonifati and Y. Velegrakis tutorial at EDBT 2011 http://www.lifl.fr/~bonifati/pubs/EDBT11MappingBenchmarkTutorial.pdf

Exercise 3- homework



- as:

Exercise 3

SOURCE-TO-TARGET TGDS

$m_1.\ \forall s,n: NYSE(s,n) \rightarrow \exists I\colon Company(I,n,s)$

$m_2.\ \forall n,c,a,pi: Public\text{-}Company(n,c) \wedge Public\text{-}Grant(a,pi,n) \rightarrow$
$\exists I,S\colon Company(I,n,S) \wedge Grant(a,I)$

$m_3.\ \forall i,n,s: NSF\text{-}Grantee(i,n,s) \rightarrow Company(i,n,s)$

$m_4.\ \forall a,c: NSF\text{-}Grant(a,c) \rightarrow Grant(a,c)$

TARGET TGDS

$t_1.\ \forall a,c: Grant(a,c) \rightarrow \exists N,S\colon Company(c,N,S)$

TARGET EGDS

$e_1.\ \forall n,n',i,i',s: Company(i,n,s) \wedge Company(i',n',s) \rightarrow (i = i') \wedge (n = n')$

Exercise 3

- Assume you have the following source instance I:

NYSE

name	symbol
Google	GOOG
Yahoo!	YHOO

Public-Company

name	city
Apple	Cup
Adobe	SJ

Public-Grant

company	investigator	amount
Apple	Mike B.	25,000
Adobe	Anne C.	50,000

NSF-Grantee

id	name	symbol
23	Yahoo!	YHOO
25	Adobe	ADBE

NSF-Grant

company	amount
23	18,000
25	50,000

Homework I - Exo 3

- Q3.1: Say whether the following solution is

Company

id	name	symbol
N1	Google	GOOG
N2	Yahoo	YHOO
I1	Apple	S1
I2	Adobe	S2
23	Yahoo!	YHOO
25	Adobe	ADBE

Grant

amount	company
25,000	I1
50,000	I2
18,000	23
50,000	25

niversal solution? a non-universal solution? other?

Homework I - Exo 3

- Q3.2: Say whether the following solution is

Company

id	name	symbol
N1	Google	GOOG
I1	Apple	S1
I2	Adobe	S2
23	Yahoo!	YHOO
25	Adobe	ADBE

Grant

amount	company
25,000	I1
50,000	I2
18,000	23
50,000	25

- a universal solution? a non-universal solution? other?

Homework I - Exo 3

- Q3.3: Say whether the following solution is

Company

id	name	symbol
N1	Google	GOOG
I1	Apple	NULL
23	Yahoo!	YHOO
25	Adobe	ADBE

Grant

amount	company
25,000	I1
18,000	23
50,000	25

  - a universal solution? a non-universal solution? other?

Exercise 3

- Q3.4: Say whether the following solution is

Company

id	name	symbol
N1	Google	GOOG
I1	Apple	NULL
23	Yahoo!	YHOO
25	Adobe	ADBE

Grant

amount	company
25,000	I1
18,000	I2
50,000	25
80,000	N1

- a universal solution? a non-universal solution? other?