ILLINOIS INSTITUTE OF TECHNOLOGY

Towards Rigorous Evaluation of Data Integration Systems – It's All About the Tools

Boris Glavic[1]

P. Arocena[2], R. Ciucanu[3], G. Mecca[4], R. J. Miller[2], P. Papotti[5], D. Santoro[4]

IIT[1]

University of Toronto[2]

Université Blaise Pascal[3]

Università della Basilicata[4]

Arizona State University[5]

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

1) **Empirical Evaluation of Integration Systems**
2) iBench
3) BART
4) Success Stories
5) Demo
6) Conclusions and Future Work

Overview

- **Challenges of evaluating integration systems**
  - **Diversity of tasks**
    - Various types of **metadata** used by integration tasks
  - **Quality** is as important as performance
    - Often requires "***gold standard***" solution
- **Goal**: make empirical evaluations …
    - … more **robust**, **repeatable**, **shareable**, and **broad**
    - … less **painful** and **time-consuming**
- **This talk:**
  - **iBench** – a flexible metadata generator
  - **BART** – generating data quality errors

Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Challenges of evaluating integration systems**
  - **Diversity of tasks**
    - Various types of **metadata** used by integration tasks

Patterson [CACM 2012]
"When a field has good benchmarks, we settle debates and the field makes rapid progress."

  - **iBench** – a flexible metadata generator
  - **BART** – generating data quality errors

Integration Tasks

ILLINOIS INSTITUTE OF TECHNOLOGY

Many integration tasks work with metadata:

- **Data Exchange**
  - *Input*: **Schemas**, **Constraints**, (Source Instance), **Mappings**
  - *Output*: Executable Transformations, (Target Instance)
- **Schema Mapping Generation**
  - *Input*: **Schemas**, **Constraints**, Instance Data, **Correspondences**
  - *Output*: **Mappings**, Transformations
- **Schema Matching**
  - *Input*: **Schemas**, (Instance Data), (**Constraints**)
  - *Output*: **Correspondences**
- **Constraint-based Data Cleaning**
  - *Input*: Instance Data, **Constraints**
  - *Output*: Instance Data
- **Constraint Discovery**
  - *Input*: **Schemas**, Instance Data
  - *Output*: **Constraints**
- **Virtual Data Integration**
  - *Input*: **Schemas**, Instance Data, **Mappings**, Queries
  - *Output*: Rewritten Queries, Certain Query Results
- **… and many others** (e.g., Mapping Operators, Schema Evolution, ...)

Integration Tasks

Many integration tasks work with metadata:

- **Data Exchange**
  - *Input*: **Schemas**, **Constraints**, (Source Instance), **Mappings**
  - *Output*: Executable Transformations, (Target Instance)
- **Schema Mapping Generation**
  - *Input*: **Schemas**, **Constraints**, Instance Data, **Correspondences**
  - *Output*: **Mappings**, Transformations
- **Schema Matching**

Inputs/Outputs

Metadata: Schemas, Constraints, Correspondences, Mappings

Data: Source Instance, Target Instance

- **Constraint Discovery**
  - *Input*: **Schemas**, Instance Data
  - *Output*: **Constraints**
- **Virtual Data Integration**
  - *Input*: **Schemas**, Instance Data, **Mappings**, Queries
  - *Output*: Rewritten Queries, Certain Query Results
- **… and many others** (e.g., Mapping Operators, Schema Evolution, ...)

State-of-the-art

- How are integration systems typically evaluated?
- **Small real-world integration scenarios**
  - **Advantages**:
    - Realistic ;-)
  - **Disadvantages**:
    - Not possible to scale (schema-size, data-size, …)
    - Not possible to vary parameters (e.g., mapping complexity)
- **Ad-hoc synthetic scenarios**
  - **Advantages**:
    - Can influence scale and characteristics
  - **Disadvantages**:
    - Often not very realistic metadata
    - Diversity requires huge effort

Requirements

- **We need tools to generate inputs/outputs**
  - **Scalability**
    - Generate large integration scenarios efficiently
    - Requires low user effort
  - **Control over metadata and data characteristics**
    - Size
    - Structure
    - …
  - **Generate inputs as well as gold standard outputs**
  - **Promote reproducibility**
    - Enable other researchers to regenerate metadata to repeat an experiment
    - Support researchers in understanding the generated metadata/data
    - Enable researchers to reuse generated integration scenarios

Related Work

ILLINOIS INSTITUTE OF TECHNOLOGY

- **STBenchmark** [Alexe et al. PVLDB '08]
  - Pioneered the **primitive** approach:
    - Generate metadata by combining typical micro scenarios
- **Data generators**
  - PDGF, Myriad
  - Data generators are not enough

Outline

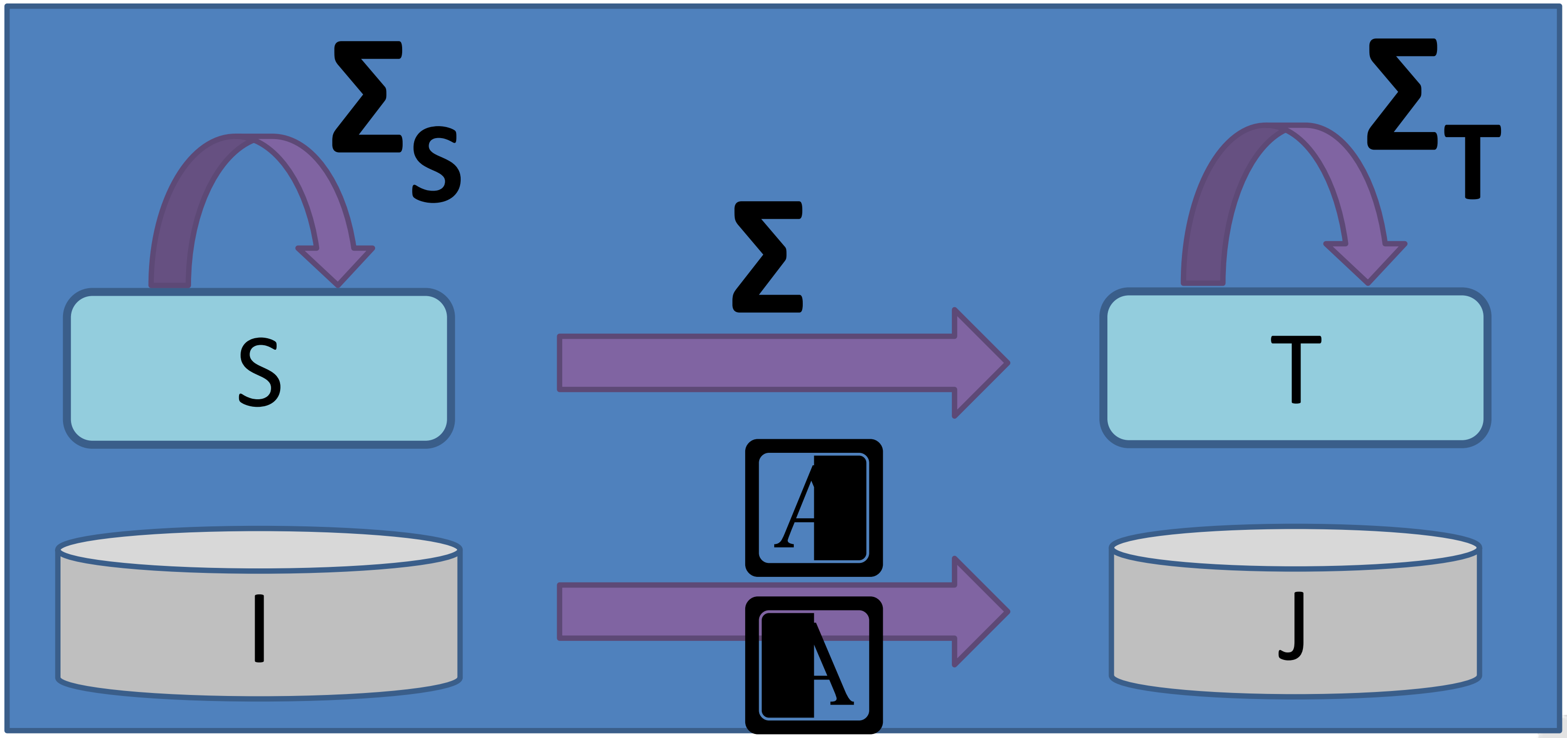
1) Empirical Evaluation of Integration Systems
2) **iBench**
3) BART
4) Success Stories
5) Demo
6) Conclusions and Future Work

iBench Overview

- **iBench** is a metadata and data generator
- **Generates synthetic integration scenarios**
  - **Metadata**
    - Schemas
    - Constraints
    - Mappings
    - Correspondences
  - **Data**
- **“Realistic” metadata**

Integration Scenarios

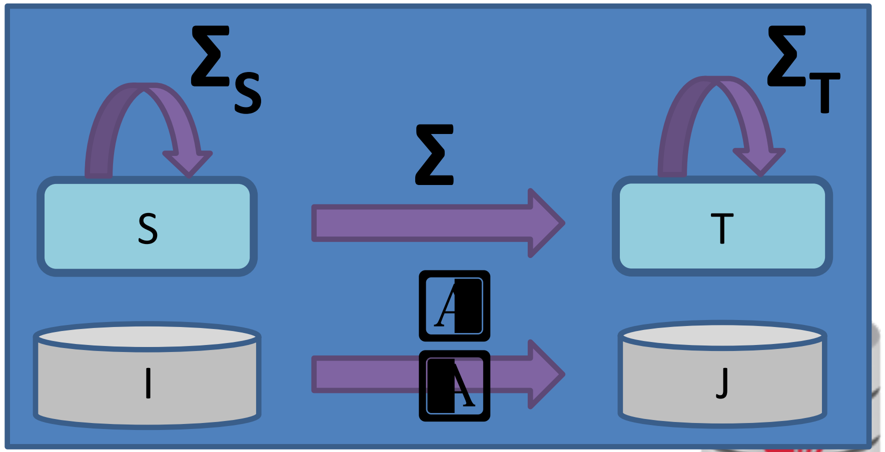
- **Integration Scenario**
  - $M = (S, T, \Sigma_S, \Sigma_T, \Sigma, I, J, \boxed{A}, \boxed{A})$



Integration Scenarios

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Integration Scenario**
  - **$M = (S, T, \Sigma_S, \Sigma_T, \Sigma, I, J, \boxed{A} \boxed{A})$**
  - **Source schema S with instance I**
  - **Target schema T with instance J**
  - **Source constraints Σ_S and target constraints Σ_T**
    - **Instance I fulfills Σ_S and instance J fulfills Σ_T**
  - **Schema mapping Σ**
    - **Instances (I,J) fulfill Σ**
  - **Transformations $\boxed{A}$ $\boxed{A}$**



iBench Input/Output

- **Inputs - Configuration**
  - **Scenario parameters Π** (min/max constraints)
    - Number of source relations
    - Number of attributes of target relations
    - …
  - **Primitive parameters**
    - Template micro-scenarios that are instantiated to create part of the output
- **Output**
  - A integration scenario **M** that fulfills the constraints of specified in the configuration
    - XML file with metadata
    - CSV files for data

Example - MD Task

- **Input**

Parameter	Source	Target
Number Relations	2-4	1-3
Number Attributes	2-10	2-10
Number of Join Attr	1-2	1-2
Number of Existentials		0-3

- **Example solution (mappings)**

- `S1(A,B,C),S2(C,D,E) -> T(A,E)`
- `S3(A,B,C,D),S4(E,A,B) -> ∃X,Y,Z T1(A,X,X),`
  `T2(A,Y,C),T3(C,B,Y,Z)`

Example - MD Task

- **Input**

Parameter	Source	Target
Number Relations	2-4	1-3
Number Attributes	2-10	2-10
Number of Join Attr	1-2	1-2
Number of Existentials		0-3

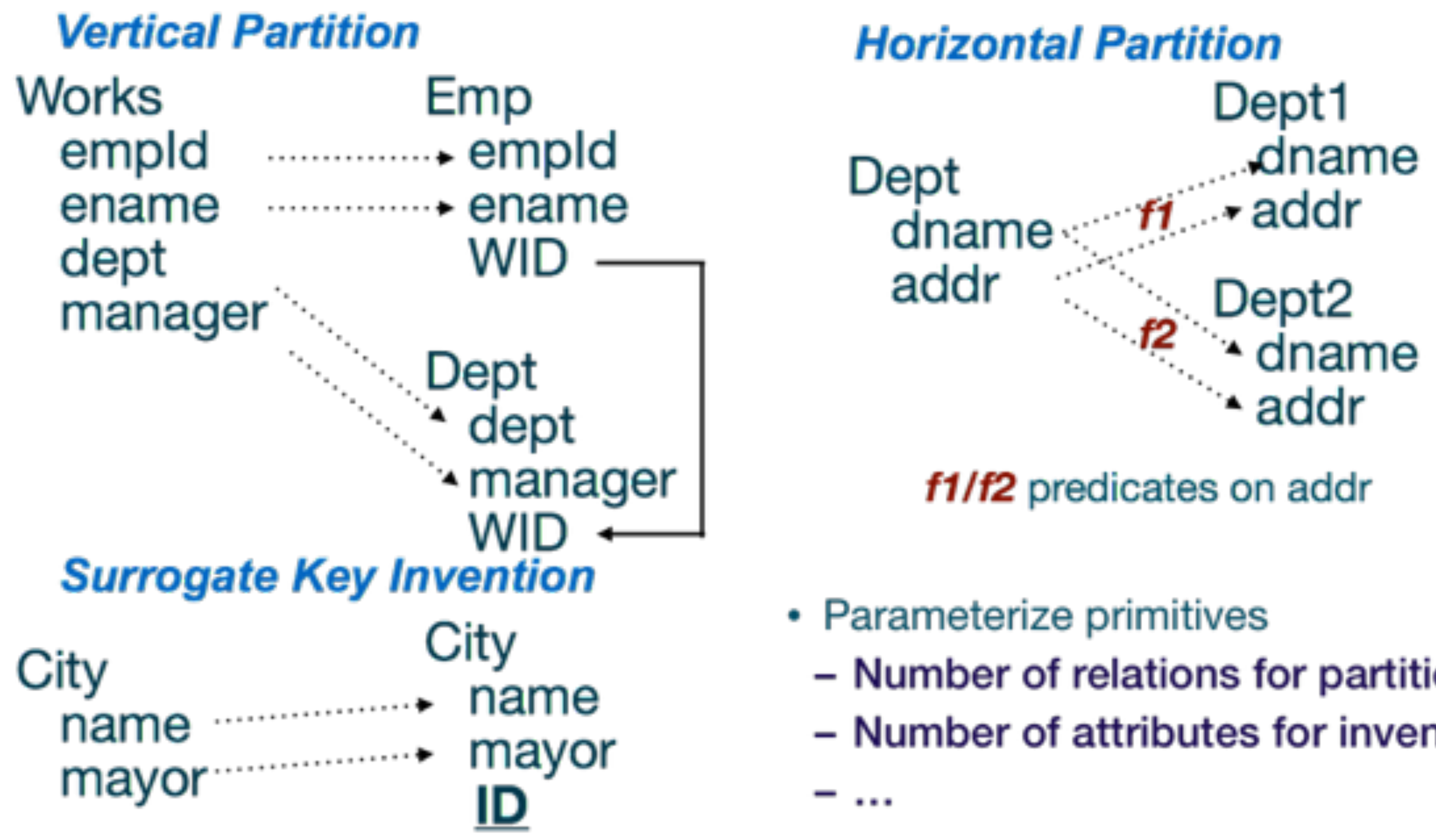
- **Example solution (mappings)**
  - `S1(A,B,C),S2(C,D,E) -> T(A,E)`
  - `S3(A,B,C,D),S4(E,A,B) -> ∃X,Y,Z T1(A,X,X),`
    `T2(A,Y,C),T3(C,B,Y,Z)`
- **Limited usefulness in practice**
  - Can we generate "realistic" scenarios?

Mapping Primitives

- **Mapping Primitives**
  - Template micro-scenarios that encode a typical schema mapping/evolution operations
    - Vertical partitioning a source relation
  - Used as building blocks for generating scenarios
- **Comprehensive Set of Primitives**
  - **Schema Evolution Primitives**
    - Mapping Adaptation [Yu, Popa VLDB05]
    - Mapping Composition [Bernstein et al. VLDBJ08]
  - **Schema Mapping Primitives**
    - *STBenchmark* [Alexe, Tan, Velegrakis PVLDB08]
      - First to propose parameterized primitives

Scenario Primitives

Example Mapping Primitives



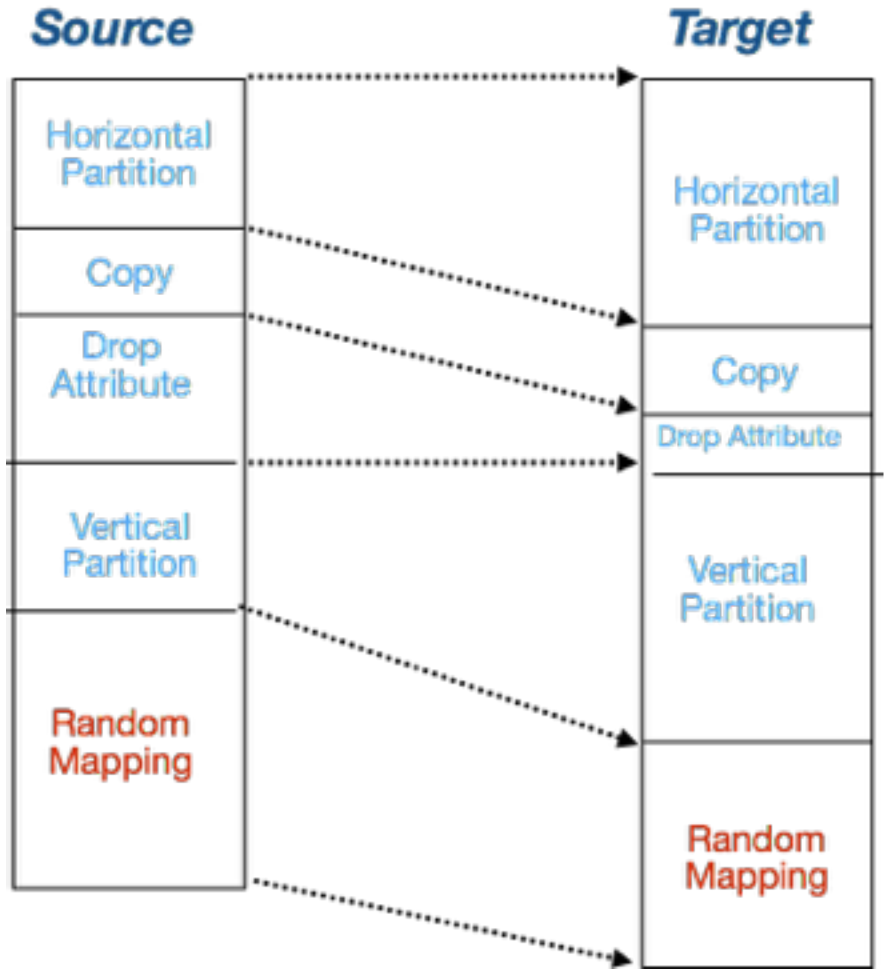
- Parameterize primitives
  - Number of relations for partitioning
  - Number of attributes for invention
  - …

29

Integration Scenario Generation

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Approach**
  - Start with empty integration scenario
  - Repeatedly add instances of primitives according to specs
  - If necessary add additional random mappings and schema elements

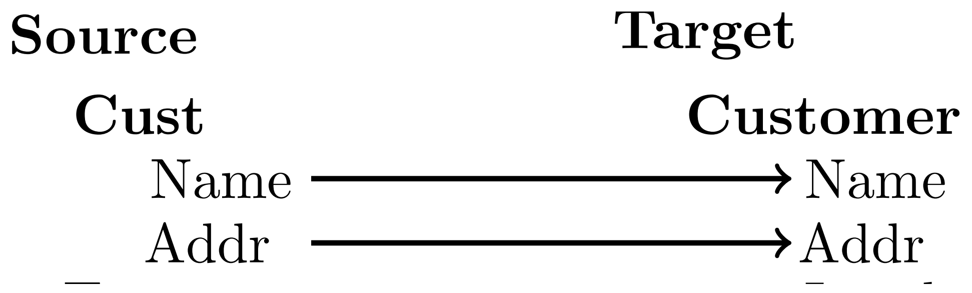


Primitive Generation

- Example Configuration
  - I want 1 copy and 1 vertical partitioning

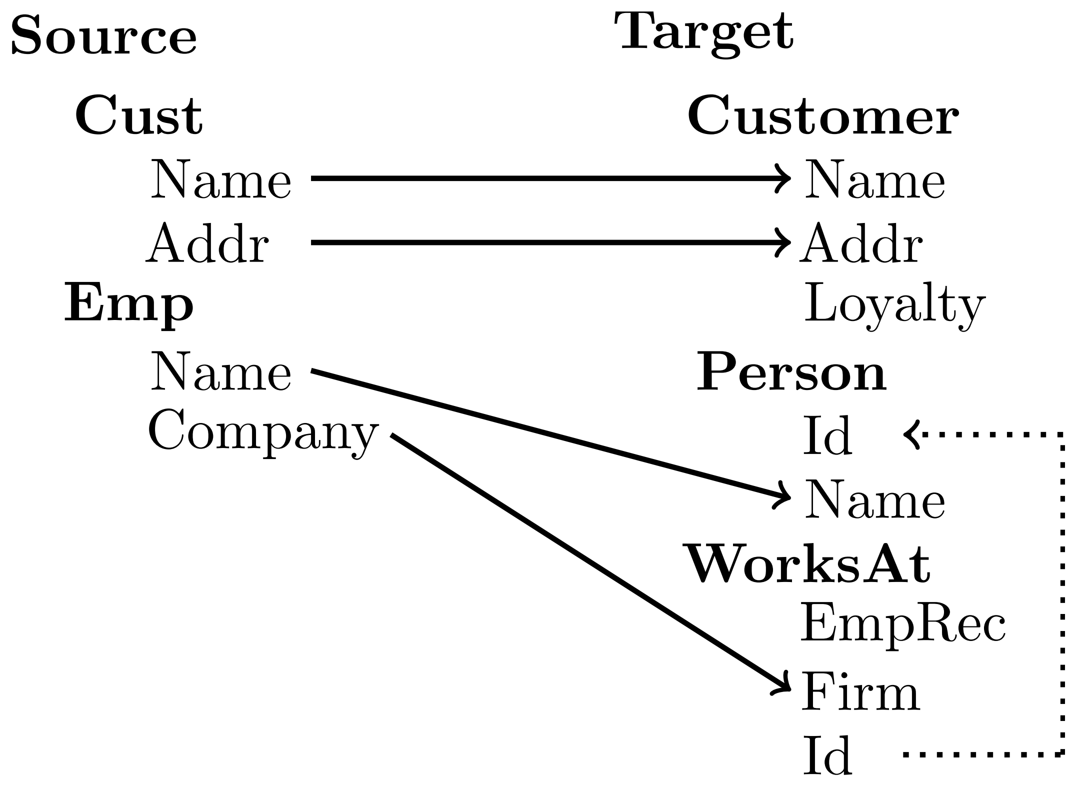
Primitive Generation

- Example Configuration
  - I want 1 copy and 1 vertical partitioning



Primitive Generation

- Example Configuration
  - I want 1 copy and 1 vertical partitioning

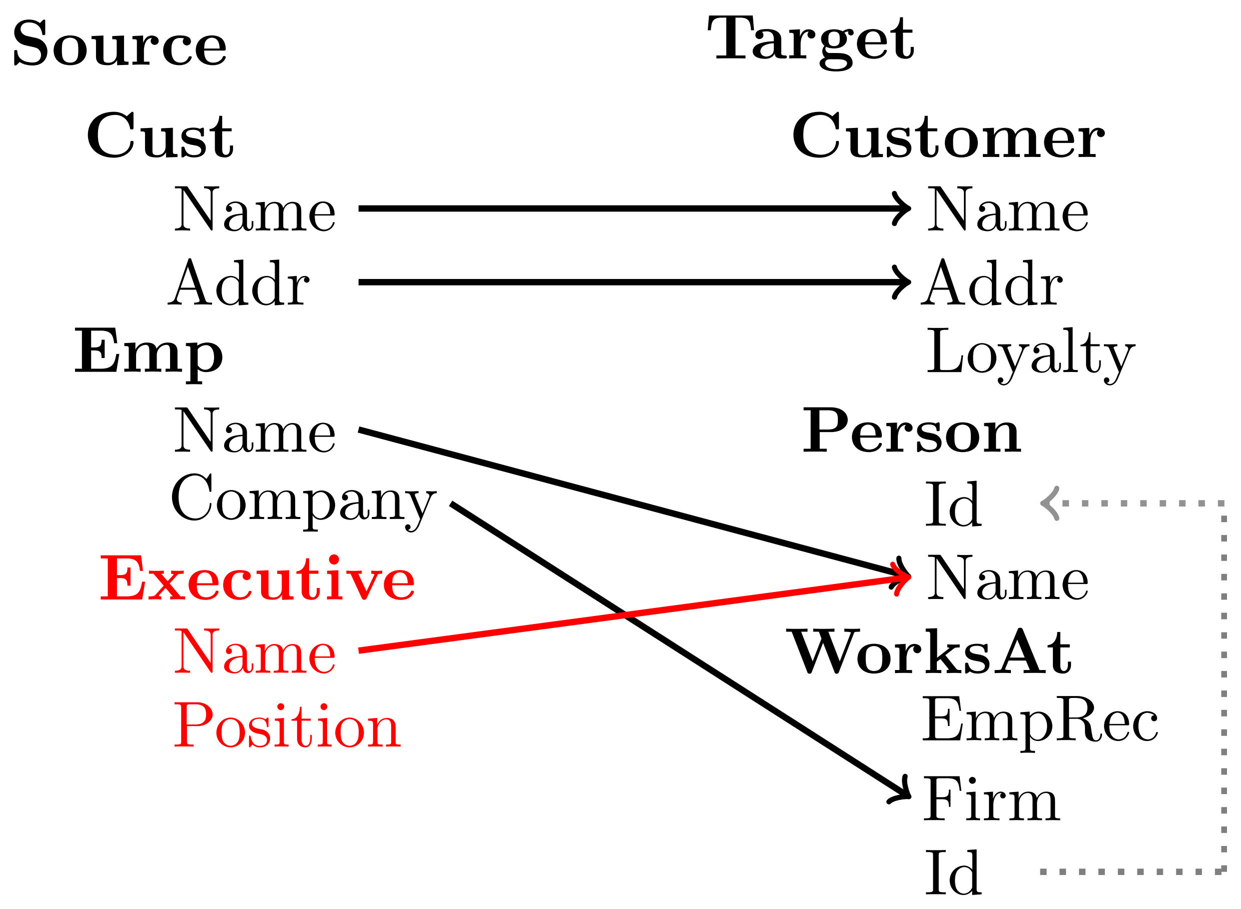


Sharing Schema Elements

- Sharing across primitives
  - Primitives cover many patterns that occur in the real world
  - however in the real world these primitives do not occur in isolation
- Enable primitives to share parts of the schema
  - Scenario parameters: *source reuse*, *target reuse*
  - Probabilistically determine whether to reuse previously generated relations

Sharing Schema Elements

- Example



User-defined Primitives

- Large number of integration scenarios have been shared by the community
  - Amalgam Test Suite (Bibliographic Schemas)
    - Four schemas - 12 possible mapping scenarios
  - Bio schemas originally used in Clio
    - Genomics Unified Schema GUS and BioSQL
  - Many others (see Bogdan Alexe's archive)
- **User defined primitive** (UDP)
  - User encodes scenario as iBench XML file
  - Such scenarios can then be declared as UDPs
    - Can be instantiated just like any build-in primitive

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

1) Empirical Evaluation of Integration Systems
2) iBench
3) **BART**
4) Success Stories
5) Demo
6) Conclusions and Future Work

Motivation

- **Evaluating constraint-based data cleaning algorithms**
  - Need dirty data (and gold standard)
  - Algorithms are sensitive to type of errors
- **Need a tool that**
  - Given a clean DB and set of constraints
  - Introduces errors that are **detectable** by the constraints
  - Provides control over how hard the errors are to repair (**repairability**)

Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

Overview

- **B**enchmarking **A**lgorithms for data **R**epairing and **T**ranslation

Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- **B**enchmarking **A**lgorithms for data **R**epairing and **T**ranslation
  - open-source error-generation system with an high level of control over the errors

Overview

- **B**enchmarking **A**lgorithms for data **R**epairing and **T**ranslation
  - open-source error-generation system with an high level of control over the errors
- **Input**: a clean database wrt a set of data-quality rules and a set of configuration parameters

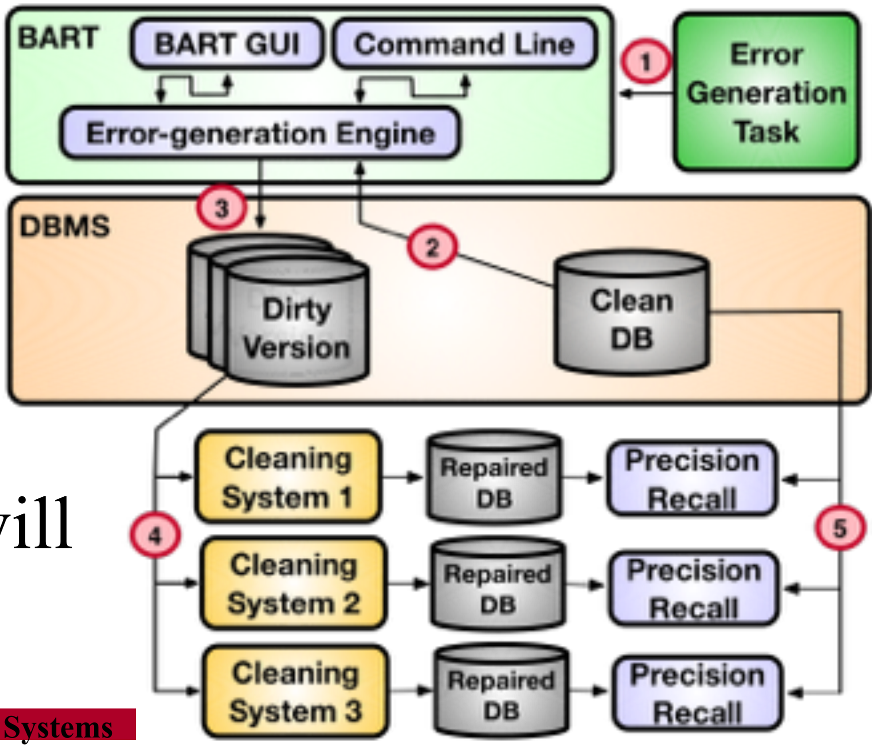
Overview

- **B**enchmarking **A**lgorithms for data **R**epairing and **T**ranslation
  - open-source error-generation system with an high level of control over the errors
- **Input**: a clean database wrt a set of data-quality rules and a set of configuration parameters
- **Output**: a dirty database (using a set of **cell changes**) and an estimate of how hard it will be to restore the original values

Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

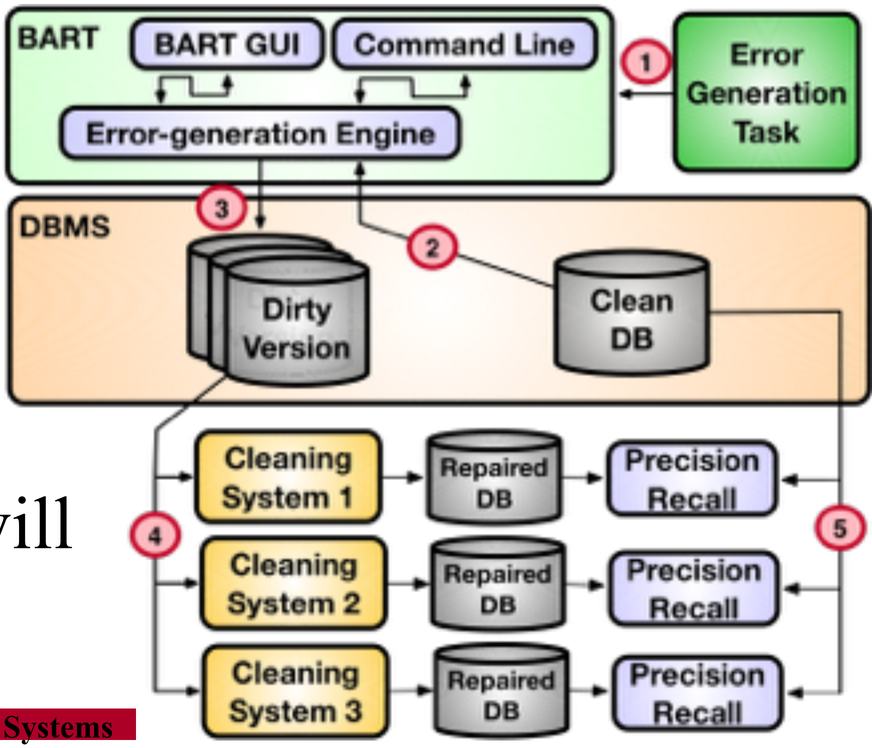
- **B**enchmarking **A**lgorithms for data **R**epairing and **T**ranslation
  - open-source error-generation system with an high level of control over the errors
- **Input**: a clean database wrt a set of data-quality rules and a set of configuration parameters
- **Output**: a dirty database (using a set of **cell changes**) and an estimate of how hard it will be to restore the original values



Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

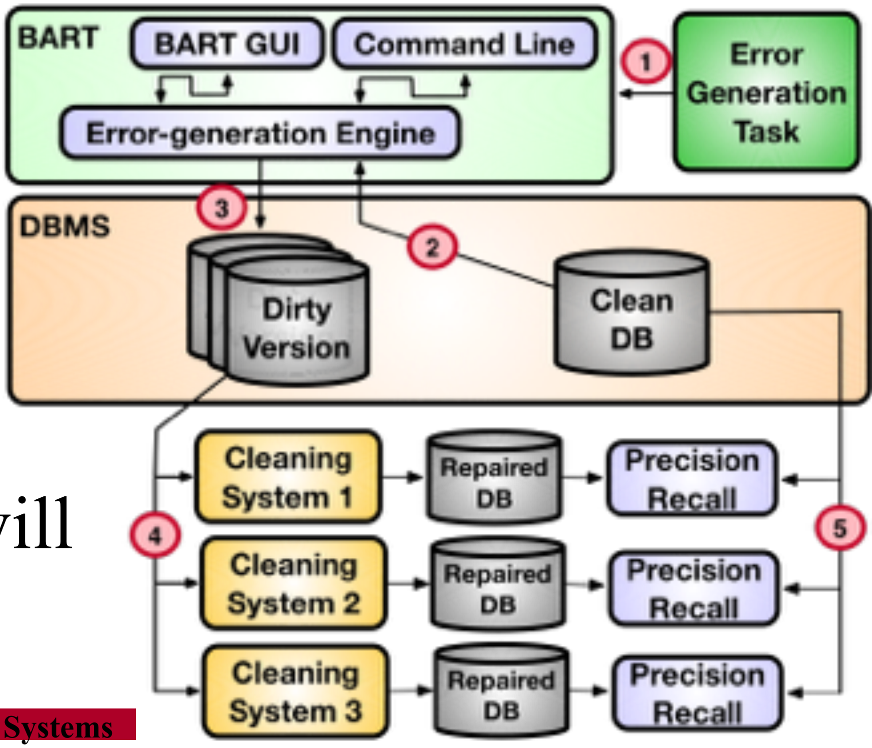
- **B**enchmarking **A**lgorithms for data **R**epairing and **T**ranslation
  - open-source error-generation system with an high level of control over the errors
- **Input**: a clean database wrt a set of data-quality rules and a set of configuration parameters
- **Output**: a dirty database (using a set of **cell changes**) and an estimate of how hard it will be to restore the original values



Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- **B**enchmarking **A**lgorithms for data **R**epairing and **T**ranslation
  - open-source error-generation system with an high level of control over the errors
- **Input**: a clean database wrt a set of data-quality rules and a set of configuration parameters
- **Output**: a dirty database (using a set of **cell changes**) and an estimate of how hard it will be to restore the original values



Creating Violations

ILLINOIS INSTITUTE OF TECHNOLOGY

- Constraint language: **denial constraints**
  - Subsumes FDs, CFDs, editing rules, …
- Update values of a cell to create a violation of a constraint
  - `t2.Team = 'Juventus'`

	Player				
	Name	Season	Team	Stadium	Goals
t1	Giovinco	2013-14	Juventus	Juventus Stadium	3
t2	Giovinco	2014-15	Toronto	BMO Field	23
t3	Pirlo	2014-15	Juventus	Juventus Stadium	5
t4	Pirlo	2015-16	N.Y. City	Yankee St.	0
t5	Vidal	2014-15	Juventus	Juventus Stadium	5
t6	Vidal	2015-16	Bayern	Allianz Arena	3

dc: ¬(Player(n, s, t, st, g), Player(n’, s’, t’, st’, g’), t=t’, st ≠ st’)

Challenges

- Error generation is an NP-complete problem
  - in the size of the DB
- How to identify cells to change efficiently?
- How to avoid interactions among introduced constraint violations?

Error Generation

- **Our approach**
  - **Sound, but not complete**
  - **Avoid interactions among cell changes**
    - Once we decide on a cell change to introduce a violation we exclude other cells involved in the violation from future changes
  - **Vio-Gen** queries
    - Derived from detection queries for denial constraints
    - Find cell to update such that the update is guaranteed to introduce a violation
    - Tuples that are almost in violation

dq: Player(n, s, t, st, g), Player(n', s', t', st', g'), t=t', st ≠ st'

vg: Player(n, s, t, st, g), Player(n', s', t', st', g'), t=t', **st = st'**

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

1) Empirical Evaluation of Integration Systems
2) iBench
3) BART
4) **Success Stories**
5) Demo
6) Conclusions and Future Work

Success Stories

- iBench has already been applied successfully by several diverse integration projects
- We have used iBench numerous times for our own evaluations
  - Our initial motivation for building iBench stemmed from our own evaluation needs

Value Invention

- **Translate mappings**
  - from expressive, less well-behaved language (SO tgds)
  - into less expressive, more well-behaved language (st-tgds)
- **Input**: schemas, integrity constraints, mappings
- **Output**: translated mappings (if possible)
- **Evaluation Goal**: how often do we succeed
- **Why iBench**: need a large number of diverse mappings to get meaningful results
- **Evaluation Approach**: generated 12.5 million integration scenarios based on randomly generated configuration file

Vagabond

- **Vagabond**
  - Finding explanations for data exchange errors
    - User marks attribute values in generated data as incorrect
    - System enumerates and ranks potential causes
- **Input**: schemas, integrity constraints, mappings, schema matches, data, **errors**
- **Output**: enumeration of causes or incremental ranking
- **Evaluation Goal**: evaluate scalability, quality
- **Why iBench**:
    - Control characteristics for scalability evaluation
    - Scale real-world examples

Mapping Discovery

- Learning mappings between schemas using statistical techniques
- **Input**: schemas, data, constraints
- **Output**: mappings
  - **University of California, Santa-Cruz**
    - Lise Getoor, Alex Memory
    - Reneé Miller
    - https://linqs.soe.ucsc.edu/people

And more …

- **Functional Dependencies Unleashed for Scalable Data Exchange**
  - [Bonifati, Ileana, Linardi - arXiv preprint arXiv:1602.00563, 2016]
  - Used iBench to compare a new chase-based data exchange algorithm to SQL-based exchange algorithm of ++Spicy
- **Approximation Algorithms for Schema-Mapping Discovery from Data**
  - [ten Cate, Kolaitis, Qian, Tan AMW 2015]
  - Approximate the Gottlob-Senellart notion
  - Kun Qian currently using iBench to evaluate effectiveness of approximation
- **Comparative Evaluation of Chase engines**
  - [Università della Basilicata, University of Oxford]
  - Using iBench to generate schemas, constraints

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

1) Empirical Evaluation of Integration Systems
2) iBench
3) BART
4) Success Stories
5) **Demo**
6) Conclusions and Future Work

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

1) Empirical Evaluation of Integration Systems
2) iBench
3) BART
4) Success Stories
5) Demo
6) **Conclusions and Future Work**

Conclusions

- **Empirical Evaluations of Integration Systems**
  - Need automated tools for robust, scalable, broad, repeatable evaluations
- **BART**
  - Controlled error generation
  - Detectable errors, measure repairability
- **iBench**
  - Comprehensive metadata generator
  - Produces inputs and outputs (gold standards) for a variety of integration tasks

Future Work

- **Data quality measures**
  - Implement complex quality measures
- **iBench**
  - More control over data generation
  - Orchestrating multiple mappings
    - Sequential: e.g., schema evolution
    - Parallel: e.g., virtual integration
- **BART**
  - Support combined mapping/cleaning scenarios
  - How to efficiently generate clean data (without having to run full cleaning algorithm)
  - Similarity measure for instances with labelled nulls/ variables

Questions?

ILLINOIS INSTITUTE OF TECHNOLOGY

- **iBench**

Webpage: http://dblab.cs.toronto.edu/project/iBench/

Code: https://bitbucket.org/ibencher/ibench/

Public Scenario Repo: https://bitbucket.org/ibencher/ibenchconfigurationsandscenarios

- **BART**

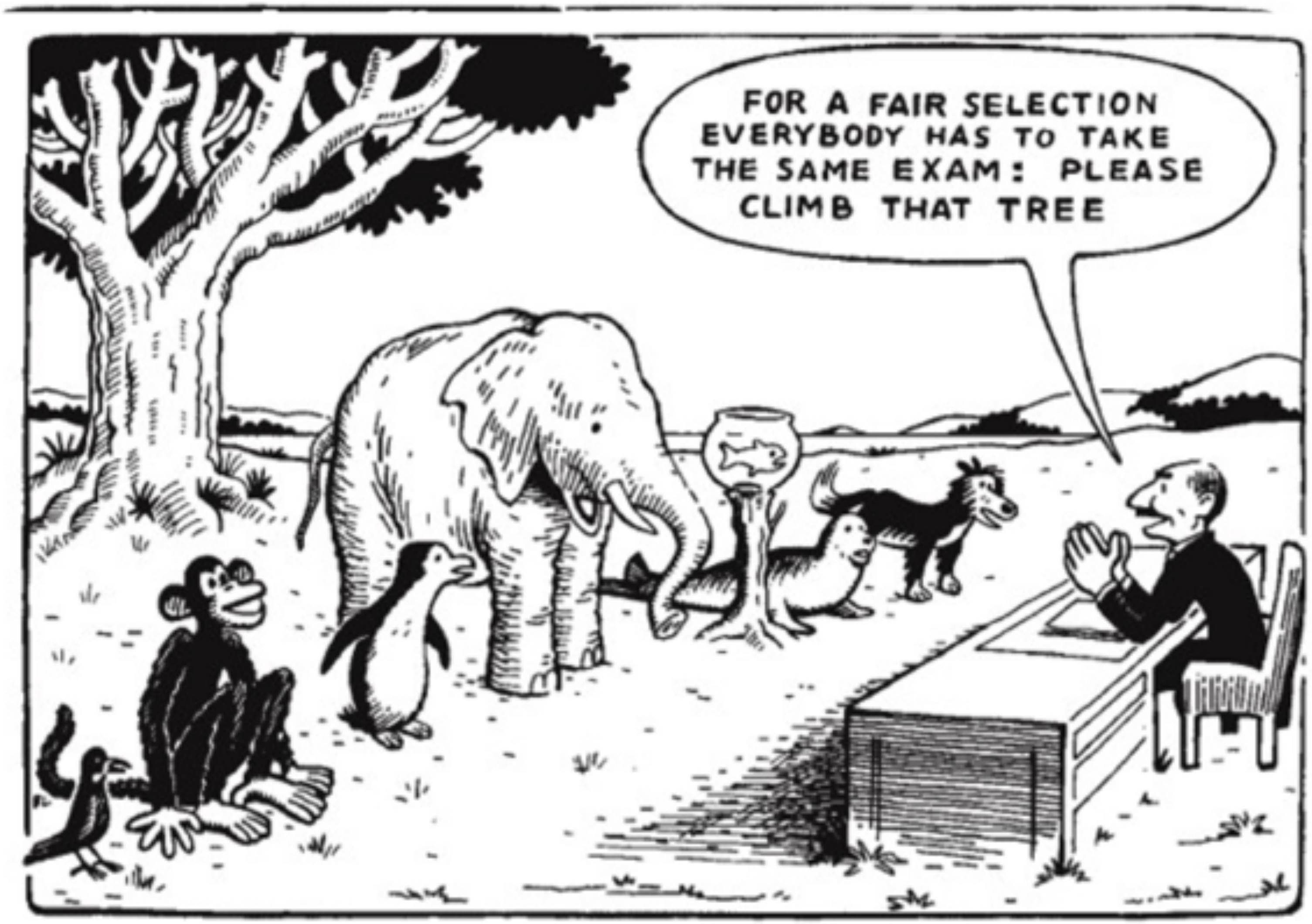
Webpage: http://www.db.unibas.it/projects/bart/

Code: https://github.com/dbunibas/BART

Example Datasets: http://www.db.unibas.it/projects/bart/files/BART-Datasets.zip

Questions?

ILLINOIS INSTITUTE OF TECHNOLOGY


FOR A FAIR SELECTION EVERYBODY HAS TO TAKE THE SAME EXAM: PLEASE CLIMB THAT TREE