Extending Dependencies with Conditions

Loreto Bravo University of Edinburgh

Wenfei Fan University of Edinburgh & Bell Laboratories

Shuai Ma University of Edinburgh

Lucent Technologies
Bell Labs Innovations

Outline

- Why Conditional Dependencies?
  - ✓ Data Cleaning
  - ✓ Schema Matching
- Conditional Inclusion Dependencies (CINDs)
  - ✓ Definition
  - ✓ Static Analysis
    - Satisfiability Problem
    - Implication Problem
    - Inference System
  - ✓ Static Analysis of CFDs+CINDs
- Satisfiability Checking Algorithms (CFDs+CINDs)
- Summary and Future Work

Motivation

- Data Cleaning
  - Real life data is dirty!
  - Specify consistency using integrity constraints
    - Inconsistencies emerge as violations of constraints
  - Constraints considered so far: traditional
    - Functional Dependencies - FD
    - Inclusion Dependencies - IND
    - . . .
- Schema matching: needed for data exchange and data integration
  - Pairings between semantically related source schema attributes and target schema attributes
  - expressed as inclusion dependencies (e.g., Clio)

Example: Amazon database

- Schema:
  - ✓ book(id, isbn, title, price, format)
  - ✓ CD(id, title, price, genre)
  - ✓ order(id, title, type, price, country, county)

book

id	isbn	title	price
a23	b32	H. Porter	17.99
a56	b65	Snow white	7.94

CD

id	title	price	genre
a12	J. Denver	17.99	country
a56	Snow White	7.94	a-book

order

id	title	type	price	country	county
a23	H. Porter	book	17.99	US	DL
a12	J. Denver	CD	7.94	UK	Reyden

Data cleaning with inclusion dependencies

- Definition of Inclusion Dependencies (INDs)
  - ✓ R1[X] ⊆ R2[Y], for any tuple t1 in R1, there must exist a tuple t2 in R2, such that t2[Y]=t1[X]
- Example Inclusion dependency:
  - ✓ **book[id, title, price] ⊆ order[id, title, price]**

book

id	isbn	title	price	format	
a23	b32	H. Porter	17.99	Hard cover	t3
a56	b65	Snow White	17.94	audio	t4

order

id	title	type	price	country	county	
a23	H. Porter	book	17.99	US	DL	t1
a12	J. Denver	CD	7.94	UK	Reyden	t2

Data cleaning meets conditions

- How to express?
  - ✓ Every book in order table must also appear in book table
- Traditional inclusion dependencies:
  - ✓ **order[id, title, price] $\subseteq$ book[id, title, price]**

order

id	title	type	price	country	county	
a23	H. Porter	book	17.99	US	DL	t1
a12	J. Denver	CD	7.94	UK	Reyden	t2

book

id	isbn	title	price	format	
a23	b32	H. Porter	17.99	Hard cover	t3
a56	b65	Snow White	17.94	audio	t4

This inclusion dependency does not make sense!

Data cleaning meets conditions

order

id	title	type	price	country	county	
a23	H. Porter	book	17.99	US	DL	t1
a12	J. Denver	CD	7.94	UK	Reyden	t2

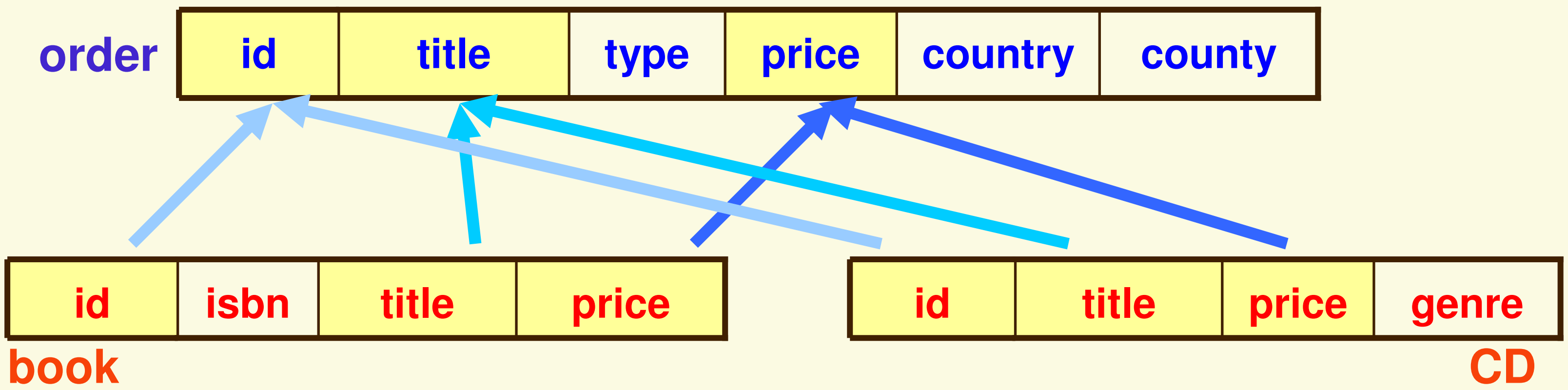
book

id	isbn	title	price	format	
a23	b32	H. Porter	17.99	Hard cover	t3
a56	b65	Snow White	17.94	audio	t4

- Conditional inclusion dependency:
  - ✓ **order[id, title, price, type =‘ book’] $\subseteq$ book[id, title, price]**

Schema matching with inclusion dependencies

- **Schema Matching:**
  - ✓ Pairings between semantically related source schema attributes and target schema attributes, which are de facto inclusion dependencies from source to target (e.g., Clio)

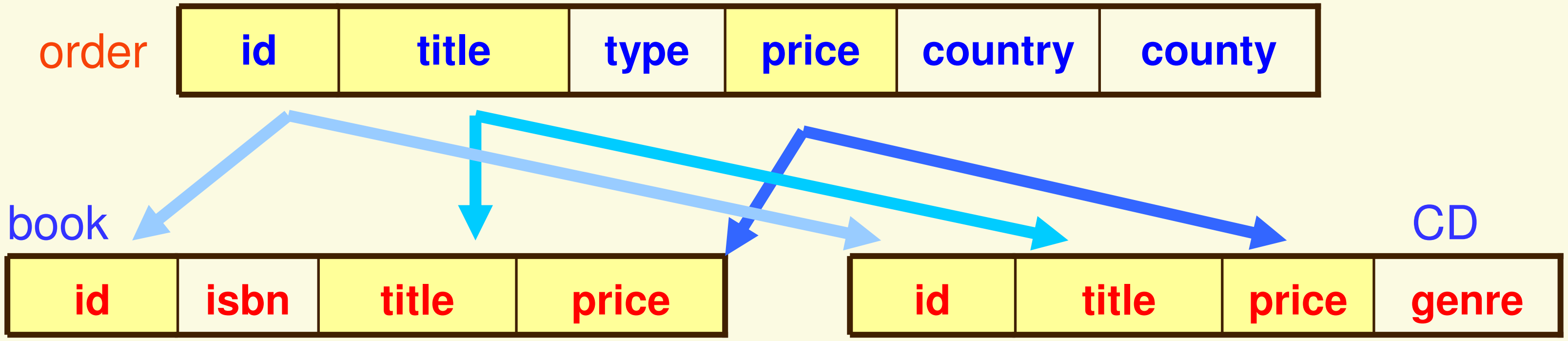


- Traditional inclusion dependencies:

book[id, title, price] $\subseteq$ order[id, title, price]

CD[id, title, price] $\subseteq$ order[id, title, price]

Schema matching meets conditions



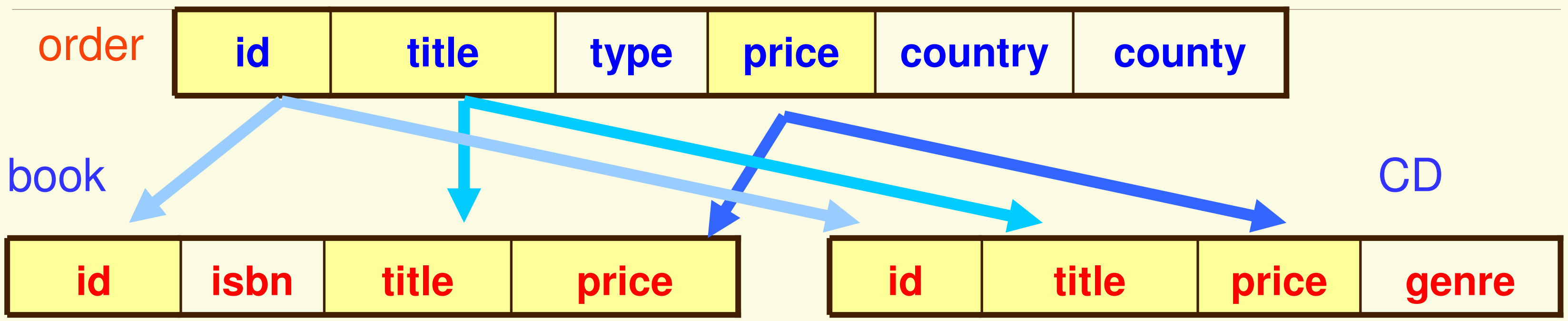
- Traditional inclusion dependencies:

  order[id, title, price] $\subseteq$ book[id, title, price]

  order[id, title, price] $\subseteq$ CD[id, title, price]

These inclusion dependencies do not make sense!

Schema matching meets conditions



Conditional inclusion dependencies:

order[id, title, price; type =‘ book’] ⊆ book[id, title, price]

order[id, title, price; type = ‘CD’] ⊆ CD[id, title, price]

The constraints do not hold on the entire order table

- order[id, title, price] ⊆ book[id, title, price] holds only if type = ‘book’
- order[id, title, price] ⊆ CD[id, title, price] holds only if type = ‘CD’

Conditional Inclusion Dependencies (CINDs)

- **(R1[X; Xp] ⊆ R2[Y; Yp], Tp):**
  - ✓ R1[X] ⊆ R2[Y]: embedded traditional IND from R1 to R2
  - ✓ attributes: X ∪ Xp ∪ Y ∪ Yp
  - ✓ Tp: a pattern tableau
  - ✓ tuples in Tp consist of constants and unnamed variable _
- **Example:**

CD[id, title, price; genre = 'a-book'] ⊆book[id, title, price; format = 'audio']

- **Corresponding CIND:**
  - ✓ (**CD[id, title, price; genre] ⊆book[id, title, price; format],** Tp)

Tp

id	title	price	genre	id	title	price	format
_	_	_	a-book	_	_	_	audio

INDs as a special case of CINDs

R1[X] ⊆ R2[Y]

- X: [A1, …, An]
- Y : [B1, …, Bn]

As a CIND: (R1[X; nil] ⊆ R2[Y; nil], Tp)

- pattern tableau Tp: a single tuple consisting of _ only

A1	…	An	B1	…	Bn
_	_	_	_	_	_

CINDs subsume traditional INDs

Static Analysis of CINDs

- Satisfiability problem $I \models \Sigma$
  - INPUT: Give a set Σ of constraints
  - Question: Does there exist a nonempty instance I satisfying Σ?
- Whether Σ itself is dirty or not
- For INDs the problem is trivially true
- For CFDs (to be seen shortly) it is NP-complete
- Good news for CINDs

Proposition: Any set of CINDs is always satisfiable

Static Analysis of CINDs

- Implication problem $\Sigma \models \varphi$
  - ✓ INPUT: set Σ of constraints and a single constraint φ
  - ✓ Question: for each instance I that satisfies Σ, **does** I also satisfy φ?
- Remove redundant constraints
- PSPACE-complete for traditional inclusion dependencies

Theorem. Complexity bounds for CINDs

- ✓ Presence of constants
- ✓ PSPACE-complete in the absence of finite domain attributes
  - Good news – The same as INDs
- ✓ EXPTIME-complete in the general setting

Finite axiomatizability of CINDs

- φ is implied by Σ iff it can be computed by the inference system
  - ✓ INDs have such Inference System
  - ✓ Good news: CINDs too!

1-Reflexivity
2-Projection and Permutation
3-Transitivity

IND Counterparts

4-Downgrading
5-Augmentation
6-Reduction

Sound and Complete in the Absence of Finite Attributes

7-F-reduction
8-F-upgrade

Finite Domain Attributes

Theorem. The above eight rules constitute a sound and complete inference system for implication analysis of CINDs

Axioms for CINDs: finite domain reduction

- New CINDs can be inferred by axioms
- (R1[X; A] ⊆ R2[Y; Yp], Tp),
  - ✓ dom(A) = { true, false}

Tp

X	A	Y	Yp	
_	true	_	d	tp1
_	false	_	d	tp2

⇩

then (R1[X; Xp] ⊆ R2[Y; Yp], tp),

X	Y	Yp
_	_	d

Static analyses: CIND vs. IND

✓ **In the absence of finite-domain attributes:**

	satisfiability	implication	finite axiom'ty
CIND	O(1)	PSPACE-complete	yes
IND	O(1)	PSPACE-complete	yes

✓**General setting with finite-domain attributes:**

	satisfiability	implication	finite axiom'ty
CIND	O(1)	EXPTIME-complete	yes
IND	O(1)	PSPACE-complete	yes

CINDs retain most complexity bounds of their traditional counterpart

Conditional Functional Dependencies (CFDs)

An extension of traditional FDs

Example: **cust([country = 44, zip] → [street])**

Name	**country**	**zip**	**street**
Bob	44	07974	Tree Ave.
Joe	44	07974	Tree Ave.
Ben	01	01202	Elem Str.
Jim	01	01202	Oak Ave.

Static analyses: CFD + CIND vs. FD + IND

	satisfiability	implication	finite axiom'ty
CFD + CIND	undecidable	undecidable	No
FD + IND	O(1)	undecidable	No

- ✓ CINDs and CFDs properly subsume FDs and INDs
- ✓ Both the satisfiability analysis and implication analysis are

  beyond reach in practice

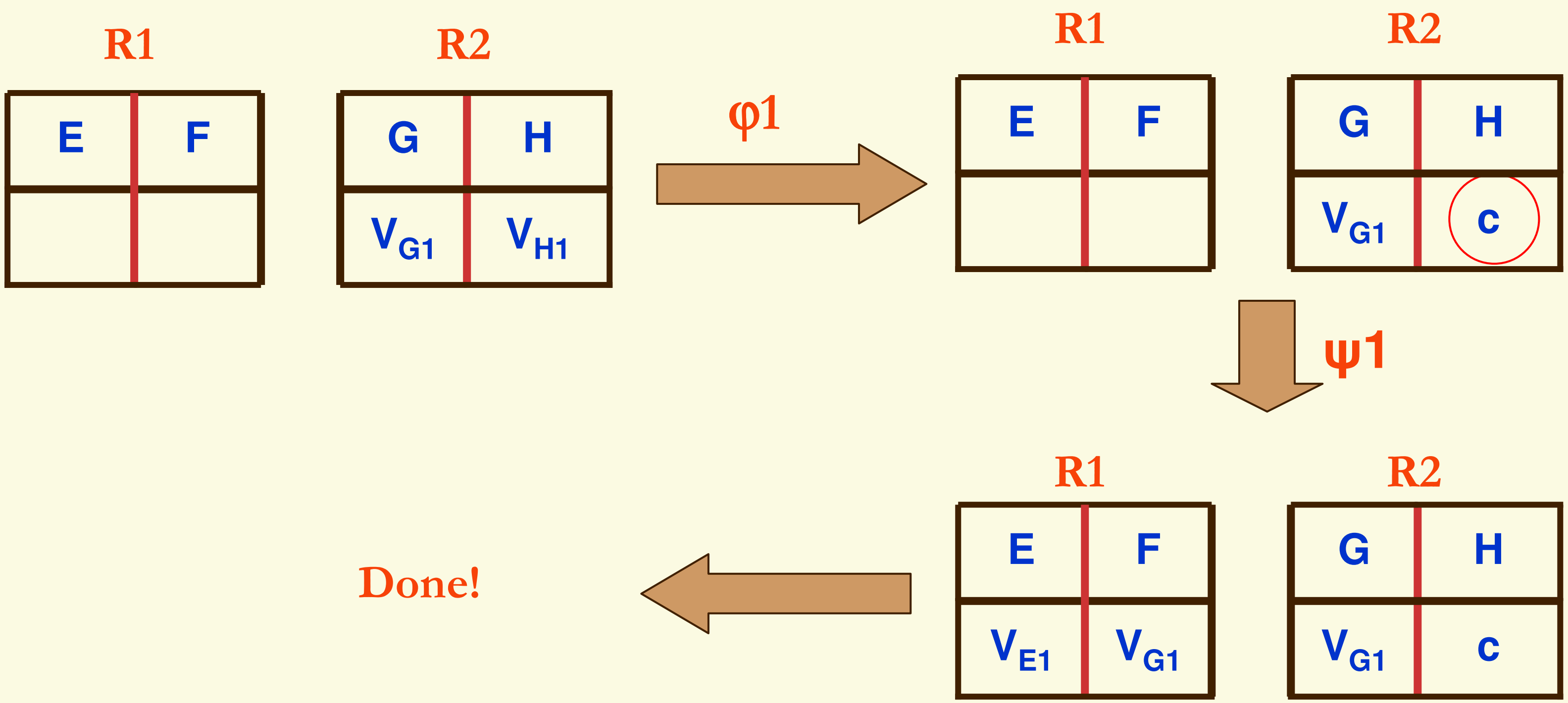
  This calls for effective heuristic methods

Satisfiability Checking Algorithms

- Before using a set of CINDs for data cleaning or schema matching we need to make sure that they make sense (that they are clean)
- We need to find heuristics to solve the satisfiability problem
  - ✓ Input: A set **Σ** of CFDs and CINDs
  - ✓ Output: true / false
- We modified and extended techniques used for FDs and INDs
  - ✓ For example: **Chase,** to build a "**canonical**" witness instance, i.e., $I \models \Sigma$

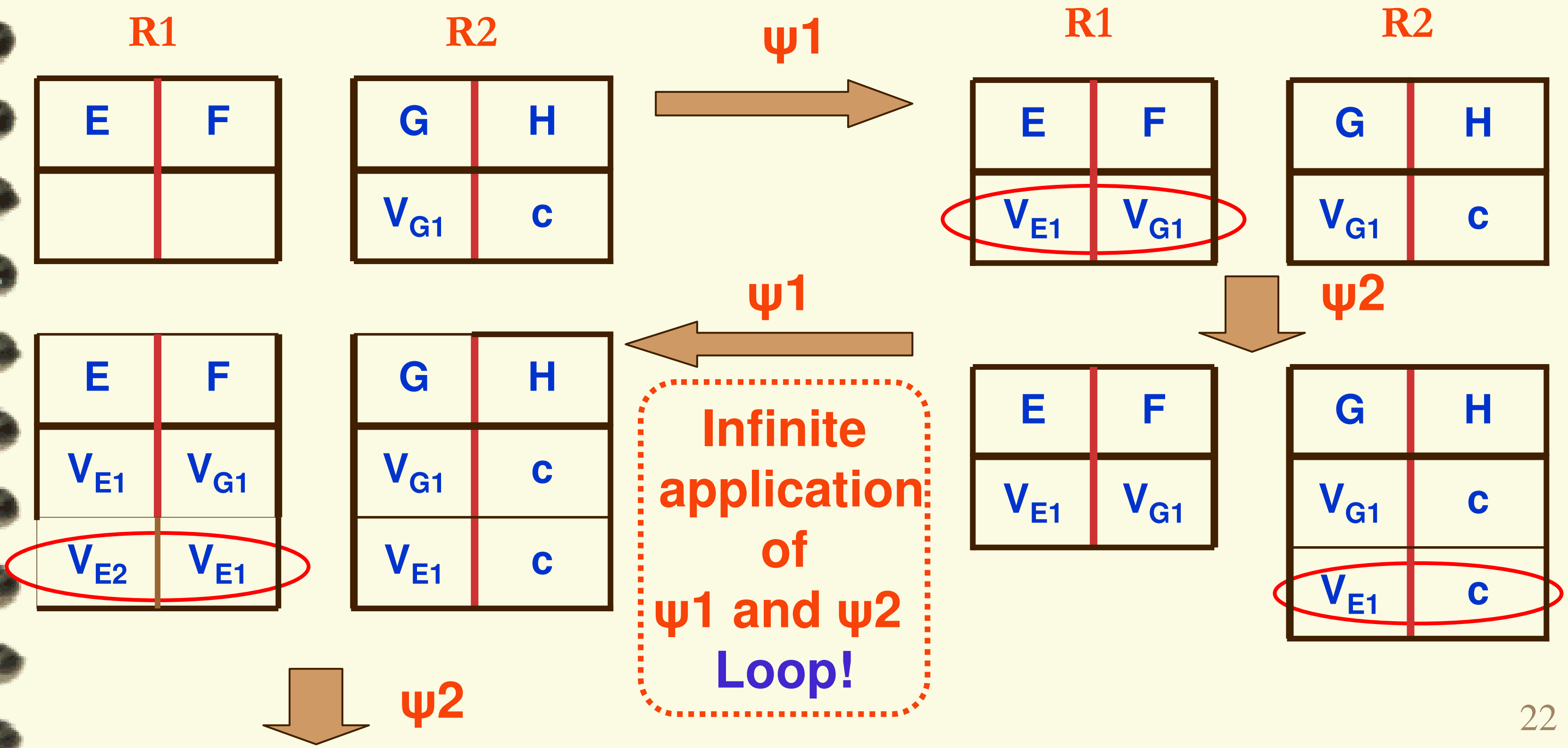
$Chase_{CFDs+CINDs}$ – Terminate case

- Σ = {φ1, ψ1}
  - ✓ φ1=(**R2(G → H)**, (_ || c)) - CFD
  - ✓ ψ1=(**R2[G; nil] ⊆ R1[F; nil]**, (_ || _)) - CIND



Chase$_{CFDs+CINDs}$ – Loop case

- Σ = {φ1, ψ1, ψ2}
  - ✓ φ1=(**R2(G → H)**, (_ || c)) - CFD
  - ✓ ψ1=(**R2[G; nil] ⊆ R1[F; nil]**, (_ || _)) - CIND
  - ✓ ψ2=(**R1[E; nil] ⊆ R2[G; nil]**, (_ || _))



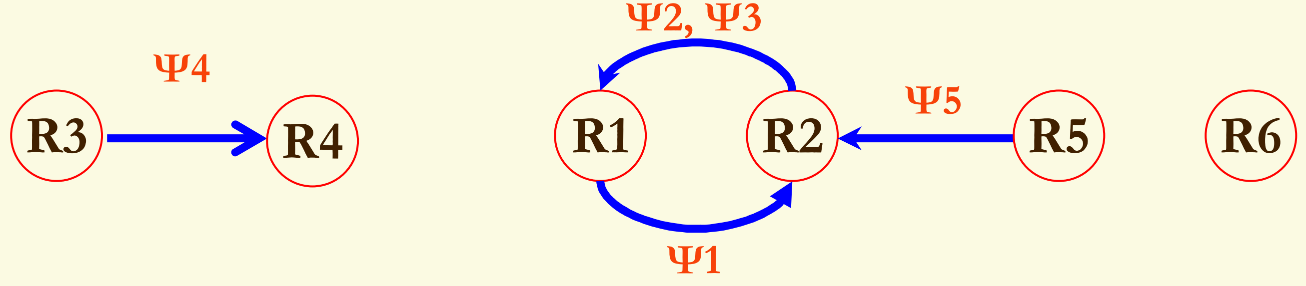
More about the checking algorithms

- Simplification of the chase:
  - ✓ The fresh variables are taken from a finite set
  - ✓ We avoid the infinite loop of the chase by limiting the size of the witness instance
- If the algorithm returns:
  - True: we know the constraints are satisfiable
  - False: there may be false negative answers – the problem is undecidable and the best we can get is a heuristic
- In order to improve accuracy of the algorithm we use:
  - ✓ Optimization techniques

Example optimization techniques

Node(Relation): related to CFDs

Edge: related to CINDS



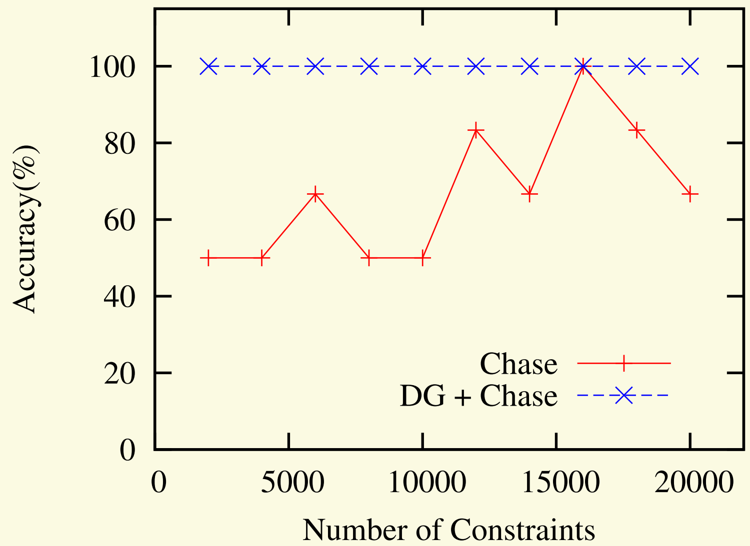
Unsatisfiability Propagation

- IF
  - ✓ CFDs on R4 is unsatisfiable
  - ✓ There is a CIND **Ψ4:** (R3[X; nil] $\subseteq$ R4[Y; Yp], tp)
- THEN
  - ✓ R3 must be empty!

CFDs+CINDs satisfiability checking - experiments

- Experimental Settings
  - ✓ Accuracy tested for satisfiable sets of CFDs and CINDs
    - The data sets where generated by ensuring the existence of a witness database that satisfies them
  - ✓ Scalability tested for random sets of CFDs and CINDs
  - ✓ Each experiment was run 6 times and the average is reported
  - ✓ # of constraints: up to 20,000
  - ✓ # of relations: up to 100
  - ✓ Ratio of finite attributes: up to 25%
  - ✓ An Intel Pentium D 3.00GHz with 1GB memory

CFDs+CINDs satisfiability checking - experiments

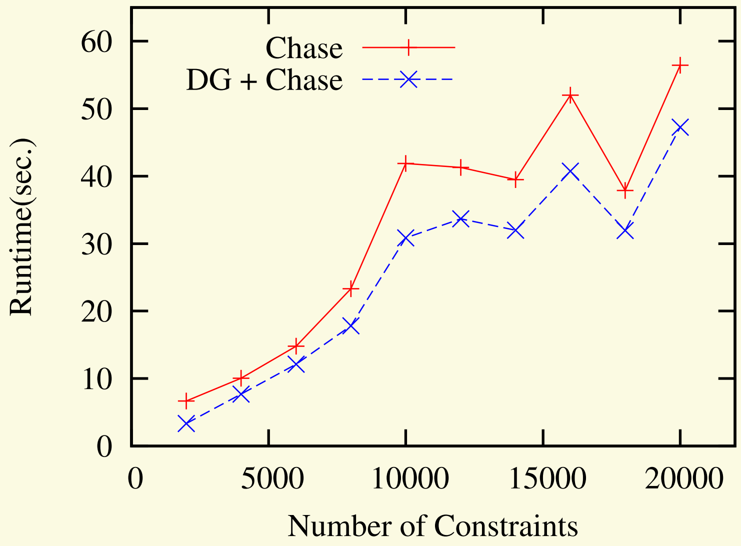


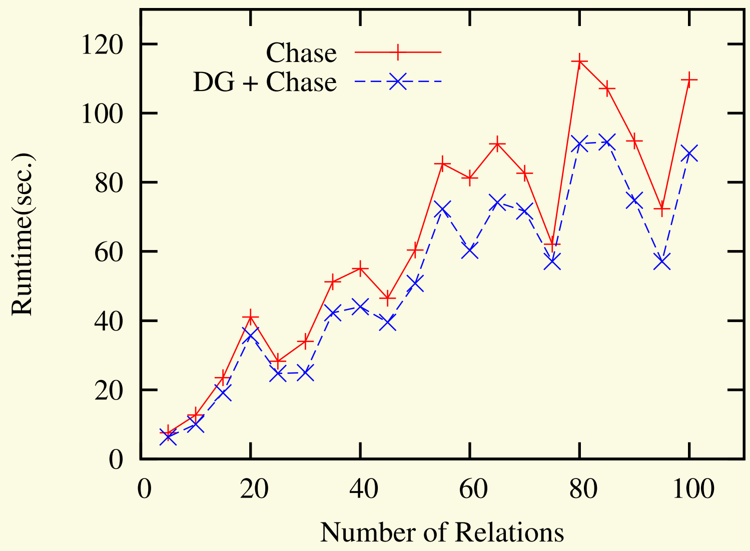
Algorithm:

1. **Chase : modified version Chase**
2. **DG+Chase: graph optimization based Chase**

Accuracy testing is based satisfiable sets of CFDs and CINDs

CFDs+CINDs satisfiability checking - experiments





Scalability testing is based on random sets of CFDs and CINDs

Summary and future work

- New constraints: conditional inclusion dependencies
  - for both data cleaning and schema matching
  - complexity bounds of satisfiability and implication analyses
  - a sound and complete inference system
- Complexity bounds for CFDs and CINDs taken together
- Heuristic satisfiability checking algorithms for CFDs and CINDs
- Open research issues:
  - Deriving schema mapping from the constraints
  - Repairing dirty data based on CFDs + CINDs
  - Discovering CFDs + CINDs

Towards a practical method for data cleaning and schema matching