CPT-S 415

Big Data

Yinghui Wu

EME B45

CPT-S 415
Big Data

Dependencies for improving data quality

- ✓ Conditional functional dependencies (CFDs)
  - Syntax and semantics
- ✓ Conditional inclusion dependencies (CINDs)
  - Syntax and semantics
- ✓ Matching dependencies for record matching (MDs)
  - Syntax and semantics

Characterizing the consistency of data

- ✓ One of the central technical problems for data consistency is how to tell whether the data is dirty or clean
- ✓ Integrity constraints (data dependencies) as data quality rules

  Inconsistencies emerge as violations of constraints
- ✓ Traditional dependencies:
  - functional dependencies
  - inclusion dependencies
  - denial constraints (a special case of full dependencies)
  - . . .

  Question: are these traditional dependencies sufficient?

Example: customer relation

✓ Schema: Cust(country, area-code, phone, street, city, zip)

✓ Instance:

country	area-code	phone	street	city	zip
44	131	1234567	Mayfield	NYC	EH4 8LE
44	131	3456789	Crichton	NYC	EH4 8LE
01	908	3456789	Mountain Ave	NYC	07974

✓ functional dependencies (FDs):

cust[country, area-code, phone] → cust[street, city, zip]

cust[country, area-code]   → cust[city]

The database satisfies the FDs. Is the data consistent?

Capturing inconsistencies in the data

- ✓ **cust ([country = 44, zip] → [street])**

  In the UK, zip code uniquely determines the street

  The constraint may not hold for other countries
- ✓ It expresses a fundamental part of the semantics of the data
- ✓ It can NOT be expressed as a traditional FD
  - It does not hold on the entire relation; instead, it holds on tuples representing UK customers only

country	area-code	phone	street	city	zip
44	131	1234567	Mayfield	NYC	EH4 8LE
44	131	3456789	Crichton	NYC	EH4 8LE
01	908	3456789	Mountain Ave	NYC	07974

Two more constraints

cust([country = 44, area-code = 131, phone] → [street, zip, city = EDI])

cust([country = 01, area-code = 908, phone] → [street, zip, city = MH])

- In the UK, if the area code is 131, then the city has to be EDI
- In the US, if the area code is 908, then the city has to be MH

✓ t1, t2 and t3 violate these constraints

- refining **cust([country, area-code, phno] → [street, city, zip])**
- combining data values and variables

id	country	Area-code	phone	street	city	zip
t1	44	131	1234567	Mayfield	NYC	EH4 8LE
t2	44	131	3456789	Crichton	NYC	EH4 8LE
t3	01	908	3456789	Mountain Ave	NYC	07974

The need for new constraints

cust([country = 44, zip] → [street])

cust([country = 44, area-code = 131, phone] → [street, zip, city = EDI])

cust([country = 01, area-code = 908, phone] → [street, zip, city = MH])

- ✓ They capture inconsistencies that traditional FDs cannot detect

  Traditional constraints were developed for schema design, not for data cleaning!

- ✓ Data integration in real-life: source constraints
  - – hold on a subset of sources
  - – hold conditionally on the integrated data
- ✓ They are NOT expressible as traditional FDs
  - – do not hold on the entire relation
  - – contain constant data values, besides logical variables

Conditional Functional Dependencies (CFDs)

An extension of traditional FDs: (R: $X \to Y$, Tp)

- ✓ $X \to Y$: embedded traditional FD on R
- ✓ Tp: a pattern tableau
  - – attributes: $X \cup Y$
  - – each tuple in Tp consists of constants and unnamed variable _

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

- ✓ (cust (country, zip → street), Tp)
- ✓ pattern tableau Tp

country	zip	street
44	_	_

Example CFDs

cust([country = 44, area-code = 131, phone] → [street, zip, city = EDI])

cust([country = 01, area-code = 908, phone] → [street, zip, city = MH])

cust([country, area-code, phone] → [street, city, zip])

as a SINGLE CFD:

- ✓ **(cust(country, area-code, phone → street, city, zip**), Tp)
- ✓ pattern tableau Tp: one tuple for each constraint

country	area-code	phone	street	city	zip
44	131	_	_	Edi	_
01	908	_	_	MH	_
_	_	_	_	_	_

CFDs subsume traditional FDs. Why?

Traditional FDs as a special case

Express

cust[country, area-code] → **cust[city]**

as a CFD:

- ✓ **(cust(country, area-code, → city**), Tp)
- ✓ pattern tableau Tp: a single tuple consisting of _ only

country	area-code	city
_	_	_

Semantics of CFDs

- ✓ a ≈ b (a matches b) if
  - – either a or b is _
  - – both a and b are constants and a = b
- ✓ tuple t1 matches t2: t1 ≈ t2

  (a, b) ≈ (a, _), but (a, b) does not match (a, c)
- ✓ DB satisfies (R: X → Y, Tp) iff for any tuple tp in the pattern tableau Tp and for any tuples t1, t2 in DB, if t1[X] = t2[X] ≈ tp[X], then t1[Y] = t2[Y] ≈ tp[Y]
  - – tp[X]: identifying the set of tuples on which the constraint tp applies, ie, { t | t[X] ≈ tp[X]}
  - – t1[Y] = t2[Y] ≈ tp[Y]: enforcing the embedded FD, and the pattern of tp

Example: violation of CFDs

cust([country = 44, zip] → [street])

country	zip	street
44	_	_

Tuples t1 and t2 violate the CFD

- ✓ t1[country, zip] = t2[country, zip] ≈ tp[country, zip]
- ✓ t1[street] ≠ t2[street]

The CFD applies to t1 and t2 since they match tp[country, zip]

id	country	area-code	phone	street	city	zip
t1	44	131	1234567	Mayfield	NYC	EH8 8LE
t2	44	131	3456789	Crichton	NYC	EH8 8LE
t3	01	908	3456789	Mountain Ave	NYC	07974

CFDs: enforcing binding of semantically related data values

Violation of CFDs by a single tuple

(cust(country, area-code → city), Tp)

id	country	area-code	city
tp1	44	131	Edi
tp2	01	908	MH
tp3	_	_	_

Tuple t1 does not satisfy the CFD

- ✓ t1[country, area-code] = t1[country, area-code] ≈ tp1[country, area-code]
- ✓ t1[city] = t1[city]; however, t1[city] does not match tp1[city]

In contrast to traditional FDs, a single tuple may violate a CFD

id	country	area-code	phone	street	city	zip
t1	44	131	1234567	Mayfield	NYC	EH8 8LE
t2	44	131	3456789	Crichton	NYC	EH8 8LE
t3	01	908	3456789	Mountain Ave	NYC	07974

Exercise

(cust(country, area-code, phno → street, city, zip), Tp)

id	country	area-code	phon	street	city	zip
tp1	44	131	_	_	Edi	_
tp2	01	908	_	_	MH	_
tp3	_	_	_	_	_	_

- ✓ Tuple t3 violates the CFD. Why?
- ✓ Tuples t1 and t4 violate the CFD. Why?

id	country	area-code	phon	street	city	zip
t1	44	131	1234567	Mayfield	Edi	EH4 8LE
t2	44	131	3456789	Mayfield	NYC	19082
t3	01	908	3456789	Mountain Ave	NYC	19082
t4	44	131	1234567	Chrichton	EDI	EH8 9LE

“Dirty” constraints?

A set of CFDs may be inconsistent!

✓ Inconsistent: **(R(A → B**), Tp)

Tp

id	A	B
tp1	_	b
tp2	_	c

In any nonempty database DB and for any tuple t in DB,

– tp1: t[B] must be b

– tp2: t[B] must be c

– Inconsistent if b and c are different

✓ inconsistent $\Sigma = \{ \varphi1, \varphi2 \}$, $\varphi1$ = **(R(A → B**), Tp1), $\varphi2$ = **(R(B → A**), Tp2)

id	A	B
tp1	true	b
tp2	false	c

id	B	A
tp3	b	false
tp4	c	true

Why?

The consistency problem

- ✓ The consistency problem for CFDs is to determine, given a set Σ of CFDs, whether or not there exists a nonempty database DB that satisfies Σ, i.e., for any φ in Σ, DB satisfies φ.

Whether or not Σ makes sense

- ✓ For traditional FDs, the consistency problem is not an issue: one can specify any FDs without worrying about their consistency
- ✓ A set of CFDs may be inconsistent!

Theorem. The consistency problem for CFDs is NP-complete.

Nontrivial: contrast this with the trivial consistency analysis of FDs!

The implication problem

The **implication problem for CFDs** is to determine, given a set Σ of CFDs and a single CFD φ, whether Σ **implies** φ, denoted by $\Sigma \models \varphi$, i.e., for any database DB, if DB satisfies Σ, then DB satisfies φ.

Example:

- ✓ $\Sigma = \{ \varphi1, \varphi2 \}$, $\varphi1$ = **(R(A → B)**, Tp1), $\varphi2$ = **(R(B → C)**, Tp2)

Tp1

id	A	B
tp1	_	b

Tp2

id	B	C
tp1	_	c

- ✓ φ = **(R(A → C)**, Tp)

id	A	C
tp	a	c

- ✓ $\Sigma \models \varphi$.

Conditional Constraints for Data Cleaning

- ✓ Conditional functional dependencies (CFDs)
  - – Syntax and semantics
  - – Static analysis: consistency and implication, axiom system
  - – SQL techniques for inconsistency detection and incremental detection
- ✓ Conditional inclusion dependencies (CINDs)
  - – Syntax and semantics
  - – Static analysis: consistency and implication
- ✓ Matching dependencies for record matching (MDs)
  - – Syntax and semantics
  - – Relative candidate keys

Example: Amazon database

- ✓ Schema:

  order(asin, title, type, price, country, county) -- source

  book(asin, isbn, title, price, format) -- target

  CD(asin, title, price, genre)

  asin: Amazon standard identification number

- ✓ Instances:

order

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

book

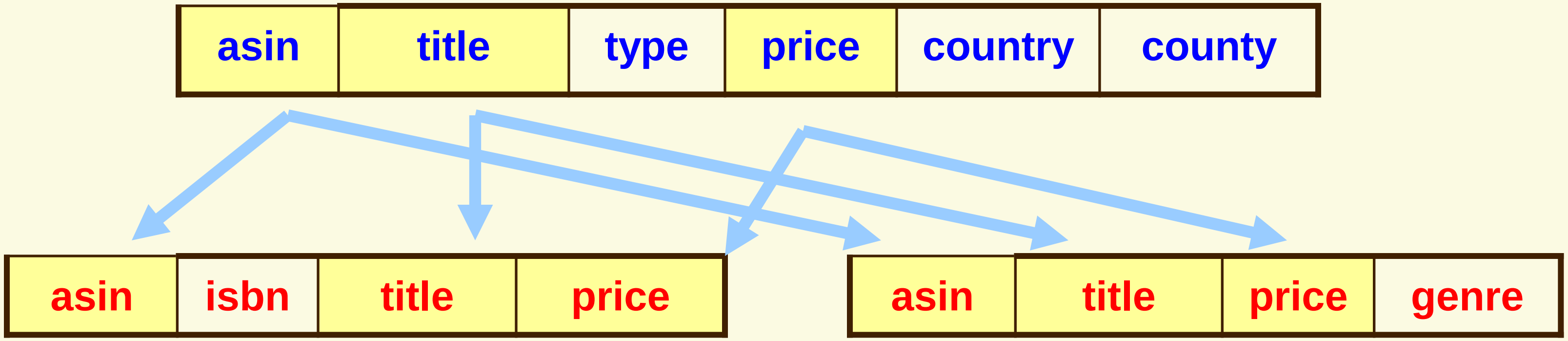
asin	isbn	title	price
a23	b32	Harry Porter	17.99
a56	b65	Snow white	7.94

CD

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

Schema matching

✓ Inclusion dependencies from source to target (e.g., Clio)



asin | title | type | price | country | county

asin | isbn | title | price

asin | title | price | genre

Do these make sense?

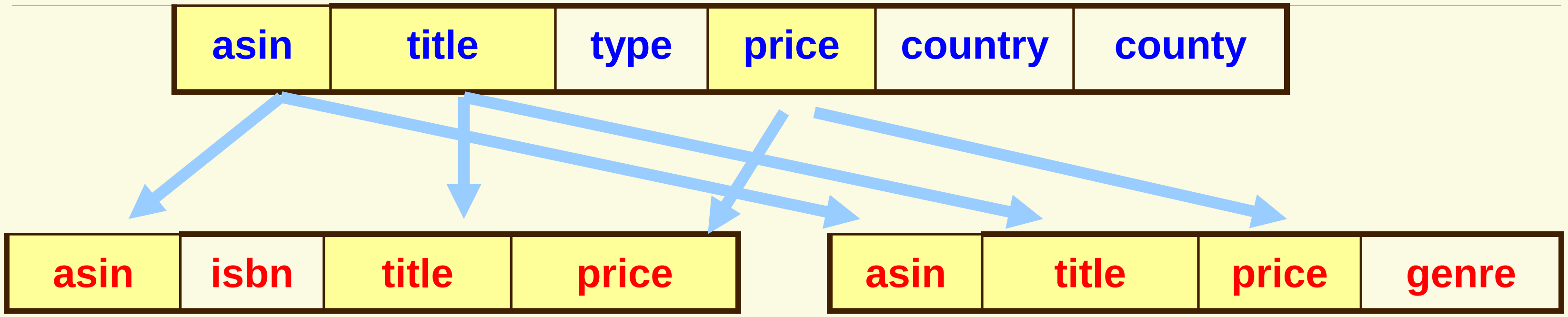
✓ Traditional inclusion dependencies:

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

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

These inclusion dependencies do not make sense!

Schema matching: dependencies with conditions



asin	title	type	price	country	county

asin	isbn	title	price

asin	title	price	genre

Conditional inclusion dependencies:

order[asin, title, price; type = book] ⊆ book[asin, title, price]

order[asin, title, price; type = CD] ⊆ CD[asin, title, price]

- ✓ order[asin, title, price] ⊆ book[asin, title, price] holds only if type = book
- ✓ order[asin, title, price] ⊆ CD[asin, title, price] holds only if type = CD

The constraints do not hold on the entire order table

Date cleaning with conditional dependencies

CIND1: order[asin, title, price; type = book] ⊆ book[asin, title, price]

CIND2: order[asin, title, price; type = CD] ⊆ CD[asin, title, price]

- ✓ Tuple t1 violates CIND1
- ✓ Tuple t2 violates CIND2, why?

order

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

book

asin	isbn	title	price
a23	b32	Harry Porter	17.99
a56	b65	Snow white	7.94

CD

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

More on data cleaning

CD

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

book

asin	isbn	title	price	format
a23	b32	Harry Porter	17.99	Hard cover
a56	b65	Snow White	17.94	audio

CD[asin, title, price; genre = ‘a-book’] ⊆ book[asin, title, price; format = ‘audio’]

- ✓ Inclusion dependency **CD[asin, title, price] ⊆ book[asin, title, price]** holds only if **genre = ‘a-book’**, i.e., when the CD is an audio boo
- ✓ In addition, the format of the corresponding book must a pattern for the referenced tuple

And what?

Conditional Inclusion Dependencies (CINDs)

(R1[X; Xp] $\subseteq$ R2[Y; Yp], Tp)

- ✓ R1[X] $\subseteq$ R2[Y]: embedded traditional IND from R1 to R2
- ✓ Tp: a pattern tableau
  - attributes: Xp $\cup$ Yp
  - tuples in Tp consist of constants and unnamed variable _

Example: express

CIND1: order[asin, title, price; type = book] $\subseteq$ book[asin, title, price]

- ✓ (**order[asin, title, price; type] $\subseteq$ book[asin, title, price; nil],** Tp)

nil: empty list

- ✓ pattern tableau Tp

type
book

Traditional CINDs as a special case

$R1[X] \subseteq R2[Y]$

- ✓ X: [A1, …, An]
- ✓ Y : [B1, …, Bn]

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

What is the pattern tableau?

- ✓ pattern tableau Tp: a single tuple ( )

CINDs subsume traditional INDs

Exercise

Express the following as CINDs:

CIND2: order[asin, title, price; type = CD] $\subseteq$ CD[asin, title, price]

CIND3: CD[asin, title, price; genre = ‘a-book’] $\subseteq$ book[asin, title, price; format = ‘audio’]

- ✓ **(order[asin, title, price; type] $\subseteq$ CD[asin, title, price; nil],** Tp)

type
CD

- ✓ **(CD[asin, title, price; genre] $\subseteq$ book[asin, title, price; format],** Tp)

genre	format
a-book	audio

Semantics of CINDs

DB = (DB1, DB2), where DBj is an instance of Rj, j = 1, 2.

DB satisfies (R1[X; Xp] $\subseteq$ R2[Y; Yp], Tp) iff for any tuples t1 in DB1, and any tuple tp in the pattern tableau Tp, if t1[Xp] $\approx$ tp[Xp], then there exists t2 in DB2 such that

- ✓ t1[Y] = t2[Y] (traditional IND semantics)
- ✓ t2[Yp] $\approx$ tp[Yp] (matching the pattern tuple on Y, Yp)

Patterns:

- ✓ t1[Xp] $\approx$ tp[Xp]: identifying the set of R1 tuples on which tp applies: { t1 | t1[Xp] $\approx$ tp[Xp] }
- ✓ t2[Yp] $\approx$ tp[Yp]: enforcing the embedded IND and the constraint specified by patterns Yp

Example

(CD[asin, title, price; genre] ⊆ book[asin, title, price; format], Tp)

genre	format
a-book	audio

The following DB satisfies the CIND

book

asin	isbn	title	price	format
a23	b32	Harry Porter	17.99	Hard cover
a56	b65	Snow white	7.94	audio

CD

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

Exercise

CIND1: (order[asin, title, price; type] ⊆ book[asin, title, price; nil], Tp)

type
book

The following DB violates CIND1. Why?

order

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

book

asin	isbn	title	price
a23	b32	Harry Porter	17.99
a56	b65	Snow white	7.94

CD

asin	title	price	genre
a12	J. Denver	17.99	country
a56	S. White	7.94	a-book

The satisfiability problem for CINDs

The consistency problem for CINDs is to determine, given a set Σ of CINDs, whether or not there exists a nonempty database DB that satisfies Σ, i.e., for any φ in Σ, DB satisfies φ.

Recall

- ✓ Any set of traditional INDs is always consistent!
- ✓ For CFDs, the satisfiability problem is intractable.

In contrast.

Theorem. Any set of CINDs is always consistent!

Despite the increased expressive power, the complexity of the satisfiability analysis does not go up.

The implication problem for CINDs

The implication problem for CINDs is to decide, given a set Σ of CINDs and a single CIND φ, whether Σ implies φ (Σ |= φ).

- ✓ For traditional INDs, the implication problem is PSPACE-complete
- ✓ For CINDs, the complexity does not hike up, to an extent:

Theorem. For CINDs containing no finite-domain attributes, the implication problem is PSPACE-complete

In the general setting, however, we have to pay a price:

Theorem. The implication problem for CINDs is EXPTIME-complete

Conditional Constraints for Data Cleaning

- Conditional functional dependencies (CFDs)
  - Syntax and semantics
  - Static analysis: consistency and implication, axiom system
  - SQL techniques for inconsistency detection and incremental detection
- Conditional inclusion dependencies (CINDs)
  - Syntax and semantics
  - Static analysis: consistency and implication
- Matching dependencies for record matching (MDs)
  - Syntax and semantics
  - Relative candidate keys

Record matching

To identify tuples from one or more *unreliable* sources that refer to *the same* real-world object.

FN	LN	address	tel	DOB	gender
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/27/97	M

FN	LN		nount
M.	Smith	10	*3,500*
…	…		..
Max	Smith		,300

Nontrivial:

✓ Real-life data is often dirty: errors in the data sources

✓ Data

different

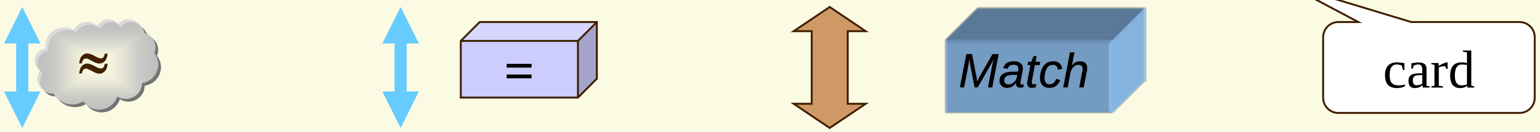
Pairwise comparison of attributes via equality only does not work!

Record linkage, entity resolution, data deduplication, merge/purge, …

Matching rules (Herndndez & Stolfo, 1995)

IF card[LN, address] = trans[LN, post] AND card[FN] and trans[FN] are *similar*, THEN *identify the two tuples*

FN	LN	address	tel	DOB	gender
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/27/97	M



FN	LN	post	phn	when	where	amount
M.	Smith	10 Oak St, EDI, EH8 9LE	null	1pm/7/7/09	EDI	*$3,500*
…	…	…	…	…	…	…
Max	Smith	PO Box 25, EDI	3256777	2pm/7/7/09	NYC	$6,300

trans

Accommodate errors in the data sources

Dependencies for record matching

$$\text{card}[\text{LN, address}] = \text{trans}[\text{LN, post}] \land \text{card}[\text{FN}] \approx \text{trans}[\text{FN}] \rightarrow \text{card}[X] \Leftrightarrow \text{trans}[Y]$$

$$\text{card}[\text{tel}] = \text{trans}[\text{phn}] \rightarrow \text{card}[\text{address}] \Leftrightarrow \text{trans}[\text{post}]$$

Identifying attributes (not necessarily entire records), across sources

X

card

FN	LN	address	tel	DOB	gender
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/27/97	M

Y

trans

FN	LN	post	phn	when	where	amount
Max	Smith	PO Box 25, EDI	3256777	2pm/7/7/09	NYC	$6,300

$2^{(m*n)}$ configurations

What attributes to compare? How to compare them?

Deducing new dependencies from given rules

$\text{card}[\text{LN}, \text{address}] = \text{trans}[\text{LN}, \text{post}] \wedge \text{card}[\text{FN}] \approx \text{trans}[\text{FN}] \rightarrow \text{card}[X] \Leftrightarrow \text{trans}[Y]$

$\text{card}[\text{tel}] = \text{trans}[\text{phn}] \rightarrow \text{card}[\text{address}] \Leftrightarrow \text{trans}[\text{post}]$

deduction

$\text{card}[\text{LN}, \text{tel}] = \text{trans}[\text{LN}, \text{phn}] \wedge \text{card}[\text{FN}] \approx \text{trans}[\text{FN}] \rightarrow \text{card}[X] \Leftrightarrow \text{trans}[Y]$

card

FN	LN	address	tel	DOB	gender
Mark	Smith	10 Oak St, EDI, EH8 9LE	3256777	10/27/97	M

Match

Radically different

trans

FN	LN	post	phn	when	where	amount
Max	Smith	PO Box 25, EDI	3256777	2pm/7/7/09	NYC	$6,300

*Matched by the deduced rule, but **NOT** by the given ones!*

Matching dependencies (MDs)

$$(R1[A_1] \approx_1 R2[B_1] \;\wedge \ldots \wedge\; R1[A_k] \approx_k R2[B_k]) \;\rightarrow\; R1[Z1] \;\Leftrightarrow\; R2[Z2]$$

R1[X], R2[Y]: entities to be identified

- ✓ (Z1, Z2): lists of attributes in (X, Y), of the same length
- ✓ $\approx_j$: similarity operator (edit distance, q-gram, jaro distance, …)
- ✓ ⇔: matching operator (identify two lists of attributes via updates)

R1[X]: card[FN, LN, address] , R2[Y]: trans[FN, LN, post]

- ✓ card[LN, address] = trans[LN, post] ∧ card[FN] ≈ trans[FN] → card[X] ⇔ trans[Y]
- ✓ card[tel] = trans[phn] → card[address] ⇔ trans[post]
- ✓ card[LN, tel] = trans[LN, phn] ∧ card[FN] ≈ trans[FN] → card[X] ⇔ trans[Y]

tel and phn are not part of X, Y

Semantic relationship on attributes across different sources

Dynamic semantics

$$\varphi = (R1[A1] \approx_1 R2[B1] \land \ldots \land R1[Ak] \approx_k R2[Bk]) \rightarrow R1[Z1] \Leftrightarrow R2[Z2]$$

(D1, D2) *satisfies* φ iff for all (t1, t2) ∈ D1,

- ✓ if $t1[A1] \approx_1 t2[B1] \land \ldots \land t1[Ak] \approx_k t2[Bk]$ in D1
  - then (t1, t2) ∈ D2, and *t1[Z1] = t2[Z2]* in D2

If (t1, t2) match the LHS, t pdated and equalized

Different from FDs?

tel	address	…
3256777	10 Oak St, EDI	

phn	post	…
3256777	PO Box 25, EDI	

D1

tel	address	…
3256777	10 Oak St, EDI, EH8 9LE	

phn	post	…
3256777	10 Oak St, EDI, EH8 9LE	

D2

Two instances are needed to cope with the dynamic semantics

An extension of functional dependencies (FDs)?

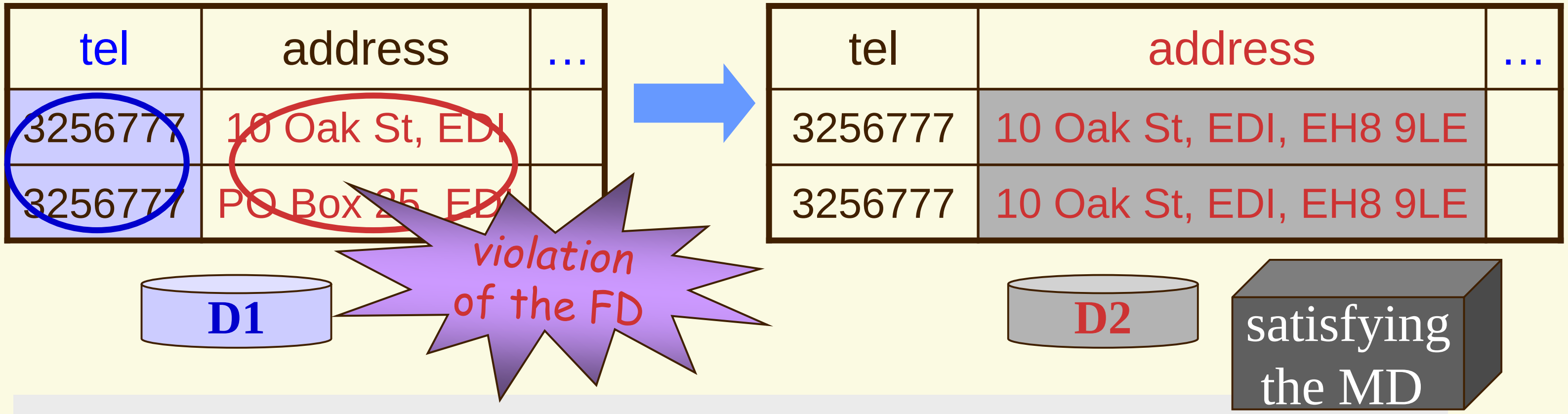
MD: $(R1[A_1] \approx_1 R2[B_1] \wedge \ldots \wedge R$ … $R1[Z1] \Leftrightarrow R2[Z2]$

FD: tel → address

developed for schema design for "clean" data

accommodate unreliable data

- ✓ similarity operators vs. equality (=) only
- ✓ across different relations (R1, R2) vs. on a single relation
- ✓ dynamic semantic (matching operator ⇔) vs. static semantics



tel	address	…
3256777	10 Oak St, EDI	
3256777	PO Box 25, EDI	

tel	address	…
3256777	10 Oak St, EDI, EH8 9LE	
3256777	10 Oak St, EDI, EH8 9LE	

A departure from traditional dependency theory

Summary and review

- ✓ What are CFDs? CINDs? Why do we need new constraints?
- ✓ What is the consistency problem? Complexity?
- ✓ What is the implication problem? Inference system? Sound and complete?
- ✓ What is record matching? Why bother?
- ✓ What are matching rules?
- ✓ A practical question: how to discover these constraints? A learning/Mining problem.

Supplementary: inference of new dependencies

The complexity of the implication problem

- ✓ For traditional FDs, the implication problem is in linear time
- ✓ In contrast, the implication problem for CFDs is intractable

Theorem. The implication problem for CFDs is coNP-complete.

Question: how about constant CFDs (without wildcard)? Would it simplify the consistency and implication analyses?

The expressive power of CFDs comes at a price

Finite axiomatizability: Flashback

Armstrong's axioms can be found in every database textbook:

- ✓ Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$
- ✓ Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$
- ✓ Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

Sound and complete for FD implication, i.e, $\Sigma \models \varphi$ iff φ can be inferred Σ from using reflexivity, augmentation, transitivity.

Question: is there a sound and complete inference system for the implication analysis of CFDs?

Finite axiomatizability of CFDs

Theorem. There is a sound and complete inference system ***I*** *for implication analysis of CFDs*

- ✓ *Sound: if Σ |- φ, i.e., φ can be proved from Σ using* ***I****, then Σ |= φ*
- ✓ *Complete: if Σ |= φ, then Σ |- φ using* ***I***

The inference system is more involved than its counterpart for traditional FDs, namely, Armstrong's axioms.

There are 5 axioms.

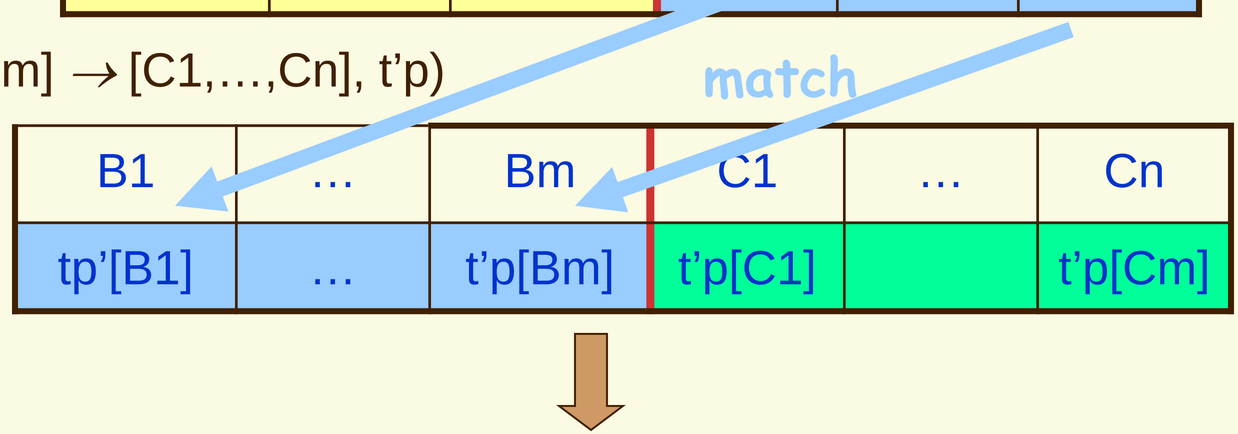
A normal form of CFDs: (R: X → A, tp), tp is a single pattern tuple.

Axioms for CFDs: transitivity

Transitivity: if ([A1,…,Ak] → [B1,…,Bm], tp)

A1	…	Ak	B1	…	Bm
tp[A1]	…	tp[Ak]	tp[B1]		tp[Bm]

and ([B1,…,Bm] → [C1,…,Cn], t’p)



B1	…	Bm	C1	…	Cn
tp’[B1]	…	t’p[Bm]	t’p[C1]		t’p[Cm]

A1	…	Ak	C1	…	Cn
tp[A1]	…	tp[Ak]	t’p[C1]		t’p[Cn]

([A1,…,Ak] → [C1,…,Cn], t’p)

Axioms for CFDs: reduction

- ✓ reduction: if ([B, X] → A, tp), tp[B] = _, and tp[A] = a

A1	…	Ak	B	A
tp[A1]	…	tp[Ak]	_	a

then (X → A, t’p)

A1	…	Ak	A
tp[A1]	…	tp[Ak]	a

Static analyses: CFD vs. FD

- ✓ **General setting:**

	satisfiability	**implication**	**finite axiom'ty**
CFD	NP-complete	coNP-complete	yes
FD	O(1)	O(n)	yes

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

	satisfiability	**implication**	**finite axiom'ty**
CFD	$O(n^2)$	$O(n^2)$	yes
FD	O(1)	O(n)	yes

- ✓ complications: finite-domain attributes

Finite axiomatizability of CINDs

- ✓ Rules for inferring IND implication:
  - Reflexivity: If R[X] ⊆ R[X]
  - Projection and Permutation: If R1[A1, …, Ak] ⊆ R2[B1, …, Bk], then R1[Ai1, …, Aik] ⊆ R2[Bi1, …, Bik],
  - Transitivity: If R1[X] ⊆ R2[Y] and R2[Y] ⊆ R3[Z], then R1[X] ⊆ R3[Z]

Sound and complete for IND implication

- ✓ CINDs retain the finite axiomatizability

Theorem. There is a sound and complete inference system for implication analysis of CINDs

There are 8 axioms.

Inference rules for CINDs

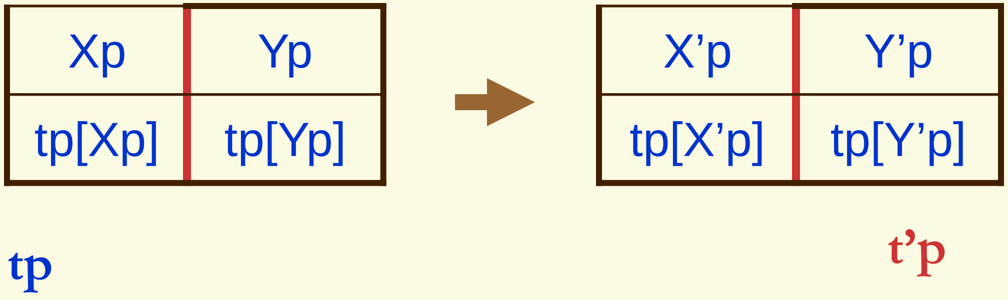
Normal form of CINDs: (R1[X; Xp] $\subseteq$ R2[Y; Yp], tp),

- ✓ tp is a single pattern tuple
- ✓ tp[A] is a constant iff A is in Xp or Yp (tp[B] = _ if B is in X or Y)

Inference rules

- ✓ Reflexivity: (R[X; nil] $\subseteq$ R[X; nil], tp), where tp = ( )

✓ Projection and permutation: If (R1[X; Xp] $\subseteq$ R2[Y; Yp], tp), then (R1[X'; X'p] $\subseteq$ R2[Y'; Y'p], t'p), for any permutation of X, Xp



Xp	Yp
tp[Xp]	tp[Yp]

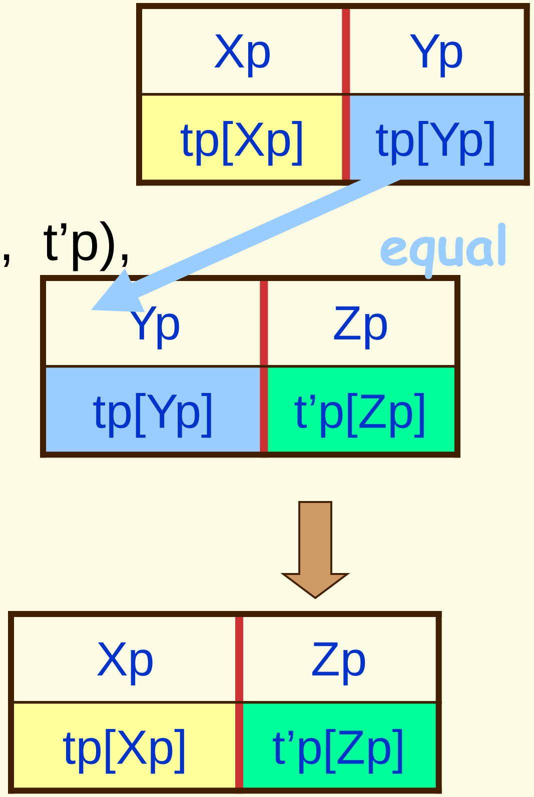
tp

X'p	Y'p
tp[X'p]	tp[Y'p]

t'p

Axioms for CINDs: transitivity

Transitivity: if (R1[X; Xp] ⊆ R2[Y; Yp], tp),



Xp	Yp
tp[Xp]	tp[Yp]

and (R2[Y; Yp] ⊆ R3[Z; Zp], t’p),

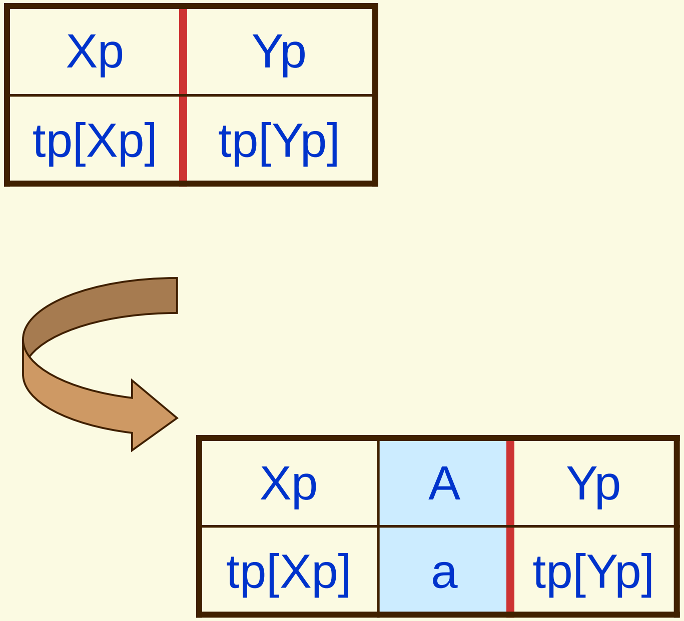
Yp	Zp
tp[Yp]	t’p[Zp]

Xp	Zp
tp[Xp]	t’p[Zp]

(R1[X; Xp] ⊆ R3[Z; Zp], t”p)

Axioms for CINDs: augmentation

- ✓ augmentation: if (R1[X; Xp] $\subseteq$ R2[Y; Yp], tp), A $\in$ attr(R1),



Xp	Yp
tp[Xp]	tp[Yp]

Xp	A	Yp
tp[Xp]	a	tp[Yp]

(R1[X; Xp, A] $\subseteq$ R2[Y; Yp], t’p)

Static analyses: CIND vs. IND

✓ **General setting:**

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

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

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

CINDs retain most complexity bounds of their traditional counterpart

CFDs and CINDs taken together

We need both CFDs and CINDs for

- ✓ data cleaning
- ✓ schema matching

Theorem. The implication problem for CFDs and CINDs is undecidable

Not surprising: The implication problem for traditional FDs and INDs is already undecidable

Theorem. The consistency problem for CFDs and CINDs is undecidable

In contrast, any set of traditional FDs and INDs is consistent!

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

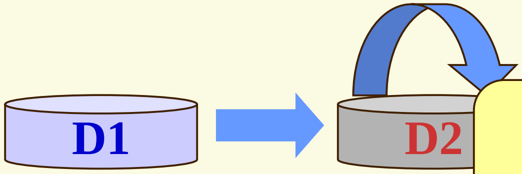
Deduction of new MDs from given MDs

Given a set Σ of MDs and a single φ, can φ be deduced from Σ?

For all (D1, D2) if

- ✓ (D1, D2) *satisfies* Σ and
- ✓ (D2, D2) *satisfies* Σ

then (D1, D2) *satisfies* φ



“fixpoint”

φ is a logical consequence of Σ

No matter how Σ is interpreted, if Σ is enforced, so is φ

Example: deduction of φ from {$\varphi 1$, $\varphi 2$}, where

φ: card[LN, tel] = trans[LN, phn] $\wedge$ card[FN] $\approx$ trans[FN] $\rightarrow$ card[X] $\Leftrightarrow$ trans[Y]

$\varphi 1$: card[tel] = trans[phn] $\rightarrow$ card[address] $\Leftrightarrow$ trans[post]

$\varphi 2$: card[LN,address] = trans[LN,post] $\wedge$ card[FN] $\approx$ trans[FN] $\rightarrow$ card[X] $\Leftrightarrow$ trans[Y]

Different from our familiar notion of implication

An inference system for MDs

Recall Armstrong's axioms for FDs

There is a finite set of axioms sound and complete for MD deduction

Example: MD φ is provable from {φ1, φ2} by using the inference system

φ1: card[tel] = trans[phn] $\rightarrow$ card[address] $\Leftrightarrow$ trans[post]

Augmentation Rule

card[LN, tel] = trans[LN, phn] $\rightarrow$ card[LN, address] $\Leftrightarrow$ trans[LN, post]

φ2: card[LN,address] = trans[LN,post] $\wedge$ card[FN] $\approx$ trans[FN] $\rightarrow$ card[X] $\Leftrightarrow$ trans[Y]

Transitivity Rule

φ: card[LN, tel] = trans[LN, phn] $\wedge$ card[FN] $\approx$ trans[FN] $\rightarrow$ card[X] $\Leftrightarrow$ trans[Y]

More involved than Armstrong's axioms (11 axioms vs. 3)

- ✓ *two relations, generic reasoning for similarity operators*