Possible and Certain Answers for Queries over Order-Incomplete Data

Antoine Amarilli[1], **M. Lamine Ba**[2]
Daniel Deutch[3], Pierre Senellart[4]

[1]Télécom ParisTech, [2]Université de Bambey

[3]Tel Aviv University, [4]École normale supérieure

Outline

Incompleteness in Data Management

- Incompleteness is a widespread problem in databases
  - Incompleteness on data values
  - Incompleteness on data ordering

Incompleteness in Data Management

- Incompleteness is a widespread problem in databases
  - Incompleteness on data values
  - Incompleteness on data ordering
- The sources of this incompleteness are varied
  - Incomplete knowledge of the world
  - Data transformation
  - Uncertain queries

Example of Order-Incomplete Data (I)

- Order-Incomplete Temporal Databases
  - Records of time-ordered connection events on different machines
    - timestamps may be missing from available data
    - machine clocks may be ill-synchronized

Example of Order-Incomplete Data (I)

- Order-Incomplete Temporal Databases
  - Records of time-ordered connection events on different machines
    - timestamps may be missing from available data
    - machine clocks may be ill-synchronized

IP	Port
10.10.1.1	80
10.10.6.2	20
10.10.1.1	443
10.10.100.5	1521
10.10.1.1	25

(a) Machine 1

IP	Port
10.10.1.1	22
10.10.11.2	80
10.10.1.1	22
10.10.5.16	515
10.10.10.1	995

(b) Machine 2

Example of Order-Incomplete data (II)

- Order-Incomplete Temporal Databases
  - Ordered data integration
    - SQL union behaves weirdly with order information

Example of Order-Incomplete data (II)

- Order-Incomplete Temporal Databases
  - Ordered data integration
    - SQL union behaves weirdly with order information

M1 $\cup$ M2

Example of Order-Incomplete data (II)

- Order-Incomplete Temporal Databases
  - Ordered data integration
    - SQL union behaves weirdly with order information

$$M1 \cup M2 \implies$$

Example of Order-Incomplete data (II)

- Order-Incomplete Temporal Databases
  - Ordered data integration
    - SQL union behaves weirdly with order information

$M1 \cup M2 \implies$

IP	Port
10.10.1.1	22
10.10.11.2	80
10.10.1.1	22
10.10.5.16	515
10.10.10.1	995
10.10.1.1	22
10.10.11.2	80
10.10.1.1	22
10.10.5.16	515
10.10.10.1	995

Example of Order-Incomplete data (II)

- Order-Incomplete Temporal Databases
  - Ordered data integration
    - SQL union behaves weirdly with order information

$\text{M1} \cup \text{M2} \implies$

IP	Port
10.10.1.1	22
10.10.11.2	80
10.10.1.1	22
10.10.5.16	515
10.10.10.1	995
10.10.1.1	22
10.10.11.2	80
10.10.1.1	22
10.10.5.16	515
10.10.10.1	995

- Result is unordered or arbitrarily ordered
  - Goal: capture all possible total orders w.r.t input orders

Order-Incomplete data example (III)

- Event log analysis
  - Port scanning: looking for specific sequences of ports, e.g., (22, 25, 80)
  - Vulnerability checking: looking for attack patterns, e.g., 443 then 25

Order-Incomplete data example (III)

- Event log analysis
  - Port scanning: looking for specific sequences of ports, e.g., (22, 25, 80)
  - Vulnerability checking: looking for attack patterns, e.g., 443 then 25

$\Pi_{\text{Port}}(\sigma_{\text{IP}=10.10.1.1}(\text{M1} \cup \text{M2}))$

Order-Incomplete data example (III)

- Event log analysis
  - Port scanning: looking for specific sequences of ports, e.g., (22, 25, 80)
  - Vulnerability checking: looking for attack patterns, e.g., 443 then 25

$\Pi_{\mathrm{Port}}(\sigma_{\mathrm{IP}=10.10.1.1}(\mathrm{M1} \cup \mathrm{M2})) \Longrightarrow$

Order-Incomplete data example (III)

- Event log analysis
  - Port scanning: looking for specific sequences of ports, e.g., (22, 25, 80)
  - Vulnerability checking: looking for attack patterns, e.g., 443 then 25

$$\Pi_{\text{Port}}(\sigma_{\text{IP}=10.10.1.1}(\text{M1} \cup \text{M2})) \Longrightarrow$$

Port
80
443
25
22
22

Order-Incomplete data example (III)

- Event log analysis
  - Port scanning: looking for specific sequences of ports, e.g., (22, 25, 80)
  - Vulnerability checking: looking for attack patterns, e.g., 443 then 25

$\Pi_{\text{Port}}(\sigma_{\text{IP}=10.10.1.1}(\text{M1} \cup \text{M2})) \Longrightarrow$

Port
80
443
25
22
22

- Resulting sequence is unordered or with unknown order
  - Goal: checking possibility and certainty of answers to queries on order-incomplete data

Outline

Order-Incomplete Data Model (I)

- Representation system
  - Relational databases (assuming bag semantics)
  - Partial order on tuple identifiers of each relation

Order-Incomplete Data Model (I)

- Representation system
  - Relational databases (assuming bag semantics)
  - Partial order on tuple identifiers of each relation

#	Port
1	22
2	80
3	22
4	515

$$
\begin{tikzcd}
 & 1 \arrow[dl] \arrow[d] \\
2 \arrow[dr] & 3 \arrow[d] \\
 & 4
\end{tikzcd}
$$

Partially ordered relation (po-relation)

Order-Incomplete Data Model (II)

- Query language

Order-Incomplete Data Model (II)

- Query language
  - PosRA: Positive relational algebra
    - selection
    - projection
    - product
    - union

Order-Incomplete Data Model (III)

- Selection
  - returns all tuples satisfying a given predicate
  - resulting partial order is a restriction of the input one

Order-Incomplete Data Model (III)

- Selection
  - returns all tuples satisfying a given predicate
  - resulting partial order is a restriction of the input one

#	Port
1	22
2	80
3	22
4	515

$$\begin{tikzcd}
 & 1 \arrow[dl] \arrow[d] \\
2 \arrow[dr] & 3 \arrow[d] \\
 & 4
\end{tikzcd}$$

Order-Incomplete Data Model (III)

- Selection
  - returns all tuples satisfying a given predicate
  - resulting partial order is a restriction of the input one

#	Port
1	22
2	80
3	22
4	515

$\sigma_{\text{Port in [22, 80]}}$

$$\begin{tikzcd} & 1 \arrow[dl] \arrow[d] \\ 2 \arrow[dr] & 3 \arrow[d] \\ & 4 \end{tikzcd}$$

Order-Incomplete Data Model (III)

- Selection
  - returns all tuples satisfying a given predicate
  - resulting partial order is a restriction of the input one

#	Port
1	22
2	80
3	22
4	515

$\sigma_{\text{Port in [22, 80]}} \Longrightarrow$

$$
\begin{tikzcd}
& 1 \arrow[dl] \arrow[d] \\
2 \arrow[dr] & 3 \arrow[d] \\
& 4
\end{tikzcd}
$$

- Selection
  - returns all tuples satisfying a given predicate
  - resulting partial order is a restriction of the input one

#	Port
1	22
2	80
3	22
4	515

$\sigma_{\text{Port in [22, 80]}} \Longrightarrow$

#	Port
1	22
2	80
3	22

$$\begin{tikzcd} & 1 \arrow[dl] \arrow[d] \\ 2 \arrow[dr] & 3 \arrow[d] \\ & 4 \end{tikzcd}$$

$$\begin{tikzcd} & 1 \arrow[dl] \arrow[d] \\ 2 & 3 \end{tikzcd}$$

Order-Incomplete Data Model (IV)

- Projection
  - returns all the input tuples with only the specified attributes
  - does not change the input tuple ordering

Order-Incomplete Data Model (IV)

- Projection
  - returns all the input tuples with only the specified attributes
  - does not change the input tuple ordering

#	IP	Port
1	10.10.1.1	22
2	10.10.11.2	80
3	10.10.1.1	22
4	10.10.5.16	515
5	10.10.10.1	995

Order-Incomplete Data Model (IV)

- Projection
  - returns all the input tuples with only the specified attributes
  - does not change the input tuple ordering

#	IP	Port
1	10.10.1.1	22
2	10.10.11.2	80
3	10.10.1.1	22
4	10.10.5.16	515
5	10.10.10.1	995

Π_{Port}

Order-Incomplete Data Model (IV)

- Projection
  - returns all the input tuples with only the specified attributes
  - does not change the input tuple ordering

#	IP	Port
1	10.10.1.1	22
2	10.10.11.2	80
3	10.10.1.1	22
4	10.10.5.16	515
5	10.10.10.1	995

$\Pi_{\text{Port}} \Longrightarrow$

Order-Incomplete Data Model (IV)

- Projection
  - returns all the input tuples with only the specified attributes
  - does not change the input tuple ordering

#	IP	Port
1	10.10.1.1	22
2	10.10.11.2	80
3	10.10.1.1	22
4	10.10.5.16	515
5	10.10.10.1	995

$\Pi_{\text{Port}} \Longrightarrow$

#	Port
1	22
2	80
3	22
4	515
5	995

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics $\Rightarrow$ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics $\Rightarrow$ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

#	Port
1	80
2	443
3	25

#	Port
1'	22
2'	22

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics $\Rightarrow$ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

#	Port
1	80
2	443
3	25

$\cup$

#	Port
1'	22
2'	22

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics $\Rightarrow$ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

#	Port
1	80
2	443
3	25

$\bigcup \qquad \Longrightarrow$

#	Port
1'	22
2'	22

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics ⇒ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

#	Port
1	80
2	443
3	25

$\cup$

#	Port
1'	22
2'	22

$\Longrightarrow$

#	Port
1	80
2	443
3	25
1'	22
2'	22

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics $\Rightarrow$ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

#	Port
1	80
2	443
3	25

$\cup$

#	Port
1'	22
2'	22

$\Longrightarrow$

#	Port
1	80
2	443
3	25
1'	22
2'	22

Parallel composition

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics $\Rightarrow$ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

#	Port
1	80
2	443
3	25

$\cup$

#	Port
1'	22
2'	22

$\Longrightarrow$

#	Port
1	80
2	443
3	25
1'	22
2'	22

1 1'

2 2'

3

Parallel composition

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics ⇒ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

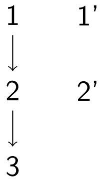
#	Port
1	80
2	443
3	25

$\cup$

#	Port
1'	22
2'	22

$\Longrightarrow$

#	Port
1	80
2	443
3	25
1'	22
2'	22



Parallel composition

Order-Incomplete Data Model (V)

- Union
  - introduces uncertainty on the resulting order
  - merges tuples from the input relations into a single relation
  - minimal semantics $\Rightarrow$ imposing minimal order constraints
    - resulting order is the parallel composition of the input orders

#	Port
1	80
2	443
3	25

$\cup$

#	Port
1'	22
2'	22

$\Longrightarrow$

#	Port
1	80
2	443
3	25
1'	22
2'	22

$$\begin{tikzcd} 1 \arrow[d] & 1' \arrow[d] \\ 2 \arrow[d] & 2' \\ 3 & \end{tikzcd}$$

Parallel composition

Order-Incomplete Data Model (VI)

- Product

Order-Incomplete Data Model (VI)

- Product
  - introduces uncertainty on the resulting order

Order-Incomplete Data Model (VI)

- Product
  - introduces uncertainty on the resulting order
  - two semantics: direct product and lexicographic product

Order-Incomplete Data Model (VII)

- Direct product

Order-Incomplete Data Model (VII)

- Direct product
  - applies usual relational product for the resulting relation

Order-Incomplete Data Model (VII)

- Direct product
  - applies usual relational product for the resulting relation
  - direct ordering (or minimal semantics) $\Rightarrow$ comparability holds only if both components of both identifiers compare in the same way

Order-Incomplete Data Model (VII)

- Direct product
  - applies usual relational product for the resulting relation
  - direct ordering (or minimal semantics) $\Rightarrow$ comparability holds only if both components of both identifiers compare in the same way

#	Port	
1	80	$\downarrow$
2	443	

$\times_{\mathrm{DIR}}$

#	Port	
1'	22	$\downarrow$
2'	22	

Order-Incomplete Data Model (VII)

- Direct product
  - applies usual relational product for the resulting relation
  - direct ordering (or minimal semantics) $\Rightarrow$ comparability holds only if both components of both identifiers compare in the same way

#	Port
1	80
2	443

$\times_{\mathrm{DIR}}$ $\Longrightarrow$

#	Port
1'	22
2'	22

Order-Incomplete Data Model (VII)

- Direct product
  - applies usual relational product for the resulting relation
  - direct ordering (or minimal semantics) $\Rightarrow$ comparability holds only if both components of both identifiers compare in the same way

#	Port
1	80
2	443

$\times_{\text{DIR}}$

#	Port
1'	22
2'	22

$\Longrightarrow$

#1	#2	Port1	Port2
1	1'	80	22
2	1'	443	22
1	2'	80	22
2	2'	443	22

Order-Incomplete Data Model (VII)

- Direct product
  - applies usual relational product for the resulting relation
  - direct ordering (or minimal semantics) $\Rightarrow$ comparability holds only if both components of both identifiers compare in the same way

#	Port
1	80
2	443

$\times_{\text{DIR}}$

#	Port
1'	22
2'	22

$\Longrightarrow$

#1	#2	Port1	Port2
1	1'	80	22
2	1'	443	22
1	2'	80	22
2	2'	443	22

Direct order

Order-Incomplete Data Model (VII)

- Direct product
  - applies usual relational product for the resulting relation
  - direct ordering (or minimal semantics) $\Rightarrow$ comparability holds only if both components of both identifiers compare in the same way

#	Port
1	80
2	443

$\times_{\mathrm{DIR}}$

#	Port
1'	22
2'	22

$\Longrightarrow$

#1	#2	Port1	Port2
1	1'	80	22
2	1'	443	22
1	2'	80	22
2	2'	443	22

(1, 1')

(1, 2') (2, 1')

(2, 2')

Direct order

Order-Incomplete Data Model (VII)

- Direct product
  - applies usual relational product for the resulting relation
  - direct ordering (or minimal semantics) $\Rightarrow$ comparability holds only if both components of both identifiers compare in the same way

#	Port
1	80
2	443

$\times_{\text{DIR}}$

#	Port
1'	22
2'	22

$\Longrightarrow$

#1	#2	Port1	Port2
1	1'	80	22
2	1'	443	22
1	2'	80	22
2	2'	443	22

$$\begin{tikzcd}
 & (1, 1') \arrow[dl] \arrow[dr] & \\
(1, 2') \arrow[dr] & & (2, 1') \arrow[dl] \\
 & (2, 2') &
\end{tikzcd}$$

Direct order

Order-Incomplete Data Model (VIII)

- Lexicographic product
  - applies usual product for the resulting relation
  - lexicographic order $\Rightarrow$ comparability holds only if
    - the first components of both pairs are comparable, or
    - the second components are comparable and the first are equal

Order-Incomplete Data Model (VIII)

- Lexicographic product
  - applies usual product for the resulting relation
  - lexicographic order ⇒ comparability holds only if
    - the first components of both pairs are comparable, or
    - the second components are comparable and the first are equal

#	Port
1	80
2	443

#	Port
1'	22
2'	22

Order-Incomplete Data Model (VIII)

- Lexicographic product
  - applies usual product for the resulting relation
  - lexicographic order ⇒ comparability holds only if
    - the first components of both pairs are comparable, or
    - the second components are comparable and the first are equal

#	Port
1	80
2	443

$\times_{\mathrm{LEX}}$

#	Port
1'	22
2'	22

Order-Incomplete Data Model (VIII)

- Lexicographic product
  - applies usual product for the resulting relation
  - lexicographic order $\Rightarrow$ comparability holds only if
    - the first components of both pairs are comparable, or
    - the second components are comparable and the first are equal

#	Port
1	80
2	443

$\times_{\mathrm{LEX}}$ $\Rightarrow$

#	Port
1'	22
2'	22

Order-Incomplete Data Model (VIII)

- Lexicographic product
  - applies usual product for the resulting relation
  - lexicographic order $\Rightarrow$ comparability holds only if
    - the first components of both pairs are comparable, or
    - the second components are comparable and the first are equal

#	Port
1	80
2	443

$\times_{\text{LEX}}$

#	Port
1'	22
2'	22

$\Rightarrow$

#1	#2	Port1	Port2
1	1'	80	22
2	1'	443	22
1	2'	80	22
2	2	443	22

Order-Incomplete Data Model (VIII)

- Lexicographic product
  - applies usual product for the resulting relation
  - lexicographic order $\Rightarrow$ comparability holds only if
    - the first components of both pairs are comparable, or
    - the second components are comparable and the first are equal

#	Port
1	80
2	443

$\times_{\mathrm{LEX}}$

#	Port
1'	22
2'	22

$\Rightarrow$

#1	#2	Port1	Port2
1	1'	80	22
2	1'	443	22
1	2'	80	22
2	2	443	22

(1, 1')

(1, 2') (2, 1')

(2, 2')

Order-Incomplete Data Model (VIII)

- Lexicographic product
  - applies usual product for the resulting relation
  - lexicographic order ⇒ comparability holds only if
    - the first components of both pairs are comparable, or
    - the second components are comparable and the first are equal

#	Port
1	80
2	443

↓

$\times_{\text{LEX}}$

#	Port
1'	22
2'	22

↓

⇒

#1	#2	Port1	Port2
1	1'	80	22
2	1'	443	22
1	2'	80	22
2	2	443	22

(1, 1')

(1, 2') (2, 1')

(2, 2')

lex order

Order-Incomplete Data Model (VIII)

- Lexicographic product
  - applies usual product for the resulting relation
  - lexicographic order $\Rightarrow$ comparability holds only if
    - the first components of both pairs are comparable, or
    - the second components are comparable and the first are equal

#	Port
1	80
2	443

$\times_{\text{LEX}}$

#	Port
1'	22
2'	22

$\Rightarrow$

#1	#2	Port1	Port2
1	1'	80	22
2	1'	443	22
1	2'	80	22
2	2	443	22

$$\begin{tikzcd} & (1, 1') \arrow[dl] \arrow[dr] & \\ (1, 2') \arrow[rr] \arrow[dr] & & (2, 1') \arrow[dl] \\ & (2, 2') & \end{tikzcd}$$

lex order

Order-Incomplete Data Model (IX)

- Query language
  - PosRA: Positive relational algebra
    - $\bigcup$, $\times_{\mathrm{DIR}}$ and $\times_{\mathrm{LEX}}$ introduce uncertainty on the total order
    - possible-world semantics $\Rightarrow$ enumerating all possible total orders
    - a po-relation captures all possible totally ordered relations

Order-Incomplete Data Model (IX)

- Query language
  - PosRA: Positive relational algebra
    - $\bigcup$, $\times_{\text{DIR}}$ and $\times_{\text{LEX}}$ introduce uncertainty on the total order
    - possible-world semantics $\Rightarrow$ enumerating all possible total orders
    - a po-relation captures all possible totally ordered relations

#	Port
1	80
2	443
3	25
1'	22
2'	22

$$\begin{array}{cc} 1 & 1' \\ \downarrow & \downarrow \\ 2 & 2' \\ \downarrow & \\ 3 & \end{array}$$

Order-Incomplete Data Model (IX)

- Query language
  - PosRA: Positive relational algebra
    - $\bigcup$, $\times_{\mathrm{DIR}}$ and $\times_{\mathrm{LEX}}$ introduce uncertainty on the total order
    - possible-world semantics $\Rightarrow$ enumerating all possible total orders
    - a po-relation captures all possible totally ordered relations

#	Port
1	80
2	443
3	25
1'	22
2'	22

$$1 \rightarrow 2 \rightarrow 3 \qquad 1' \rightarrow 2'$$

#	Port
1	80
2	443
3	25
1'	22
2'	22

#	Port
1'	22
2'	22
1	80
2	443
3	25

#	Port
1'	22
1	80
2	443
3	25
2'	22

#	Port
1'	22
1	80
2	443
2'	22
3	25

#	Port
1	88
2	443
1'	22
2'	22
3	25

#	Port
1	88
1'	22
2	443
2'	22
3	25

...

Order-Incomplete Data Model (X)

- Query language
  - PosRA: Positive relational algebra
    - $\bigcup$, $\times_{\text{DIR}}$ and $\times_{\text{LEX}}$ introduce uncertainty on the total order
    - possible-world semantics $\Rightarrow$ enumerating all possible total orders
    - a po-relation captures all possible totally ordered relations

Order-Incomplete Data Model (X)

- Query language
  - PosRA: Positive relational algebra
    - $\bigcup$, $\times_{\mathrm{DIR}}$ and $\times_{\mathrm{LEX}}$ introduce uncertainty on the total order
    - possible-world semantics $\Rightarrow$ enumerating all possible total orders
    - a po-relation captures all possible totally ordered relations

$$\begin{tikzcd} \text{po-relation} \arrow[r, "\text{query}"] \arrow[d, "\text{pw}"'] & \text{po-relation} \arrow[d, "\text{pw}"] \\ \text{totally ordered relations} \arrow[r, "\text{query one by one}"] & \text{totally ordered relations} \end{tikzcd}$$

Order-Incomplete Data Model (XI)

- Query language
  - PosRA: Positive relational algebra

Proposition

For any fixed PosRA query Q, given a po-database D, we can construct the po-relation $Q(D)$ in PTIME in the size of D.

Order-Incomplete Data Model (XII)

- Query language (extended!)
  - PosRA: Positive relational algebra
    - selection
    - projection
    - product
    - union
  - PosRA+acc: PosRA with order-aware accumulation
    - concatenation
    - top-k
    - more complex accumulation

Outline

Complexity of Possibility and Certainty (I)

- For PosRA+acc queries, a concise encoding is not easy to find
- Instead, we study the complexity of possibility and certainty problems

Complexity of Possibility and Certainty (I)

- For PosRA+acc queries, a concise encoding is not easy to find
- Instead, we study the complexity of possibility and certainty problems

`POSS`. Given $L \in M$, is L a possible world of W?

`CERT`. Given $L \in M$, is L the only possible world of W?

Complexity of Possibility and Certainty (II)

- Main complexity results (data complexity)

Complexity of Possibility and Certainty (II)

- Main complexity results (data complexity)

Theorem

- POSS is NP-complete for PosRA and PosRA+acc queries

Complexity of Possibility and Certainty (II)

- Main complexity results (data complexity)

Theorem

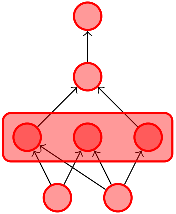
- `POSS` is NP-complete for PosRA and PosRA+acc queries

Theorem

- `CERT` is PTIME for PosRA queries
- `CERT` is coNP-complete for PosRA+acc queries

Complexity of Possibility and Certainty (II)

- Easy cases
  - Bounded width (limits the sets of pairwise incomparable tuples)
    - e.g., totally ordered set (width equals to 1)

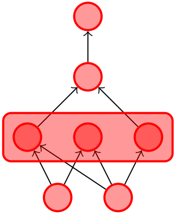


Complexity of Possibility and Certainty (II)

- Easy cases
  - Bounded width (limits the sets of pairwise incomparable tuples)
    - e.g., totally ordered set (width equals to 1)

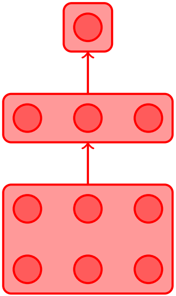
Theorem

- POSS and CERT are PTIME
  - no accumulation
  - accumulation on bounded domains



Complexity of Possibility and Certainty (II)

- Easy cases
  - Bounded ia-width (class of pairwise incomparable tuples)
    - e.g., unordered set (only one class)

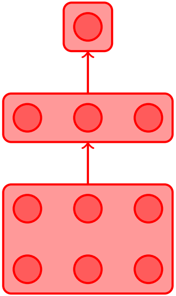


Complexity of Possibility and Certainty (II)

- Easy cases
  - Bounded ia-width (class of pairwise incomparable tuples)
    - e.g., unordered set (only one class)

Theorem

- POSS and CERT are PTIME
  - no accumulation
  - accumulation on bounded domains



Outline

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

#	Port
1	80
2	20
3	20

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

#	Port
1	80
2	20
3	20

dupElim

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

#	Port
1	80
2	20
3	20

dupElim $\Longrightarrow$

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

#	Port
1	80
2	20
3	20

dupElim $\Longrightarrow$

#	Port
1	80
2	20

dupElim succeeds!

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

#	Port
1	20
2	80
3	20

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

#	Port
1	20
2	80
3	20

no id-sets

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

#	Port
1	20
2	80
3	20

$\Longrightarrow$

no id-sets

Extensions

- dupElim: duplicate elimination
  - merge tuples with duplicate values
  - uncertainty if merged values have different orderings
  - semantics of order-aware dupElim via indistinguishable sets (id-sets)
    - merge succeeds if duplicate tuples form id-sets

#	Port
1	20
2	80
3	20

$\Longrightarrow$ dupElim fails

no id-sets

Extension

- Group-by
  - Extension of accumulation
  - First, group tuples by their value on some attributes
  - Then, perform accumulation within each group

Outline

Conclusion

- Bag semantics of positive relational algebra on order-incomplete data
  - order-preserving operators
  - capturing all possible worlds of order-uncertain relations
- Concise representation model: po-relations
- Accumulation as last operation
- Complexity of deciding possibility and certainty

Conclusion

- Bag semantics of positive relational algebra on order-incomplete data
  - order-preserving operators
  - capturing all possible worlds of order-uncertain relations
- Concise representation model: po-relations
- Accumulation as last operation
- Complexity of deciding possibility and certainty

- More operators?
- Set semantics?
- Combine uncertainty on order with that on values
- Other results for POSS/CERT?
- Preliminary results on typological sorting by Amarilli and Paperman.

Conclusion

- Bag semantics of positive relational algebra on order-incomplete data
  - order-preserving operators
  - capturing all possible worlds of order-uncertain relations
- Concise representation model: po-relations
- Accumulation as last operation
- Complexity of deciding possibility and certainty

- More operators?
- Set semantics?
- Combine uncertainty on order with that on values
- Other results for POSS/CERT?
- Preliminary results on typological sorting by Amarilli and Paperman.

Thank for your attention !