Learning Rules for Collective Entity Resolution

Zhiliang (Leon) Xiang

Cardiff University

Entity Resolution (ER)

Entity resolution (*ironically goes with different names: deduplication, record linkage, entity matching…*)

- to identify **different constants** representing **the same entity**
- usually in structured/semi-structured data sources, e.g. **database**, knowledge graph

Entity Resolution (ER)

Entity resolution (*ironically goes with different names: deduplication, record linkage, entity matching…*)

- to identify **different constants** representing **the same entity**
- usually in structured/semi-structured data sources, e.g. **database**, knowledge graph

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	123456	brufen
p2	John Smith	32	12345-6	aspirin

Entity Resolution (ER)

Entity resolution (*ironically goes with different names: deduplication, record linkage, entity matching…*)

- to identify **different constants** representing **the same entity**
- usually in structured/semi-structured data sources, e.g. **database**, knowledge graph

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	123456	brufen
p2	John Smith	32	12345-6	aspirin

Entity Resolution (ER)

Entity resolution (*ironically goes with different names: deduplication, record linkage, entity matching…*)

- to identify **different constants** representing **the same entity**
- usually in structured/semi-structured data sources, e.g. **database**, knowledge graph

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	123456	brufen
p2	John Smith	32	12345-6	aspirin

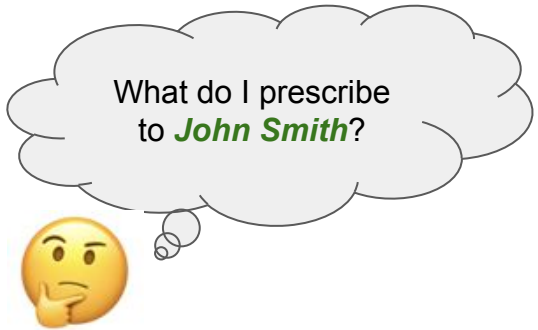
Entity Resolution (ER)

Entity resolution (*ironically goes with different names: deduplication, record linkage, entity matching…*)

- to identify **different constants** representing **the same entity**
- usually in structured/semi-structured data sources, e.g. **database**, knowledge graph

Patient

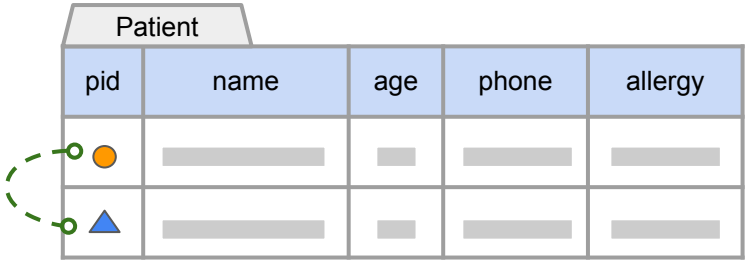
pid	name	age	phone	allergy
p1	J.Smith	32	123456	brufen
p2	John Smith	32	12345-6	aspirin



Critical to **data quality** and **decision making**!

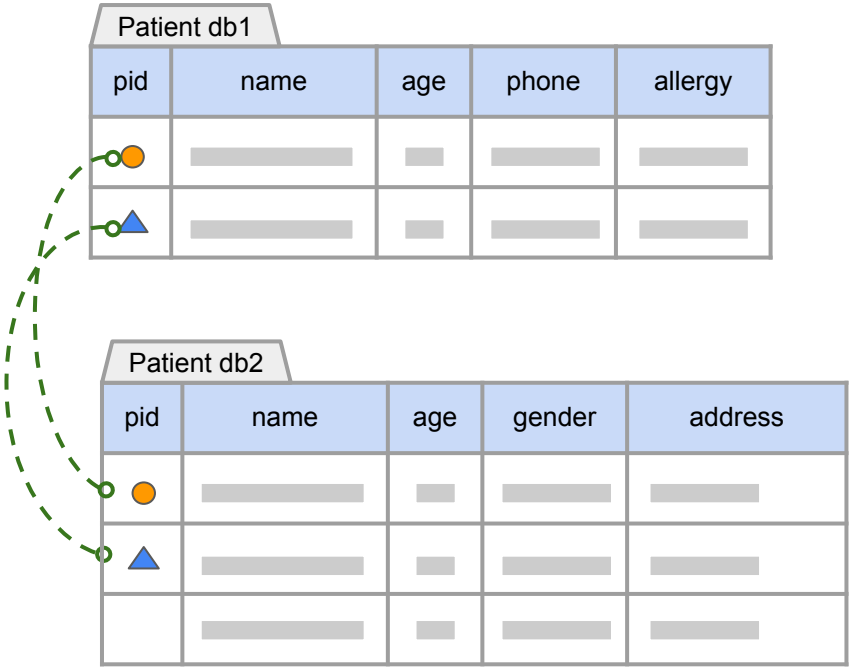
Entity Resolution (ER)

Traditional: within **single-table** or **table pair** (same entity type)



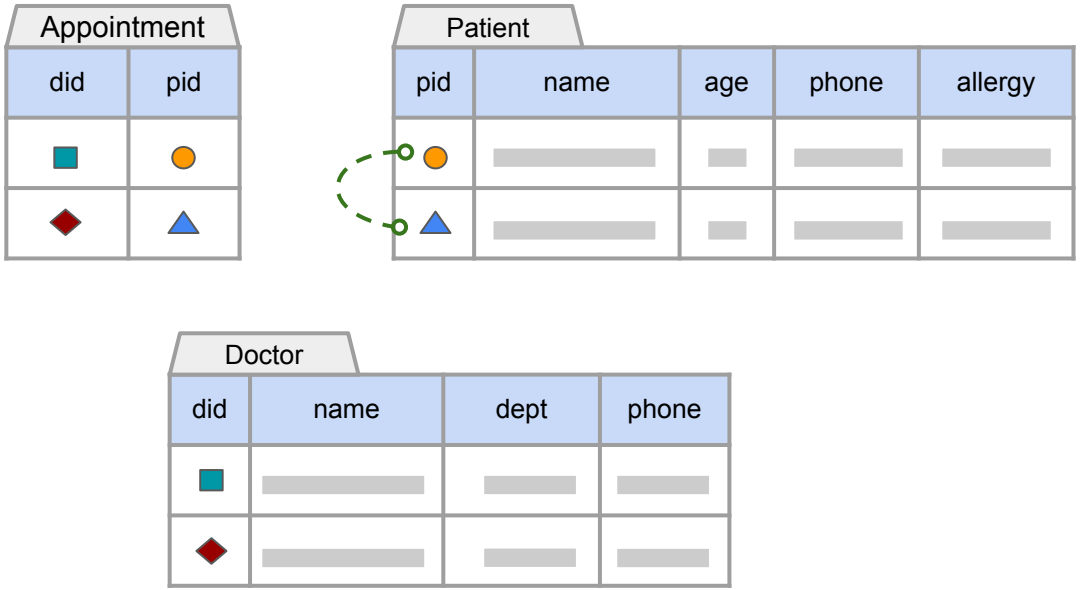
Entity Resolution (ER)

Traditional: within **single-table** or **table pair** (same entity type)



Entity Resolution (ER)

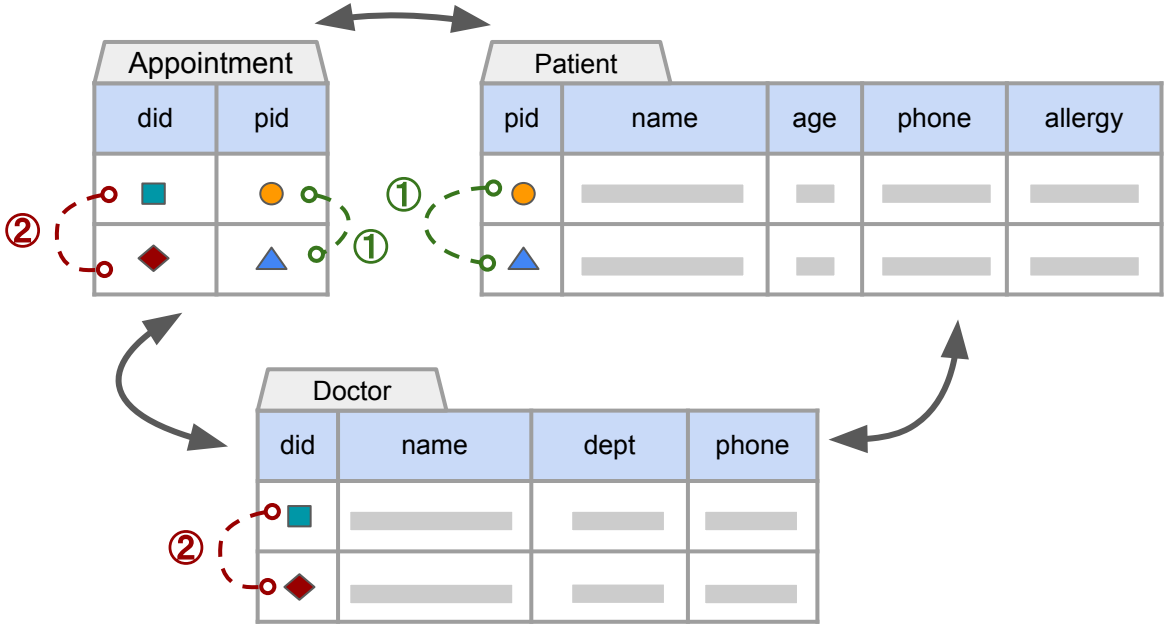
Traditional: within **single-table** or **table pair** (same entity type)



Entity Resolution (ER)

Traditional: within **single-table** or **table pair** (same entity type)

Collective: across **multiple tables**, exploit **inter-dependencies** between tables



Previous work

Neural-based methods excel in pairwise setting, but …

- Failed to capture **transitivity**, predictions may be **inconsistent**
- Not clear how to support **collective ER**
- **Explanation** matters

Previous work

Novel rule-based framework:

- ***LACE (A Logical Approach to Collective Entity Resolution)*** *[Bienvenu et al. 2022]*

ASP-based Implementation:

- ***ASPEn system*** *[Xiang et al. 2024]*

Featuring

- **declarative:** logical **rules** and **constraints**
- **collective:** natural to capture **interdependencies** between tables
- **explainable**: "**why** are *J. Smith* and *John Smith* deemed to be the **same** patient?"

Previous work

Novel rule-based framework:

- ***LACE (A Logical Approach to Collective Entity Resolution)*** *[Bienvenu et al. 2022]*

ASP-based Implementation:

- ***ASPEn system*** *[Xiang et al. 2024]*

Featuring

- **declarative:** logical **rules** and **constraints**
- **collective:** natural to capture **interdependencies** between tables
- **explainable**: "**why** are *J. Smith* and *John Smith* deemed to be the **same** patient?"

Expert knowledge is hard to obtain

Rule-based approach

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

ER rules L $x[pid]=y[pid] <= Patient(x), Patient(y), x[name]\approx y[name], x[age,phone]=y[age,phone]$

Rule-based approach

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

ER rules L $x[\mathbf{pid}]=y[\mathbf{pid}] \leftarrow Patient(x), Patient(y), x[\mathbf{name}] \approx y[\mathbf{name}], x[\mathbf{age},\mathbf{phone}]=y[\mathbf{age},\mathbf{phone}]$

Rule-based approach

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

ER rules L $x[\mathbf{pid}]=y[\mathbf{pid}] <= Patient(x), Patient(y), x[\mathbf{name}]\approx y[\mathbf{name}], x[\mathbf{age},\mathbf{phone}]=y[\mathbf{age},\mathbf{phone}]$

If … then … are the same

pid of two patients are the same if they have **similar names**, and **identical age and phone number**

Rule-based approach

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

ER rules L $x[\mathbf{pid}]=y[\mathbf{pid}] <= Patient(x), Patient(y), x[\mathbf{name}]\approx y[\mathbf{name}], x[\mathbf{age},\mathbf{phone}]=y[\mathbf{age},\mathbf{phone}]$

⇕ If … then … are the same

pid of two patients are the same if they have **similar names**, and **identical age and phone number**

⇕ database query

```
select x.pid, y.pid from patient as x, patient as y
        where x.name ≈ y.name, x.age = y.age, x.phone = y.phone
```

Rule-based approach

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

ER rules L $x[\mathbf{pid}]=y[\mathbf{pid}] \leftarrow Patient(x), Patient(y), x[\mathbf{name}]\approx y[\mathbf{name}], x[\mathbf{age},\mathbf{phone}]=y[\mathbf{age},\mathbf{phone}]$

Query answers

result

?

Learning ER rule

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

ER rules L

Query answers

result

pid	pid
p1	p2
p3	p4

Learning ER rule $\sqsubseteq$ Query reverse engineering

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

ER rules L

?

Query answers

result

pid	pid
p1	p2
p3	p4

Searching for the **rules (queries)** give rise to the **ER answers** on a **database instance**

Research aim and questions

Learning ER rules $\sqsubseteq$ Query reverse engineering (QRE) $\sqsubseteq$ Inductive logic programming (ILP)

Research aim and questions

Learning ER rules $\sqsubseteq$ Query reverse engineering (QRE) $\sqsubseteq$ Inductive logic programming (ILP)

General goal:
taking inspirations from **ILP** and **QRE** to develop a **theoretically sound** method on **learning collective ER rules**

Research aim and questions

Learning ER rules ⊑ Query reverse engineering (QRE) ⊑ Inductive logic programming (ILP)

General goal:
taking inspirations from **ILP** and **QRE** to develop a **theoretically sound** method on **learning collective ER rules**

Searching for the **rules (queries)** give rise to the **ER answers** on a **database instance**

1. What **language**?
2. What is an **ER answer** under such language?
3. What are the **structure** of interest?
4. How can we **characterise** the structure to **facilitate searching**?

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: tuple relational calculus + LACE framework

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: **tuple relational calculus** + LACE framework

Schema $\mathcal{S} = \{Doctor[did, name, dept, phone], \ldots, Appointment[did, pid]\}$

order invariant

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: **tuple relational calculus** + LACE framework

Schema $\mathcal{S} = \{Doctor[did, name, dept, phone], \ldots, Appointment[did, pid]\}$

order invariant

Tuple E.g. *{pid: J.Smith, age: 32, phone:12345, allergy: brupen}*

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: **tuple relational calculus** + LACE framework

Specification L

$$x[pid]=y[pid] <= Patient(x), Patient(y), x[name]\approx y[name], x[age,phone]=y[age,phone]$$

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: **tuple relational calculus** + LACE framework

Specification L

$$x[pid]=y[pid] <= \textbf{Patient(x), Patient(y)}, x[name]\approx y[name], x[age,phone]=y[age,phone]$$

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: **tuple relational calculus** + LACE framework

Specification L

$x[pid]=y[pid] <= Patient(x), Patient(y), \mathbf{x[name] \approx y[name], x[age,phone]=y[age,phone]}$

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: tuple relational calculus + **LACE framework**

Specification L

Hard rule: **evident** conditions for **mandatory** merge

$x[pid]=y[pid] <= Patient(x), Patient(y), x[name] \approx y[name], x[age,phone]=y[age,phone]$

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: tuple relational calculus + **LACE framework**

Specification L

Hard rule: **evident** conditions for **mandatory** merge

$x[pid]=y[pid] <= Patient(x), Patient(y), x[name] \approx y[name], x[age,phone]=y[age,phone]$

Soft rule: **less certain** conditions for **likely** merges

$x[name]=y[name] <\sim Doctor(x), Doctor(y), App(w), App(z), x[did]=w[did],$
$y[did]=z[did], w[pid]=z[pid], x[dept,phone]=y[dept,phone]$

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: tuple relational calculus + **LACE framework**

Specification L

Hard rule: **evident** conditions for **mandatory** merge

$x[pid]=y[pid] <= Patient(x), Patient(y), x[name] \approx y[name], x[age,phone]=y[age,phone]$

Soft rule: **less certain** conditions for **likely** merges

$x[name]=y[name] <\sim Doctor(x), Doctor(y), App(w), App(z), x[did]=w[did],$
$y[did]=z[did], w[pid]=z[pid], x[dept,phone]=y[dept,phone]$

Previously $EqV(<i,2>,<i',2>)$

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Rule language: tuple relational calculus + **LACE framework**

Specification L

Hard rule: **evident** conditions for **mandatory** merge

$x[pid]=y[pid] <= Patient(x), Patient(y), x[name] \approx y[name], x[age,phone]=y[age,phone]$

Soft rule: **less certain** conditions for **likely** merges

$x[name]=y[name] <\sim Doctor(x), Doctor(y), App(w), App(z), x[did]=w[did],$
$y[did]=z[did], w[pid]=z[pid], x[dept,phone]=y[dept,phone]$

Constraint: enforce consistency

$\perp <= Patient(x), Patient(y), x[pid]=y[pid], x[name] \neq y[name]$

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Specification L

Hard rule: $x[pid]=y[pid] \leftarrow Patient(x), Patient(y), x[name] \approx y[name], x[age,phone]=y[age,phone]$

Soft rule: $x[name]=y[name] \leftarrow\sim Doctor(x), Doctor(y), App(w), App(z), x[did]=w[did], y[did]=z[did], w[pid]=z[pid], x[dept,phone]=y[dept,phone]$

Constraint: $\perp \leftarrow Patient(x), Patient(y), x[pid]=y[pid], x[name] \neq y[name]$

EqRel E $E = \{\}$

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Specification L

Hard rule: $x[pid]=y[pid] \leftarrow Patient(x), Patient(y), x[name] \approx y[name], x[age,phone]=y[age,phone]$

Soft rule: $x[name]=y[name] \leftarrow\sim Doctor(x), Doctor(y), App(w), App(z), x[did]=w[did], y[did]=z[did], w[pid]=z[pid], x[dept,phone]=y[dept,phone]$

Constraint: $\bot \leftarrow Patient(x), Patient(y), x[pid]=y[pid], x[name] \neq y[name]$

EqRel E $\quad E = \{(p1,p2), (p3,p4), (d1,d2)\} \longrightarrow$ A *solution* is an E s.t. D_E satisfies **H** and **C**

The language

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

Specification L

Hard rule: $x[pid]=y[pid] <= Patient(x), Patient(y), x[name] \approx y[name], x[age,phone]=y[age,phone]$

Soft rule: $x[did]=y[did] <= \ldots y[did,pid], y[did,pid]=z[did,\ldots$

Observation: ER rule only talks about **conditions** are **relevant to the head**

Connectivity

Definition 1: *Connected rule*

An ER rule is **connected** iff there does not exist disjoint variables in its rule body.

- x and y are the same tuple variable
- if $x \neq y$ and there exists an atom(s) $x[A] \otimes y[B]$
- there exists atoms $x[A] = c$ and $y[B] = c$
- there exists a tuple variable z, s.t. x is connected to z and z is connected to y

Connectivity

Definition 1: *Connected rule*

An ER rule is **connected** iff there does not exist disjoint variables in its rule body.

- x and y are the same tuple variable
- if $x \neq y$ and there exists an atom(s) $x[A] \otimes y[B]$
- there exists atoms $x[A] = c$ and $y[B] = c$
- there exists a tuple variable z, s.t. x is connected to z and z is connected to y

Ensure all body conditions **are relevant** and **at least one comparison**

Connectivity

Database D

Appointment

did	pid
d1	p1
d2	p2
d3	p3
d3	p4

Patient

pid	name	age	phone	allergy
p1	J.Smith	32	12345	brupen
p2	John Smith	32	12345	aspirin
p3	Jerry Smith	32	6789	ibuprofen
p4	J.Smith	32	6789	ibuprofen

Doctor

did	name	dept	phone
d1	ブラックジャック	surgeon	66677
d2	Black jack	surgeon	66677
d3	Tezuka	skin	77333

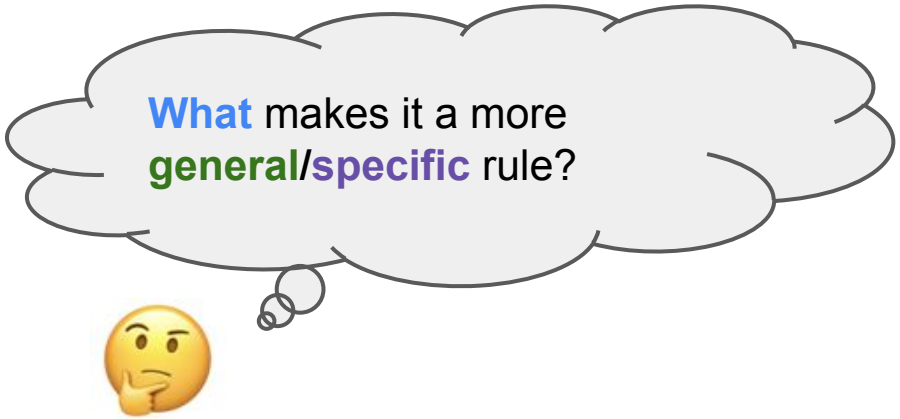
Connected: *x[did]=y[did] <~Doctor(x), Doctor(y), App(w), App(z), x[did]=w[did], y[did]=z[did], w[pid]=z[pid], x[dept,phone]=y[dept,phone]*

Not Connected:

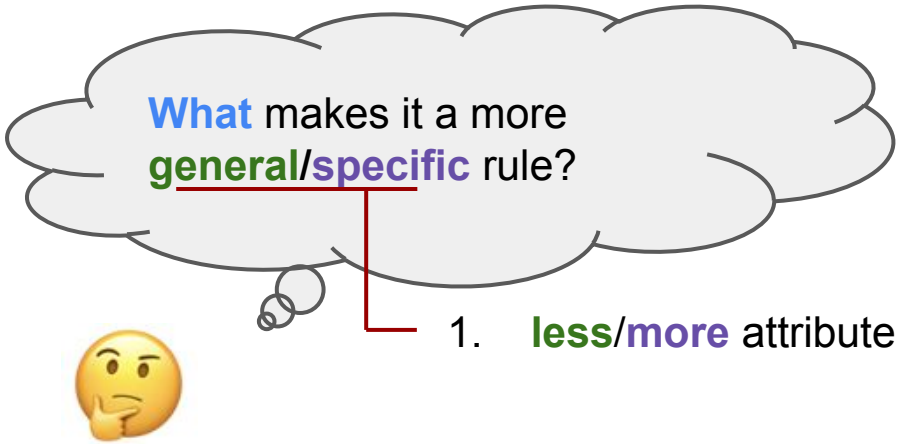
x[did]=y[did] <~Doctor(x), Doctor(y), Patient(w), Patient(z), w[name]≈z[name].

x[did]=y[did] <~Doctor(x), Doctor(y), App(w), App(z), x[did]=w[did], y[did]=z[did] .

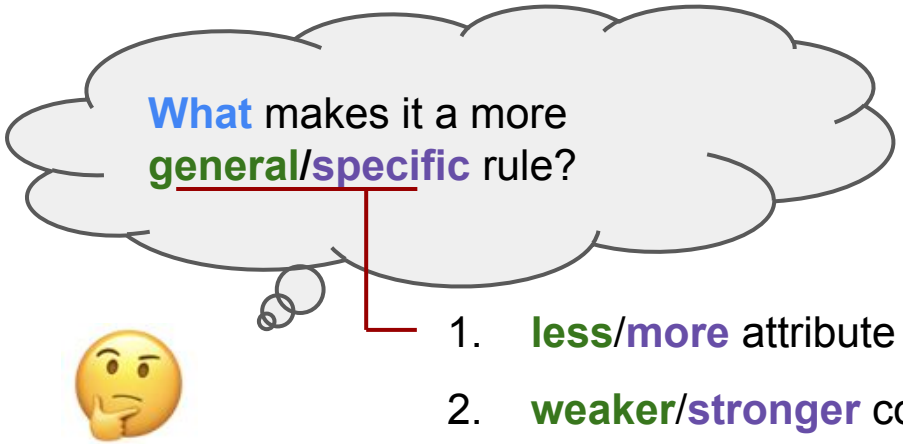
Subsumption order



Subsumption order



Subsumption order



1. **less**/**more** attribute comparisons
2. **weaker**/**stronger** comparisons applied

Subsumption order

Example: Consider following ER rules over schema $\mathcal{S}=\{R[A,B]\}$

$r_1: \; x[A] = y[A] \leftarrow R(x), R(y), x[B] = y[B].$

$r_2: \; x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[B] = z[B].$

$r_3: \; x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[A] = z[A], y[B] = z[B].$

From previous intuition, it should be the case that $r_3 \succ r_2 \succ r_1$

But r_3 is actually **the same** rule as r_1 ?

Subsumption order

Example: Consider following ER rules over schema $\mathcal{S}=\{R[A,B]\}$

$r_1: x[A] = y[A] \leftarrow R(x), R(y), x[B] = y[B].$

$r_2: x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[B] = z[B].$

$r_3: x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[A] = z[A], y[B] = z[B].$

From previous intuition, it should be the case that r_1

But r_3 is actually **the same** rule as r_1 ?

In **ILP**, subsumption can be defined as **substitution**

Subsumption order

Definition 2: *subsumption order* $\geq$

$r' \geq r$ (more specific) if there exist a mapping $g : var(r) \rightarrow var(r')$, s.t. the rule r_g satisfies:

- *head*(r_g) = *head*(r')
- Every relational atom $a \in$ *body*(r_g) is in *body*(r')
- *body*$(r_g) \subseteq$ *body*(r') or for every comparison atom $a \in$ *body*(r_g), there exists $a' \in$ *body*(r'), s.t. $a' \vDash a$.

Subsumption order

Definition 2: *subsumption order* $\geq$

$r' \geq r$ (more specific) if there exist a mapping $g : var(r) \rightarrow var(r')$, s.t. the rule r_g satisfies:

- *head*(r_g) = *head*(r')
- Every relational atom $a \in$ *body*(r_g) is in *body*(r')
- *body*$(r_g) \subseteq$ *body*(r') or for every comparison atom $a \in$ *body*(r_g), there exists $a' \in$ *body*(r'), s.t. $a' \vDash a$.

Subsumption order

Example: Consider following ER rules over database schema $\mathcal{S} = \{R[A,B]\}$

$r_1 : x[A] = y[A] \leftarrow R(x), R(y), x[B] = y[B].$

$r_2 : x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[B] = z[B].$

$r_3 : x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[A] = z[A], y[B] = z[B].$

- For $\boldsymbol{r_3 \geq r_1}$, let $g_{r_1 \to r_3} = \{x \to x, y \to y\}$, we have $r_g = r_1$.

Subsumption order

Example: Consider following ER rules over database schema $\mathcal{S} = \{R[A,B]\}$

$r_1 : \; x[A] = y[A] \leftarrow R(x), R(y), x[B] = y[B].$

$r_2 : \; x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[B] = z[B].$

$r_3 : \; x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[A] = z[A], y[B] = z[B].$

- For $\boldsymbol{r_3 \geq r_1}$, let $g_{r_1 \to r_3} = \{x \to x, y \to y\}$, we have $r_g = r_1$.
- For $\boldsymbol{r_1 \geq r_3}$, let $g_{r_3 \to r_1} = \{x \to x, y \to y, z \to y\}$,

  we have $r_g = x[A] = y[A] \leftarrow R(x), R(y), R(y), x[B] = y[B], y[A] = y[A], y[B] = y[B] \Leftrightarrow r_1$.

Subsumption order

Example: Consider following ER rules over database schema $\mathcal{S} = \{R[A,B]\}$

$r_1 : \; x[A] = y[A] \leftarrow R(x), R(y), x[B] = y[B].$

$r_2 : \; x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[B] = z[B].$

$r_3 : \; x[A] = y[A] \leftarrow R(x), R(y), R(z), x[B] = y[B], y[A] = z[A], y[B] = z[B].$

- For $\boldsymbol{r_3 \geq r_1}$, let $g_{r_1 \to r_3} = \{x \to x, y \to y\}$, we have $r_g = r_1$.
- For $\boldsymbol{r_1 \geq r_3}$, let $g_{r_3 \to r_1} = \{x \to x, y \to y, z \to y\}$,

  we have $r_g = x[A] = y[A] \leftarrow R(x), R(y), R(y), x[B] = y[B], y[A] = y[A], y[B] = y[B] \Leftrightarrow r_1$.
- For $\boldsymbol{r_1 \geq r_2}$, let $g_{r_2 \to r_1} = \{x \to x, y \to y, z \to y\}$,

  we have $r_g = x[A] = y[A] \leftarrow R(x), R(y), R(y), x[B] = y[B], y[B] = y[B] \Leftrightarrow r_1$.

Subsumption order

Definition 2: *subsumption order* $\geq$

$r' \geq r$ (more specific) iff there exist a mapping $g : var(r) \rightarrow var(r')$, s.t. the rule r_g satisfies:

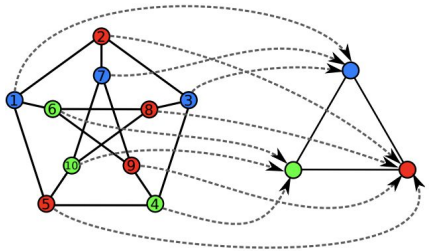
- *head*(r_g) = *head*(r')
- Every relational atom $a \in$ *body*(r_g) is in *body*(r')
- *body*$(r_g) \subseteq$ *body*(r') or for every comparison atom $a \in$ *body*(r_g), there exists $a' \in$ *body*(r'), s.t. $a' \vDash a$.

In fact, this also aligns with **homomorphism** theorem in **database theory.**

r' is subsumed by r ⇔

existences of mapping g ⇔

answers of (body query of) r' is **contained by** answers of r



Subsumption order

Theorem:

Fixing a rule r_h. Let Γ be the set of all possible **connected** rules built from r_h. Then the subsumption ordered set **$<\Gamma, \geq>$ is a lattice**.

Subsumption order

Theorem:

Fixing a rule r_h. Let Γ be the set of all possible **connected** rules built from r_h. Then the subsumption ordered set **$<\Gamma,\geq>$ is a lattice**.

glb (least generalisation) and **lub (greatest specialisation)**

exist for any **finite set** of rules built from r_h

Refinement operator

An ***(upward) refinement operator***
over the quasi-ordered space $\langle \Gamma, \succeq \rangle$ is a function ρ, s.t. $\rho(r) \subseteq \{r' | r' \succeq r\}$, for every $r \in \Gamma$.

Refinement operator

An ***(upward) refinement operator***
over the quasi-ordered space $\langle \Gamma, \succeq \rangle$ is a function ρ, s.t. $\rho(r) \subseteq \{r' | r' \succeq r\}$, for every $r \in \Gamma$.

n-step refinements

- For $n = 1$, $\rho_1(r) = \rho(r)$.
- For $n \geq 2$, $\rho_n(r) = \{r' | r \in \rho_{n-1}(r), r' \in \rho(r)\}$.

Refinements $\rho^*(r) = \rho_1(r) \cup \ldots \cup \rho_n(r)$.

Refinement operator

An ***(upward) refinement operator***
over the quasi-ordered space $\langle \Gamma, \succeq \rangle$ is a function ρ, s.t. $\rho(r) \subseteq \{r' | r' \succeq r\}$, for every $r \in \Gamma$.

n-step refinements

- For $n = 1$, $\rho_1(r) = \rho(r)$.
- For $n \geq 2$, $\rho_n(r) = \{r' | r \in \rho_{n-1}(r), r' \in \rho(r)\}$.

Refinements $\rho^*(r) = \rho_1(r) \cup \ldots \cup \rho_n(r)$.

Desired properties

- **locally finite**: each application results to a finite set.
- **complete**: for every r' s.t. $r' \succ r$, there exists $r^* \in \rho^*(r)$, s.t. $r^* \sim r'$.
- **proper**: every $r' \in \rho(r)$ is strictly more specific than r.

Refinement operator

An ***(upward) refinement operator***
over the quasi-ordered space $\langle \Gamma, \succeq \rangle$ is a function ρ, s.t. $\rho(r) \subseteq \{r' | r' \succeq r\}$, for every $r \in \Gamma$.

n-step refinements

- For $n = 1$, $\rho_1(r) = \rho(r)$.
- For $n \geq 2$, $\rho_n(r) = \{r' | r \in \rho_{n-1}(r), r' \in \rho(r)\}$.

Refinements $\rho^*(r) = \rho_1(r) \cup \ldots \cup \rho_n(r)$.

Desired properties

- **locally finite**: each application results to a finite set.
- **complete**: for every r' s.t. $r' \succ r$, there exists $r^* \in \rho^*(r)$, s.t. $r^* \sim r'$.
- **proper**: every $r' \in \rho(r)$ is strictly more specific than r.

Refinement operator for ER rules

Given r, we consider the 1-step refinements $\rho(r)$ as follows:

- No comparison on (compatible) attributes A, B exists
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{x[A] \approx y[B]\}$.

Refinement operator for ER rules

Given r, we consider the 1-step refinements $\rho(r)$ as follows:

- No comparison on (compatible) attributes A, B exists
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{x[A] \approx y[B]\}$.
- Exists $x[A] \approx y[B]$
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) / \{x[A] \approx y[B]\} \cup \{x[A] = y[B]\}$

Refinement operator for ER rules

Given r, we consider the 1-step refinements $\rho(r)$ as follows:

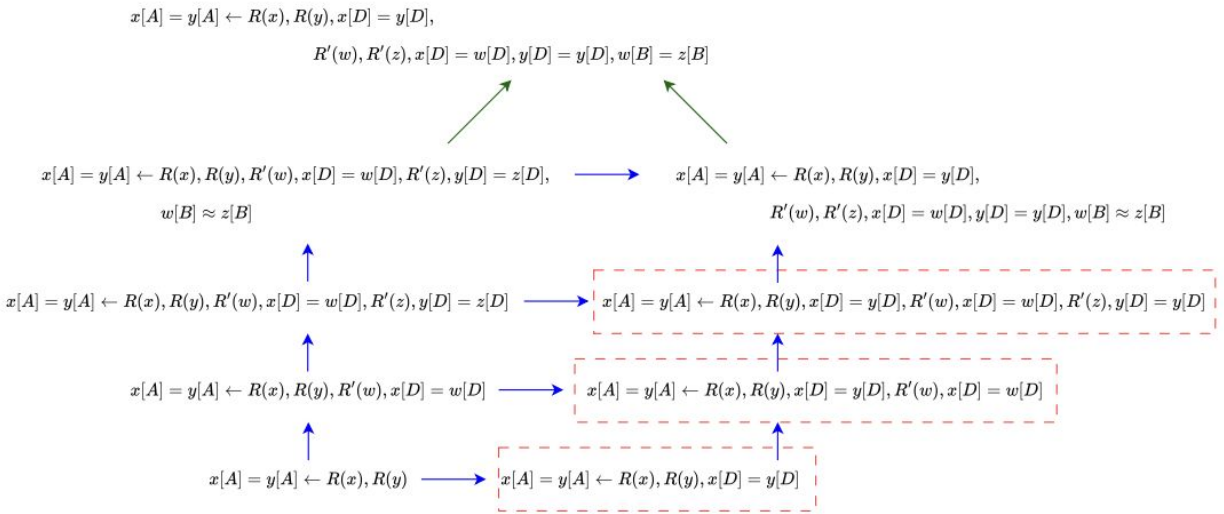
- No comparison on (compatible) attributes A, B exists
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{x[A] \approx y[B]\}$.
- Exists $x[A] \approx y[B]$
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \,/\, \{x[A] \approx y[B]\} \cup \{x[A] = y[B]\}$
- Exists $x[A] = y[B]$ or $y[B] = c$.
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{x[A] = c\}$ or $body(r) \cup \{y[B] = c\}$

Refinement operator for ER rules

Given r, we consider the 1-step refinements $\rho(r)$ as follows:

- No comparison on (compatible) attributes A, B exists
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{x[A] \approx y[B]\}$.
- Exists $x[A] \approx y[B]$
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \,/\, \{x[A] \approx y[B]\} \cup \{x[A] = y[B]\}$
- Exists $x[A] = y[B]$ or $y[B] = c$.
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{x[A] = c\}$ or $body(r) \cup \{y[B] = c\}$
- Exist table reference from R to R' on attribute A
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{R'(y), x[A] = y[A]\}$.

Refinement operator for ER rules



Refinements example to merge $R[A]$ on schema $\mathcal{S} = \{R[A, D], R'[D, B]\}$

Refinement operator for ER rules

Given r, we consider the 1-step refinements $\rho(r)$ as follows:

- No comparison on (compatible) attributes A, B exists
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{x[A] \approx y[B]\}$.
- Exists $x[A] \approx y[B]$
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) / \{x[A] \approx y[B]\} \cup \{x[A] = y[B]\}$
- Exists $x[A] = y[B]$ or $y[B] = c$.
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{x[A] = c\}$ or $body(r) \cup \{y[B] = c\}$
- Exist table reference from R to R' on attribute A
  - Let $\rho(r)$ contains r', s.t. $body(r') = body(r) \cup \{R'(y), x[A] = y[A]\}$.

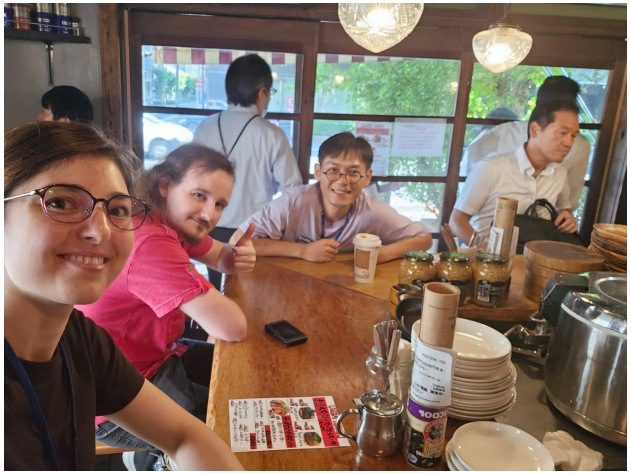
(at a glance) if consider only **finite set of constants**, our ρ is **locally finite**, and **complete**, though **not proper**.

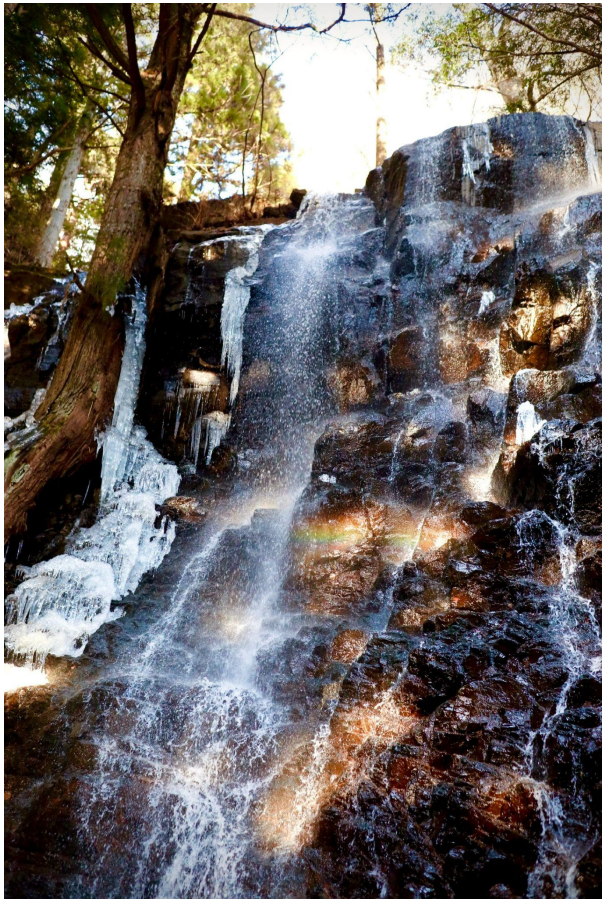
Conclusion

- We study **learning ER rules** under **LACE** framework using **tuple relational calculus**.
- We defined **connectivity** of ER rules restricting conditions to be **relevant**.
- We defined **subsumption ordering search space** of all possible rules, which induces **a lattice structure** and **guarantee** that if one rule is **more general** than the other, then **set containment holds** for **answer sets** of the rules.
- We defined an **upward refinement operator** for navigating the subsumption ordered rule space, and showed that such operator is **locally finite** and **complete**, though **not proper**.

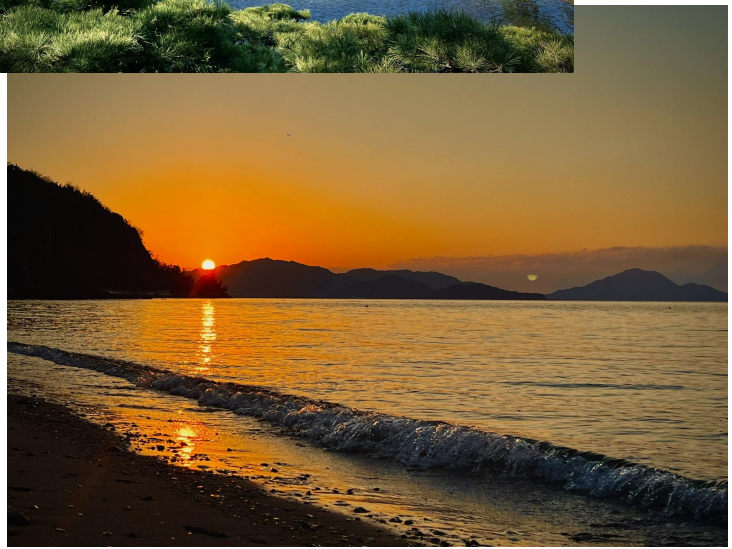
Ongoing & future work

- Characterise **hard** and **soft** rules under a subsumption ordered rule set
- **Learning algorithm** taking into account **LACE semantics**:
  - Meta program *[Inoue et al. 2019 IJCAI]*
  - Fitting Algorithms for Conjunctive Queries *[ten Cate et al. 2024 SIGMOD]*
- Experiments





富士河口湖



高松&男木島

日光