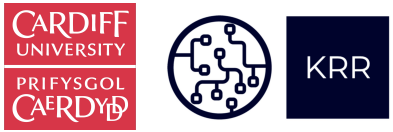
ASPEN: ASP-Based System for Collective Entity Resolution

Zhiliang (Leon) Xiang, Meghyn Bienvenu, Gianluca Cima,
Víctor Gutiérrez Basulto, Yazmín Ibáñez-García



Cardiff Knowledge Representation & Reasoning (KRR) Group

the LaBRI Research Lab

Sapienza University of Rome

Background

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

  Example: medical records of patients

  John Smith

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

  Example: medical records of patients

  John Smith is allergic to *aspirin*

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

  Example: medical records of patients

  John Smith is allergic to *aspirin*

  J. Smith

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

  Example: medical records of patients

  John Smith is allergic to *aspirin*

  J. Smith is allergic to *ibuprofen*

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

  Example: medical records of patients

  John Smith is allergic to *aspirin*

  J. Smith is allergic to *ibuprofen*

- **Central task** in **data quality**, **long-standing challenge**

  erroneous/missing values, syntactic variants

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

  Example: medical records of patients

  J. Smith is allergic to *ibuprofen*, ***John Smith*** is allergic to *aspirin*

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

  Example 1: medical database

  J. Smith is allergic to *ibuprofen*, ***John Smith*** is allergic to *aspirin*

  Example 2: bibliographical database

  Zhiliang Xiang has a paper in *KR*, ***Zhiliang Xiang*** has a paper in *Politics*

Entity Resolution (ER)

- **Entity resolution (ER):** identify **different constants** denoting **the same thing**

  Example: medical database

  J. Smith is allergic to *ibuprofen*, ***John Smith*** is allergic to *aspirin*

- **Central task** in **data quality**, **long-standing challenge**

  erroneous/missing values, syntactic variants

Entity Resolution (ER)

In **database (DB) context**, many approaches: **ML**, **probabilistic**, **logic**

A recent ER framework: LACE

A Logical Approach to Collective Entity Resolution

[Bienvenu et al. PODs 2022, KR&IJCAI 2023]

- **declarative:** logical **rules** and **constraints**
- **collective: interdependencies**, *doctor* merges trigger hospital merges, vice versa
- **explainable**: "**why** are *J. Smith* and *John Smith* deemed to be the **same** patient?"

LACE specification

Album

id	artist	title	barcode
ab1	a3	Born to Die	22334455
ab2	a2	Born to DIe (de)	
ab3	a1	Born to DIE	22334455
ab4	a2	Born to DIE (deluxe)	22334466

Artist

id	name	born year
a1	Lana Del Rey	21/6/1985
a2	LANA DEL REY	21/6/1985
a3	Lana D. Rey	211/6/1985

LACE specification

A **LACE specification Σ** over **database D** takes the form **(*H* , *S* , *C*)**

Album

id	artist	title	barcode
ab1	a3	Born to Die	22334455
ab2	a2	Born to DIe (de)	
ab3	a1	Born to DIE	22334455
ab4	a2	Born to DIE (deluxe)	22334466

Artist

id	name	born year
a1	Lana Del Rey	21/6/1985
a2	LANA DEL REY	21/6/1985
a3	Lana D. Rey	211/6/1985

LACE specification

A **LACE specification Σ** over **database D** takes the form (***H*** **,** ***S*** **,** ***C***)

Album

id	artist	title	barcode
ab1	a3	Born to Die	22334455
ab2	a2	Born to DIe (de)	
ab3	a1	Born to DIE	22334455
ab4	a2	Born to DIE (deluxe)	22334466

Artist

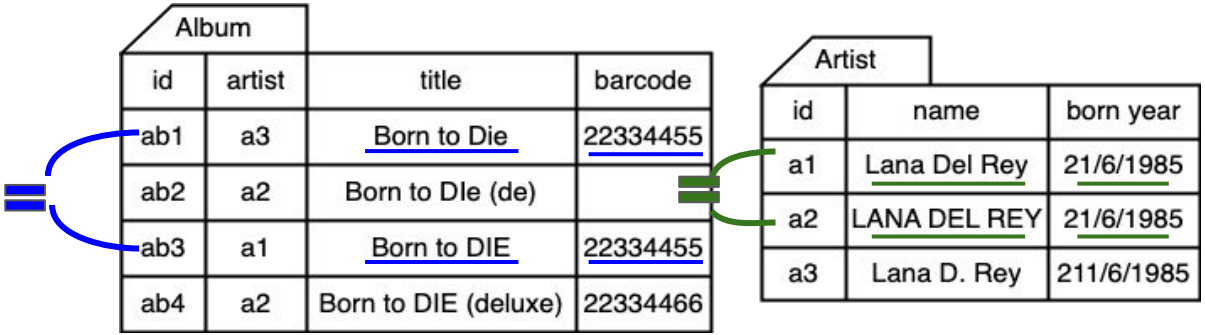
id	name	born year
a1	Lana Del Rey	21/6/1985
a2	LANA DEL REY	21/6/1985
a3	Lana D. Rey	211/6/1985

H: hard rules indicate **mandatory** merges

- $Album(x,a,t,\mathbf{b}) \wedge Album(y,a',t',\mathbf{b}) \wedge \mathbf{t \approx t'} \Rightarrow Eq(x,y)$

LACE specification

A **LACE specification Σ** over **database D** takes the form (***H*** , ***S*** , ***C***)



Album

id	artist	title	barcode
ab1	a3	Born to Die	22334455
ab2	a2	Born to DIe (de)	
ab3	a1	Born to DIE	22334455
ab4	a2	Born to DIE (deluxe)	22334466

Artist

id	name	born year
a1	Lana Del Rey	21/6/1985
a2	LANA DEL REY	21/6/1985
a3	Lana D. Rey	211/6/1985

H: hard rules indicate **mandatory** merges

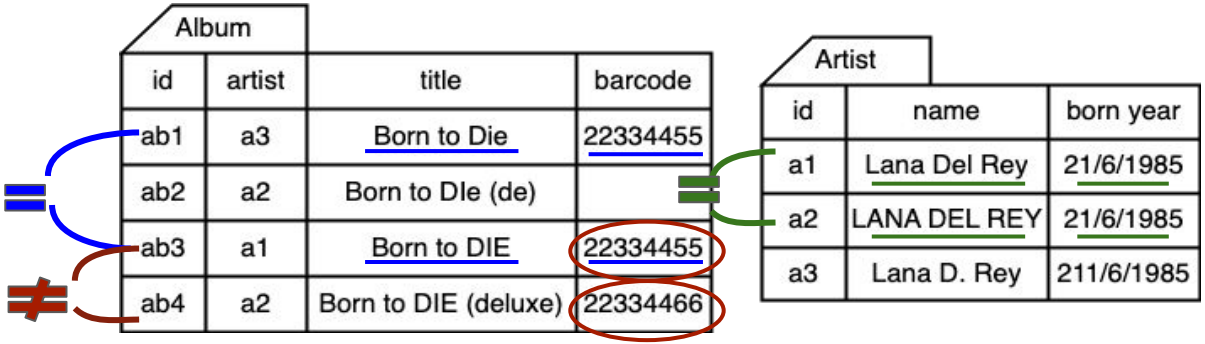
- *Album(x,a,t,**b**) ∧ Album(y,a',t',**b**) ∧ **t≈t' =>** Eq(x,y)*

S: soft rules indicate **likely** merges

- *Artist(x,n,**b**) ∧ Artist(y,n',**b**) ∧ **n≈n' ~>** Eq(x,y)*
- *Album(x,**a**,t,b) ∧ Album(y,**a**,t',b') ∧ **t≈t' ~>** Eq(x,y)*

LACE specification

A **LACE specification Σ** over **database D** takes the form (***H*** , ***S*** , ***C***)



Album

id	artist	title	barcode
ab1	a3	Born to Die	22334455
ab2	a2	Born to DIe (de)	
ab3	a1	Born to DIE	22334455
ab4	a2	Born to DIE (deluxe)	22334466

Artist

id	name	born year
a1	Lana Del Rey	21/6/1985
a2	LANA DEL REY	21/6/1985
a3	Lana D. Rey	211/6/1985

H: hard rules indicate **mandatory** merges

- $Album(x,a,t,\mathbf{b}) \wedge Album(y,a',t',\mathbf{b}) \wedge \mathbf{t \approx t'} \Rightarrow Eq(x,y)$

S: soft rules indicate **likely** merges

- $Artist(x,n,\mathbf{b}) \wedge Artist(y,n',\mathbf{b}) \wedge \mathbf{n \approx n'} \rightsquigarrow Eq(x,y)$
- $Album(x,\mathbf{a},t,b) \wedge Album(y,\mathbf{a},t',b') \wedge \mathbf{t \approx t'} \rightsquigarrow Eq(x,y)$

C: constraints enforce **consistency**: $Album(\mathbf{x},a,t,\mathbf{b}) \wedge Album(\mathbf{x},a',t',\mathbf{b'}) \wedge \mathbf{b \neq b'} \Rightarrow \bot$

Semantics of LACE

Semantics (informal):

- ***Eq/2*** is **reflexive**, **symmetric** and **transitive**, induce ***equivalence relation E***

Semantics of LACE

Semantics (informal):

- ***Eq/2*** is **reflexive**, **symmetric** and **transitive**, induce ***equivalence relation E***
  - Evaluate rule bodies on **induced database D_E**

Semantics of LACE

Semantics (informal):

- ***Eq/2*** is **reflexive**, **symmetric** and **transitive**, induce ***equivalence relation E***
  - Evaluate rule bodies on **induced database D_E**
  - Construct **step-by-step**

Semantics of LACE

Semantics (informal):

- ***Eq/2*** is **reflexive**, **symmetric** and **transitive**, induce ***equivalence relation E***
  - Evaluate rule bodies on **induced database D_E**
  - Construct **step-by-step**
  - **Solution** for *(D,Σ)* is an **equivalence relation *E*** *(over constants in D)*, s.t. **D_E satisfies H and C**

Multiple solutions -> focus on the best ones, (**set-inclusion**) **maximal solution**

Computational problems in LACE

Main decision problems:

- ***existence*** *and* ***recognition*** *of* ***(maximal) solutions***
- **possible** / **certain** merge: present in **some** / **all** maximal solutions

Computational problems in LACE

Main decision problems:

- ***existence*** *and* ***recognition*** *of* ***(maximal) solutions***
- **possible** / **certain** merge: present in **some** / **all** maximal solutions

Can be **correctly** computed via **ASP** !

Computational problems in LACE

Main decision problems:

- ***existence*** *and* ***recognition*** *of* ***(maximal) solutions***
- **possible** / **certain** merge: present in **some** / **all** maximal solutions

Can be **correctly** computed via **ASP** !

Computational problems in LACE

Main decision problems:

- ***existence** and **recognition** of **(maximal) solutions***
- **possible** / **certain** merge: present in **some** / **all** maximal solutions

Can be **correctly** computed via **ASP** ! But **implementation** is **absent** …

Computational problems in LACE

Main decision problems:

- ***existence*** *and* ***recognition*** *of* ***(maximal) solutions***
- **possible** / **certain** merge: present in **some** / **all** maximal solutions

Can be **correctly** computed via **ASP** ! But **implementation** is **absent** …

Goal: develop ASP-based system as implementation

ASP-Based System to ER

Construct ASP encoding

Album

id	artist	title	barcode
ab1	a3	Born to Die	22334455
ab2	a2	Born to DIe (de)	
ab3	a1	Born to DIE	22334455
ab4	a2	Born to DIE (deluxe)	22334466

Artist

id	name	born year
a1	Lana Del Rey	21/6/1985
a2	LANA DEL REY	21/6/1985
a3	Lana D. Rey	211/6/1985

$$Album(x,\mathbf{a},t,b) \wedge Album(y,\mathbf{a},t',b') \wedge t \approx t' \rightsquigarrow Eq(x,y)$$

Construct ASP encoding

- **Similarity atom** expanded with an **extra argument** denoting **threshold**

Album(x,a,t,b) ∧ Album(y,a,t',b') ∧ ***t≈t'*** *~> Eq(x,y)*

{eq(X,Y)}:- album(X,A,T,B), album(Y,A',T',B'), eq(A,A'), ***sim(T,T',S), S>=90****.*

Construct ASP encoding

- **Similarity atom** expanded with an **extra argument** denoting **threshold**
- Replace **join variable** by ***eq*-atom with distinguished variables**

*Album(x,**a**,t,b) ∧ Album(y,**a**,t',b') ∧ **t≈t'** ~> Eq(x,y)*

*{eq(X,Y)}:- album(X,**A**,T,B), album(Y,**A'**,T',B'), **eq(A,A')**, **sim(T,T',S)**, **S>=90**.*

Construct ASP encoding

- **Similarity atom** expanded with an **extra argument** denoting **threshold**
- Replace **join variable** by ***eq*-atom with distinguished variables**
- Encode **soft rules** using **choice head**

Album(x,a,t,b) ∧ Album(y,a,t',b') ∧ t≈t' ~> Eq(x,y)

{eq(X,Y)}:- album(X,A,T,B), album(Y,A',T',B'), eq(A,A'), sim(T,T',S), S>=90.

Construct ASP encoding

- **Similarity atom** expanded with an **extra argument** denoting **threshold**
- Replace **join variable** by ***eq*-atom with distinguished variables**
- Encode **soft rules** using **choice head**

*Album(x,**a**,t,b) ∧ Album(y,**a**,t',b') ∧ **t≈t' ~>** Eq(x,y)*

*{eq(X,Y)}:- album(X,**A**,T,B), album(Y,**A'**,T',B'), **eq(A,A')**, **sim(T,T',S), S>=90**.*

- Axiomatise ***eq*/2**:

Reflexivity: *eq(X,X) :- album(X,A,T,B).* **Symmetry:** *eq(X,Y) :- eq(Y,X).*

Transitivity: *eq(X,Y):- eq(X,Z), eq(Z,Y).*

Reasoning using ASP Encoding

Use the **ASP encoding** of a LACE specification (***H*** **,** ***S*** **,** ***C***), we can:

- Test **solution existence**, **generate one** or **more** if exist
- Generate **maximal solution** (s) with **set preference**
- Compute **possible merges** with brave reasoning
- Compute **certain merges**?

Reasoning using ASP Encoding

Use the **ASP encoding** of a LACE specification (***H , S , C***), we can:

- Test **solution existence**, **generate one** or **more** if exist
- Generate **maximal solution** (s) with **set preference**
- Compute **possible merges** with brave reasoning
- Compute **certain merges**?
  - Meta programming *[Gebser et al, TPLP'2011]*
  - WASP *[Alviano et al, AIJ'2023]*

Reasoning using ASP Encoding

Use the **ASP encoding** of a LACE specification (***H*** **,** ***S*** **,** ***C***), we can:

- Test **solution existence**, **generate one** or **more** if exist
- Generate **maximal solution** (s) with **set preference**
- Compute **possible/certain merges** (brave reasoning/?)

Transform **ASP encoding** to compute **approximate** set of **certain**/**possible** merges

Π_2^p-c NP-c

- **Lower bound (LB):** drop ***S*** and ***C***
- **Upper bound (UB)**: drop ***C***, “harden” ***S***

polynomial, with **unique** stable model!

Important ingredient: similarity facts

- **Not present** in **database**
- **Similarity measure**: call an external **scoring** function (e.g. edit distance) to compare constants and apply **threshold**
- **Naive**: score **all pair** of constants (quadratic), **infeasible** as data scales up

Important ingredient: similarity facts

- **Similarity measure**: call a **scoring** function (e.g. edit distance) to compare constants and apply **threshold**
- **Not present** in **database**
- **Naive**: score **all pair** of constants, **infeasible** as data scales up (quadratic)

Similarity computation

Observation:

E.g. *eq(X,Y):- album(X,**A**,T,B), album(Y,**A'**,T',B'), **eq(A,A')**, **sim(T,T',S), S>=90**.*

only need to compare similarity of ***titles*** of those albums have the **same artist**.

Similarity computation

Observation:

E.g. *eq(X,Y):- album(X,**A**,T,B), album(Y,**A'**,T',B'), **eq(A,A')**, **sim(T,T',S), S>=90**.*

only need to compare similarity of ***titles*** of those albums have the **same artist**.

Similarity computation

Observation:

E.g. *eq(X,Y):- album(X,**A**,T,B), album(Y,**A'**,T',B'), **eq(A,A')**, **sim(T,T',S), S>=90**.*

only need to compare similarity of ***titles*** of those albums have the **same artist**.

Intuition:

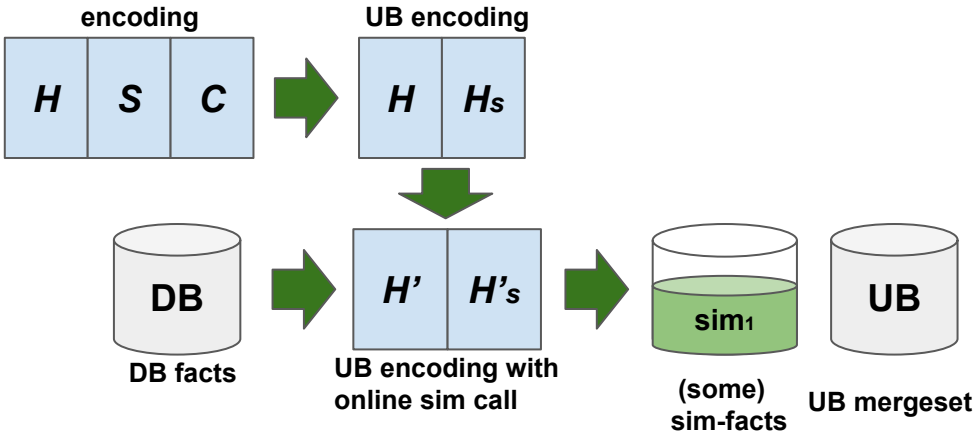
- take into account **structure of an encoding**
- compare all those may actually be **useful**, so that we can use the output to **compute all possible mergesets**!

Similarity computation

Phase 1: Compute a **UB encoding** making **online sim calls**

- Transform to **upper bound encoding**
- Make **online calls** to similarity functions

*E.g. eq(X,Y):- artist(X,N,B), artist(Y,N',B), **@sim(N,N')>=95**.*

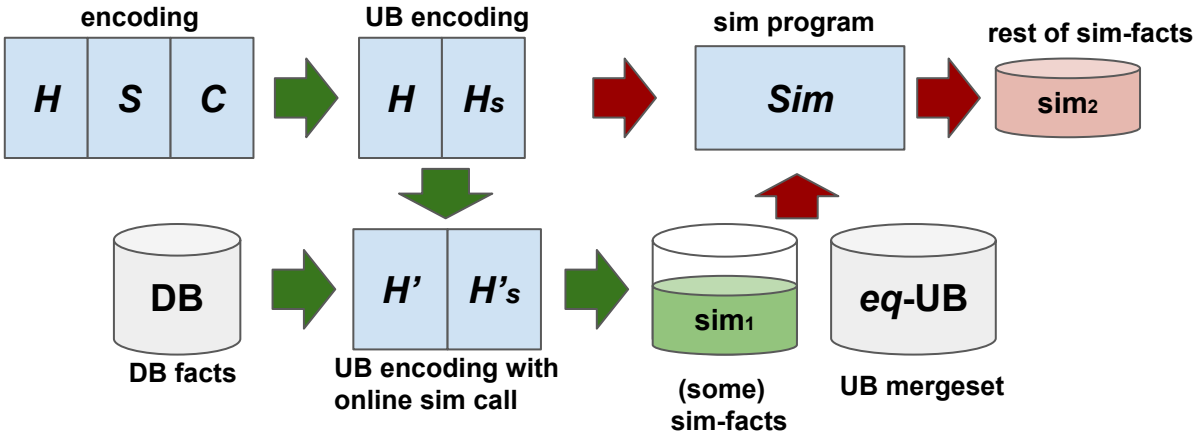


Similarity computation

Phase 1: Compute a **UB encoding** that makes **online sim calls**

Phase 2: **Retrieve** those have **not yet compared** using UB and **similarity program** (another transformation of UB encoding), then **evaluate** and **materialise**

E.g. ***sim(N,N')*** *:- artist(X,N,B), artist(Y,N',B),* ***eq(X,Y)****.*

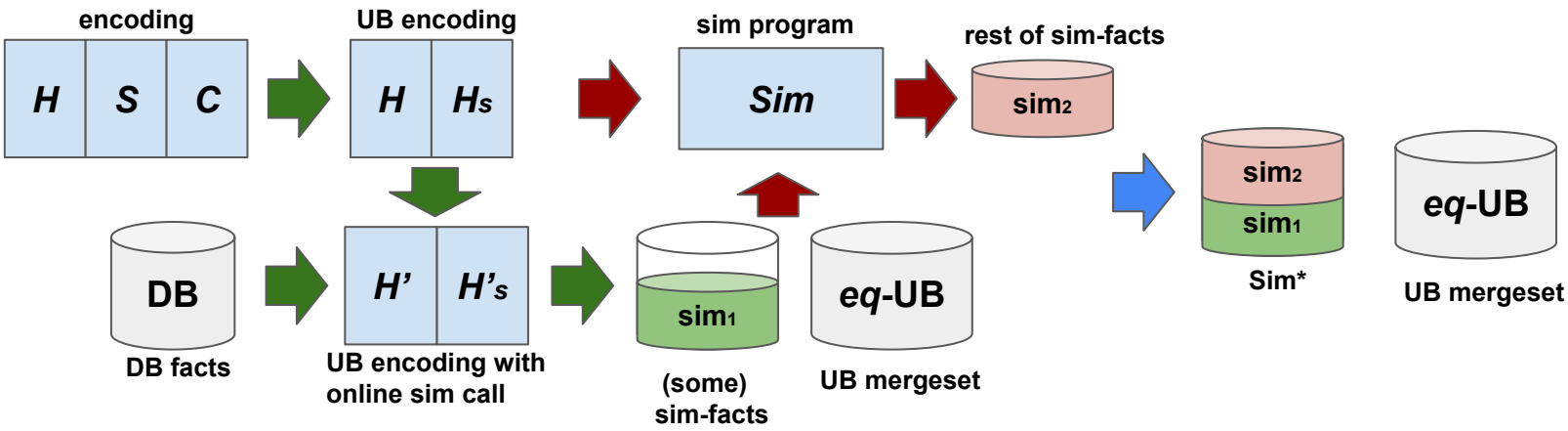


Similarity computation

Final output:

Sim* (Sim-facts required to compute UB) **& UB eq-facts**

Both can be reused later for computation of (diff'nt) mergesets!

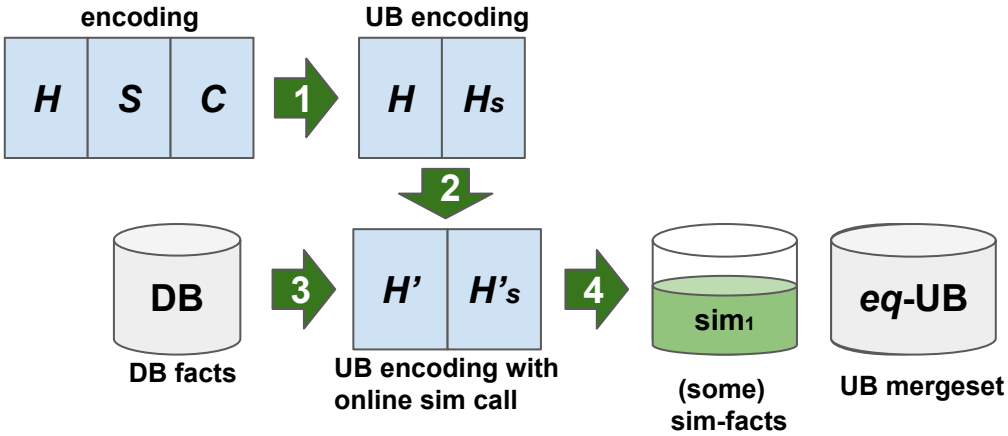


Similarity computation

Phase 1: Compute a **UB encoding** that makes **online sim calls**

- Transform to **upper bound encoding**
- Make **online calls** to external similarity functions

*E.g. eq(X,Y):- album(X,A,T,B), album(Y,A',T',B'), eq(A,A'), **@sim(T,T')>=90**.*

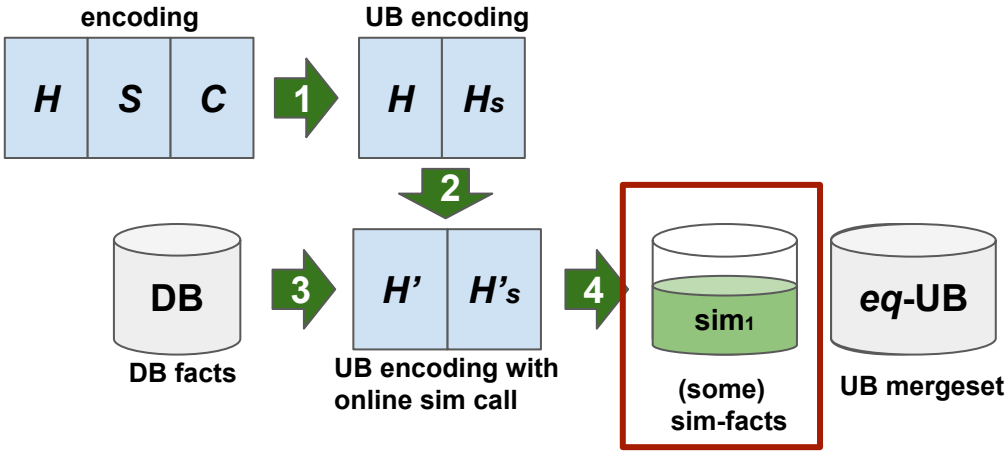


Similarity computation

Phase 1: Compute a **UB encoding** that makes **online sim calls**

- Transform to **upper bound encoding**
- Make **online calls** to external similarity functions

*E.g. eq(X,Y):- album(X,A,T,B), album(Y,A',T',B'), eq(A,A'), **@sim(T,T')>=90**.*

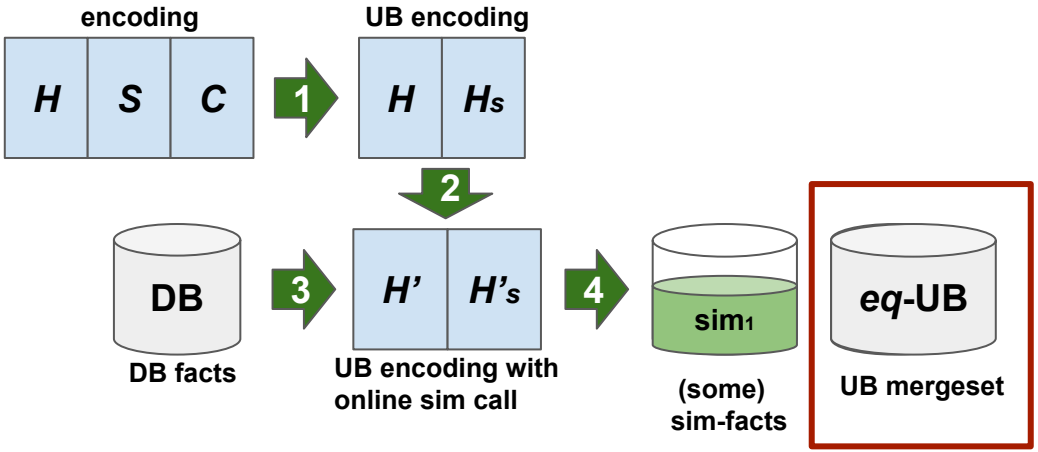


Similarity computation

Phase 1: Compute a **UB encoding** that makes **online sim calls**

- Transform to **upper bound encoding**
- Make **online calls** to external similarity functions

*E.g. eq(X,Y):- album(X,A,T,B), album(Y,A',T',B'), eq(A,A'), **@sim(T,T')>=90**.*

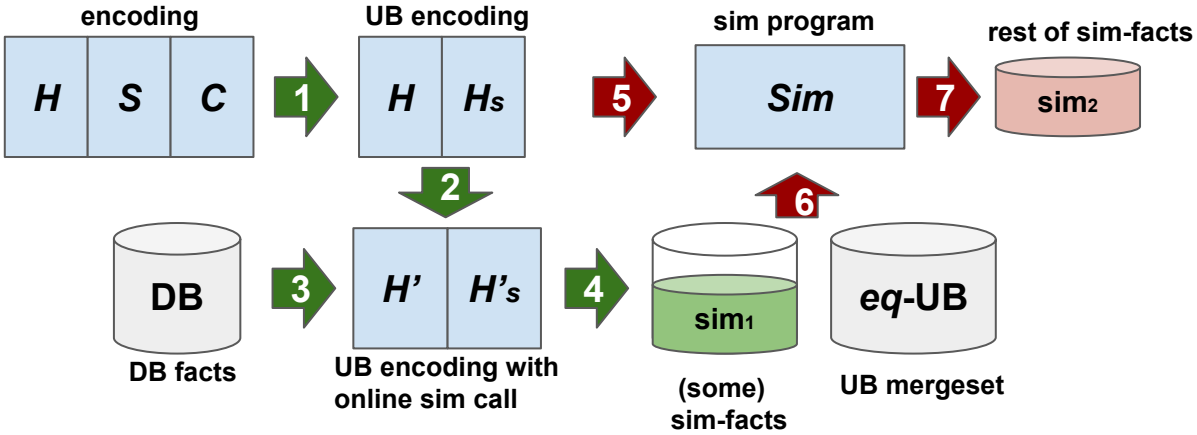


Similarity computation

Phase 1: Compute a **UB encoding** that makes **online sim calls**

Phase 2: **Retrieve** those have **not yet compared** using **UB mergeset** and **similarity program** (another transformation of the encoding), then **evaluate similarities** and **materialise**

E.g. ***sim(T,T')***:- *album(X,A,T,B), album(Y,A',T',B'), eq(A,A'),* ***eq(X,Y)***.

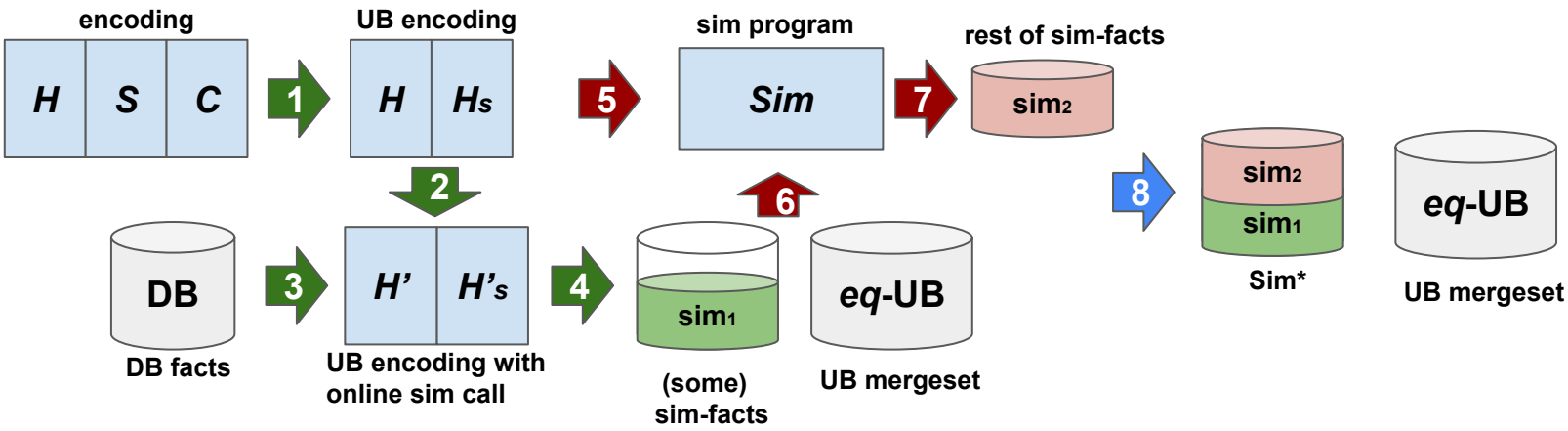


Similarity computation

Final output:

Sim* (all those may actually be **useful** to compute mergesets) **&** **UB eq-facts**

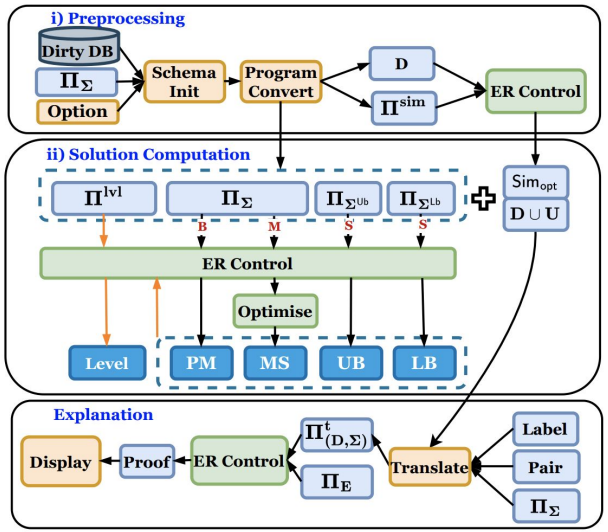
Both can be reused later for computation of (diff'nt) mergesets with proved correctness!



The ASPEn system

Input:

- Database, ASP Encoding
- Options specify desired output



Output:

- Merge lower/upper bound (LB/UB)
- Exact set of possible merge (PM)
- **Fixed # of** maximal solutions (**MS**, with asprin *[Brewka et al. AAAI'2015]*)
- Graphical explanation of a possible merge (s) (xclingo *[Cabalar et al. ICLP'2020]*)

Experimental setup

Datasets:

- 2 **single-table benchmarks** (DBLP-ACM, CORA)
- 4 **multi-table datasets** IMDB (movie), (**synthetic duplicates**) MUSIC, Pokemon

Rules and constraints:

- **Handcrafted**
- #Rule: 9 to 57
- #Constraint: 1 to 8

Name	#Rec	#Rel	#At	#Ref	#Dup
Dblp	5k	2	5^b	0	2.2k
Cora	1.9k	1	17	0	64k
Imdb	30k	5	22	4	6k
Mu	41k	11	72	12	15k
MuC	41k	11	72	12	15k
Poke	240k	20	104	20	4k

Experimental setup

- **Quality measure**: **F1/ Precision/ Recall**
- **Efficiency**: **Running times** / **Memory usage** (sim computation)

Experimental setup

- **Environments**: *clingo 5.5 Python API, a single 3.8GHz AMD Ryzen Threadripper 5965WX core and 128 GB of RAM*
- **Quality measure**: **F1/ Precision/ Recall**
- **Efficiency**: **Running times** (seconds) / **Memory usage** (sim computation)

Experiment 1: Sim computation

- Baseline: **naive computation** of sim
- Consistently **much more memory-efficient**
- **Runtimes** got **twice better** for **larger datasets**

Data	#At	#Cat	t_{cs}	t_{opt}	M_{cs}	M_{opt}	MRed.
Dblp	5	10.2M	**96.3**	531.4	512Mb	256Mb	50
Cora	17	0.8M	**5.49**	932.9	32Mb	4Mb	87.5
Imdb	22	19.6M	**89.5**	598.9	512Mb	128Mb	75
Mu	72	143.6M	**664.03**	772.3	4Gb	256Mb	93.8
MuC	72	147.1M	867.5	**446.4**	4Gb	256Mb	93.8
Poke	104	769.9M	9,419	**4,142**	32Gb	128Mb	99.6

Experiments: Main results

- **Consistently superior quality** compared to baselines (Magellan, JeDAI), **esp. multi-table**
- **Disadvantageous runtime** (*expensive preprocessing*)

Data	Method	F_1	(P / R)	t_o	t_g	t_s
Dblp	Magellan	79.97	89.80 72.08	**1.71**	-	-
	JedAI	95.02	**100** 90.51	49.16	-	-
	LB	46.09	97.10 30.21	531.63	**0.23**	**0.0039**
	MS-1	**96.21**	95.36 97.07	532.17	0.45	0.32
	PM	95.15	92.85 **97.57**	538.5	0.35	6.75
	UB	91.11	85.50 **97.57**	531.41	-	-
Cora	Magellan	79.70	93.09 69.68	131.13	-	-
	JedAI	90.53	95.53 87.72	**40.92**	-	-
	LB	83.57	**99.80** 71.87	1,008	**7.85**	**0.29**
	MS-1	**95.55**	94.25 96.79	1,031	20.88	10.50
	PM	**95.55**	94.25 96.79	1,857.6	20.19	837.58
	UB	87.50	79.66 **97.05**	999.87	-	-

single table

Data	Method	F_1	(P / R)	t_o	t_g	t_s
MuC	Magellan	55.54	97.51 38.83	66.87	-	-
	JedAI	32.75	73.95 21.02	**7.88**	-	-
	LB	53.95	**99.79** 36.97	474.56	**28.01**	**0.062**
	MS-1	**84.10**	88.11 80.44	562.37	113.64	2.33
	PM	83.87	87.59 80.46	893.44	113.26	333.78
	UB	83.55	86.01 **81.24**	446.4	-	-
Poke	Magellan	7.01	3.97 29.74	260.96	-	-
	JedAI	2.1	1.08 46.56	**23.46**	-	-
	LB	28.00	**100** 16.27	4,144	**2.29**	**0.018**
	MS-1	**88.71**	92.88 **84.90**	4,271.8	127.67	2.17
	PM	**88.71**	92.88 **84.90**	4,296	129.04	25.84
	UB	77.00	70.43 **84.90**	4,142	-	-

multi-table

Exp1: Sim computation

- Compared **our sim computation** with the **naive one (commonly used)**

Data	#At	#Cat	t_{cs}	t_{opt}	M_{cs}	M_{opt}	MRed.
Dblp	5	10.2M	**96.3**	531.4	512Mb	256Mb	50
Cora	17	0.8M	**5.49**	932.9	32Mb	4Mb	87.5
Imdb	22	19.6M	**89.5**	598.9	512Mb	128Mb	75
Mu	72	143.6M	**664.03**	772.3	4Gb	256Mb	93.8
MuC	72	147.1M	867.5	**446.4**	4Gb	256Mb	93.8
Poke	104	769.9M	9,419	**4,142**	32Gb	128Mb	99.6

M*{cs,opt}** - memory usage, ***MRed. - memory reduction %
t*{cs,opt}** - running times, ***#Cat - no. cross product of constants

Exp1: Sim computation

- Compared **our sim computation** with the **naive one (commonly used)**
- Consistently **much more memory-efficient**

Data	#At	#Cat	t_{cs}	t_{opt}	M_{cs}	M_{opt}	MRed.
Dblp	5	10.2M	**96.3**	531.4	512Mb	256Mb	50
Cora	17	0.8M	**5.49**	932.9	32Mb	4Mb	87.5
Imdb	22	19.6M	**89.5**	598.9	512Mb	128Mb	75
Mu	72	143.6M	**664.03**	772.3	4Gb	256Mb	93.8
MuC	72	147.1M	867.5	**446.4**	4Gb	256Mb	93.8
Poke	104	769.9M	9,419	**4,142**	32Gb	128Mb	99.6

M*{cs,opt}** - memory usage, ***MRed. - memory reduction %
t*{cs,opt}** - running times, ***#Cat - no. cross product of constants

Exp1: Sim computation

- Compared **our sim computation** with the **naive one (commonly used)**
- Consistently **much more memory-efficient**
- **Running times** got **twice better** for **larger datasets**

Data	#At	#Cat	t_{cs}	t_{opt}	M_{cs}	M_{opt}	MRed.
Dblp	5	10.2M	**96.3**	531.4	512Mb	256Mb	50
Cora	17	0.8M	**5.49**	932.9	32Mb	4Mb	87.5
Imdb	22	19.6M	**89.5**	598.9	512Mb	128Mb	75
Mu	72	143.6M	**664.03**	772.3	4Gb	256Mb	93.8
MuC	72	147.1M	867.5	**446.4**	4Gb	256Mb	93.8
Poke	104	769.9M	9,419	**4,142**	32Gb	128Mb	99.6

M*{cs,opt}** - memory usage, ***MRed. - memory reduction %
t*{cs,opt}** - running times, ***#Cat - no. cross product of constants

Exp2: Main results

- Compared **MS-1** with **2 rule-based baselines** (Magellan/JedAI) in DB community

Data	Method	F_1	(P	R)	t_o	t_g	t_s
Dblp	Magellan	79.97	89.80	72.08	**1.71**	-	-
	JedAI	95.02	**100**	90.51	49.16	-	-
	LB	46.09	97.10	30.21	531.63	**0.23**	**0.0039**
	MS-1	**96.21**	95.36	97.07	532.17	0.45	0.32
	PM	95.15	92.85	**97.57**	538.5	0.35	6.75
	UB	91.11	85.50	**97.57**	531.41	-	-
Cora	Magellan	79.70	93.09	69.68	131.13	-	-
	JedAI	90.53	95.53	87.72	**40.92**	-	-
	LB	83.57	**99.80**	71.87	1,008	**7.85**	**0.29**
	MS-1	**95.55**	94.25	96.79	1,031	20.88	10.50
	PM	**95.55**	94.25	96.79	1,857.6	20.19	837.58
	UB	87.50	79.66	**97.05**	999.87	-	-

single table

Data	Method	F_1	(P	R)	t_o	t_g	t_s
MuC	Magellan	55.54	97.51	38.83	66.87	-	-
	JedAI	32.75	73.95	21.02	**7.88**	-	-
	LB	53.95	**99.79**	36.97	474.56	**28.01**	**0.062**
	MS-1	**84.10**	88.11	80.44	562.37	113.64	2.33
	PM	83.87	87.59	80.46	893.44	113.26	333.78
	UB	83.55	86.01	**81.24**	446.4	-	-
Poke	Magellan	7.01	3.97	29.74	260.96	-	-
	JedAI	2.1	1.08	46.56	**23.46**	-	-
	LB	28.00	**100**	16.27	4,144	**2.29**	**0.018**
	MS-1	**88.71**	92.88	**84.90**	4,271.8	127.67	2.17
	PM	**88.71**	92.88	**84.90**	4,296	129.04	25.84
	UB	77.00	70.43	**84.90**	4,142	-	-

multi-table

Exp2: Main results

- Compared **MS-1** with **2 rule-based baselines** (Magellan/JedAI) in DB community
- **Consistently superior quality** compared to baselines **esp. multi-table**

Data	Method	F_1	(P / R)	t_o	t_g	t_s
Dblp	Magellan	79.97	89.80 / 72.08	**1.71**	-	-
	JedAI	95.02	**100** / 90.51	49.16	-	-
	LB	46.09	97.10 / 30.21	531.63	**0.23**	**0.0039**
	MS-1	**96.21**	95.36 / 97.07	532.17	0.45	0.32
	PM	95.15	92.85 / **97.57**	538.5	0.35	6.75
	UB	91.11	85.50 / **97.57**	531.41	-	-
Cora	Magellan	79.70	93.09 / 69.68	131.13	-	-
	JedAI	90.53	95.53 / 87.72	**40.92**	-	-
	LB	83.57	**99.80** / 71.87	1,008	**7.85**	**0.29**
	MS-1	**95.55**	94.25 / 96.79	1,031	20.88	10.50
	PM	**95.55**	94.25 / 96.79	1,857.6	20.19	837.58
	UB	87.50	79.66 / **97.05**	999.87	-	-

single table

Data	Method	F_1	(P / R)	t_o	t_g	t_s
MuC	Magellan	55.54	97.51 / 38.83	66.87	-	-
	JedAI	32.75	73.95 / 21.02	**7.88**	-	-
	LB	53.95	**99.79** / 36.97	474.56	**28.01**	**0.062**
	MS-1	**84.10**	83.11 / 80.44	562.37	113.64	2.33
	PM	83.87	87.59 / 80.46	893.44	113.26	333.78
	UB	83.55	86.01 / **81.24**	446.4	-	-
Poke	Magellan	7.01	3.97 / 29.74	260.96	-	-
	JedAI	2.1	1.08 / 46.56	**23.46**	-	-
	LB	28.00	**100** / 16.27	4,144	**2.29**	**0.018**
	MS-1	**88.71**	92.88 / **84.90**	4,271.8	127.67	2.17
	PM	**88.71**	92.88 / **84.90**	4,296	129.04	25.84
	UB	77.00	70.43 / **84.90**	4,142	-	-

multi-table

Exp2: Main results

- Compared **MS-1** with **2 rule-based baselines** (Magellan/JedAI) in DB community
- **Consistently superior quality** compared to baselines **esp. multi-table**
- **Disadvantageous runtime** (*expensive preprocessing*)

Data	Method	F_1	(P / R)	t_o	t_g	t_s
Dblp	Magellan	79.97	89.80 / 72.08	**1.71**	-	-
	JedAI	95.02	**100** / 90.51	49.16	-	-
	LB	46.09	97.10 / 30.21	531.63	**0.23**	**0.0039**
	MS-1	**96.21**	95.36 / 97.07	532.17	0.45	0.32
	PM	95.15	92.85 / **97.57**	538.5	0.35	6.75
	UB	91.11	85.50 / **97.57**	531.41	-	-
Cora	Magellan	79.70	93.09 / 69.68	131.13	-	-
	JedAI	90.53	95.53 / 87.72	**40.92**	-	-
	LB	83.57	**99.80** / 71.87	1,008	**7.85**	**0.29**
	MS-1	**95.55**	94.25 / 96.79	1,031	20.88	10.50
	PM	**95.55**	94.25 / 96.79	1,857.6	20.19	837.58
	UB	87.50	79.66 / **97.05**	999.87	-	-

single table

Data	Method	F_1	(P / R)	t_o	t_g	t_s
MuC	Magellan	55.54	97.51 / 38.83	66.87	-	-
	JedAI	32.75	73.95 / 21.02	**7.88**	-	-
	LB	53.95	**99.79** / 36.97	474.56	**28.01**	**0.062**
	MS-1	**84.10**	83.11 / 80.44	562.37	113.64	2.33
	PM	83.87	87.59 / 80.46	893.44	113.26	333.78
	UB	83.55	86.01 / **81.24**	446.4	-	-
Poke	Magellan	7.01	3.97 / 29.74	260.96	-	-
	JedAI	2.1	1.08 / 46.56	**23.46**	-	-
	LB	28.00	**100** / 16.27	4,144	**2.29**	**0.018**
	MS-1	**88.71**	92.88 / **84.90**	4,271.8	127.67	2.17
	PM	**88.71**	92.88 / **84.90**	4,296	129.04	25.84
	UB	77.00	70.43 / **84.90**	4,142	-	-

multi-table

Exp3: Effect of recursion

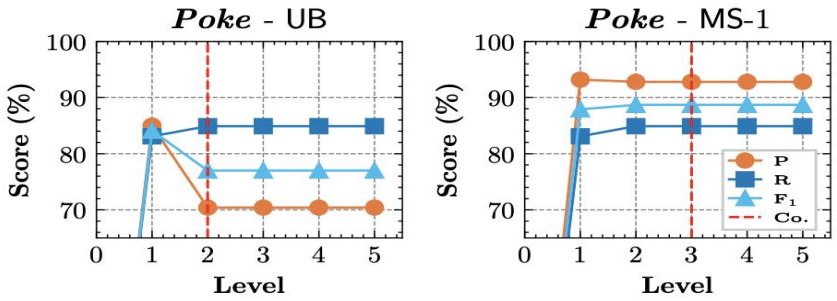
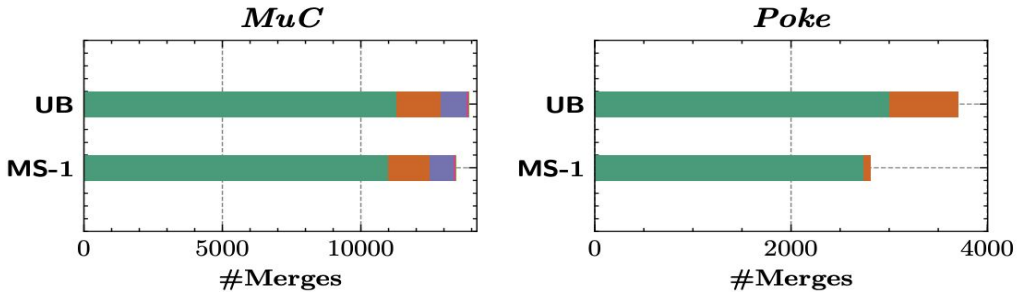


Figure 3: Impact of Recursion on Accuracy



- Most merges in 1st recursion 'level', but **multiple 'levels' to converge**
- **Presence of constraints is crucial** to prevent false positive in the recursive setting

Experiments: scalability & comparison with Datalog

- **Competitive to Datalog engine**, performances trends **differ among datasets**
- **While keeping other 2 factors the same, varying**:
  - **Datasize |D|** -> the **larger** the **slower**
  - **% of duplicates (Du%)** -> the **higher** the **slower**
  - **Similarity thresholds** -> the **lower** the **slower**
- **Grounding times** are more sensitive to **datasize**, **solving times** are more sensitive to **% of duplicates** and **similarity thresholds**

In Arxiv version

Conclusion

- **ASPEn System**, a **declarative**, **collective**, **explainable** solution to **ER** based on **LACE**
- Similarity computation consistently **much more memory-efficient**, more **scalable** to **large datasets**
- Experiment on **complex real-world datasets** shows it is **promising**, achieving **high accuracy** in **real-life ER** scenarios,
- Showcase **ER** as an **exciting** but **challenging** application in the wild for **ASP**

Conclusion

- **ASPEn System**, a **declarative**, **collective**, **explainable** solution to **ER** based on **LACE**
- **Efficiently computable approximations** to possible/certain merges
- Proposed similarity computation consistently **much more memory-efficient**, more **scalable** to **large datasets**
- Experiment on **complex real-world datasets** shows it is **promising**, achieving **high accuracy** in **real-life ER** scenarios
- Showcase **ER** as an **exciting** but **challenging** application in the wild for **ASP**

Thank you for your attention!

repo of the source code/arxiv full detail