Combining Global and Local Merges in Logic-based Entity Resolution

Meghyn Bienvenu[1] , **Gianluca Cima**[2] , **Víctor Gutiérrez-Basulto**[3] , **Yazmín Ibáñez-García**[3]
[1]Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800
[2]Department of Computer, Control and Management Engineering, Sapienza University of Rome
[3]School of Computer Science & Informatics, Cardiff University
meghyn.bienvenu@labri.fr, cima@diag.uniroma1.it, {gutierrezbasultov, ibanezgarciay}@cardiff.ac.uk

Abstract

In the recently proposed LACE framework for collective entity resolution, logical rules and constraints are used to identify pairs of entity references (e.g. author or paper ids) that denote the same entity. This identification is global: all occurrences of those entity references (possibly across multiple database tuples) are deemed equal and can be merged. By contrast, a local form of merge is often more natural when identifying pairs of data values, e.g. some occurrences of 'J. Smith' may be equated with 'Joe Smith', while others should merge with 'Jane Smith'. This motivates us to extend LACE with local merges of values and explore the computational properties of the resulting formalism.

1 Introduction

Entity resolution (ER) is a data quality management task aiming at identifying database different constants (of the same type) that refer to the same real-world entity (Singla and Domingos 2006). Given the fundamental nature of this problem, several variants of ER (also known as record linkage or deduplication) have been investigated. *Collective* entity resolution (Bhattacharya and Getoor 2007; Deng et al. 2022)) considers the joint resolution (match, merge) of entity references or values of multiple types across multiple tables, e.g. using the merge of two authors to infer that two paper ids have to be merged as well. Various approaches to collective ER, with different formal foundations, have been developed: probabilistic approaches, deep learning approaches, and approaches based on rules and constraints, see (Christophides et al. 2021) for a survey.

We have recently proposed LACE (Bienvenu, Cima, and Gutiérrez-Basulto 2022), a declarative framework for collective ER based upon logical rules and constraints. LACE employs hard and soft rules to define mandatory and possible merges. The semantics of LACE is dynamic: ER solutions are generated by sequences of rule applications, where rules are evaluated over the current induced database, taking into account all previously derived merges. This makes it possible to support recursive scenarios (e.g. a merge of authors triggers a merge of papers which in turn enables another merge of authors), while ensuring that all merges have a (non-circular) derivation. The semantics is also global in the sense that *all* occurrences of the matched constants are merged, rather than only those constant occurrences used in deriving the match. Such a global semantics is well suited for merging constants that are entity references (e.g. authors or paper ids) and has been used in other prominent logic-based approaches (Arasu, Ré, and Suciu 2009; Burdick et al. 2016). However, for merging attribute values (e.g. author names), a local semantics, which considers the context in which a value occurs, is more appropriate. Indeed, a local semantics allows some occurrences of 'J. Smith' to be matched to 'Joe Smith' and others to 'Jane Smith', without (wrongly) equating the latter two constants. *Matching dependencies* (Bertossi, Kolahi, and Lakshmanan 2013; Fan 2008; Fan et al. 2009) are an example of a principled logical formalism for merging values.

To the best of our knowledge, there is currently no ER framework that supports both global and local merges. This motivates us to introduce LACE$^+$, an extension of LACE with local merges of values, in which local merges may enable global merges, and vice versa. In particular, local merges can resolve constraint violations which would otherwise block desirable global merges. LACE$^+$ extends LACE's syntax by adding hard and soft rules for values, but it departs from LACE semantics by considering sets of constants, rather than single constants, as arguments in induced databases. Intuitively, such a set of constants provides alternative representations of the same information, e.g. different forms of a name. The semantic treatment of local merges within LACE$^+$ aligns with the design of the generic ER framework Swoosh (Benjelloun et al. 2009).

Our main contributions are the introduction of the new LACE$^+$ framework and the exploration of its computational properties. Our complexity analysis shows that the addition of local merges does not increase the data complexity of the considered reasoning tasks. We also show how an existing answer set programming (ASP) encoding of ER solutions in LACE can be extended to handle local merges of values. We refer readers to (Bienvenu, Cima, and Gutiérrez-Basulto 2022) for more details on related work, to (Bienvenu, Cima, and Gutiérrez-Basulto 2023) for an extension of LACE with repairs, and to (Bienvenu et al. 2023) for omitted proofs.

2 Preliminaries

Databases We assume that *constants* are drawn from three infinite and pairwise disjoint sets: a set $\mathbf{O}$ of *object constants* (or *objects*), serving as references to real-world entities (e.g.

paper and author ids), a set $\mathbf{V}$ of *value constants* (or *values*) from the considered datatypes (e.g. strings for names of authors and paper titles, dates for time of publication), and a set $\mathbf{TID}$ of *tuple identifiers (tids)*.

A *(database) schema* $\mathcal{S}$ consists of a finite set of *relation symbols*, each having an associated arity $k \in \mathbb{N}$ and type vector $\{\mathbf{O}, \mathbf{V}\}^k$. We use $R/k \in \mathcal{S}$ to indicate that the relation symbol R from $\mathcal{S}$ has arity k, and denote by $\mathbf{type}(R, i)$ the ith element of R's type vector. If $\mathbf{type}(R, i) = \mathbf{O}$ (resp. $\mathbf{V}$), we call i an *object (resp. value) position* of R.

A *($\mathbf{TID}$-annotated) $\mathcal{S}$-database* is a finite set D of *facts* of the form $R(t, c_1, \ldots, c_k)$, where $R/k \in \mathcal{S}$, $t \in \mathbf{TID}$, and $c_i \in \mathbf{type}(R, i)$ for every $1 \leq i \leq k$. We require that each $t \in \mathbf{TID}$ occurs in at most one fact of D. We say that t (resp. c_i) occurs in position 0 (resp. $i \in \{1, \ldots, k\}$) of $R(t, c_1, \ldots, c_k)$, and slightly abusing notation, use t and $t[j]$ respectively to refer to the unique fact having tid t, and to the constant in the jth position of that fact. The set of constants (resp. objects) occurring in D is denoted $\mathsf{Dom}(D)$ (resp. $\mathsf{Obj}(D)$), and the set $\mathsf{Cells}(D)$ of *(value) cells* of D is defined as $\{\langle t, i\rangle \mid R(t, c_1, \ldots, c_k) \in D, \mathbf{type}(R, i) = \mathbf{V}\}$.

Queries In the setting of $\mathbf{TID}$-annotated $\mathcal{S}$-databases, a *conjunctive query (CQ)* has the form $q(\vec{x}) = \exists \vec{y}. \varphi(\vec{x}, \vec{y})$, where $\vec{x}$ and $\vec{y}$ are disjoint tuples of variables, and $\varphi(\vec{x}, \vec{y})$ is a conjunction of relational atoms of the form $R(u_0, u_1, \ldots, u_k)$, where $R/k \in \mathcal{S}$ and $u_i \in \mathbf{O} \cup \mathbf{V} \cup \mathbf{TID} \cup \vec{x} \cup \vec{y}$ for $0 \leq i \leq k$. When formulating entity resolution rules and constraints, we shall also consider extended forms of CQs that may contain inequality atoms or atoms built from a set of binary *similarity predicates*. Note that such atoms will not contain the tid position and have a fixed meaning[1]. As usual, the *arity* of $q(\vec{x})$ is the length of $\vec{x}$, and queries of arity 0 are called *Boolean*. Given an n-ary query $q(x_1, \ldots, x_n)$ and n-tuple of constants $\vec{c} = (c_1, \ldots, c_n)$, we denote by $q[\vec{c}]$ the Boolean query obtained by replacing each x_i by c_i. We use $\mathsf{vars}(q)$ (resp. $\mathsf{cons}(q)$) for the set of variables (resp. constants) in q.

Constraints Our framework will also employ denial constraints (DCs) (Bertossi 2011; Fan and Geerts 2012). A *denial constraint* over a schema $\mathcal{S}$ takes the form $\exists \vec{y}. \varphi(\vec{y}) \rightarrow \bot$, where $\varphi(\vec{y})$ is a Boolean CQ with inequalities, whose relational atoms use relation symbols from $\mathcal{S}$. We impose the standard safety condition: each variable occurring in an inequality atom must also occur in some relational atom. Denial constraints notably generalize the well-known class of *functional dependencies (FDs)*. To simplify the presentation, we sometimes omit the initial quantifiers from DCs.

Equivalence Relations We recall that an *equivalence relation* on a set S is a binary relation on S that is reflexive, symmetric, and transitive. We use $\mathsf{EqRel}(P, S)$ for the smallest equivalence relation on S that extends P.

3 LACE^{+} Framework

This section presents and illustrates LACE^{+}, an extension of the LACE framework to handle local merges of values.

3.1 Syntax of LACE^{+} Specifications

As in LACE, we consider *hard and soft rules for objects (over schema $\mathcal{S}$)*, which take respectively the forms:

$$q(x, y) \Rightarrow \mathsf{EqO}(x, y) \quad q(x, y) \dashrightarrow \mathsf{EqO}(x, y)$$

where $q(x, y)$ is a CQ whose atoms may use relation symbols from $\mathcal{S}$ as well as similarity predicates and whose free variables x and y occur only in object positions. Intuitively, the above hard (resp. soft) rule states that (o_1, o_2) being an answer to q provides sufficient (resp. reasonable) evidence for concluding that o_1 and o_2 refer to the same real-world entity. The special relation symbol EqO (not in $\mathcal{S}$) is used to store such merged pairs of object constants.

To handle local identifications of values, we introduce *hard and soft rules for values (over $\mathcal{S}$)*, which take the forms:

$$q(x_t, y_t) \Rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$$
$$q(x_t, y_t) \dashrightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$$

where $q(x_t, y_t)$ is a CQ whose atoms may use relation symbols from $\mathcal{S}$ as well as similarity predicates, variables x_t and y_t each occur once in q in position 0 of (not necessarily distinct) relational atoms with relations $R_x \in \mathcal{S}$ and $R_y \in \mathcal{S}$, respectively, and i and j are value positions of R_x and R_y, respectively. Intuitively, such a hard (resp. soft) rule states that a pair of tids (t_1, t_2) being an answer to q provides sufficient (resp. reasonable) evidence for concluding that the values in cells $\langle x_t, i\rangle$ and $\langle y_t, j\rangle$ are non-identical representations of the same information. The special relation symbol EqV (not in $\mathcal{S}$ and distinct from EqO) is used to store pairs of value cells which have been merged.

Definition 1. *A* LACE^{+} *entity resolution (ER) specification Σ for schema $\mathcal{S}$ takes the form $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$, where $\Gamma_O = \Gamma_h^o \cup \Gamma_s^o$ is a finite set of hard and soft rules for objects, $\Gamma_V = \Gamma_h^v \cup \Gamma_s^v$ is a finite set of hard and soft rules for values, and Δ is a finite set of denial constraints, all over $\mathcal{S}$.*

Example 1. *The schema $\mathcal{S}_{\mathsf{ex}}$, database D_{ex}, and ER specification $\Sigma_{\mathsf{ex}} = \langle \Gamma_{\mathsf{ex}}^O, \Gamma_{\mathsf{ex}}^V, \Delta_{\mathsf{ex}}\rangle$ of our running example are given in Figure 1. Informally, the denial constraint δ_1 is an FD saying that an author id is associated with at most one author name, while the constraint δ_2 forbids the existence of a paper written by the chair of the conference in which the paper was published. The hard rule ρ_1^o states that if two author ids have the same name and the same institution, then they refer to the same author. The soft rule σ_1^o states that authors who wrote a paper in common and have similar names are likely to be the same. Finally, the hard rule ρ_1^v locally merges similar names associated with the same author id.*

3.2 Semantics of LACE^{+} Specifications

In a nutshell, the semantics is based upon considering sequences of rule applications that result in a database that satisfies the hard rule and denial constraints. Every such sequence gives rise to a solution, which takes the form of a pair of equivalence relations $\langle E, V\rangle$, specifying which objects and cells have been merged. Importantly, rules and constraints are evaluated w.r.t. the induced database, taking into account previously derived merges of objects and cells.

[1] The extension of similarity predicates is typically defined by applying some similarity metric, e.g. edit distance, and keeping those pairs of values whose score exceeds a given threshold.

Author(tid, aid, name, inst)

tid	aid	name	inst
t_1	a_1	J. Smith	Sapienza
t_2	a_2	Joe Smith	Oxford
t_3	a_3	J. Smith	NYU
t_4	a_4	Joe Smith	NYU
t_5	a_5	Joe Smith	Sapienza
t_6	a_6	Min Lee	CNRS
t_7	a_7	M. Lee	UTokyo
t_8	a_8	Myriam Lee	Cardiff

Paper(tid, pid, title, conf, ch)

tid	pid	title	conf	ch
t_9	p_1	Logical Framework for ER	PODS'21	a_6
t_{10}	p_2	Rule-based approach to ER	ICDE'19	a_4
t_{11}	p_3	Query Answering over DLs	KR'22	a_1
t_{12}	p_4	CQA over DL Ontologies	IJCAI'21	a_1
t_{13}	p_5	Semantic Data Integration	AAAI'22	a_8

The sim predicate $\approx$ is such that the names of authors a_1, a_2, a_3, a_4, and a_5 are pairwise similar, and both the names of authors a_6 and a_8 are similar to the name of author a_7

Wrote(tid, aid, pid)

tid	aid	pid
t_{14}	a_1	p_1
t_{15}	a_2	p_1
t_{16}	a_3	p_2
t_{17}	a_6	p_3
t_{18}	a_7	p_3
t_{19}	a_7	p_4
t_{20}	a_8	p_4
t_{21}	a_6	p_5

$$\delta_1 = \text{Author}(t, a, n, i) \land \text{Author}(t', a, n', i') \land n \neq n' \rightarrow \bot; \qquad \rho_1^o = \text{Author}(t, x, n, i) \land \text{Author}(t', y, n, i) \Rightarrow \mathsf{EqO}(x, y)$$

$$\delta_2 = \text{Paper}(t, p, ti, c, a) \land \text{Wrote}(t', a, p) \rightarrow \bot; \quad \rho_1^v = \text{Author}(x, a, n, i) \land \text{Author}(y, a, n', i') \land n \approx n' \Rightarrow \mathsf{EqV}(\langle x, 2\rangle, \langle y, 2\rangle)$$

$$\sigma_1^o = \text{Author}(t, x, n, i) \land \text{Author}(t', y, n', i') \land n \approx n' \land \text{Wrote}(t'', x, p) \land \text{Wrote}(t''', y, p) \dashrightarrow \mathsf{EqO}(x, y)$$

Figure 1: Schema $\mathcal{S}_{\mathsf{ex}}$, $\mathcal{S}_{\mathsf{ex}}$-database D_{ex}, and ER specification $\Sigma_{\mathsf{ex}} = \langle \Gamma^O_{\mathsf{ex}}, \Gamma^V_{\mathsf{ex}}, \Delta_{\mathsf{ex}}\rangle$ with $\Gamma^O_{\mathsf{ex}} = \{\rho_1^o, \sigma_1^o\}$, $\Gamma^V_{\mathsf{ex}} = \{\rho_1^v\}$, and $\Delta_{\mathsf{ex}} = \{\delta_1, \delta_2\}$.

In the original LACE framework, solutions consist of a single equivalence relation over objects, and induced databases are simply defined as the result of replacing every object with a representative of its equivalence class. Such an approach cannot however accommodate local identifications of values. For this reason, we shall work with an extended form of database, where the arguments are *sets of constants*.

Definition 2. *Given an $\mathcal{S}$-database D, equivalence relation E over $\mathsf{Obj}(D)$, and equivalence relation V over $\mathsf{Cells}(D)$, we denote by $D_{E,V}$ the* (extended) database induced by *D, E,* and *V, which is obtained from D by replacing:*

- *each tid t with the singleton set $\{t\}$,*
- *each occurrence of $o \in \mathsf{Obj}(D)$ by $\{o' \mid (o, o') \in E\}$,*
- *each value in a cell $\langle t, i\rangle \in \mathsf{Cells}(D)$ with the set of values $\{t'[i'] \mid (\langle t, i\rangle, \langle t', i'\rangle) \in V\}$.*

It remains to specify how queries in rule bodies and constraints are to be evaluated over such induced databases. First, we need to say how similarity predicates are extended to sets of constants. We propose that $C_1 \approx C_2$ is satisfied whenever there are $c_1 \in C_1$ and $c_2 \in C_2$ such that $c_1 \approx c_2$, since the elements of a set provide different possible representations of a value. Second, we must take care when handling join variables in value positions. Requiring all occurrences of a variable to map to the same set is too strong, e.g. it forbids us from matching {J. Smith, Joe Smith} with {J. Smith}. We require instead that the intersection of all sets of constants assigned to a given variable is non-empty.

Definition 3. *A Boolean query q (possibly containing similarity and inequality atoms) is* satisfied *in $D_{E,V}$, denoted $D_{E,V} \models q$, if there exists a function $h : \mathsf{vars}(q) \cup \mathsf{cons}(q) \rightarrow 2^{\mathsf{Dom}(D)} \setminus \{\emptyset\}$ and functions $g_\pi : \{0, \ldots, k\} \rightarrow 2^{\mathsf{Dom}(D)}$ for each k-ary relational atom $\pi \in q$, such that:*

1. *h is determined by the g_π: for every $a \in \mathsf{cons}(q)$, $h(a) = \{a\}$, and for every $z \in \mathsf{vars}(q)$, $h(z)$ is the intersection of all sets $g_\pi(i)$ such that z is the ith argument of π;*
2. *for every relational atom $\pi = R(u_0, u_1, \ldots, u_k) \in q$, $R(g_\pi(0), g_\pi(1), \ldots, g_\pi(k)) \in D_{E,V}$, and for every $1 \leq i \leq k$, if $u_i \in \mathsf{cons}(q)$, then $u_i \in g_\pi(i)$;*
3. *for every inequality atom $z \neq z' \in q$: $h(z) \cap h(z') = \emptyset$;*
4. *for every similarity atom $u \approx u' \in q$: there exist $c \in h(u)$ and $c' \in h(u')$ such that $c \approx c'$.*

For non-Boolean queries, the set $q(D_{E,V})$ of answers to $q(\vec{x})$ contains those tuples $\vec{c}$ such that $D_{E,V} \models q[\vec{c}]$.

Observe that the functions g_π make it possible to map the same variable z to different sets, with Point 1 ensuring these sets have a non-empty intersection, $h(z)$. It is this intersected set, storing the common values for z, that is used to evaluate inequality and similarity atoms. Note that when constants occur in relational atoms, the sets assigned to a constant's position must contain that constant.

The preceding definition of satisfaction of queries is straightforwardly extended to constraints and rules:

- $D_{E,V} \models \exists \vec{y}. \varphi(\vec{y}) \rightarrow \bot$ iff $D_{E,V} \not\models \exists \vec{y}. \varphi(\vec{y})$
- $D_{E,V} \models q(x, y) \rightarrow \mathsf{EqO}(x, y)$ iff $q(D_{E,V}) \subseteq E$
- $D_{E,V} \models q(x_t, y_t) \rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$ iff $(t_1, t_2) \in q(D_{E,V})$ implies $(\langle t_1, i\rangle, \langle t_2, j\rangle) \in V$;

where symbol $\rightarrow$ can be instantiated by either $\Rightarrow$ or $\dashrightarrow$. We write $D_{E,V} \models \Lambda$ iff $D_{E,V} \models \lambda$ for every $\lambda \in \Lambda$.

With these notions in hand, we can formally define solutions of LACE$^+$ specifications.

Definition 4. *Given an ER specification $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$ over schema $\mathcal{S}$ and an $\mathcal{S}$-database D, we call $\langle E, V\rangle$ a* candidate solution for *(D, Σ) if it satisfies one of the following:*

- *$E = \mathsf{EqRel}(\emptyset, \mathsf{Obj}(D))$ and $V = \mathsf{EqRel}(\emptyset, \mathsf{Cells}(D))$;*
- *$E = \mathsf{EqRel}(E' \cup \{(o, o')\}, \mathsf{Obj}(D))$, where $\langle E', V\rangle$ is a candidate solution for (D, Σ) and $(o, o') \in q(D_{E,V})$ for some $q(x, y) \rightarrow \mathsf{EqO}(x, y) \in \Gamma_O$;*
- *$V = \mathsf{EqRel}(V' \cup \{(\langle t, i\rangle, \langle t', i'\rangle)\}, \mathsf{Cells}(D))$, where $\langle E, V'\rangle$ is a candidate solution for (D, Σ) and $(t, t') \in q(D_{E,V})$ for some $q(x_t, y_t) \rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, i'\rangle) \in \Gamma_V$.*

If also $D_{E,V} \models \Gamma_h^o \cup \Gamma_h^v \cup \Delta$, then $\langle E, V\rangle$ is a solution *for (D, Σ). We use $\mathsf{Sol}(D, \Sigma)$ for the set of solutions for (D, Σ).*

We illustrate solutions and the utility of local merges:

Example 2. *Starting from database D_{ex}, we can apply the soft rule σ_1^o to merge author ids a_1 and a_2 (more formally, we minimally extend the initial trivial equivalence relation E to include (a_1, a_2)). The resulting induced instance is obtained by replacing* all *occurrences of a_1 and a_2 by $\{a_1, a_2\}$. Note that the constraint δ_1 is now violated, since t_1 and t_2 match on aid, but have different names. In the original* LACE *framework, this would prevent (a_1, a_2) from belonging to any solution. However, thanks to the hard rule for values ρ_1^v, we can resolve this violation. Indeed, ρ_1^v is applicable and allows us to (locally) merge the names in facts t_1 and t_2. The new induced database contains {J. Smith, Joe Smith} in the name position of t_1 and t_2, but the names for t_3, t_4, t_5 remain as before. Note the importance of performing a local rather than a global merge: if we had grouped J. Smith with Joe Smith* everywhere*, this would force a merge of a_3 with a_4 due to the hard rule ρ_1^o, which would in turn violate δ_2, again resulting in no solution containing (a_1, a_2). Following the local merge of the names of t_1 and t_2, the hard rule ρ_1^o becomes applicable and allows us (actually, forces us) to merge (globally) author ids a_1 and a_5. We let $\langle E_{\mathsf{ex}}, V_{\mathsf{ex}}\rangle$ be the equivalence relations obtained from the preceding rule applications. As the instance induced by $\langle E_{\mathsf{ex}}, V_{\mathsf{ex}}\rangle$ satisfies all hard rules and constraints, $\langle E_{\mathsf{ex}}, V_{\mathsf{ex}}\rangle$ is a solution. Another solution is the pair of trivial equivalence relations, since D_{ex} satisfies the constraints and hard rules.*

As in LACE, we shall compare solutions based upon set inclusion, to maximize the discovered merges.

Definition 5. *A solution $\langle E, V\rangle$ for (D, Σ) is a* maximal solution *for (D, Σ) if there exists no solution $\langle E', V'\rangle$ for (D, Σ) such that $E \cup V \subsetneq E' \cup V'$. We denote by $\mathsf{MaxSol}(D, \Sigma)$ the set of maximal solutions for (D, Σ).*

Example 3. *The solution $\langle E_{\mathsf{ex}}, V_{\mathsf{ex}}\rangle$ described in Example 2 is not optimal as the soft rule σ_1^o can be applied to get (a_6, a_7) or (a_7, a_8). Notice, however, that it is not possible to include both merges, otherwise by transitivity, a_6, a_7, a_8 would all be replaced by $\{a_6, a_7, a_8\}$, which would violate denial δ_1 due to paper p_5. We have two maximal solutions: a first that extends $\langle E_{\mathsf{ex}}, V_{\mathsf{ex}}\rangle$ with (a_6, a_7) and the corresponding pair of names cells $(\langle t_6, 2\rangle, \langle t_7, 2\rangle)$ (due to ρ_1^v), and a second that extends $\langle E_{\mathsf{ex}}, V_{\mathsf{ex}}\rangle$ with (a_7, a_8) and the corresponding name cells $(\langle t_6, 2\rangle, \langle t_7, 2\rangle)$ (again due to ρ_1^v).*

The LACE⁺ framework properly generalizes the one in (Bienvenu, Cima, and Gutiérrez-Basulto 2022): if we take $\Sigma = \langle \Gamma_O, \emptyset, \Delta\rangle$ (i.e. no rules for values), then E is a solution for (D, Σ) in the original LACE framework iff $\langle E \cap (\mathbf{O} \times \mathbf{O}), \mathsf{EqRel}(\emptyset, \mathsf{Cells}(D))\rangle \in \mathsf{Sol}(D, \Sigma)$.

More interestingly, we show that it is in fact possible to simulate global merges using local merges.

Theorem 1. *For every ER specification $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$ over $\mathcal{S}$, there exists a specification $\Sigma' = \langle \emptyset, \Gamma'_V, \Delta\rangle$ (over a modified $\mathcal{S}$, with all object positions changed to value positions, and all object constants treated as value constants) such that for every $\mathcal{S}$-database D: $\mathsf{Sol}(D, \Sigma') = \{\langle \emptyset, V \cup V_E\rangle \mid \langle E, V\rangle \in \mathsf{Sol}(D, \Sigma)\}$, where V_E contains all pairs $(\langle t, i\rangle, \langle t', j\rangle)$ such that $(t[i], t'[j]) \in E$.*

4 Computational Aspects

We briefly explore the computational properties of LACE⁺. As in (Bienvenu, Cima, and Gutiérrez-Basulto 2022), we are interested in the *data complexity* of the following decision problems: REC (resp. MAXREC) which checks if $\langle E, V\rangle \in \mathsf{Sol}(D, \Sigma)$ (resp. $\langle E, V\rangle \in \mathsf{MaxSol}(D, \Sigma)$), EXISTENCE which determines if $\mathsf{Sol}(D, \Sigma) \neq \emptyset$, CERTMERGE (resp. POSSMERGE) which checks if a candidate merge belongs to $E \cup V$ for all (resp. some) $\langle E, V\rangle \in \mathsf{MaxSol}(D, \Sigma)$, and CERTANS (resp. POSSANS) which checks whether $\vec{c} \in q(D_{E,V})$ for all (resp. some) $\langle E, V\rangle \in \mathsf{MaxSol}(D, \Sigma)$. Interestingly, we show that incorporating local merges does not affect the complexity of all the above decision problems.

Theorem 2. REC *is* P*-complete;* MAXREC *is coNP-complete;* EXISTENCE, POSSMERGE, *and* POSSANS *are NP-complete;* CERTMERGE *and* CERTANS *are* Π_2^p*-complete. For specifications that do not use inequality atoms in denial constraints,* REC, MAXREC, *and* EXISTENCE *are* P*-complete;* POSSMERGE *and* POSSANS *are NP-complete;* CERTMERGE *and* CERTANS *are coNP-complete.*

Due to Theorem 1, all lower bounds hold even for specifications that do not contain any rules for objects.

We extend the ASP encoding from (Bienvenu, Cima, and Gutiérrez-Basulto 2022) to obtain a normal logic program Π_{Sol} whose stable models capture LACE⁺ solutions:

Theorem 3. *For every database D and specification $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$: $\langle E, V\rangle \in \mathsf{Sol}(D, \Sigma)$ iff $E = \{(a, b) \mid EqO(a, b) \in M\}$ and $V = \{(\langle t, i\rangle, \langle t', i'\rangle) \mid EqV(t, i, t', i') \in M\}$ for a stable model M of (Π_{Sol}, D).*

We sketch here how rules for values are handled. Basically, every hard rule $q(x_t, y_t) \Rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$ is translated into the ASP rule $EqV(x_t, i, y_t, j) \leftarrow \hat{q}(x_t, y_t)$. To define $\hat{q}$, we use $\mathsf{vpos}(v)$ (resp. opos) for the set of pairs (u_t, i) such that v occurs in a value (resp. object) position i in atom $R(u_t, v_1, \ldots, v_k) \in q$. The query $\hat{q}$ is obtained from q by replacing each occurrence (u_t, i) of a non-distinguished variable v in q with a fresh variable $v_{(u_t,i)}$, and then:

- for every join variable v in q, take fresh variables u'_t, k, v' and add to $\hat{q}$ the set of atoms $\{EqV(u_t, i, u'_t, k) \mid (u_t, i) \in \mathsf{vpos}(v)\} \cup \{EqO(v_{(u_t,i)}, v') \mid (u_t, i) \in \mathsf{opos}(v)\}$;
- for each atom $\alpha = v \approx w$, take fresh variables v', w' and replace α by the set of atoms $\{Val(u_t, i, v') | (u_t, i) \in \mathsf{vpos}(v)\} \cup \{Val(u'_t, j, w') | (u'_t, j) \in \mathsf{vpos}(w)\} \cup \{v' \approx w'\}$, where Val is a predicate defined by the rule:

  $$Val(u_t, i, v) \leftarrow EqV(u_t, i, u'_t, j), Proj(u'_t, j, v), \quad \text{and}$$

  ground atoms $Proj(t, i, c)$ of $Proj/3$ encode $t[i] = c$.

Soft rules for values are handled similarly: we use the same modified body $\hat{q}$, but then enable a choice between producing $EqV(x_t, i, y_t, j)$ or not applying the rule (adding a blocking fact $NEqV(x_t, i, y_t, j)$). Additionally, Π_{Sol} will contain rules that encode object rules (producing EqO facts), rules that ensure EqV and EqO are equivalence relations, and rules that enforce the satisfaction of the denial constraints.

Acknowledgements

This work has been supported by the ANR AI Chair INTENDED (ANR-19-CHIA-0014), by MUR under the PNRR project FAIR (PE0000013), and by the Royal Society (IES\R3\193236).

References

Arasu, A.; Ré, C.; and Suciu, D. 2009. Large-scale deduplication with constraints using dedupalog. In *Proceedings of the Twenty-Fifth International Conference on Data Engineering (ICDE 2009)*, 952–963.

Benjelloun, O.; Garcia-Molina, H.; Menestrina, D.; Su, Q.; Whang, S. E.; and Widom, J. 2009. Swoosh: a generic approach to entity resolution. *The VLDB Journal* 18(1):255–276.

Bertossi, L. E.; Kolahi, S.; and Lakshmanan, L. V. S. 2013. Data cleaning and query answering with matching dependencies and matching functions. *Theory of Computing Systems* 52(3):441–482.

Bertossi, L. E. 2011. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Bhattacharya, I., and Getoor, L. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data* 1(1):5.

Bienvenu, M.; Cima, G.; Gutiérrez-Basulto, V.; and Ibáñez-García, Y. 2023. Combining global and local merges in logic-based entity resolution. Long version with appendix. Available at https://arxiv.org/abs/2305.16926.

Bienvenu, M.; Cima, G.; and Gutiérrez-Basulto, V. 2022. LACE: A logical approach to collective entity resolution. In *Proceedings of the Forty-First ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2022)*, 379–391.

Bienvenu, M.; Cima, G.; and Gutiérrez-Basulto, V. 2023. REPLACE: A logical framework for combining collective entity resolution and repairing. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI 2023)*.

Burdick, D.; Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W. 2016. A declarative framework for linking entities. *ACM Transactions on Database Systems* 41(3):17:1–17:38.

Christophides, V.; Efthymiou, V.; Palpanas, T.; Papadakis, G.; and Stefanidis, K. 2021. An overview of end-to-end entity resolution for big data. *ACM Computing Surveys* 53(6):127:1–127:42.

Deng, T.; Fan, W.; Lu, P.; Luo, X.; Zhu, X.; and An, W. 2022. Deep and collective entity resolution in parallel. In *Proceedings of the Thirty-Eighth IEEE International Conference on Data Engineering (ICDE 2022)*, 2060–2072.

Fan, W., and Geerts, F. 2012. *Foundations of Data Quality Management*. Morgan & Claypool Publishers.

Fan, W.; Jia, X.; Li, J.; and Ma, S. 2009. Reasoning about record matching rules. *Proceedings of the VLDB Endowment* 2(1):407–418.

Fan, W. 2008. Dependencies revisited for improving data quality. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2008)*, 159–170. ACM.

Singla, P., and Domingos, P. M. 2006. Entity resolution with markov logic. In *Proceedings of the Sixth IEEE International Conference on Data Mining (ICDM 2006)*, 572–582.