Recent Advances in Logic-Based Entity Resolution

Meghyn Bienvenu
Univ. Bordeaux, CNRS, LaBRI

Gianluca Cima
Sapienza University of Rome

Víctor Gutiérrez-Basulto
Cardiff University

Yazmín Ibáñez-García
Cardiff University

Zhiliang Xiang
Cardiff University

ABSTRACT

Entity resolution (ER) is a central task in data quality, which is concerned with identifying pairs of distinct constants or tuples that refer to the same real-world entity. Declarative approaches, based upon logical rules and constraints, are a natural choice for tackling complex, collective ER tasks involving the joint resolution of multiple entity types across multiple tables. This paper provides an overview of recent advances in logic-based entity resolution, with a particular focus on the LACE framework, first introduced at PODS'22 and subsequently extended with additional features (IJCAI'23, KR'23) and equipped with an answer set programming-based implementation (KR'24, KR'25).

1 Introduction

Entity resolution (ER) is a key data quality task that seeks to identify distinct constants that refer to the same real-world entity [54]. A wide range of ER approaches have been proposed, differing in their assumptions, the nature of the data they handle, and the techniques they employ [21]. In the context of relational databases, ER has traditionally focused on matching records based on field-level similarity [47], which is why it is also known as record linkage [35]. For example, in a bibliographic database, *two author records might be matched if their email addresses are similar*. A more expressive and general approach, known as collective ER [9, 24], performs joint resolution of entity references or values of multiple types across multiple tables. For instance, *merging two author entities may lead to the inference that their associated paper IDs should also be merged*. Most existing approaches to ER focus on single-pass matching of tuples within a single table or between a pair of tables, and machine learning methods have obtained remarkable results [41] for such settings. On the other hand, declarative methods based on logical rules and constraints are able to capture and exploit relational dependencies, making them well-suited for handling complex multi-relational settings arising in collective ER.

In this paper, we examine declarative approaches to collective entity resolution[1], with a particular focus on the LACE framework. We designed LACE to satisfy three natural desiderata: being *collective*, *declarative*, and *justifiable*. Specifically, our approach (i) supports complex interdependencies between merges of different entities, (ii) adopts a declarative language based on logical rules and constraints, and (iii) provides clear justifications for why two constants are considered to represent the same entity. While the collective and declarative aspects have received considerable attention in the literature, the notion of justifiability remains relatively underexplored, despite being a crucial step toward building more advanced explanation capabilities and, ultimately, more responsible technologies [42].

As a declarative language, LACE shares several design principles with existing logic-based frameworks. Inspired by the Dedupalog framework [2], it employs both hard and soft rules to specify conditions under which pairs of constants must or may be merged. For instance, statements such as *every paper has a single corresponding author* and *conferences with similar names are likely to be the same* can be naturally expressed using hard and soft rules, respectively. Beyond rules, LACE specifications can also include denial constraints [7], which ensure the consistency of the resulting instance by restricting inadmissible combinations of merges. For example, one may enforce that *an author's name can appear only once in the list of authors of a paper*. In line with entity resolution approaches based on matching dependencies (MDs) [8, 27, 32] and their extensions, such as relational MDs [4, 5] or entity-enhancing rules (REEs) [24, 33], LACE adopts a dynamic semantics in which rule bodies are evaluated over the evolving instance, i.e., the instance resulting from merges that have already been derived.

[1]For a more extensive discussion of ER methods, the interested reader is referred to [21].

The dynamic semantics is key to obtaining a collective yet justifiable framework: merges can trigger additional merges, potentially in a recursive manner, while still guaranteeing that all merges occurring in a solution are *justified*, in the sense that it is possible to trace back how each merge was obtained via a sequence of rule applications.

Another important design consideration concerns the nature of the constants being merged. When constants represent entity references (e.g., author names or paper IDs), a global semantics, where all occurrences of matched constants are merged (not just those directly involved in deriving the match) is particularly well suited. In contrast, when dealing with data values, a local merging semantics, one that considers the context in which a value occurs, is often more appropriate. For example, some occurrences of 'J. Smith' may refer to 'Joe Smith', while others may correspond to 'Jane Smith'; merging all instances globally in such cases would lead to incorrect resolutions. Various efforts have already been dedicated to studying both approaches. For instance, MDs provide a principled logical formalism for merging values [8, 27, 32], adopting a local semantics. On the other hand, Dedupalog, MRLs, and the declarative framework for *entity linking* [17] (henceforth referred to as EL) rely on a global semantics. Despite substantial work on each of these approaches, most existing frameworks focus on one or the other. This was also the case for the Lace framework, which initially only supported global merges of entity references [11] but was subsequently extended [13] to also handle local merges of data values. Note that, contemporaneously to Lace, the CERQ framework supporting both local and global merges was independently developed [26].

Another distinction among logic-based approaches to ER lies in the nature of their solutions or outputs. A key aspect in this regard is whether the approach produces a single solution or a set of alternative solutions.[2] As in the EL approach [17], we consider not just one solution but a space of preferred solutions. In Lace, this space arises naturally from the role of denial constraints, which restrict which merges can co-occur, thereby introducing meaningful choices. Also in line with EL, we can naturally define the notions of *certain* and *possible* merges, referring respectively to those merges that appear in all, or in some, of the preferred solutions.

[2]Here, a solution can be roughly viewed as a set of constant pairs that are judged to refer to the same entity. Naturally, outside logic-based approaches, solutions may take other forms, for example, expressing the likelihood that two constants refer to the same entity.

We argue that the successful adoption of any ER framework depends on the availability of an accompanying implementation. To this end, we have developed the answer set programming (ASP)-based systems ASPen and ASPen$^+$, grounded in the foundational result that Lace solutions can be faithfully encoded as ASP stable models. ASPen$^+$ supports the full range of Lace features and further extends the framework by exploring various optimality criteria; not only prioritizing solutions that maximize the number of merges (w.r.t. set inclusion), but also enabling other natural forms of preference.

Organization After reviewing the necessary background in Section 2, we introduce in Section 3 the fundamentals of Lace, including its syntax, semantics, properties, and alternative optimality criteria. In Section 4, we define the central decision problems and analyse their computational complexity. Section 5 discusses the practical implementation of Lace using the answer set programming systems ASPen and ASPen$^+$. In Section 6, we present Replace, a holistic framework that integrates ER and repairing. Section 7 overviews the broader landscape of logic-based ER approaches. Finally, Section 8 offers perspectives for future work.

2 Preliminaries

Databases We assume that *constants* are drawn from three infinite and pairwise disjoint sets: a set O of *object constants* (or *objects*), serving as references to real-world entities (e.g. paper and author ids), a set 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 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 $\{\mathsf{O}, \mathsf{V}\}^k$. 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 $\mathbf{type}(R, i) = \mathsf{O}$ (resp. V), we call i an *object (resp. value) position* of R.

An *$\mathcal{S}$-database* is a finite set D of *facts* of the form $R(t, c_1, \ldots, c_k)$, where $R/k \in \mathcal{S}$, $t \in \mathsf{TID}$, and $c_i \in \mathbf{type}(R, i)$ for $1 \leq i \leq k$. We require that each $t \in \mathsf{TID}$ occurs in at most one fact of D. Abusing notation, we will sometimes use t to refer the unique fact with tid t and use $t[j]$ for the constant in the jth position of t (tid arguments occupy position 0, and 'regular' arguments of R/k are in positions $1, \ldots, k$). 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 $\{\langle t, i\rangle \mid t \in D, t[I] \in \mathsf{V}\}$.

Queries We consider *conjunctive queries (CQs)* of the form $q(\mathbf{x}) = \exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ and $\mathbf{y}$ are disjoint tuples of variables, and $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of relational atoms $R(u_0, u_1, \ldots, u_k)$, where $R/k \in \mathcal{S}$, $u_0 \in \mathsf{TID} \cup \mathbf{x} \cup \mathbf{y}$, and for every $1 \leq i \leq k$: $u_i \in \mathsf{O} \cup \mathsf{V} \cup \mathbf{x} \cup \mathbf{y}$ and $u_i \in \mathsf{O} \cup \mathsf{V}$ implies $u_i \in \mathbf{type}(R, i)$. 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 relations*. Note that such atoms will not contain the tid position and have a fixed meaning[3]. Moreover, we impose a standard safety condition: each variable occurring in an inequality or similarity atom must also occur in some relational atom (in a value position, in the case of similarity atoms). As usual, the *arity* of $q(\mathbf{x})$ is the length of $\mathbf{x}$, and queries of arity 0 are called *Boolean*. Given an n-ary query $q(x_1, \ldots, x_n)$ and n-tuple of constants $\mathbf{c} = (c_1, \ldots, c_n)$, we denote by $q[\mathbf{c}]$ the Boolean query obtained by replacing each x_i by c_i. We denote by $\mathsf{vars}(q)$ (resp. $\mathsf{cons}(q)$) the set of variables (resp. constants) in q. and will use set notation for queries when convenient.

Constraints Our framework will also employ *denial constraints* (DCs) [7, 28] which take the form $\exists \mathbf{y}.\varphi(\mathbf{y}) \rightarrow \bot$, where $\exists \mathbf{y}.\varphi(\mathbf{y})$ is a Boolean CQ with inequalities, whose relational atoms use relation symbols from the considered schema $\mathcal{S}$. DCs notably generalize the well-known class of *functional dependencies (FDs)*. To simplify the presentation, we sometimes omit the quantifiers from DCs.

3 LACE Framework

In this section, we present LACE[4], a Logical Approach to Collective Entity resolution, designed to satisfy the desiderata laid out in Section 1.

3.1 Syntax of LACE

In LACE, rules are used to describe conditions under which two constants must or may be identified (we use the term 'merge' to speak of identified pairs). These come in two flavours, depending on whether the considered constants are objects or values.

Rules for Objects To formalize the resolution of object pairs (i.e., references to real-world entities) that denote the same underlying entity, we employ *hard and soft rules for objects* (over a schema $\mathcal{S}$), which take respectively the following 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 relations 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. Note that rules for objects use a special relation symbol EqO (not in schema $\mathcal{S}$) in the rule head.

Rules for Values To formalize *local identifications* between distinct representations of the same information, we introduce *hard and soft rules for values*, which take respectively the following 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 the considered schema $\mathcal{S}$ as well as similarity relations, 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. Rules for values use head relation EqV (not in $\mathcal{S}$ and distinct from EqO).

To refer to a generic (hard or soft) rule, we use the arrow symbol $\rightarrow$ (which can stand for $\Rightarrow$ or $\dashrightarrow$). For the sake of brevity, we usually omit existential quantifiers of variables in rule bodies.

ER Specifications In addition to rules for indicating mandatory or likely merges, LACE specifications may also include denial constraints, which serve to define what counts as a legal (or consistent) database and can help to block incorrect merges.

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*

[3]Similarity relations are typically defined by applying a similarity metric, e.g. edit distance, and keeping those pairs of values whose score exceeds a given threshold.

[4]We present here the version of LACE from [13], which extends the original LACE framework [11] to support local merges of values, rather than only global merges of objects, as was the case in [11]. Note that this version of the framework is referred to as LACE$^+$ in [13].

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

As the aim is to identify pairs of objects (resp. occurrences of values) that denote the same real-world entity (resp. represent the same value), we will be interested in solutions taking the form of a *pair of equivalence relations* $\langle E, V\rangle$, over the sets $\mathsf{Obj}(D)$ and $\mathsf{Cells}(D)$ respectively, giving the merged pairs of objects and value cells. To satisfy our desiderata, we must ensure that the set of merges present in a solution can be justified by appealing to the rules and is coherent w.r.t. the expressed constraints. The idea will thus be to define solutions in terms of sequences of rule applications that lead to a database satisfying all hard rules and denial constraints. Importantly, rule bodies and constraints will be evaluated with respect to the database induced by the current pair of equivalence relations, in order to exploit the previously derived object and cell merges.

To make this formal, we need to clarify the notion of 'induced database' obtained by modifying the initial database to 'implement' a given set of merges. We might be tempted to simply pick a representative from each equivalence class and replace every constant with its representative. While such an approach can be used to treat global merges of (as was done in [11]), it cannot account for the local nature of cell-level merges 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 $h : \mathsf{vars}(q) \cup \mathsf{cons}(q) \to 2^{\mathsf{Dom}(D)} \setminus \{\emptyset\}$ and functions $g_\pi : \{0, \ldots, k\} \to 2^{\mathsf{Dom}(D)}$ for each k-ary relational atom $\pi \in q$, such that:*

1. *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 atom $z \neq z' \in q$: $h(z) \cap h(z') = \emptyset$;*
4. *for every 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 $q(\mathbf{x})$ contains tuples $\mathbf{c}$ s.t. $D_{E,V} \models q[\mathbf{c}]$.

Observe that the functions g_π make it possible to map different occurrences of the same variable z to different sets of constants, 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 a constant c occurs in a relational atom, the set assigned to the position where c occurs must contain c.

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

- $D_{E,V} \models \exists \mathbf{y}.\varphi(\mathbf{y}) \to \bot$ iff $D_{E,V} \not\models \exists \mathbf{y}.\varphi(\mathbf{y})$
- $D_{E,V} \models q(x, y) \to \mathsf{EqO}(x, y)$ iff $q(D_{E,V}) \subseteq E$
- $D_{E,V} \models q(x_t, y_t) \to \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$ 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 $\to$ may be either $\Rightarrow$ or $\dashrightarrow$. We write $D_{E,V} \models \Lambda$ iff $D_{E,V} \models \lambda$ for every $\lambda \in \Lambda$.

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

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

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

tid	pid	title	conf	chr
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

$\delta_1 = \text{Author}(t,a,n,i) \wedge \text{Author}(t',a,n',i') \wedge n \neq n' \rightarrow \bot$

$\delta_2 = \text{Paper}(t,p,ti,c,a) \wedge \text{Wrote}(t',a,p) \rightarrow \bot$

$\rho_1^o = \text{Author}(t,x,n,i) \wedge \text{Author}(t',y,n,i) \Rightarrow \mathsf{EqO}(x,y)$

$\rho_1^v = \text{Author}(t,a,n,i) \wedge \text{Author}(t',a,n',i') \wedge n \approx n' \Rightarrow \mathsf{EqV}(\langle t,2\rangle, \langle t',2\rangle)$

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

Figure 1: Schema $\mathcal{S}_{\mathsf{ex}}$, database D_{ex}, and 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 $\Delta_{\mathsf{ex}} = \{\delta_1, \delta_2\}$. Similarity relation $\approx$ is defined so names of authors a_1, a_2, a_3, a_4, and a_5 are all pairwise similar, and the names of a_6 and a_8 are both similar to the name of a_7 (but not similar to each other).

With these notions in hand, we can formally define solutions to an ER specification and dataset. The definition employs the notation $\mathsf{EqRel}(P,S)$, giving the least equivalence relation on set S that contains all pairs in P (i.e. minimally extending P to satisfy reflexivity, symmetry, and transitivity).

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 three conditions:*

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

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

Observe that by construction, every merge occurring in a solution can be justified by providing the sequence of rule applications and closure operations [11] that led to the merge being incorporated into the solution (alternatively, such steps may be presented as a proof tree, as formalized and illustrated in the long version of [56]). Importantly, as rule bodies do not involve any kind of negation (in particular, no $\neq$-atoms), 'later' merges cannot invalidate the reasons for performing an 'earlier' merge.

We note that a database-specification pair may admit zero, one, or several solutions. The absence of solutions arises from constraint violations (either initially present or introduced by the hard rules) which cannot be repaired solely through permitted merges. The existence of multiple solutions is due to some combinations of merges not being possible without violating the constraints, leading to a choice of which possible merges to include. We return to our running example[5] to 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. If local merges were not permitted (as was the case 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*

[5] Additional examples of Lace specifications and solutions can be found in [11, 12, 56, 57].

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 database 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 the initial database D_{ex} satisfies the constraints and hard rules.

Rather than considering all solutions, it is natural to restrict attention to the 'best' ones. We shall therefore focus on solutions that are maximal w.r.t. set inclusion, i.e. derive as many merges as possible subject to the constraints. Alternative optimality criteria can also be considered, see Section 3.4.

DEFINITION 5. *The set* $\mathsf{MaxSol}(D, \Sigma)$ *of* maximal solutions *for* (D, Σ) *contains those* $\langle E, V\rangle \in \mathsf{Sol}(D, \Sigma)$ *for which there is no solution* $\langle E', V'\rangle \in \mathsf{Sol}(D, \Sigma)$ *with* $E \cup V \subsetneq E' \cup V'$.

EXAMPLE 3. *The solution* $\langle E_{\mathsf{ex}}, V_{\mathsf{ex}}\rangle$ *described in Example 2 is not maximal 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: the first 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 the second extends* $\langle E_{\mathsf{ex}}, V_{\mathsf{ex}}\rangle$ *with* (a_7, a_8) *and the corresponding name cells* $(\langle t_6, 2\rangle, \langle t_7, 2\rangle)$ *(via* ρ_1^v*).*

3.3 Properties of the Framework

We briefly highlight some interesting properties of the LACE framework.

Simulating Hard Rules We can show that hard rules can be simulated by soft rules in combination with denial constraints, provided that we allow denial constraints to use atoms with similarity relations. Specifically, a hard rule $\rho^o = \varphi(x, y, \mathbf{z}) \Rightarrow \mathsf{EqO}(x, y)$ can be replaced by a soft rule $\sigma_{\rho^o} = \varphi(x, y, \mathbf{z}) \dashrightarrow \mathsf{EqO}(x, y)$ and DC $\delta_{\rho^o} = \varphi(x, y, \mathbf{z}) \wedge x \neq y \rightarrow \bot$. Similarly, $\rho^v = \varphi(x_t, y_t, \mathbf{z}) \Rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$ is replaced by $\sigma_{\rho^v} = \varphi(x_t, y_t, \mathbf{z}) \dashrightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$ and $\delta_{\rho^v} = \varphi(x_t, y_t, \mathbf{z}) \wedge u_{x_t,i} \neq u_{y_t,j} \rightarrow \bot$, with $u_{x_t,i}$ (resp. $u_{y_t,j}$) the ith (resp. jth) argument of the atom in $\varphi(x_t, y_t, \mathbf{z})$ with tid x_t (resp. y_t).

THEOREM 1. *Consider an ER specification* $\Sigma = \langle \Gamma_h^o \cup \Gamma_s^o, \Gamma_h^v \cup \Gamma_s^v, \Delta\rangle$ *over* $\mathcal{S}$, *and define* Σ' *as the specification* $\langle \Gamma_{h\rightsquigarrow s}^o, \Gamma_{h\rightsquigarrow s}^v, \Delta'\rangle$ *with* $\Gamma_{h\rightsquigarrow s}^o = \Gamma_s^o \cup \{\sigma_{\rho^o} \mid \rho^o \in \Gamma_h^o\}$, $\Gamma_{h\rightsquigarrow s}^v = \Gamma_s^v \cup \{\sigma_{\rho^v} \mid \rho^v \in \Gamma_h^v\}$, *and* $\Delta' = \Delta \cup \{\delta_\rho \mid \rho \in \Gamma_h^o \cup \Gamma_h^v\}$. *Then* $\mathsf{Sol}(D, \Sigma) = \mathsf{Sol}(D, \Sigma')$ *for each* $\mathcal{S}$*-database* D.

Local Can Simulate Global Interestingly, we can show that it is possible to simulate global merges of objects using local value merges. To formulate the result, we will use $\mathcal{S}_{\mathsf{V}}$ for the schema with the same relations as $\mathcal{S}$ but with all object positions changed to value positions, and use D_{V} for the dataset D but with all object constants treated as value constants.

THEOREM 2. *For every ER specification* $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$ *over schema* $\mathcal{S}$, *there exists a specification* $\Sigma' = \langle \emptyset, \Gamma'_V, \Delta\rangle$ *over* $\mathcal{S}_{\mathsf{V}}$ *s.t. for every* $\mathcal{S}$*-database* D: $\mathsf{Sol}(D_{\mathsf{V}}, \Sigma') = \{\langle \emptyset, V \cup V_E\rangle \mid \langle E, V\rangle \in \mathsf{Sol}(D, \Sigma)\}$, *where* $V_E = \{(\langle t, i\rangle, \langle t', j\rangle) \mid (t[i], t'[j]) \in E\}$.

PROOF SKETCH. Every rule $q(x, y) \rightarrow \mathsf{EqO}(x, y)$ is replaced by a rule $q(x_t, y_t) \rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$, where $\langle x_t, i\rangle$ (resp. $\langle y_t, j\rangle$) is any position that contains x (resp. y) in q. Additionally, we include all rules $P(x_t, \mathbf{u}_1^{i-1}, z, \mathbf{u}_{i+1}^k) \wedge P'(y_t, \mathbf{v}_1^{j-1}, z, \mathbf{v}_{j+1}^\ell) \Rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, j\rangle)$ such that $\mathbf{u}_1^{i-1}$ abbreviates the tuple of distinct variables $u_1, \ldots, u_{i-1}$ (likewise for $\mathbf{u}_{i+1}^k$, $\mathbf{v}_1^{j-1}$, $\mathbf{v}_{j+1}^\ell$) and i and j are object positions of P/k and P'/ℓ w.r.t. the original schema $\mathcal{S}$. □

Note that there can be no analogous translation from local to global, for the simple reason that the equivalence relation for objects cannot distinguish between different occurrences of a same constant.

3.4 Alternative Optimality Criteria

While focusing on solutions with a $\subseteq$-maximal set of merges is arguably reasonable, there are other natural optimality criteria that can be used instead. For example, we may want to give more importance to a merge that is supported by multiple rules, or compare solutions based upon soft rule violations. To formalize these criteria, we will use the notion of active pair: (o, o') (resp. $(\langle t, i\rangle, \langle t', i'\rangle)$) is *active* in $D_{E,V}$ w.r.t. $q(x, y) \rightarrow \mathsf{EqO}(x, y)$ (resp. $q(x_t, y_t) \rightarrow \mathsf{EqV}(\langle x_t, i\rangle, \langle y_t, i'\rangle)$) if $(o, o') \in q(D_{E,V})$ (resp. $(t, t') \in D_{E,V}$). We then define $\mathbf{ap}(D, E, V, \Gamma)$ as the set of all (μ, ρ) such that pair μ is active in $D_{E,V}$ w.r.t. rule $\rho \in \Gamma$. Our proposed optimality criteria are obtained by associating each solution $\langle E, V\rangle$ with one of the following sets (using Γ for $\Gamma_O \cup \Gamma_V$):

$$\text{EQ}(E, V) = E \cup V$$
$$\text{SP}(E, V) = \{(\mu, \rho) \in \mathbf{ap}(D, E, V, \Gamma) \mid \mu \in E \cup V\}$$
$$\text{AB}(E, V) = \{\mu \mid (\mu, \rho) \in \mathbf{ap}(D, E, V, \Gamma), \mu \notin E \cup V\}$$
$$\text{VIO}(E, V) = \{(\mu, \rho) \in \mathbf{ap}(D, E, V, \Gamma) \mid \mu \notin E \cup V\}$$

Observe that $\text{SP}(E,V)$ refines $\text{EQ}(E,V)$ by indicating the supporting rules for merges. Likewise, $\text{AB}(E,V)$ gives only the active but absent pairs, while $\text{VIO}(E,V)$ records which soft rules the absent pair violates. The resulting optimality criteria are:

- maxES/maxEC: maximize $\text{EQ}(E,V)$
- maxSS/maxSC: maximize $\text{SP}(E,V)$
- minAS/minAC: minimize $\text{AB}(E,V)$
- minVS/minVC: minimize $\text{VIO}(E,V)$

where the final S (resp. C) indicates comparison using set inclusion (resp. set cardinality). For example, a solution $\langle E,V\rangle$ is minVC-optimal if there is no other solution $\langle E',V'\rangle$ such that $\text{VIO}(E',V') < \text{VIO}(E,V)$. Note that maxES-optimal solutions correspond to the maximal solutions of Definition 5. It can be shown that the optimality criteria give rise to different sets of optimal solutions, except for maxES and maxSS, which actually coincide. Thus, there are overall seven distinct optimality criteria.

4 Complexity Results

In this section, we analyze the computational complexity of the main decision problems associated with the framework. Since the database size is typically order of magnitude larger than the other components, we focus on the *data complexity* measure, i.e. the complexity w.r.t. the size of the database D (and also $\langle E,V\rangle$ for those problems that require it).

We start with the solution recognition problem (Rec): decide if a given $\langle E,V\rangle$ belongs to $\mathsf{Sol}(D,\Sigma)$. To solve this problem, it is enough to verify that $D_{E,V} \models \Gamma_h^o \cup \Gamma_h^v \cup \Delta$ and that $\langle E,V\rangle$ is a candidate solution for (D,Σ). The latter can be done by starting with the empty set of merges and then repeatedly applying the rules in Σ to check whether some $\alpha \in E \cup V$ can be added to the current set.

Theorem 3. *Rec is P-complete.*

By contrast, the problem of deciding whether there exists a solution (Existence) is intractable.

Theorem 4. *Existence is NP-complete.*

Proof sketch. The upper bound is trivial: guess $\langle E,V\rangle$ and check if it is a solution for (D,Σ).

The lower bound is by reduction from the 3CNF problem. Consider $\phi = c_1 \wedge \ldots \wedge c_m$ over the variables $x_1,\ldots,x_n$, where $c_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$. Denote by $x_{i,j}$ the variable of $\ell_{i,j}$, and set $s_{i,j} = t$ if $\ell_{i,j} = x_{i,j}$ and $s_{i,j} = f$ if $\ell_{i,j} = \neg x_{i,j}$. We encode ϕ with a database comprising the following facts:

$$\{V(t_{x_i}, x_i) \mid 1 \le i \le n\} \cup$$
$$\{Prec(t_{p_i}, x_i, x_{i+1}) \mid 1 \le i < n\} \cup$$
$$\{R_{s_{i,1}s_{i,2}s_{i,3}}(t_{c_i}, x_{i,1}, x_{i,2}, x_{i,3}) \mid 1 \le i \le m\} \cup$$
$$\{FV(t_f, x_1), LV(t_l, x_n), T(t_1, 1), F(t_0, 0)\} \cup$$
$$\{Q(t_{Q_0}, 0), Q(t_{Q_1}, 1), C_1(t_{C_1}, c_1), C_2(t_{C_2}, c_2)\}.$$

For instance, a clause of the form $c_i = x_k \vee \neg x_z \vee x_w$ is represented as $R_{tft}(t_{c_i}, x_k, x_z, x_w)$. The fixed ER specification Σ_{3CNF} has soft rules $V(t_1,x) \wedge Q(t_2,y) \wedge FV(t_3,x) \dashrightarrow \mathsf{EqO}(x,y)$, $V(t_1,x) \wedge Q(t_2,y) \wedge Prec(t_3,x_p,x) \wedge Q(t_4,x_p) \dashrightarrow \mathsf{EqO}(x,y)$, and $C_1(t_1,x) \wedge C_2(t_2,y) \wedge Q(t_3,z) \wedge LV(t_4,z) \dashrightarrow \mathsf{EqO}(x,y)$. The first two allow variables to merge with either 0 or 1. Once every variable has been assigned a truth value, the third rule merges c_1 and c_2. The DCs in Σ_{3CNF} ensure the merges yield a proper truth assignment ($F(t_1,y) \wedge T(t_2,y) \rightarrow \bot$) not violating any clause ($R_{tft}(t_1,y_1,y_2,y_3) \wedge F(t_2,y_1) \wedge T(t_3,y_2) \wedge F(t_4,y_3) \rightarrow \bot$, and similarly for other clause types). Finally, $C_1(t_1,y_1) \wedge C_2(t_2,y_2) \wedge y_1 \neq y_2 \rightarrow \bot$ requires c_1 and c_2 to be merged, which means a truth assignment is generated. □

Another central problem is deciding whether $\langle E,V\rangle \in \mathsf{MaxSol}(D,\Sigma)$ (MaxRec). The coNP upper bound is easy (guess $\langle E',V'\rangle$, check that $\langle E',V'\rangle \in \mathsf{Sol}(D,\Sigma)$ and $E \cup V \subsetneq E' \cup V'$), and a coNP lower bound can be obtained by slightly extending the one for Existence. The basic idea is to introduce a new fact $C(t^*,e,e')$ and new soft rule $C(t,x,y) \rightarrow \mathsf{EqO}(x,y)$. The first soft rule is then replaced by $V(t_1,x) \wedge Q(t_2,y) \wedge FV(t_3,x) \wedge C(t_4,z,z) \dashrightarrow \mathsf{EqO}(x,y)$, allowing x_1 to merge with either 0 or 1 (and enabling such merges for later variables) only if e and e' have been previously merged. As a result, $\langle \mathsf{EqRel}(\emptyset, \mathsf{Obj}(D^\phi)), \mathsf{EqRel}(\emptyset, \mathsf{Cells}(D^\phi))\rangle$ is a maximal solution for $(D^\phi, \Sigma'_{\text{3CNF}})$ iff ϕ is unsatisfiable.

Theorem 5. *MaxRec is coNP-complete.*

Other key tasks involve formal reasoning over the maximal solutions of a database-specification pair. We start with two tasks that enable a *credulous* form of reasoning: deciding whether (*i*) a given merge α is such that $\alpha \in \langle E,V\rangle$ for *some* $\langle E,V\rangle \in \mathsf{MaxSol}(D,\Sigma)$ (PossMerge) and (*ii*) a given CQ q and tuple $\mathbf{c}$ is such that $\mathbf{c} \in q(D_{E,V})$ for *some* $\langle E,V\rangle \in \mathsf{MaxSol}(D,\Sigma)$ (PossAns).

Theorem 6. *Both PossMerge and PossAns are NP-complete.*

Proof sketch. The upper bounds are based on guessing $\langle E,V\rangle$ and then checking that $\langle E,V\rangle \in$

$\mathsf{Sol}(D,\Sigma)$ and $\alpha \in E \cup V$ (resp. $\mathbf{c} \in q(D_{E,V})$). Obviously, if $\alpha \in \langle E,V\rangle$ (resp. $\mathbf{c} \in q(D_{E,V})$) for $\langle E,V\rangle \in \mathsf{Sol}(D,\Sigma)$, then $\alpha \in \langle \mathcal{E},\mathcal{V}\rangle$ (resp. $\mathbf{c} \in q(D_{\mathcal{E},\mathcal{V}})$) for some $\langle \mathcal{E},\mathcal{V}\rangle \in \mathsf{MaxSol}(D,\Sigma)$ with $E \cup V \subsetneq \mathcal{E} \cup \mathcal{V}$. For the lower bounds, consider the specification obtained from the one in the proof of Theorem 4 by removing the last denial constraint. Then, (c_1, c_2) is a possible merge (resp. (c_1) is a possible answer to $q(x) = C_1(x) \wedge C_2(x)$) iff ϕ is satisfiable. □

We now investigate a *skeptical* form of reasoning through the decision problems CertMerge and CertAns, defined as PossMerge and PossAns, respectively, but with the additional requirement that $\alpha \in \langle E,V\rangle$ for *all* $\langle E,V\rangle \in \mathsf{MaxSol}(D,\Sigma)$ in the case of CertMerge, and that $\mathbf{c} \in q(D_{E,V})$ for *all* $\langle E,V\rangle \in \mathsf{MaxSol}(D,\Sigma)$ in the case of CertAns. This problem can be solved by guessing $\langle E,V\rangle$, and then checking $\langle E,V\rangle \in \mathsf{MaxSol}(D,\Sigma)$ and $\alpha \notin E \cup V$ (resp. $\mathbf{c} \notin q(D_{E,V})$), which puts the problems in Π_2^p. The main difference with PossMerge and PossAns is that one needs to check that the guessed pair is a *maximal* solution, not just a solution.

Theorem 7. *Both CertMerge and CertAns are Π_2^p-complete.*

All of our lower bounds hold already for fixed ER specifications having only rules for objects, and, moreover, they can be adapted to apply to ER specifications using only FDs as denial constraints.

We now consider the case of *restricted* specifications, i.e. ER specifications whose DCs do not use inequality atoms. Such $\neq$-free DCs are widely used in ontologies, e.g. to express class disjointness, and are available in popular ontology languages such as *DL-Lite* [19]. While the results in Theorems 3 and 6 apply already to restricted ER specifications, the other tasks become easier under standard complexity assumptions. Intuitively, this is because constraint violations are preserved under merges.

Theorem 8. *For restricted ER specifications, Existence and MaxRec are P-complete, while CertMerge and CertAns are coNP-complete.*

For the other optimality criteria discussed in Section 3.4, we studied the complexity of recognizing optimal solutions [57]. The next result shows that this problem is coNP-complete for all the optimality criteria, just as in the case of maxES (Theorem 5).

Theorem 9. *For all defined optimality criteria, recognition of optimal solutions is coNP-complete.*

Interestingly, for restricted ER specifications, while the problem is P-complete for the optimality criteria based on set-inclusion (as for maxES, see Theorem 8), the problem remains intractable for the optimality criteria based on set cardinality.

Theorem 10. *For restricted ER specifications, recognition of optimal solutions is P-complete (resp. coNP-complete) for all the optimality criteria based on set inclusion (resp. set cardinality).*

5 LACE Implementation

In order to explore the practical interest of Lace, we have implemented the framework using answer set programming (ASP), a well-studied paradigm for declarative problem solving [43]. The suitability of employing ASP for tackling data quality tasks has been previously demonstrated by work on data cleaning with (relational) MDs [4, 5] and consistent query answering [25, 45].

5.1 ASP Encoding of LACE specifications

Given an ER specification Σ, the ASP encoding of Σ is a program Π_Σ containing an ASP rule for each (hard or soft) rule in Σ. Predicates eqo and eqv are used to store merges of objects and values respectively, and additional rules are used to ensure that eqo and eqv are equivalence relations.

Rather than providing the full encoding, which requires introducing quite a lot of notation, we shall describe the main ideas underlying the encoding. A hard rule for objects will have head atom $\mathtt{eqo}(X,Y)$, enforcing that X and Y are merged. A soft rule for objects can be elegantly encoded using a choice rule (in ASP parlance) whose head $\{\mathtt{eqo}(X,Y)\}$ allows but does not require that $\mathtt{eqo}(X,Y)$ is made true when the rule body holds. Rules for values are instead use head atom $\mathtt{eqv}(T,I,T',J)$ to encode merges of cells, again with choice rules used to capture the semantics of soft rules for values.

The translation of Lace rule bodies into the corresponding ASP rule bodies is a bit more involved, as we need to simulate the effect of evaluating the rule body over the induced instance. This is accomplished by instantiating every variable position with a distinct variable, then using adding eqo and eqv to enforce that the required positions 'join' in the induced database. The encoding of rule bodies must also ensure that similarity atoms are evaluated using the common set of values for the compared variables (cf. Definition 3).

As the following result shows, the stable models of the ASP program capture Lace solutions:

Theorem 11. *For every database D and specification $\Sigma = \langle \Gamma_O, \Gamma_V, \Delta\rangle$: $\langle E,V\rangle \in \mathsf{Sol}(D,\Sigma)$ iff $E = \{(a,b) \mid \mathtt{eqo}(a,b) \in M\}$ and $V =$*

$\{(\langle t, i\rangle, \langle t', i'\rangle) \mid \texttt{eqv}(t, i, t', i') \in M\}$ *for a stable model* M *of* (Π_Σ, D).

5.2 Similarity Computation

Differently from the original ASP encoding in [11], similarity relations are implemented with a predicate $\texttt{sim}_i(X, Y, S)$, where X and Y are instantiated with the constants to be assessed for similarity, and S with a similarity score. This allows the same similarity measure to be used in different rules with different thresholds (specified using comparison atoms, e.g. $S > 95$, in the rule body). In the specifications used in our evaluation, we employ similarity atoms based upon Levenshtein distance for numerical constants, Jaro–Winkler distance for short strings, and TF–IDF cosine score for long textual values.

A key challenge is how to efficiently evaluate the $\texttt{sim}_i$ atoms, since the input dataset does not contain $\texttt{sim}_i$ facts, which must instead be computed via external functions (e.g., string similarity measures). A naïve approach is to precompute the set of all facts $\texttt{sim}_i(c, d, s)$ where c and d are compatible values and $s = f_i(c, d)$, with f_i the function underlying the relation $\texttt{sim}_i$. Although this only needs polynomially many function calls in $|D|$, it is prohibitively costly on even moderately-sized databases.

This led us to explore a more sophisticated strategy for similarity computation [56], which exploits the program structure to better identify which pairs of facts need to be compared. Roughly speaking, this is achieved by considering different simplifications of the original program, coupled with online calls to the external functions. The empirical evaluation conducted in [56] shows that this strategy can achieve substantial improvements in runtime and memory efficiency, especially on larger datasets.

5.3 ASPEN and ASPEN$^+$

The systems ASPen [56] and ASPen$^+$ [57] implement the Lace framework, employing the ASP encoding described in Section 5.1 and exploiting the capabilities of the `clingo` ASP solver [36] to generate and reason about ER solutions. By utilizing the diverse reasoning modes available in `clingo`, ASPen can produce one or more (maximal) solutions as well as other relevant merge sets, like the set of possible merges and an approximation of the set of certain merges. Explainability is achieved through integration with `xclingo` [18], which enables the generation of proof trees that justify individual merges by explicitly tracing the rule applications that led to a merge being derived.

ASPen$^+$ extends ASPen's functionality by introducing support for local merges, thereby enabling context-specific value resolution in addition to global merges. Furthermore, ASPen$^+$ implements the set of optimality criteria described in Section 3.4, allowing the selection of preferred solutions according to various optimality criteria. This optimization capability is realized through the `asprin` framework [15, 16], which provides declarative preference handling within ASP.

5.4 Experiments

ASPen and ASPen$^+$ are publicly available[6], and experimental results show strong performance across real-world ER benchmarks and synthetic data, each with ground truth merges. Both systems have been evaluated against existing open-source systems, Magellan and JedAI [38, 48], that support rule-based ER, which we take as our baselines.

Effectiveness and Scalability Across all datasets, ASPen and ASPen$^+$ achieved consistently higher F1 scores than the baseline systems, with performance gaps of up to 86% on multi-relational datasets where traditional ER approaches perform poorly. These quality gains, however, came with a cost in computational efficiency: both baselines were significantly faster than the ASP-based ones.

The scalability analysis indicates that runtime is sensitive to several factors. Increasing the *data size* or the *duplicate ratio*, or reducing the *similarity thresholds*, leads to substantial slowdowns. In extreme cases, these variations caused up to a 300-fold increase in execution time. This reflects the higher computational complexity of the ASP-based approach, particularly under parameter settings that expand the search space of possible merges.

Impact of Local Merges With global semantics alone, ASPen programs often admit no solution when all natural FDs were included in specifications, especially in noisy datasets. ASPen$^+$, by supporting local semantics, accommodated all FDs and achieved higher-quality results, even outperforming ASPen in cases where both could satisfy the constraints. Overall, this analysis shows that local semantics provides greater robustness to data variability and constraint interactions, enabling more complete and accurate ER solutions.

Optimality Criteria On datasets with few null values and little variation in values, criteria that maximize the number of merges (maxE and maxS) achieved the highest F1 scores because they focus on coverage. On noisier datasets, criteria that minimize rule violations (minA and minV) performed

[6] https://github.com/zl-xiang/Aspen

better, as their emphasis on precision reduced incorrect merges. In all settings, solutions based on set inclusion were usually faster to compute, while solutions based on the number of merges were often closer to the gold standard. The improvement in quality from using the latter often required significantly higher computation times, making the former more suitable when resources are limited.

6 Combining ER with Repairs

Real-world databases may suffer from multiple data quality issues. Some constraint violations may result from the use of different constants for the same entity, and thus may be resolved through merging constants, but others may stem from the presence of erroneous facts and can only be resolved by repairing the data, i.e. removing or modifying facts. A pipeline approach, applying ER and repairing methods in sequence, may miss useful synergies. For example, by merging two constants, we may resolve an FD violation without the need to delete facts, while conversely, deleting incorrect facts may enable some desirable merges. This suggests the interest of developing holistic approaches to jointly deduplicating and repairing data, an idea which was been advocated in [23, 34] but remains little explored.

These considerations motivated us to propose the REPLACE framework [12], an extension of LACE that allows for both merge and fact deletion operations. Fact deletions make it possible to obtain meaningful solutions when $\mathsf{Sol}(D, \Sigma) = \emptyset$, but also to discover additional merges that were blocked due to constraint violations. The REPLACE framework employs the same form of specifications as LACE, but redefines the notion of solution, which now takes the form[7] $\langle R, E, V\rangle$, with E, V equivalence relations as before and R is a set of facts to delete from D.

DEFINITION 6. *Given a specification Σ over $\mathcal{S}$ and $\mathcal{S}$-database D, we call $\langle R, E, V\rangle$ a* REP-solution for (D, Σ) *if $R \subseteq D$ and $\langle E, V\rangle \in \mathsf{Sol}(D \setminus R, \Sigma)$.*

Similarly to LACE, we naturally prefer solutions that contain more merges. However, we also want to retain as much information as possible, hence should minimize fact deletions, as is done when defining repairs. These two criteria may conflict, as deleting more facts may enable more merges. This lead us to consider three natural ways to compare REP-solutions: give priority to maximizing merges (MER), give priority to minimizing deletions (DEL), or adopt the Pareto principle and accord equal priority to both criteria (PAR). Using $X \in \{\text{MER}, \text{DEL}, \text{PAR}\}$ for comparison, we obtain the set $\mathsf{Sol}_X^{\text{REP}}(D, \Sigma)$ of $\preceq_X$-optimal REP-solutions.

It is easily verified that we always have $\mathsf{Sol}_{\text{MER}}^{\text{REP}}(D, \Sigma) \subseteq \mathsf{Sol}_{\text{PAR}}^{\text{REP}}(D, \Sigma)$ and $\mathsf{Sol}_{\text{DEL}}^{\text{REP}}(D, \Sigma) \subseteq \mathsf{Sol}_{\text{PAR}}^{\text{REP}}(D, \Sigma)$, while the converse inclusions do not hold in general. We further observe that $\langle \emptyset, E, V\rangle \in \mathsf{Sol}_{\text{DEL}}^{\text{REP}}(D, \Sigma)$ iff $\langle \emptyset, E, V\rangle \in \mathsf{Sol}_{\text{PAR}}^{\text{REP}}(D, \Sigma)$ iff $\langle E, V\rangle \in \mathsf{MaxSol}(D, \Sigma)$. Thus, maximal solutions in LACE are special cases of $\preceq_{\text{DEL}}$- and $\preceq_{\text{PAR}}$-optimal solutions (an analogous property does not hold for $\preceq_{\text{MER}}$). REP-solutions can also be related to the subset repairs employed in consistent query answering [3, 7, 20]. Indeed, if we consider (D, Σ) with $\Sigma = \langle \emptyset, \emptyset, \Delta\rangle$, then $\mathsf{Sol}_{\text{MER}}^{\text{REP}}(D, \Sigma)$, $\mathsf{Sol}_{\text{DEL}}^{\text{REP}}(D, \Sigma)$, and $\mathsf{Sol}_{\text{PAR}}^{\text{REP}}(D, \Sigma)$ coincide and contain only solutions of the form $(R, \mathsf{trivE}, \mathsf{trivCells})$, with trivE and $\mathsf{trivCells}$ the trivial equivalence relations over $\mathsf{Obj}(D)$ and $\mathsf{Cells}(D)$. It is readily verified that $\langle R, \mathsf{trivE}, \mathsf{trivCells}\rangle \in \mathsf{Sol}_{\text{MER}}^{\text{REP}}(D, \Sigma) = \mathsf{Sol}_{\text{DEL}}^{\text{REP}}(D, \Sigma) = \mathsf{Sol}_{\text{PAR}}^{\text{REP}}(D, \Sigma)$ iff $D \setminus R$ is a repair.

The complexity analysis of REPLACE carried out in [12] reveals that in almost all cases, the addition of delete operations to LACE does not affect the complexity of recognizing (maximal / optimal) solutions or certain and possible answers.

7 Overview of Logic-Based ER Methods

As a foundational and multifaceted task in computer science, entity resolution has been tackled using a variety of different approaches, including probabilistic models, (deep) learning techniques, and logical methods [21]. In this section, we provide a brief overview of logic-based approaches to ER and related problems[8]. We will compare these works by considering (i) which ER problem is tackled and what constitutes a solution, (ii) what kind of rules and/or constraints are employed, (iii) the nature of the semantics (static vs. dynamic, local and/or global merges), and (iv) the existence of an accompanying implementation.

Dedupalog [2] was the first logic-based framework targeting collective ER. It employs soft and hard datalog-style rules and also allows for rules with negated heads, to indicate likely non-merges. Dedupalog allows for recursive rules, but due to the static semantics, it is unclear how to extract a non-circular derivation of produced merges. The semantics can be characterized as global, as solutions in Dedu-

[7] As REPLACE [12] extends the original LACE framework [11], it only supports global merges. Definition 6 adapts the notion of solution from [12] to accommodate local merges. It can be verified that the complexity results in [12] hold also for this modified definition.

[8] A detailed account of early logic-based approaches and their precursors can be found in Chapter 4 of [29].

palog define equivalence relations over entity references from designated relations. Solutions are required to satisfy all hard rules and should minimize violations of soft rules. However, for efficiency reasons, the Dedupalog system (not publicly available) generates a single approximately optimal solution.

Matching dependencies (MDs) specify conditions under which pairs of attribute values in database facts must be matched [27, 29, 32], i.e., made equal. Formally, an MD is an expression of the form

$$R_1[\vec{X}_1] \approx R_2[\vec{X}_2] \;\; \rightarrow \;\; R_1[Y_1] \doteq R_2[Y_2],$$

which states that if the projections of an R_1-fact t_1 and an R_2-fact t_2 onto attributes $\vec{X}_1$ and $\vec{X}_2$ are pairwise similar, then the Y_1-value of t_1 and the Y_2-value of t_2 must be made equal. Relational MDs [4, 5] generalize MDs by allowing additional atoms in the body, supporting collective scenarios. (Relational) MDs are equipped with a dynamic semantics: when the body condition is satisfied, values are (locally) updated to ensure satisfaction of the head. In [8], this is formalized using a chase-like procedure that repairs violations of MDs, using matching functions to determine the resulting value when two values are matched (rather than using the set of merged constants). Although (relational) MDs can be viewed as hard constraints (since they must be satisfied), the order in which rules are applied affects the outcome, as value modifications may lead to multiple possible solutions.

More recently, entity-enhancing rules (REEs) have been introduced [33], which extend both relational MDs and (conditional) functional dependencies by incorporating machine learning predicates and attribute-value comparisons. In the context of entity resolution, REEs focus on the global matching of tuple IDs through a chase-like procedure, which if successful, yields a unique updated data instance in which all REEs are satisfied (REEs are thus treated as hard rules). Although the framework can infer (in)equalities among tuple cells, the presence of multiple representations of a data value is regarded as an error that must be resolved by the end user. Aside from entity resolution, REEs can also be used for other data quality tasks, like conflict resolution and data imputation.

The recently proposed CERQ framework [26] considers ER in the setting of knowledge bases consisting of facts, tuple-generating dependencies (tgds), and equality-generating dependencies (egds). Intuitively, the egds act as hard rules for objects and values, and the tgds support (open-world) inference of new facts. The chase-based semantics is dynamic and supports both local and global merges, where the notion of instance takes a very similar form to our notion of induced database (having, in particular, sets of values in value positions). Interestingly, although developed independently, the semantics for the satisfaction of queries and rules over instances with sets of constants shares the same principle adopted in Lace. As the CERQ framework does not consider soft rules and denial constraints, it is possible to define the notion of universal solution as the preferred output (which can be used to support conjunctive query answering when the chase terminates). Finally, we mention that the CERQ framework does not have an implementation.

A declarative framework for entity linking (EL) based upon link-to-source constraints was presented in [17]. In contrast to ER, whose aim is to infer which entities that correspond to the same real-world object, this work is concerned with discovering other kinds of binary relations linking pairs of entities. As a result, link relations are not constrained to equivalence relations, and it is not evident how one can force relations to act as equivalence relations to adapt the framework to handle collective ER (in particular, recursive ER scenarios, cf. discussion in [11]). The static semantics characterizes a space of maximal solutions, from which notions of certain and possible links are defined.

There have also been several efforts to develop practical systems for ER based upon declarative formalisms. Prominent examples include the open-source systems Magellan [38] and JeDAI [48], which address simpler ER settings that match tuples from a pair of tables (or within a single table). Both systems support syntactically simple rules, formulated as single-pass conditions based on similarity measures between attribute values of tuple pairs. ERBlox [5] is a system for collective ER based upon relational MDs. It represents one of the earliest efforts to combine machine learning techniques with declarative approaches to ER. In particular, ERBlox employs ML techniques to construct a classifier that identifies blocks of duplicate candidates, over which MDs are subsequently applied for entity resolution. More recently, building on the REE framework, the industrial-scale data-cleaning system Rock [6] has been developed to address a range of data quality issues, including collective ER. These systems address scalability challenges through blocking strategies and/or parallelization and offer further functionalities to simplify usage, e.g. user interfaces, default settings, debugging, use of external KBs, support for semi-structured or unstructured data. They also showcase the interest of combining ML and declarative approaches.

8 Perspectives

We have briefly surveyed recent developments in logic-based entity resolution, as exemplified by the LACE framework. Key foundational and conceptual advances include: a differentiated treatment of object and value merges using global and local semantics, the use of dynamic semantics to enable justifiability in the presence of recursive rules, and the consideration of a space of (optimal) solutions, which can be explored using certain and possible merges and query answers. Valuable insights have also been gained from experimenting with implementations of logic-based ER, underscoring the importance of dedicated optimisations and the interest of combining logical and ML methods. Despite these important advances, many interesting foundational and practical questions remain to be tackled. We mention a few items high on our agenda:

Representation of Query Answers One of the original motivations for developing LACE and its successor REPLACE was to be able to evaluate queries w.r.t. a space of (REP-)solutions, in the spirit of consistent query answering. Note however that it is not at all obvious how best to present query results in a way that makes clear which constants have been merged and avoids returning distinct yet equivalent answer tuples. For example, if we pose the query $\exists v.P(v, x, y)$ to a database containing $P(t, c_1, c_2)$ and $P(t', c_3, c_4)$, with (c_1, c_3) and (c_2, c_4) certain object merges, then we get four certain answers: (c_1, c_2), (c_1, c_4), (c_3, c_2), and (c_3, c_4). However, we would tempted to return instead a single answer tuple consisting of *sets of constants*: $(\{c_1, c_3\}, \{c_2, c_4\})$. This idea motivated the notion of *most informative (certain or possible) answers* [12], but the definition only handles global merges and lacks a practical algorithm.

Different Forms of Explanations The notions of justification and proof trees that have been defined for LACE [11, 56] can be used to explain to users how a given merge was obtained in a given solution. It would be interesting however to consider additional forms of explanations that concern the whole space of (maximal or optimal) solutions. For example, how can we justify why a given merge (or answer) is certain, or why a possible merge (or answer) is *not* certain? Some first ideas for how to formalize such explanations might be gleaned from prior work on explaining query (non)answers under repair-based semantics [10].

Integration with Ontologies It would also be relevant to extend LACE to the setting of ontology-based data access [51, 58], in which an ontology is used to provide a convenient vocabulary for query formulation and to specify domain knowledge, which can be exploited when answering queries. To the best of our knowledge, the only work that has considered entity resolution in the context of ontologies is the recent work on CERQ [26]. However, it is non-trivial to incorporate soft ER rules and denial constraints into the latter framework. Moreover, there are other interesting questions to explore, such as how to accommodate mappings that link the data to the ontology and which may involve the creation of new entity-referring constants.

Implementation of Holistic Approaches We plan to build upon ASPEN to develop an implementation of the REPLACE framework, also drawing inspiration from existing ASP-based implementations of consistent query answering [25, 45]. The computation of certain answers to queries, which has not yet been incorporated into ASPEN, is an important functionality that will require care to implement due to its higher complexity (Π_2^p).

Scalability While our experiments show that ASPEN can successfully handle some real-world ER scenarios, scalability remains an issue. We plan to explore the potential of employing specialized data structures or custom procedures for handling equivalence relations, as has been considered for Datalog reasoners [46, 52]. As parallelization has been successfully employed in some rule-based ER systems [24], another promising direction is developing parallel algorithms, building on prior work in parallel Datalog reasoning [1, 50] and ASP solving [37].

Learning ER Specifications A major barrier to adopting logic-based approaches is the difficulty of obtaining accurate ER rules. Most existing work on rule learning for ER targets single-pass matching rules within one or two tables [39, 40, 53], although there has been some relevant recent research on discovering entity-enhancing rules [30, 31]. Moreover, the related questions of learning conjunctive queries [14, 55] and mining database constraints [22, 44, 49] have also been extensively investigated. In addition to learning ER specifications, it is also interesting to continue to explore the use of machine-learning methods for tuning specifications (e.g. setting the score thresholds for similarity relations) and for obtaining more informative similarity measures (cf. use of ML predicates in [24]).

Acknowledgments

This work has been supported by the ANR AI Chair INTENDED (ANR-19-CHIA-0014) and by MUR under the PNRR project FAIR (PE0000013).

References

[1] T. Ajileye and B. Motik. Materialisation and data partitioning algorithms for distributed RDF systems. *J. Web Semant.*, 2022.

[2] A. Arasu, C. Ré, and D. Suciu. Large-scale deduplication with constraints using dedupalog. In *Proc. of ICDE*, 2009.

[3] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. of PODS*, 1999.

[4] Z. Bahmani and L. E. Bertossi. Enforcing relational matching dependencies with datalog for entity resolution. In *Proc. of FLAIRS*, 2017.

[5] Z. Bahmani, L. E. Bertossi, and N. Vasiloglou. Erblox: Combining matching dependencies with machine learning for entity resolution. *Int. J. Approx. Reason.*, 2017.

[6] Z. Bao, B. Binbin, W. Fan, D. Li, M. Li, K. Lin, W. Lin, P. Liu, P. Liu, Z. Lv, M. Ouyang, C. Sun, S. Tang, Y. Wang, Q. Wei, X. Wu, M. Xie, J. Zhang, R. Zhao, J. Zhu, and Y. Zhu. Rock: Cleaning data with both ML and logic rules. *Proc. VLDB Endow.*, 2024.

[7] L. E. Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, 2011.

[8] L. E. Bertossi, S. Kolahi, and L. V. S. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. *Theory Comput. Syst.*, 2013.

[9] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Trans. Knowl. Discov. Data*, 2007.

[10] M. Bienvenu, C. Bourgaux, and F. Goasdoué. Computing and explaining query answers over inconsistent dl-lite knowledge bases. *J. Artif. Intell. Res.*, 2019.

[11] M. Bienvenu, G. Cima, and V. Gutiérrez-Basulto. LACE: A logical approach to collective entity resolution. In *Proc. of PODS*, 2022.

[12] M. Bienvenu, G. Cima, and V. Gutiérrez-Basulto. REPLACE: A logical framework for combining collective entity resolution and repairing. In *Proc. of IJCAI*, 2023. Long version available at `https://orca.cardiff.ac.uk/id/eprint/159626/1/main.pdf`.

[13] M. Bienvenu, G. Cima, V. Gutiérrez-Basulto, and Y. Ibáñez-García. Combining global and local merges in logic-based entity resolution. In *Proc. of KR*, 2023. Long version available at `https://arxiv.org/pdf/2305.16926`.

[14] A. Bonifati, R. Ciucanu, and S. Staworko. Learning join queries from user examples. *ACM Trans. Database Syst.*, 2016.

[15] G. Brewka, J. P. Delgrande, J. Romero, and T. Schaub. asprin: Customizing answer set preferences without a headache. In *Proc. of AAAI*, 2015.

[16] G. Brewka, J. P. Delgrande, J. Romero, and T. Schaub. A general framework for preferences in answer set programming. *Artif. Intell.*, 2023.

[17] D. Burdick, R. Fagin, P. G. Kolaitis, L. Popa, and W. Tan. A declarative framework for linking entities. *ACM Trans. Database Syst.*, 2016.

[18] P. Cabalar, J. Fandinno, and B. Muñiz. A system for explainable answer set programming. In *Proc. of ICLP*, 2020.

[19] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.*, 2007.

[20] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 2005.

[21] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.*, 2021.

[22] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *Proc. VLDB Endow.*, 2013.

[23] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *Proc. of ICDE*, 2013.

[24] T. Deng, W. Fan, P. Lu, X. Luo, X. Zhu, and W. An. Deep and collective entity resolution in parallel. In *Proc. of ICDE*, 2022.

[25] T. Eiter, M. Fink, G. Greco, and D. Lembo. Repair localization for query answering from inconsistent databases. *ACM Trans. Database Syst.*, 2008.

[26] R. Fagin, P. G. Kolaitis, D. Lembo, L. Popa, and F. Scafoglieri. A framework for combining entity resolution and query answering in knowledge bases. In *Proc. of KR*, 2023.

[27] W. Fan. Dependencies revisited for improving data quality. In *Proc. of PODS*, 2008.

[28] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Morgan & Claypool

Publishers, 2012.
[29] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.
[30] W. Fan, Z. Han, Y. Wang, and M. Xie. Parallel rule discovery from large datasets by sampling. In *Proc. of SIGMOD*, 2022.
[31] W. Fan, Z. Han, M. Xie, and G. Zhang. Discovering top-k relevant and diversified rules. *Proc. ACM Manag. Data*, 2024.
[32] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *Proc. VLDB Endow.*, 2009.
[33] W. Fan, P. Lu, and C. Tian. Unifying logic rules and machine learning for entity enhancing. *Sci. China Inf. Sci.*, 2020.
[34] W. Fan, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. *ACM J. Data Inf. Qual.*, 2014.
[35] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *J. Amer. Statist. Assoc.*, 1969.
[36] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.
[37] M. Gebser, B. Kaufmann, and T. Schaub. Multi-threaded ASP solving with clasp. *Theory Pract. Log. Program.*, 2012.
[38] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra. Magellan: Toward building entity matching management systems. *Proc. VLDB Endow.*, 2016.
[39] I. K. Koumarelas, T. Papenbrock, and F. Naumann. Mdedup: Duplicate detection with matching dependencies. *Proc. VLDB Endow.*, 2020.
[40] L. Li, J. Li, and H. Gao. Rule-based method for entity resolution. *IEEE Trans. Knowl. Data Eng.*, 2015.
[41] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 2020.
[42] Y. Li, J. Li, Y. Suhara, J. Wang, W. Hirota, and W. Tan. Deep entity matching: Challenges and opportunities. *ACM J. Data Inf. Qual.*, 2021.
[43] V. Lifschitz. *Answer Set Programming*. Springer, 2019.
[44] E. Livshits, A. Heidari, I. F. Ilyas, and B. Kimelfeld. Approximate denial constraints. *Proc. VLDB Endow.*, 2020.
[45] M. Manna, F. Ricca, and G. Terracina. Consistent query answering via ASP from different perspectives: Theory and practice. *Theory Pract. Log. Program.*, 2013.
[46] P. Nappa, D. Zhao, P. Subotic, and B. Scholz. Fast parallel equivalence relations in a datalog compiler. In *Proc. of PACT*, 2019.
[47] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 1959.
[48] G. Papadakis, G. M. Mandilaras, L. Gagliardelli, G. Simonini, E. Thanos, G. Giannakopoulos, S. Bergamaschi, T. Palpanas, and M. Koubarakis. Three-dimensional entity resolution with JedAI. *Inf. Syst.*, 2020.
[49] E. H. M. Pena, F. Porto, and F. Naumann. Fast algorithms for denial constraint discovery. *Proc. VLDB Endow.*, 2022.
[50] S. Perri, F. Ricca, and M. Sirianni. Parallel instantiation of ASP programs: techniques and experiments. *Theory Pract. Log. Program.*, 2013.
[51] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *Journal of Data Semantics*, 2008.
[52] A. Sahebolamri, L. Barrett, S. Moore, and K. K. Micinski. Bring your own data structures to datalog. *Proc. ACM Program. Lang.*, 2023.
[53] R. Singh, V. V. Meduri, A. K. Elmagarmid, S. Madden, P. Papotti, J. Quiané-Ruiz, A. Solar-Lezama, and N. Tang. Synthesizing entity matching rules by examples. *Proc. VLDB Endow.*, 2017.
[54] P. Singla and P. M. Domingos. Entity resolution with markov logic. In *Proc. of ICDM*, 2006.
[55] B. ten Cate, M. Funk, J. C. Jung, and C. Lutz. Fitting algorithms for conjunctive queries. *SIGMOD Rec.*, 2023.
[56] Z. Xiang, M. Bienvenu, G. Cima, V. Gutiérrez-Basulto, and Y. Ibáñez-García. ASPEN: ASP-based system for collective entity resolution. In *Proc. of KR*, 2024. Long version available at `https://arxiv.org/pdf/2408.06961`.
[57] Z. Xiang, M. Bienvenu, G. Cima, V. Gutiérrez-Basulto, and Y. Ibáñez-García. Advances in logic-based entity resolution: Enhancing ASPen with local merges and optimality criteria. In *Proc. of KR*, 2025. Long version available at `https:`

//github.com/zl-xiang/ASPEnP/blob/main/KR_2025_Supplementary_343.pdf.

[58] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. Ontology-based data access: A survey. In *Proc. of IJCAI*, 2018.