A Declarative Framework for Linking Entities

Doug Burdick[1], Ronald Fagin[1], Phokion G. Kolaitis[2,1], Lucian Popa[1], and Wang-Chiew Tan[2]

1 **IBM Research – Almaden**
2 **UC Santa Cruz**

Abstract

The aim of this paper is to introduce and develop a truly declarative framework for entity linking and, in particular, for entity resolution. As in some earlier approaches, our framework is based on the systematic use of constraints. However, the constraints we adopt are link-to-source constraints, unlike in earlier approaches where source-to-link constraints were used to dictate how to generate links. Our approach makes it possible to focus entirely on the intended properties of the outcome of entity linking, thus separating the constraints from any procedure of how to achieve that outcome. The core language consists of link-to-source constraints that specify the desired properties of a link relation in terms of source relations and built-in predicates such as similarity measures. A key feature of the link-to-source constraints is that they employ disjunction, which enables the declarative listing of all the reasons as to why two entities should be linked. We also consider extensions of the core language that capture collective entity resolution, by allowing inter-dependence between links.

We identify a class of "good" solutions for entity linking specifications, which we call *maximum-value solutions* and which capture the strength of a link by counting the reasons that justify it. We study natural algorithmic problems associated with these solutions, including the problem of enumerating the "good" solutions, and the problem of finding the certain links, which are the links that appear in every "good" solution. We show that these problems are tractable for the core language, but may become intractable once we allow inter-dependence between link relations. We also make some surprising connections between our declarative framework, which is deterministic, and probabilistic approaches such as ones based on Markov Logic Networks.

1998 ACM Subject Classification H.2.5 Heterogeneous Databases

Keywords and phrases entity linking, entity resolution, constraints, certain links

Digital Object Identifier 10.4230/LIPIcs.ICDT.2015.25

1 Introduction

Entity linking is a long-standing research problem that has received considerable attention over the years. The most extensively investigated case of entity linking is entity resolution, which is the problem of linking pieces of information occurring in one or more, possibly heterogeneous, datasets that refer to the same real-world object (entity). Entity resolution is known under various names: record linkage, data deduplication, reference reconciliation, merge-purge (see, e.g., [9, 11, 15, 22, 26]). Much of entity resolution research has focused on developing the algorithms, similarity measures, and the general methodologies for matching entities, while at the same time significant engineering effort has been devoted to experimenting and tuning the resulting systems.

In recent years, we have seen several new efforts aimed at raising the level of abstraction in entity resolution systems. These efforts, ranging from the earlier AJAX framework [17] to the more recent Dedupalog [2] and HIL [21] languages, represent attempts to specify, in

18th International Conference on Database Theory (ICDT'15).
Editors: Marcelo Arenas and Martín Ugarte; pp. 25–43

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a more declarative way, the basic ingredients of an entity resolution process. In particular, instead of using lower-level implementation algorithms, they employ SQL-like constructs or constraints expressed in logical formalisms as components of a high-level language. A common characteristic in these approaches is the use of *source-to-link* constraints, that is, constraints that specify the direct creation of the links from the source data. In turn, this feature has the consequence that operational semantics are used, hence the meaning of a specification in such a language is some link relation resulting from the operational semantics. For example, HIL uses SQL-like statements to express the creation of links from a set of sources and provides only an operational procedure to interpret such statements. As for Dedupalog, the specification has the form of a Datalog-style program with constraints of two types: hard constraints and soft constraints. The goal is to find a link relation that satisfies the hard constraints, and that minimizes the number of soft constraints violated. Since this turns out to be a computationally intractable problem, the semantics of Dedupalog is given by an algorithm that, in many cases, produces an approximately optimal result.

In this paper, we take a different approach to declarative entity linking (and, in particular, to declarative entity resolution), where we clearly separate the specification from the implementation, and also ensure that the implementation always satisfies the specification. Our goal is to provide a clean and expressive specification language, with rigorous semantics, which can serve as a foundation for the implementation or evaluation of entity linking systems. Two salient features of our framework are as follows.

First, we consider entity resolution as a general problem of defining links between source values. A (binary) link is modeled as a binary table that relates pairs of values from the given source relations. While, as described earlier, entity resolution is typically confined to the problem of matching entities representing the same real-world object, our framework allows linking entities that are not necessarily of the same type; in particular, a link relation need not be an equivalence relation. In other words, the same type of specification is used not only to match person records across multiple databases (which is a typical entity resolution application), but also to link a subsidiary company with its parent company or to link a CEO with his/her company.

Second, as in some of the earlier approaches, our specification language is based on constraints. However, the constraints we adopt are *link-to-source* constraints, unlike in earlier approaches where source-to-link constraints were used to dictate how to populate the link relations. Our approach makes it possible to focus entirely on the intended properties of the outcome of entity linking, thus separating the constraints from any procedure of how to achieve that outcome. The core language $\mathcal{L}_\mathbf{0}$ consists of link-to-source constraints that specify the desired properties of link relations in terms of the source relations and built-in predicates, such as similarity measures. We also consider extensions of $\mathcal{L}_\mathbf{0}$ in which other link relations may be used in the specification of link relations, thus allowing a link to also depend on other links. We distinguish two such extensions, namely, the language $\mathcal{L}_\mathbf{1}$ in which no recursion is allowed in the specification (i.e., no link relation depends on itself) and the language $\mathcal{L}_\mathbf{2}$ in which recursion is allowed; these extensions capture what is usually called *collective entity resolution* [6], where inter-dependence between links is allowed. A key feature of the link-to-source constraints is that, in their most general form, they are *disjunctive constraints* that enable the declarative listing of all the reasons as to why two entities are linked. In addition, our specification languages make use of inclusion dependencies that specify the provenance of the links w.r.t. the source data, and also allow for functional dependencies that specify when a link relation is many-to-one or one-to-one.

Since all our constraints are link-to-source, they always admit *solutions*, that is, link

relations that satisfy all the constraints at hand. (The empty link instance is always a solution, albeit not necessarily a desirable one.) Therefore, one of the main questions that has to be addressed is: what are the "good" solutions out of the space of all possible solutions? Moreover, since multiple good solutions may exist for a given specification and a given source instance, a related important problem is that of identifying the *certain links* and the *ambiguous links*, that is, those links that appear in every good solution and, respectively, in some, but not in every, good solution. From a practical point of view, the certain links are the links that should be kept, while the ambiguous links are the links that must be inspected by a human. In particular, examination of the ambiguous links may lead to a revised specification that will result into fewer ambiguous links. Thus, producing the ambiguous links is an important computational task.

As a first candidate of a class of "good" solutions, we consider *maximal solutions*, where goodness is maximality among solutions w.r.t. set containment. For each fixed entity linking specification in the core language $\mathcal{L}_\mathbf{0}$, we show that there is a polynomial-delay algorithm that, for each source instance I, enumerates all of the maximal solutions for I. (A *polynomial-delay algorithm* [24] for a problem is an algorithm that, given an input, generates all solutions to the problem, one after another, where the first solution is generated in polynomial time, and the next solution is generated in polynomial time after the previous solution.) We also show that there are polynomial-time algorithms that, given a source instance I, compute the certain links and the ambiguous links for I w.r.t. the class of maximal solutions. However, we point out that, in practice, there are too many maximal solutions, which implies that quite often there are too few certain links, if any. In other words, the semantics given by maximal solutions is too coarse-grained and does not have enough differentiating power among solutions. In view of the above, we refine the semantics by considering a subclass of maximal solutions that we call *maximum-value solutions*, which maximize the total strength of links. Under this semantics, the strength of a link is measured by counting the disjuncts and existential witnesses that "justify" the existence of a link. For each fixed entity linking specification in the core language $\mathcal{L}_\mathbf{0}$, we show that there is a polynomial-delay algorithm that, for each source instance I, enumerates all of the maximum-value solutions for I. We also show that there are polynomial-time algorithms that, for each source instance I, compute the certain links and the ambiguous links for I w.r.t. the class of maximum-value solutions.

We also establish that some existing probabilistic approaches for entity resolution can be captured, in a precise sense, by entity linking specifications in $\mathcal{L}_\mathbf{0}$ under a suitable extension of the maximum-value semantics that allows for weight functions. We start with a well-known class of probabilistic matching algorithms that originated with Fellegi and Sunter [15] and is at the core of many commercial systems, including IBM's QualityStage [23]. We show that the core logic behind these matching algorithms is captured by a fragment of $\mathcal{L}_\mathbf{0}$ where each disjunct in the matching constraint consists of a single atomic formula. We then consider the richer probabilistic framework of Markov Logic Networks (MLNs) [29], which in general allows for arbitrary first-order formulas to be interpreted in a probabilistic sense. We show that a class of *linear MLNs* that is useful for entity resolution [30] is captured by a fragment of $\mathcal{L}_\mathbf{0}$ (under the same extended semantics). Thus, rather surprisingly, a purely probabilistic approach (based on MLNs) can be captured in a deterministic way. As a byproduct, we show that for linear MLNs, there is a polynomial-delay algorithm for enumerating the maximum probability worlds, and polynomial-time algorithms for computing the certain and ambiguous links (w.r.t. the class of maximum probability worlds). To the best of our knowledge, these are the first polynomial-time results for MLN-based entity resolution.

The state of affairs turns out to be dramatically different for the extended language $\mathcal{L}_\mathbf{1}$

that allows dependence of a link on other links, but disallows recursive interdependence between links. To begin with, we show that there is a fixed entity linking specification in $\mathcal{L}_1$ for which the following problem is NP-complete: given a source instance I and a positive integer k, is there a solution for I whose value is at least k? Consequently, there is no polynomial-delay algorithm for enumerating the maximum-value solutions, unless P = NP. Moreover, we show that there is a fixed entity linking specification in $\mathcal{L}_1$ for which there are no polynomial-time algorithms for telling whether a link is certain or ambiguous w.r.t. the class of maximum-value solutions, unless NP = coNP. It should be noted that the intractability of recognizing the certain links and the ambiguous links is established by using results about the computational complexity of recognizing *frozen variables* in constraint satisfaction problems [25]. On the positive side, we identify a large syntactic fragment of $\mathcal{L}_1$ for which there is a polynomial-delay algorithm for enumerating maximum-value solutions, as well as polynomial-time algorithms for computing the certain links and the ambiguous links.

The complete proofs of all our results will appear in a full version of this paper.

2 A Declarative Framework for Linking Entities: Basics

A source relational schema is a finite sequence $\mathbf{S} = \langle R_1, \ldots, R_m \rangle$ of relation symbols, each of a fixed arity. When attribute names are not essential, we may identify attributes by their position. A *source instance* I over $\mathbf{S}$ is a sequence $(R_1^I, \ldots, R_m^I)$, where each R_i^I is a finite relation of the same arity as R_i. We often use R_i to denote both the relation symbol and the relation R_i^I that interprets it. Additionally, a *link schema* is a finite sequence $\mathbf{L} = \langle L_1, \ldots, L_n \rangle$ of link symbols, where each L_i is binary. A *link instance* J over $\mathbf{L}$ is a sequence $(L_1^J, \ldots, L_n^J)$ of finite binary relations. For a relation T (either source or link) and a tuple t in T, we denote by $T(t)$ the association between T and t and refer to it as a *fact*. When T is a link relation, we may refer to $T(t)$ as a *link*. An instance can be conveniently represented by its set of facts. Given instances K and K', we say that K is a subinstance of K' and write $K \subseteq K'$ if the set of facts in K is a subset of the set of facts in K'. We write $K \subset K'$ if this subset relationship is strict.

We specify a link relation, implicitly, by defining the properties that it must satisfy. For each link symbol L, there are three sets of constraints, as follows. The first set contains at most one *matching constraint* of the form

$$(m_L) \quad L(x, y) \rightarrow \forall \mathbf{u}(\psi(x, y, \mathbf{u}) \rightarrow \alpha_1 \vee \ldots \vee \alpha_k),$$

where $\psi(x, y, \mathbf{u})$ is a (possibly empty) conjunction of atomic formulas over $\mathbf{S}$, the universally quantified variables $\mathbf{u}$ must occur in ψ, and where $\alpha_i ::= \exists \mathbf{z}\ \phi_i(x, y, \mathbf{u}, \mathbf{z})$. Each ϕ_i is a conjunction of source atomic formulas, along with equalities and other built-in or user-defined boolean predicates (such as similarity or string containment). Also, note that x and y are universally quantified, but for simplicity of notation we omit their quantifiers.

The intuition behind the use of disjunction in the matching constraint is that it lists all the possible matching conditions (i.e., α_1, ..., α_k) for why a link $L(x, y)$ may exist. If a link $L(x, y)$ exists, then one or more of those reasons must be true. We do not require a matching constraint to be given for each link; for those links without a matching constraint, the link relation is implicitly defined by the rest of the constraints. We will give concrete examples of matching constraints shortly. We will also explain the role of the universal quantification $\forall \mathbf{u}$ and of the formula $\psi(x, y, \mathbf{u})$.

The second set of constraints, for a given link symbol L, consists of an inclusion dependency of the form $L[X] \subseteq R[A]$ and an inclusion dependency of the form $L[Y] \subseteq R'[A']$. Here,

X and Y denote the first and the second attribute of L, while A and A' denote attributes in source relations R and R'. As usual, $R[A]$ denotes the projection of R on A. The two dependencies specify an upper bound for the set of links that can appear in L: every link in L will be a pair relating a value in $R[A]$ with a value in $R'[A']$. Finally, the third set of constraints, for a given link symbol L, with attributes X and Y, consists of zero, one or both of the functional dependencies $L : X \to Y$ and $L : Y \to X$. Functional dependencies encode basic cardinality constraints on the result of entity linking. (See also [21] for the significance of such constraints in practice.) The presence of one functional dependency means that the links are required to be many-to-one, that is, an entity on one side must be linked with at most one entity on the other side. The presence of both functional dependencies means that the links must be one-to-one.

We use the term $\mathcal{L}_\mathbf{0}$ for the above language, consisting of matching constraints, and inclusion and functional dependencies. Later, we also consider extensions to $\mathcal{L}_\mathbf{0}$ (such as allowing for inter-dependencies among the links). We use the term *entity linking specification* (in $\mathcal{L}_\mathbf{0}$) for a triple $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ where $\mathbf{L}$ is a link schema, $\mathbf{S}$ is a source schema, and Σ is a set of constraints containing, for each link symbol L in $\mathbf{L}$, (1) at most one matching constraint, (2) two inclusion dependencies, and (3) zero, one or two functional dependencies, all as defined above.

▶ **Definition 1.** Let $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ be an entity linking specification, and I be a source instance. A *solution* for I w.r.t. $\mathcal{E}$ is a link instance J such that $(J, I) \models \Sigma$, where (J, I) is the instance over the schema $\mathbf{L} \cup \mathbf{S}$ obtained by taking the union of the facts in J and I.

▶ **Example 2.** In this scenario, we link subsidiaries in one database with parent companies in another database. Consider the following source schema $\mathbf{S}$:

$$\texttt{Subsid}(sid, sname, location) \qquad \texttt{Company}(cid, cname, hdqrt) \qquad \texttt{Exec}(eid, cid, name, title)$$

This source schema includes the relation symbols `Subsid` from the first database, and `Company` and `Exec` from the second database. A source instance I for $\mathbf{S}$ is given below as a set of facts:

$\texttt{Subsid}(s_1, \text{“Citibank N.A.”}, \text{“New York”})$
$\texttt{Subsid}(s_2, \text{“CIT Bank”}, \text{“Salt Lake City”})$
$\texttt{Company}(c_1, \text{“Citigroup Inc”}, \text{“New York”})$
$\texttt{Company}(c_2, \text{“CIT Group Inc”}, \text{“New York”})$
$\texttt{Exec}(e_1, c_1, \text{“E. McQuade”}, \text{“CEO, Citibank N.A.”})$

The intention is to generate links between subsidiary ids and corresponding company ids. Thus, the link schema $\mathbf{L}$ consists of a single link symbol $\texttt{L}(sid, cid)$. In the scenario,“Citibank N.A.” is the name of a true subsidiary of “Citigroup Inc”, while “CIT Bank” is the name of a true subsidiary of “CIT Group Inc”. We note that this is a real-life example, and “Citigroup Inc” and “CIT Group Inc” are two different financial institutions. The goal of entity linking is to identify links such as $L(s_1, c_1)$ and $L(s_2, c_2)$, given the available evidence.

A possible entity linking specification that exploits the available attributes and the relationship between the source tables is $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$, where Σ consists of a matching constraint:

$$\texttt{L}(sid, cid) \to \forall sn, loc, cn, hd\ (\texttt{Subsid}(sid, sn, loc) \wedge \texttt{Company}(cid, cn, hd) \to (sn \sim cn) \vee \exists e, n, t(\texttt{Exec}(e, cid, n, t) \wedge\ contains(t, sn))),$$

two inclusion dependencies $\texttt{L}[sid] \subseteq \texttt{Subsid}[sid]$, $\texttt{L}[cid] \subseteq \texttt{Company}[cid]$, and the functional dependency $\texttt{L} : sid \to cid$. While the inclusion dependencies specify where `L` is allowed to take values from, the functional dependency gives the additional requirement on `L` that the

links must be many-to-one from *sid* to *cid*. Thus, every subsidiary must link to at most one company, but the converse need not hold. The matching constraint gives the actual matching logic and includes a listing of all the possible matching conditions for why a link may exist. Concretely, if a subsidiary id and a company id are linked, then it must be that one of the two matching conditions holds: (1) there is an overlap in the names, as specified by $sn \sim cn$, or (2) there is some executive working for the company and this executive has a title that contains the subsidiary's name.

The universally quantified conjunction `Subsid` (sid, sn, loc) $\wedge$ `Company` (cid, cn, hd) gives the *context* surrounding the occurrences of *sid* and *cid* in the source relations. In general, the matching conditions can refer to any variable in the context (e.g., sn, cn), and each matching constraint's disjunction must be true for *every* instantiation of the universal variables. For example, if a subsidiary id (sid) is associated with two or more subsidiary names (sn) in the source relation `Subsid`, then the disjunction of the two matching conditions must hold for every such name. Thus, we consider every name variation of the subsidiary; if for some variation sn the matching conditions do not hold, then that may be an indication that we do not have a true subsidiary.

The following are solutions for I w.r.t. $\mathcal{E}$:

$$J_1 = \{\mathtt{L}(s_1, c_1), \mathtt{L}(s_2, c_1)\} \qquad J_2 = \{\mathtt{L}(s_1, c_1), \mathtt{L}(s_2, c_2)\}$$
$$J_3 = \{\mathtt{L}(s_1, c_2), \mathtt{L}(s_2, c_1)\} \qquad J_4 = \{\mathtt{L}(s_1, c_2), \mathtt{L}(s_2, c_2)\}$$

We assume here that the name overlap predicate $\sim$ evaluates to true for all pairs of subsidiary name and company name occurring in our instance I (thus, "Citibank N.A." $\sim$ "Citigroup Inc" but also "Citibank N.A." $\sim$ "CIT Group Inc", and so on). Note that the link $\mathtt{L}(s_1, c_1)$ satisfies both the $\sim$ predicate and the the `Exec`-based condition, while other links satisfy only the $\sim$ predicate. The link instance $J_5 = \{\mathtt{L}(s_1, c_1), \mathtt{L}(s_1, c_2)\}$ is not a solution, since it violates the functional dependency. Finally, we note that every subinstance of a solution is always a solution. ◀

The above example shows that, in general, we allow matching of entities that are not necessarily of the same type and where a link relation is not necessarily an equivalence relation.

A key feature of the language is that matching constraints do not "force" the existence of the links. They form only a necessary condition for the existence of the links. This is a departure from the more traditional approaches based on *source-to-link* rules of the form $\alpha \rightarrow L$, which eagerly populate (or require) links in L whenever the matching condition α is true. However, when other constraints are considered (e.g., functional dependencies), the links in L may become invalid. As a result, any specification that includes source-to-link rules will likely have no solutions. In contrast, our notion of entity linking specification always has solutions. A large part of this paper will then be focused on identifying a subset of "good" solutions among all the possible solutions.

Before we proceed to define concrete classes of "good" solutions, we first define the notions of certain, possible and ambiguous links. These notions can be defined, generally, w.r.t. an arbitrary *class* of solutions, that is, w.r.t. a subset C of solutions that satisfy some property. We may also refer to the solutions in a class C as C-solutions.

▶ **Definition 3.** Assume a class C of solutions and an entity linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$. Then, given a source instance I:

(i) The set of *certain links* for I w.r.t. C and $\mathcal{E}$ is the set of links that appear in every C-solution J for I w.r.t. $\mathcal{E}$.

(ii) The set of *possible links* for I w.r.t. C and $\mathcal{E}$ is the set of links that appear in some C-solution J for I w.r.t. $\mathcal{E}$.

(iii) The set of *ambiguous* links for I w.r.t. C and $\mathcal{E}$ is given by the set difference between the possible and the certain links for I w.r.t. $\mathcal{E}$.

3 A Naive Semantics Based on Maximal Solutions

The first class of "good" solutions that we investigate is the class of maximal solutions, where "goodness" of a solution is defined as maximality w.r.t. set containment.

▶ **Definition 4.** Assume an entity linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$. Given a source instance I, a *maximal solution* for I w.r.t. $\mathcal{E}$ is a link instance J such that: (1) (J, I) satisfies Σ, and (2) there is no J' such that $J \subset J'$ and (J', I) satisfies Σ.

▶ **Example 5.** We revisit Example 2. The solutions J_1, J_2, J_3, J_4 are maximal for the given source instance I (and w.r.t. the given $\mathcal{E}$), because in each of the four instances, we cannot add any further links over the *sid*- and *cid*-values in I without violating the functional dependency. It can also be verified that these four instances are all the maximal solutions for I. ◀

Maximal solutions vs. repairs. We next show a connection with source-to-link constraints and the framework of repairs [4], which we shall use later in this section. Given an entity linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ in $\mathcal{L}_\mathbf{0}$, we first extract a source-to-link specification $\mathcal{M} = (\mathbf{S}, \mathbf{L}, \Sigma')$ as follows. For each matching constraint m_L in Σ, and given the inclusion dependencies $L[X] \subseteq R[A]$ and $L[Y] \subseteq R'[A']$, we add the following source-to-link constraint in Σ':

$$(m'_L) \quad R(\ldots, x, \ldots) \wedge R'(\ldots, y, \ldots) \wedge \ (\forall \mathbf{u}(\psi(x, y, \mathbf{u}) \rightarrow \alpha_1 \vee \ldots \vee \alpha_k)) \ \rightarrow L(x, y)$$

In the above, the occurrence of x in the R-atom is in the position of attribute A and, similarly, the occurrence of y in the R'-atom is in the position of attribute A'. Intuitively, the formula m'_L inverts the direction of the implication in m_L. For every pair x, y of values, with x coming from $R[A]$ and y coming from $R'[A']$, we check that the left-hand side of m'_L is satisfied in the source. If that is the case then m'_L requires the addition of an appropriate link in L. We can formally define this process of adding links by using the chase as follows.

First, we note that $\mathcal{M}$ can be seen as a schema mapping or data exchange setting [12] where the link schema plays the role of a target schema. The constraints in Σ' are a particular case of first-order tgds [3], that is, source-to-target tgds where the left-hand side of the tgd can contain an arbitrary first-order formula (rather than just a conjunction of atomic formulas). As shown in [3], the chase with first-order tgds behaves in the same way as the chase with regular source-to-target tgds. In particular, it terminates in polynomial time in the size of the source instance. Furthermore, since there are no existentially quantified variables in L, each m'_L is a *full* tgd; hence, the chase produces no nulls and its result for a given source instance I is unique.

Let us denote the result of the chase by $U = chase_\mathcal{M}(I)$. Intuitively, U contains all the links that are possible based on just the matching constraints and inclusion dependencies. However, when we consider the additional functional dependencies in Σ, not all the links in U are possible due to conflicts. Instead we must consider subinstances of U that are consistent. The maximal subinstances of U that are consistent are also known as the *subset repairs* [4] of U.

As it turns out, the subset repairs of $U = chase_{\mathcal{M}}(I)$ w.r.t. the functional dependencies are precisely the maximal solutions w.r.t. the original entity linking specification.

▶ **Proposition 6.** *Assume an entity linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ in $\mathcal{L}_\mathbf{0}$, and let $\mathcal{M} = (\mathbf{S}, \mathbf{L}, \Sigma')$ be the source-to-link specification constructed from $\mathcal{E}$. Furthermore, let F be the set of functional dependencies in Σ. Then, for every source instance I, the set of maximal solutions for I w.r.t. $\mathcal{E}$ is the same as the set of subset repairs of $U = chase_{\mathcal{M}}(I)$ w.r.t. F.*

Based on the previous proposition and known results about the consistent answers of projection-free queries [8], we immediately obtain the following tractability results.

▶ **Theorem 7.** *Let $\mathcal{E}$ be an entity linking specification in $\mathcal{L}_\mathbf{0}$. Then:*

- *There is a polynomial-delay algorithm that, given a source instance I, enumerates all the maximal solutions for I w.r.t. $\mathcal{E}$.*
- *There is a polynomial-time algorithm that, given a source instance I, computes the set of certain links for I w.r.t. the class of maximal solutions and $\mathcal{E}$.*
- *There is a polynomial-time algorithm that, given a source instance I, computes the set of ambiguous links for I w.r.t. the class of maximal solutions and $\mathcal{E}$.*

Proposition 6 provides a useful connection between an entirely declarative specification, based on maximal solutions w.r.t. $\mathcal{E}$, and a more procedural approach, based on chasing with $\mathcal{M}$ and then applying repairs. It also gives us polynomial-time algorithms for the three problems of interest. While this connection with repairs is directly applicable for $\mathcal{L}_\mathbf{0}$ and the semantics of maximal solutions, the situation becomes more complex for the more refined semantics that we consider later, where we will need to employ graph-based techniques to handle link values.

Deficiency of maximal solutions. We now point out the main deficiency of the semantics based on maximal solutions: in general, there may be too many maximal solutions and, hence, too few certain links. Intuitively, the semantics given by maximal solutions is too coarse-grained and does not have enough discriminating power to identify the "good" links. Consider our scenario in Example 2. We showed that there are four maximal solutions, J_1, J_2, J_3, and J_4, for the given source instance I. It can be easily seen that the set of certain links in this example is empty: there is no link that appears in all four maximal solutions and, hence, no link qualifies as a certain link. On the flip side, every link that occurs in one of the four maximal solutions is possible (and ambiguous). However, some links are clearly stronger than others. In particular, the link $L(s_1, c_1)$ relating "Citibank N.A." to "Citigroup Inc." satisfies both the $\sim$ predicate and the `Exec`-based matching condition, while the other links satisfy only the $\sim$ predicate. Intuitively, there is evidence that suggests that $L(s_1, c_1)$ is a strong link that should be differentiated from the other links.

To address the above issue, we next refine the class of "good" solutions by assigning value to links, which in turn will increase the power of discriminating among the links. In particular, it will allow to increase the number of links that qualify as certain, thus reducing ambiguity.

4 Maximum-Value Solutions

We now consider a variation of the core language $\mathcal{L}_\mathbf{0}$ that allows us to differentiate among the links in a solution, based on the evidence supporting each link. More precisely, for each link fact $L(a, b)$ in a solution, we count the number of disjuncts that are satisfied among

all the possible disjuncts $\alpha_1, \ldots, \alpha_k$ in the matching constraint m_L. We also count, for each satisfied disjunct $\alpha_i = \exists \mathbf{z}\ \phi_i$, the number of different instantiations of the existentially quantified variables $\mathbf{z}$ that witness the satisfaction of ϕ_i. Intuitively, the larger these numbers are, the better the links are.

While syntactically similar to $\mathcal{L}_\mathbf{0}$, the new language will be semantically different due to the presence of counting. In particular, in the new language, one cannot drop disjuncts that are logically redundant, since such disjuncts may be important for measuring the strength of the links, so dropping them would change the semantics. To make this behavior explicit, in the notation for a matching constraint m_L, we replace $\vee$ with a new symbol $\oplus$ as follows:

$$(m_L) \quad L(x,y) \rightarrow \forall \mathbf{u}(\psi(x,y,\mathbf{u}) \rightarrow \alpha_1 \oplus \ldots \oplus \alpha_k).$$

Syntactically, everything else is the same as in $\mathcal{L}_\mathbf{0}$. We call the resulting language $\mathcal{L}_\mathbf{0}(\oplus)$.

While the notion of a solution is the same as for $\mathcal{L}_\mathbf{0}$ and continues to be based on logical satisfaction (where $\oplus$ is interpreted as $\vee$), the notion of a "good" solution in $\mathcal{L}_\mathbf{0}(\oplus)$ will now change to reflect the strength or the value of the links. Concretely, we will identify, among the maximal solutions, a subclass of solutions that additionally maximize the total value of the links.

Assume an entity linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ in $\mathcal{L}_\mathbf{0}(\oplus)$. Let I be a source instance and J be a solution for I w.r.t. $\mathcal{E}$. We define the value of a link $L(a,b)$ in the solution J as follows.

First, if there is no matching constraint m_L for L, we take the value of $L(a,b)$ to be 1. This is the case when there are no direct requirements on the link, other than the inclusion and functional dependencies (if any). Furthermore, the link is consistent with other links in the given solution (since it appears in the solution). Giving it a value of 1 (as opposed to 0, for example) ensures that the total value of a solution strictly increases with an increase in the number of links. Assume now that there is a matching constraint m_L for L. Since (J, I) satisfies m_L, it must be that I satisfies the right-hand side of m_L where x and y are instantiated with a and b. Assume first that there is no instantiation $\mathbf{u_0}$ of the vector of universally quantified variables $\mathbf{u}$ such that $I \models \psi(a,b,\mathbf{u}_0)$. This means that the matching constraint for $L(a,b)$ is satisfied for vacuous reasons. For the same reasons as above (in the case of no matching constraint), we take the value of the link to also be 1. In all other cases, we let the value of the link be:

$$\mathrm{Val}(L(a,b)) = \min_{\mathbf{u}_0}\Big(\sum_{\alpha_i, \mathbf{z}_0} 1\Big). \tag{1}$$

In the above, $\mathbf{u}_0$ ranges over all the distinct instantiations of the vector of universally quantified variables $\mathbf{u}$ such that $I \models \psi(a,b,\mathbf{u}_0)$. We take the minimum, over all such $\mathbf{u}_0$, of the *strength* with which the source instance I satisfies the disjunction $\alpha_1 \vee \ldots \vee \alpha_k$. This strength is defined as a sum that gives a value of 1 for *every* disjunct α_i such that I satisfies $\alpha_i(a,b,\mathbf{u}_0)$ and, moreover, for *every* distinct instantiation $\mathbf{z}_0$ of the vector $\mathbf{z}$ of existentially quantified variables of α_i that makes this satisfaction hold. (Recall that α_i is, in general, of the form $\exists \mathbf{z}\ \phi_i(x,y,\mathbf{u},\mathbf{z})$.) In the case when the existentially quantified variables are missing, then we count only 1 per disjunct.

Intuitively, the sum in formula (1) calculates the matching strength by counting the number of satisfied disjuncts together with the evidence (i.e., the number of existential witnesses), while the minimum guarantees that we take the weakest matching strength among all $\mathbf{u}_0$.

The value of a solution J, denoted by $\mathrm{Val}(J)$, is then the sum of the values of the links in J.

▶ **Definition 8.** Assume an entity linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ in $\mathcal{L}_\mathbf{0}(\oplus)$. Given a source instance I, a *maximum-value solution* for I w.r.t. $\mathcal{E}$ is a link instance J such that: (1) (J, I) satisfies Σ, and (2) for every J' such that (J', I) satisfies Σ, we have that $\mathrm{Val}(J') \leq \mathrm{Val}(J)$.

▶ **Example 9.** Recall Example 2. By applying formula (1), the values of the individual links that can be formed between subsidiary ids and company ids, based on the matching constraint, are:

$$\mathrm{Val}(\mathtt{L}(s_1, c_1)) = 2 \qquad \mathrm{Val}(\mathtt{L}(s_1, c_2)) = \mathrm{Val}(\mathtt{L}(s_2, c_1)) = \mathrm{Val}(\mathtt{L}(s_2, c_2)) = 1$$

The value of 2, for the link $\mathtt{L}(s_1, c_1)$, is obtained as follows. First, for the given s_1 and c_1, there is only one way to instantiate the universally quantified variables sn, loc, cn, and hd in the matching constraint. (This is because there is only one tuple for s_1 in `Subsid`, and one tuple for c_1 in `Company`.) Hence, the min in the formula (1) is applied over a single element. Then, it can be seen that both disjuncts in the matching constraint are satisfied for s_1 and c_1. The first disjunct contributes a value of 1, since the disjunct is simply the atomic formula $sn \sim cn$. The second disjunct also contributes a value of 1, since there is only one way to instantiate the existential variables in the `Exec`-based condition (with the values corresponding for "E. McQuade"). Thus, the total strength with which the disjuncts are satisfied is 2 and, hence, the value of the link is 2. A similar evaluation takes place for the other three links, with the difference that only the first disjunct is satisfied.

Consider the earlier solutions J_1, J_2, J_3, and J_4, which were shown to be the maximal solutions. By summing up the values of their links, we obtain that $\mathrm{Val}(J_1) = \mathrm{Val}(J_2) = 3$, while $\mathrm{Val}(J_3) = \mathrm{Val}(J_4) = 2$. So, J_1 and J_2 are maximum-value solutions, while J_3 and J_4 are not. It can also be seen that there is now one certain link, namely $\mathtt{L}(s_1, c_1)$, which appears in both J_1 and J_2 and correctly relates "Citibank N.A." with "Citigroup Inc". This in contrast with the case of the maximal solutions semantics where we had zero certain links. Also, the two links $\mathtt{L}(s_2, c_1)$ and $\mathtt{L}(s_2, c_2)$, relating "CIT Bank" with either "Citigroup Inc." or "CIT Group Inc." are now ambiguous, whereas in the case of the maximal solutions semantics all four links were ambiguous. Finally, the ambiguity of $\mathtt{L}(s_2, c_1)$ and $\mathtt{L}(s_2, c_2)$ is, intuitively, the best we can achieve here, since there is not enough information to differentiate between the two links, based on the given specification. A human user is needed at this point to further refine the entity linking specification, possibly by using additional information (e.g., additional attributes or relations). ◀

A simple but important observation for $\mathcal{L}_\mathbf{0}(\oplus)$ is that, even though $\mathrm{Val}(L(a, b))$ was defined relative to a solution J (in which $L(a, b)$ occurs), the actual value of $\mathrm{Val}(L(a, b))$ is independent of J. This is so because, in $\mathcal{L}_\mathbf{0}(\oplus)$, the formula ψ and the disjuncts α_1, …, α_k are over the source schema. In Section 6, we will consider richer languages, where the α's can also depend on link predicates. Even though the same definitions of value and maximum-value solutions continue to apply for the richer languages, there we will have that $\mathrm{Val}(L(a, b))$ depends, in general, on the choice of the solution J in which it occurs.

▶ **Proposition 10.** *If $\mathcal{E}$ is an entity linking specification in $\mathcal{L}_\mathbf{0}(\oplus)$ and I is a source instance, then every maximum-value solution for I w.r.t. $\mathcal{E}$ is also a maximal solution for I w.r.t. $\mathcal{E}$.*

The proposition is an immediate consequence of the fact that in $\mathcal{L}_\mathbf{0}(\oplus)$, the value of a link is independent of the solution in which it occurs, and is at least one. The reverse inclusion does not hold, as seen in Example 9. Thus, for $\mathcal{L}_\mathbf{0}(\oplus)$, maximum-value solutions form a strict subclass of maximal solutions. As a consequence, the set of certain links over maximum-value solutions is often a strict superset of the certain links over maximal solutions.

We give next the main complexity result of this section, stating the tractability of $\mathcal{L}_{\mathbf{0}}(\oplus)$. In contrast with Theorem 7, which follows from results on data repairs, the proof of the following theorem is of a different nature and makes use of maximum-weight matching type of algorithms. In particular, it is based on extensions of results from [10, 16, 27].

▶ **Theorem 11.** *Let $\mathcal{E}$ be an entity linking specification in $\mathcal{L}_{\mathbf{0}}(\oplus)$. Then:*

- *There is a polynomial-delay algorithm that, given a source instance I, enumerates all the maximum-value solutions for I w.r.t. $\mathcal{E}$.*
- *There is a polynomial-time algorithm that, given a source instance I, computes the certain links for I w.r.t. the class of maximum-value solutions and $\mathcal{E}$.*
- *There is a polynomial-time algorithm that, given a source instance I, computes the ambiguous links for I w.r.t. the class of maximum-value solutions and $\mathcal{E}$.*

5 Connection to Probabilistic Approaches

In this section, we investigate the relationship between our declarative framework based on disjunctive matching constraints and existing probabilistic methods for entity resolution.

We start by introducing a simple yet powerful extension of $\mathcal{L}_{\mathbf{0}}(\oplus)$ that incorporates weights and which we call $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$. For each matching constraint

$$(m_L) \quad L(x, y) \rightarrow \forall \mathbf{u}(\psi(x, y, \mathbf{u}) \rightarrow \alpha_1 \oplus \ldots \oplus \alpha_k),$$

and for each disjunct $\alpha_i ::= \exists \mathbf{z}\ \phi_i(x, y, \mathbf{u}, \mathbf{z})$ there is now a weight function $w_{\phi_i}(x, y, \mathbf{u}, \mathbf{z})$ that returns a number. Intuitively, with each disjunct that returns true or false we also have a function that computes a weight (or a score) for that disjunct. The semantics of $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ is the same as that of $\mathcal{L}_{\mathbf{0}}(\oplus)$ except that when counting existential witnesses for each disjunct we also multiply by the number returned by the weight function for that disjunct. Theorem 11 goes through when we replace $\mathcal{L}_{\mathbf{0}}(\oplus)$ by $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$, by the same proof.

5.1 Comparison to Probabilistic Matching

The first connection we make is to a well-known class of probabilistic matching algorithms that has originated with Fellegi and Sunter [15] and is at the core of many commercial systems including IBM's QualityStage [23], which we use as a representative example.

The probabilistic matching algorithm in QualityStage approaches record matching in three steps. First, it applies pairwise comparison functions over the individual attributes (or fields) in the two records to be compared. For each pair of attributes, the function returns a score based on two probabilities (that must be learned or given to the system a priori): the "match" probability m, which is the probability that two fields match given that it is known that the two records match, and the "unmatch" (or accidental match) probability u, which is the probability that two fields match but the records do not match. Secondly, the algorithm aggregates the scores returned by individual comparison functions by taking a weighted sum, where each comparison function has its own weight (also to be learned or given to the system a priori). Finally, a link is returned if it has high-enough aggregated score (higher than a threshold, which also must be learned or tuned).

We show that the first two steps in the above algorithm can be captured by a single disjunctive matching constraint, while the third one can be captured as an implementation step. We use a canonical example for deduplication of mailing lists.

▶ **Example 12.** The source schema $\mathbf{S}$ consists of two relation symbols: `MasterList`, representing a master list of mailing addresses, and `NewList`, a list with new mailing addresses

that must be deduplicated against the first one. Both relations are assumed to have the same schema, including personal attributes (e.g., last name `ln`, first name `fn`, etc.) and also address attributes (e.g., street name `street`, etc.). Furthermore, we assume each record has been assigned a record id (`rid`) and the deduplication problem is one of linking an `rid` from the new list to a unique `rid` in the master list. The core functionality of a QualityStage algorithm can be logically expressed by an entity linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ in $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$, where Σ consists of a single matching constraint:

$$\begin{aligned}
&\mathtt{L}(rid_1, rid_2) \rightarrow \\
&\qquad \forall ln_1, fn_1, \ldots, street_1, ln_2, fn_2, \ldots, street_2 \\
&\qquad (\mathtt{NewList}(rid_1, ln_1, fn_1, street_1, \ldots) \wedge \mathtt{MasterList}(rid_2, ln_2, fn_2, street_2, \ldots) \\
&\qquad\quad \rightarrow \mathtt{SOUNDEX}(ln_1, ln_2) \oplus \mathtt{SOUNDEX}(fn_1, fn_2) \oplus \ldots \oplus \mathtt{EDIT}(street_1, street_2)),
\end{aligned}$$

along with the obvious inclusion dependencies. Although QualityStage does not have cardinality constraints, it is natural to add to our specification the functional dependency $\mathtt{L} : \mathtt{rid}_1 \rightarrow \mathtt{rid}_2$. In the above matching constraint, each disjunct calls a QualityStage built-in comparison function (e.g., `SOUNDEX`, which compares how similar two names sound, or edit distance `EDIT`), by passing the arguments to be compared. In turn, each call to a QualityStage comparison function returns a weight that depends on the given arguments and also on the aforementioned probabilities m and u for the particular attribute. The weight of each possible link is then the sum of the weights for all the disjuncts. Maximum-value solutions are obtained as solutions (containing non-conflicting links) that maximize the total value. (QualityStage has the additional requirement that only links whose weights are above a certain threshold are considered possible. This can be easily added in an implementation on top of our maximum-value semantics.)

We note that the above matching constraint uses only a fragment of $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ where each disjunct is a simple atomic formula with no existential quantification and no conjunction. ◀

5.2 Comparison to Markov Logic Networks for Entity Resolution

We now connect to a richer probabilistic framework, that of Markov Logic Networks (MLNs) [29], which in general allows for arbitrary first-order formulas to be interpreted in a probabilistic sense. We show that a class of MLNs that is useful for entity resolution [30] is captured, in a precise sense, by the fragment of $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ with no universal quantification. Thus, rather surprisingly, a purely probabilistic approach (based on MLNs) can be captured in a deterministic way (via $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$). We make use of this correspondence to obtain a polynomial-delay algorithm for enumerating the maximum-probability worlds in the MLN setting, and polynomial-time algorithms for finding the certain and ambiguous links over maximum-probability worlds. These are the first polynomial-time results, to the best of our knowledge, for MLN-based entity resolution.

5.2.1 Markov Logic Networks: Preliminaries

The fragment of MLNs that we consider is defined as follows. For simplicity of discussion, we assume that there is one single link symbol L; the same definitions extend immediately to the case of multiple link symbols in the schema. A *linear MLN* $\mathcal{M}$ is a set of formulas $\sigma_i \rightarrow L(x, y)$ (for $1 \leq i \leq n$), each with a weight w_i, where σ_i is a conjunction of atomic formulas over the source, and where the free variables of σ_i include x and y. Examples of linear MLN formulas for entity resolution appear in [30], with the provision that the role of the link relation is played there by the `Equals` predicate. Later we also consider

extensions of linear MLNs where a link symbol may also appear in the left-hand side, thus allowing for inter-dependencies among the links. We assume that the same requirements we have for the presence of inclusion dependencies involving the link relation L in our entity linking specifications are also required in the MLN setup; also, as with our entity linking specifications, there may be functional dependencies on L.

Note that the formulas $\sigma_i \to L(x, y)$ in linear MLNs are source-to-link constraints; thus, they fall in the category of rules that eagerly populate the link relations. As discussed earlier in Section 2, such specifications may not have solutions. However, in the MLN framework, these formulas are not required to be satisfied in a hard logical sense but rather in a probabilistic sense, which allows for violations and which we explain next.

Fix a source instance I. For each "possible world", that is, choice of link instance L_0 for L satisfying the inclusion dependencies w.r.t. I and the functional dependencies on L, we assign a probability to that possible world, based on the source-to-link formulas in $\mathcal{M}$ and their weights, as follows. Let K be an instance over the combined source and link schema, and let γ be a formula over the combined schema. A *(K, γ)-valuation* (or simply *valuation*, if K and γ are fixed or understood) is a function v from the free variables of γ to members of the domain of K. Denote by $|\gamma|$ the number of valuations that make γ true in K. Then the probability assigned to a link instance L_0 (for a given source instance I) is proportional (see also [30]) to $e^{w_1|\sigma_1 \to L| + \cdots + w_n|\sigma_n \to L|}$, where the role of L is played by L_0. (These probabilities are scaled so that they sum up to 1, over all choices for L_0.) Intuitively, the probability of a world increases with the number of valuations that make a formula in $\mathcal{M}$ true and also with the weight of the formula.

Define a link (a tuple over the L schema) to be *certain* (w.r.t. the class of maximum-probability worlds) if it is in every maximum-probability world w.r.t. $\mathcal{M}$ (for the given source instance I). Similarly, we define *ambiguous* links.

5.2.2 Translation to $\mathcal{L}_0(\oplus, \mathbf{w})$

Given the linear MLN with formulas $\sigma_i \to L(x, y)$ with weight w_i, for $1 \leq i \leq n$, we define the *corresponding entity linking specification in* $\mathcal{L}_0(\oplus, \mathbf{w})$ to consist of the matching constraint $L(x, y) \to \exists \mathbf{z_1} \sigma_1 \oplus \cdots \oplus \exists \mathbf{z_n} \sigma_n$, where $\mathbf{z_i}$ consists of the free variables of σ_i other than x and y, and where the disjunct $\exists \mathbf{z_i} \sigma_i$ has weight w_i, for $1 \leq i \leq n$. Also, this corresponding entity linking specification has the same inclusion dependencies and functional dependencies on L as the linear MLN. Note that the weight function for each disjunct σ_i is a constant, whereas for QualityStage we needed in general nonconstant weight functions for each disjunct in a matching constraint.

Let $\mathcal{M}$ be an MLN, and let $\mathcal{E}$ be the corresponding entity linking specification in $\mathcal{L}_0(\oplus, \mathbf{w})$. For a given source instance I, let us denote the set of maximum-probability worlds w.r.t. $\mathcal{M}$ by Max Probability Worlds$_{\mathcal{M}}(I)$, the set of solutions w.r.t. $\mathcal{E}$ by Solutions$_{\mathcal{E}}(I)$, and the set of maximum-value solutions w.r.t. $\mathcal{E}$ by Max Value Solutions$_{\mathcal{E}}(I)$.We have the following result, interrelating maximum-value solutions and maximum-probability worlds.

▶ **Theorem 13.** *Let $\mathcal{M}$ be a linear MLN, let $\mathcal{E}$ be the corresponding entity linking specification in $\mathcal{L}_0(\oplus, \mathbf{w})$, and let I be a source instance. Then:*

- *Max Value Solutions$_{\mathcal{E}}(I)$ = Max Probability Worlds$_{\mathcal{M}}(I) \cap$ Solutions$_{\mathcal{E}}(I)$.*
- *The certain links for I w.r.t. the class of maximum-probability worlds and $\mathcal{M}$ are precisely the certain links for I w.r.t. the class of maximum-value solutions and $\mathcal{E}$.*

Note that the second part of the theorem holds even though the sets of maximum-value solutions and of maximum-probability worlds do not coincide.

The proof of the second part is based on a characterization of maximum-probability worlds in terms of maximum-value solutions. (The details of this characterization will appear in the full version of the paper.) We also use that characterization to prove the analog of Theorem 11 for linear MLNs, that is, that for linear MLNs, there is a polynomial-delay algorithm for enumerating the maximum-probability worlds, and polynomial-time algorithms for finding the certain and ambiguous links.

5.3 Deterministic vs. Probabilistic: Discussion

We showed that our declarative language can capture important classes of probabilistic methods, under a suitable extension that allows for weights. While this translation is interesting in itself, we envision our language to be used as a deterministic framework (i.e., no probabilities) where the rules that govern the links are written out explicitly, by a domain expert, as semantic rules with a true/false interpretation and (primarily) with no weights (other than 1) in the disjuncts.

While probabilistic methods may provide more automation via learning algorithms to tune or learn weights, probabilities, and thresholds, these methods are also more opaque in that it is hard to explain the results other than in terms of scores or probabilities in the underlying model. These methods may also be hard to customize when the results are not satisfactory. A simple change in a parameter or a threshold may often have unintended consequences. In contrast, a high-level deterministic language (such as $\mathcal{L}_{\mathbf{0}}(\oplus)$) provides a more transparent way for linking entities where the results can be explained in terms of the rules (disjuncts) that are satisfied. When the results are not satisfactory, rather than changing some numbers, a domain expert can explicitly refine the entity linking logic by adding or removing disjuncts, or by adding, removing or changing a conjunct within a particular disjunct. We note that similar observations were made in the context of information extraction (IE) systems [7], where it is observed that rule-based IE systems are the dominant systems adopted by commercial companies for similar reasons (i.e., they are declarative, and easier to understand, to explain, and to incorporate domain knowledge).

6 More Expressive Languages

We now explore extensions of the core language $\mathcal{L}_{\mathbf{0}}$, to allow a matching constraint for a link to possibly refer to other links. These extensions allow us to express what is usually called *collective entity resolution* [6], that is, the process of creating multiple types of links *together*.

The matching constraint for a link symbol L has the same form m_L as in Section 2. However, in each disjunct $\alpha_i ::= \exists \mathbf{z}\ \phi_i(x, y, \mathbf{u}, \mathbf{z})$, the formula ϕ_i can now be a conjunction of source *and link* atomic formulas, along with equalities and other built-in or user-defined boolean predicates. If a specification is not allowed to have recursion among the link predicates, we call the resulting language $\mathcal{L}_{\mathbf{1}}$. Thus, in $\mathcal{L}_{\mathbf{1}}$, there is a hierarchy of links, where a matching constraint for a link L may call only links that are strictly lower in the hierarchy than L. When recursion is allowed, we call the language $\mathcal{L}_{\mathbf{2}}$. So $\mathcal{L}_{\mathbf{1}}$ is a sublanguage of $\mathcal{L}_{\mathbf{2}}$. The corresponding variations for maximum-value solutions, $\mathcal{L}_{\mathbf{1}}(\oplus)$ and $\mathcal{L}_{\mathbf{2}}(\oplus)$, are defined as in the case of $\mathcal{L}_{\mathbf{0}}$. We also consider the corresponding weighted versions $\mathcal{L}_{\mathbf{1}}(\oplus, \mathbf{w})$ and $\mathcal{L}_{\mathbf{2}}(\oplus, \mathbf{w})$.

▶ **Example 14.** Consider a bibliographic example where we link papers (from one database) with articles (from another database), while also linking the corresponding venues. The source schema $\mathbf{S}$ consists of $\texttt{Paper}(pid, title, venue, year)$ and $\texttt{Article}(ano, title, journal, year)$.

Here, `pid` is a unique id assigned to `Paper` records, while `venue` could be a conference, a journal, or some other place of publication. The `Article` relation represents publications that appeared in journals, and `ano` is a unique id assigned to such records. The link schema $\mathbf{L}$ consists of two relations: `PaperLink` (pid, ano) and `VenueLink` $(venue, journal)$. The first relation is intended to link paper ids from `Paper` with article numbers from `Article`, when they represent the same publication. The second relation is intended to relate `journal` values that occur in `Article` (e.g., "ACM TODS") to journal values that occur under the `venue` field in `Paper` (e.g., "TODS").

A possible entity linking specification in $\mathcal{L}_\mathbf{2}$ is $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$, where Σ contains:

$$\begin{aligned}\texttt{VenueLink}(ven, jou) \rightarrow\ & (ven \sim_1 jou) \\ \vee\ & \exists pid, t_1, y_1, ano, t_2, y_2\ (\ \texttt{Paper}(pid, t_1, ven, y_1) \\ & \qquad \wedge\ \texttt{Article}(ano, t_2, jou, y_2) \\ & \qquad \wedge\ \texttt{PaperLink}(pid, ano)\ )\end{aligned}$$

$$\begin{aligned}&\texttt{PaperLink}(pid, ano) \rightarrow \\ &\quad \forall t_1, ven, y_1, t_2, jou, y_2\ (\ \texttt{Paper}(pid, t_1, ven, y_1) \wedge \texttt{Article}(ano, t_2, jou, y_2) \\ &\qquad\qquad \rightarrow ((t_1 \sim_2 t_2) \wedge (y_1 = y_2)) \vee ((t_1 \sim_2 t_2) \wedge \texttt{VenueLink}(ven, jou))\ )\end{aligned}$$

The first constraint specifies that we may link a venue with a journal only if their string values are similar (via some similarity predicate $\sim_1$), or if there are papers and articles that have been published in the respective venue and journal and that are linked via `PaperLink`. The second constraint specifies that we may link a paper with an article only if their titles are similar (via a similarity predicate $\sim_2$) and their years of publication match exactly, or if their titles are similar and their venues of publications are linked via `VenueLink`.

Additionally, Σ includes two functional dependencies on `PaperLink`: $pid \rightarrow ano$, $ano \rightarrow pid$, to reflect that each paper id in `Paper` must match to at most one article number in `Article`, and vice-versa. We do not require any functional dependencies on `VenueLink`; thus, we could have multiple venue strings in `Paper` matching with a journal string in `Article`, and vice-versa. We also include in Σ the expected inclusion dependencies from the link attributes to the corresponding source attributes (e.g., $\texttt{PaperLink}[pid] \subseteq \texttt{Paper}[pid]$).

With a simple modification, where we remove the second disjunct in the matching constraint for `PaperLink`, we obtain a different entity linking specification that is in $\mathcal{L}_\mathbf{1}$. While the advantage of such specification is that it is non-recursive, the modified specification is more constrained: the matching conditions for `PaperLink` are stricter now (whereas before we had a disjunction of conditions). As a result, there will be less possible links for the modified specification. ◀

6.1 Results for $\mathcal{L}_1$ and $\mathcal{L}_2$

We now focus on the computational complexity of the relevant problems (computing/enumerating maximum-value solutions and computing certain and ambiguous links). We show that we hit intractability in general, even in the case of $\mathcal{L}_\mathbf{1}$, the non-recursive fragment of $\mathcal{L}_\mathbf{2}$. On the other hand, we show that there is a large syntactic fragment of $\mathcal{L}_\mathbf{1}$ that is tractable. Finally, we show that the correspondence between $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$ and MLNs breaks when we go to the richer $\mathcal{L}_\mathbf{1}(\oplus, \mathbf{w})$.

Our first result, for $\mathcal{L}_\mathbf{1}(\oplus)$, states the NP-completeness of determining whether there exists a solution of at least a given value. In turn, this implies that there is no polynomial-time algorithm to compute one maximum-value solution (unless P = NP). Hence, there is no

polynomial-delay algorithm for the problem of enumerating maximum-value solutions (again, unless P = NP).

▶ **Theorem 15.** *There is a fixed entity linking specification $\mathcal{E}$ in $\mathcal{L}_\mathbf{1}(\oplus)$ for which the following problem is* NP*-complete: Given source instance I and positive integer k, is there a solution for I w.r.t. $\mathcal{E}$ of value at least k?*

We now turn our attention to the problems of computing certain and ambiguous links. If C is a class of solutions and $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ is a fixed entity linking specification, then *recognizing certain links w.r.t. C and $\mathcal{E}$* is the following decision problem: given a source instance I and a link l, is l a certain link for I w.r.t. C and $\mathcal{E}$? The problem of *recognizing ambiguous links w.r.t. C and $\mathcal{E}$* is defined in a similar way. Here, we investigate the complexity of recognizing certain and ambiguous links for the class of all maximum-value solutions for entity linking specifications in $\mathcal{L}_\mathbf{1}(\oplus)$. The main result is that there is an entity linking specification in $\mathcal{L}_\mathbf{1}(\oplus)$ for which no polynomial-time algorithms for recognizing certain and ambiguous links exist, unless NP = coNP.

By Theorem 15, there is an entity linking specification $\mathcal{E}$ in $\mathcal{L}_\mathbf{1}(\oplus)$ such that the following problem is NP-complete: given a source instance I and a positive integer k, is there a solution for I of value at least k? For that particular specification, recognizing certain links and recognizing ambiguous links are trivial problems because no link is certain and every link is ambiguous; intuitively, this is so because $\mathcal{E}$ encodes 3-Colorability, a problem that has "symmetries".

To establish the intractability of recognizing certain and ambiguous links, we bring into the picture the concept of a *frozen variable* from constraint satisfaction. An instance of the constraint satisfaction problem consists of a set of variables, a domain of values for each variable, and a set of constraints that restrict the combinations of values that some tuples of variables may take. A *solution* to such an instance is an assignment of values to variables so that all constraints are satisfied. A variable is *frozen* if it takes the same value in all solutions of a given instance. Jonsson and Krokhin [25] showed that for every constraint satisfaction problem over a two-element domain the problem of recognizing frozen variables exhibits the following trichotomy: it is in PTIME or it is coNP-complete or it is DP-complete. Recall that DP is the class of all decision problems that can be written as the conjunction of a problem in NP and a problem in coNP; in particular, both NP and coNP are subclasses of DP (see also [28]). Constraint satisfaction problems over a two-element domain can be thought of as variants of boolean satisfiability. An important such NP-complete variant is Positive-1-in-3-SAT, which asks: given a positive 3CNF-formula φ (i.e., a 3CNF-formula in which each clause has the form $(x \vee y \vee z)$), is there a 1-in-3 satisfying truth assignment (i.e., a truth assignment that makes exactly one variable true in every clause of φ)? Theorem 6.1 in [25] implies that the following problem is DP-complete: given a positive 3CNF-formula φ and a variable x of φ, is it true that there is a 1-in-3 satisfying truth assignment for φ and the variable x is frozen? By exploiting the above result, we are able to establish the intractability of recognizing certain and ambiguous links for entity linking specifications in $\mathcal{L}_\mathbf{1}(\oplus)$.

▶ **Theorem 16.** *There is a fixed entity linking specification $\mathcal{E}$ in $\mathcal{L}_\mathbf{1}(\oplus)$ such that:*

- *Unless* NP = coNP*, there is no polynomial-time algorithm for recognizing certain links w.r.t. to the class of all maximum-value solutions and $\mathcal{E}$.*
- *Unless* NP = coNP*, there is no polynomial-time algorithm for recognizing ambiguous links w.r.t. to the class of all maximum-value solutions and $\mathcal{E}$.*

While the above complexity results for $\mathcal{L}_1$ show intractability in general, the next theorem gives special conditions under which the same problems become tractable for $\mathcal{L}_1(\oplus)$. However, if any of the conditions fails to be satisfied, then we fall back into intractability. We say that an entity linking specification is *2-level hierarchical* if the link relations each fall into one of two disjoint sets, the "top-level links" and the "bottom-level links". The right-hand side of the matching constraints for the bottom-level link relations can refer only to source relations and built-in predicates (like equality, similarity, and string containment). The right-hand side of the matching constraints for the top-level link relations can refer only to bottom-level link relations, source relations, and built-in predicates.

▶ **Theorem 17.** *Assume that the entity linking specification is 2-level hierarchical. Assume also the following three conditions.*

1. *The top-level links have no FDs.*
2. *There are no universal quantifiers in the matching constraints for the top-level links.*
3. *Each disjunct in each matching constraint for each top-level link refers to at most one bottom-level link relation.*

Then there is a polynomial-delay algorithm to enumerate the maximum-value solutions, and there are polynomial-time algorithms to compute the certain and the ambiguous links.

Furthermore, if any of the three assumptions (1), (2), or (3) is violated, then it may be NP*-complete even to decide the following: Given source instance I and positive integer k, is there a solution for I of value at least k?*

As an immediate application of the above theorem, recall the entity linking specification in $\mathcal{L}_2$ for `VenueLink` and `PaperLink` in Example 14, and the entity linking specification in $\mathcal{L}_1$ obtained from it by the modification described in the same Example 14. The entity linking specification in $\mathcal{L}_1$, where `VenueLink` is the top-level link, and `PaperLink` is the bottom-level link, satisfies the assumptions of Theorem 17, and so enjoys the desirable properties in the conclusions of (the positive part of) the theorem. Interestingly enough, it turns out that even the entity linking specification in $\mathcal{L}_2$ for this example enjoys the desirable properties.

We close this section by considering the weighted versions $\mathcal{L}_1(\oplus, \mathbf{w})$ and $\mathcal{L}_2(\oplus, \mathbf{w})$. Theorem 13 shows a precise correspondence between a linear MLN and its corresponding entity linking specification in $\mathcal{L}_0(\oplus, \mathbf{w})$. Does this correspondence carry over to $\mathcal{L}_1(\oplus, \mathbf{w})$ or $\mathcal{L}_2(\oplus, \mathbf{w})$? Let us define *extended linear MLNs* to be defined like linear MLNs, except that instead of taking σ_i to be a conjunction of atomic formulas over the source, we allow these atomic formulas to also involve another link relation. We then define the corresponding entity linking specification as before. It can be shown that the analog of Theorem 13 fails (the details will appear in the full version of the paper). Thus, the two frameworks, one based on deterministic entity linking specifications, the other based on probabilistic Markov Logic Networks, diverge when allowing for inter-dependencies among links.

7 Related Work

As mentioned in the introduction, there has been extensive earlier work on entity resolution; overviews can be found in the recent surveys [14] and [18] and the tutorial [19]. We have also made in-depth connections to existing probabilistic approaches for entity resolution. We now comment briefly on some other declarative approaches to entity resolution.

An early argument in favor of using link-centric constraints to specify links in a declarative manner appeared in [1], but no formal language, semantics or algorithms were given there. We already discussed Dedupalog [2], a high-level framework that enables collective entity

resolution through the use of Datalog-like constraints. The result of executing a Dedupalog program is an instance that satisfies the hard constraints of the program but may violate the soft constraints of the program. Thus, a Dedupalog program is only a guideline for the implementation, which is an algorithm that attempts to minimize the number of violations of soft constraints. In contrast, the constraints we use form a truly declarative specification, by stating only the necessary conditions that must be satisfied by links, thus decoupling the specification from any implementation.

The language LinQL [20] uses SQL-like syntax to define similarity predicates among string-valued attributes only. In contrast, we create links among structured entities, and the LinQL similarity functions could be used as one ingredient in our framework. Matching dependencies (MDs) were introduced in [13] to enforce equality on attribute values based on matching conditions. In effect, MDs are source-to-link constraints that may modify source relations. MDs have been given operational semantics in [5] via a variation of the chase procedure that fixes violations of a given set of MDs. Like Dedupalog, MDs look only at equivalence (same-as) type of linkage.

8 Concluding Remarks

We laid the foundation for a truly declarative entity-linking framework that is based on specifying only the desired properties of the links. We identified a class of maximum-value solutions for entity linking specifications, and studied the computational complexity of producing such solutions and identifying certain and ambiguous links. This work opens up several new directions in reasoning about entity linking specifications. These include studying the implication and equivalence of entity linking specifications (e.g., deciding when two such specifications have the same certain links), as well as delineating the expressive power of the languages we introduced. More broadly, this work may also provide a different perspective for linking heterogeneous entities in the Semantic Web.

Acknowledgements. Kolaitis is partially supported by NSF Grant IIS-1217869; Tan is partially supported by NSF grant IIS-1450560.

References

1 Bogdan Alexe, Douglas Burdick, Mauricio A. Hernández, Georgia Koutrika, Rajasekar Krishnamurthy, Lucian Popa, Ioana R. Stanoi, and Ryan Wisnesky. High-Level Rules for Integration and Analysis of Data: New Challenges. In *LNCS 8000: In Search of Elegance in the Theory and Practice of Computation*, pages 36–55, 2013.

2 A. Arasu, C. Re, and D. Suciu. Large-Scale Deduplication with Constraints using Dedupalog. In *ICDE*, pages 952–963, 2009.

3 M. Arenas, P. Barceló, R. Fagin, and L. Libkin. Solutions and Query Rewriting in Data Exchange. *Inf. Comp.*, pages 28–51, 2013.

4 Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent Query Answers in Inconsistent Databases. In *PODS*, pages 68–79, 1999.

5 Leopoldo E. Bertossi, Solmaz Kolahi, and Laks V. S. Lakshmanan. Data Cleaning and Query Answering with Matching Dependencies and Matching Functions. *Theory of Computing Systems*, 52(3):441–482, 2013.

6 Indrajit Bhattacharya and Lise Getoor. Collective Entity Resolution in Relational Data. *TKDD*, 1(1), 2007.

7 Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems! In *EMNLP*, pages 827–832, 2013.

8 Jan Chomicki and Jerzy Marcinkowski. Minimal-Change Integrity Maintenance using Tuple Deletions. *Inf. Comp.*, 197:90–121, 2005.

9 Xin Dong, Alon Y. Halevy, and Jayant Madhavan. Reference Reconciliation in Complex Information Spaces. In *SIGMOD*, pages 85–96, 2005.

10 J. Edmonds. Maximum Matching and a Polyhedron with 0,1-vertices. *Journal of Research National Bureau of Standards Section B*, 69:125–130, 1965.

11 Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE TKDE*, 19(1):1–16, 2007.

12 R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science (TCS)*, 336(1):89–124, 2005.

13 Wenfei Fan. Dependencies Revisited for Improving Data Quality. In *PODS*, pages 159–170, 2008.

14 Wenfei Fan and Floris Geerts. *Foundations of Data Quality Management*. Morgan & Claypool Publishers, 2012.

15 I. P. Fellegi and A. B. Sunter. A Theory for Record Linkage. *J. Am. Statistical Assoc.*, 64(328):1183–1210, 1969.

16 K. Fukuda and T. Matsui. Finding All the Perfect Matchings in Bipartite Graphs. *Appl. Math. Lett.*, 7(1):15–18, 1994.

17 H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Declarative Data Cleaning: Language, Model, and Algorithms. In *VLDB*, pages 371–380, 2001.

18 Venkatesh Ganti and Anish Das Sarma. *Data Cleaning: A Practical Perspective*. Morgan & Claypool Publishers, 2013.

19 Lise Getoor and Ashwin Machanavajjhala. Entity Resolution: Theory, Practice & Open Challenges. *PVLDB*, 5(12):2018–2019, 2012.

20 Oktie Hassanzadeh, Anastasios Kementsietsidis, Lipyeow Lim, Renée J. Miller, and Min Wang. A Framework for Semantic Link Discovery over Relational Data. In *CIKM*, pages 1027–1036, 2009.

21 Mauricio A. Hernández, Georgia Koutrika, Rajasekar Krishnamurthy, Lucian Popa, and Ryan Wisnesky. HIL: A High-Level Scripting Language for Entity Integration. In *EDBT*, pages 549–560, 2013.

22 Mauricio A. Hernández and Salvatore J. Stolfo. The Merge/Purge Problem for Large Databases. In *SIGMOD*, pages 127–138, 1995.

23 IBM InfoSphere QualityStage. `http://www.ibm.com/software/products/en/ibminfoqual`.

24 D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis. On Generating All Maximal Independent Sets. *Inf. Process. Lett.*, 27(3):119–123, 1988.

25 Peter Jonsson and Andrei A. Krokhin. Recognizing Frozen Variables in Constraint Satisfaction Problems. *Theoretical Computer Science (TCS)*, 329(1-3):93–113, 2004.

26 Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. Record Linkage: Similarity Measures and Algorithms. In *SIGMOD*, pages 802–803, 2006.

27 K.G. Murty. An Algorithm for Ranking All the Assignments in Order of Increasing Cost. *Operations Research*, 16(3):682–687, 1968.

28 C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

29 Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, 2006.

30 Parag Singla and Pedro Domingos. Entity Resolution with Markov Logic. In *ICDM*, pages 572–582, 2006.