A Declarative Framework for Linking Entities

DOUGLAS BURDICK and RONALD FAGIN, IBM Research – Almaden
PHOKION G. KOLAITIS, UC Santa Cruz & IBM Research – Almaden
LUCIAN POPA, IBM Research – Almaden
WANG-CHIEW TAN, UC Santa Cruz

We introduce and develop a declarative framework for entity linking and, in particular, for entity resolution. As in some earlier approaches, our framework is based on a 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 two entities should be linked. We also consider extensions of the core language that capture collective entity resolution by allowing interdependencies among the link relations.

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 interdependencies among the 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.

CCS Concepts: • **Information systems → Information integration; Deduplication; Entity resolution;**

Additional Key Words and Phrases: Entity linking, constraints, certain links, maximum-value solutions, Markov logic networks, maximum-probability worlds

ACM Reference Format:
Douglas Burdick, Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang-Chiew Tan. 2016. A declarative framework for linking entities. ACM Trans. Database Syst. 41, 3, Article 17 (June 2016), 38 pages.
DOI: http://dx.doi.org/10.1145/2894748

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

This article is the expanded version of the ICDT 2015 paper [Burdick et al. 2015].
Kolaitis is partially supported by NSF Grant IIS-1217869; Tan is partially supported by NSF grants IIS-1450560 and IIS-1524382.
Authors' addresses: D. Burdick, R. Fagin, and L. Popa, IBM Research – Almaden, 650 Harry Road, San Jose, CA 95120; emails: {drburdic, fagin, lpopa}@us.ibm.com; P. G. Kolaitis and W.-C. Tan, UC Santa Cruz, 1156 High Street, Santa Cruz, CA 95060; emails: {kolaitis, tan}@cs.ucsc.edu.

DOI: http://dx.doi.org/10.1145/2894748

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, and merge-purge (see, e.g., Dong et al. [2005], Elmagarmid et al. [2007], Fellegi and Sunter [1969], Hernández and Stolfo [1995], and Koudas et al. [2006]). Much of entity resolution research has focused on developing the algorithms, similarity measures, and 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 [Galhardas et al. 2001] to the more recent Dedupalog [Arasu et al. 2009] and HIL [Hernández et al. 2013] languages, represent attempts to specify, in 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 article, 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 or 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. Specifically,

a matching constraint in $\mathcal{L}_\mathbf{0}$ is an expression of the form

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

where L is a link relation, $\psi(x, y, \mathbf{u})$ is a (possibly empty) conjunction of source atomic formulas, the universally quantified variables $\mathbf{u}$ must occur in ψ, and each formula α_i is of the form $\exists \mathbf{z_i}\ \phi_i(x, y, \mathbf{u}, \mathbf{z_i})$ with ϕ_i a conjunction of source atomic formulas, equalities, and other built-in or user-defined predicates. The *existential fragment* $\exists\mathcal{L}_\mathbf{0}$ of $\mathcal{L}_\mathbf{0}$ consists of matching constraints of the form

$$\forall x \forall y (L(x, y) \rightarrow \alpha_1 \vee \ldots \vee \alpha_k),$$

where each formula α_i is of the form $\exists \mathbf{z_i}\ \phi_i(x, y, \mathbf{z_i})$ with ϕ_i as before. A key feature of the matching constraints in $\mathcal{L}_\mathbf{0}$ is that, in their most general form, they are *disjunctive constraints* that enable the declarative listing of all the reasons two entities are linked.

We also consider extensions of $\mathcal{L}_\mathbf{0}$ (and hence of $\exists\mathcal{L}_\mathbf{0}$) 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* [Bhattacharya and Getoor 2007], which allows for interdependencies among the various types of links.

In addition to the matching constraints, 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 in fewer ambiguous links. Thus, producing the ambiguous links is an important computational task.

As a first candidate for a class of "good" solutions, we consider *maximal solutions*, where goodness means 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* [Johnson et al. 1988] 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 this, 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 investigate the relationship between our declarative framework and other approaches to entity resolution. We first show that, under the maximal solutions semantics, the existential fragment $\exists\mathcal{L}_\mathbf{0}$ of $\mathcal{L}_\mathbf{0}$ can capture, in a precise sense, the entity resolution fragment of the HIL language in Hernández et al. [2013]. We also point out that $\exists\mathcal{L}_\mathbf{0}$ (and hence HIL) is a proper fragment of $\mathcal{L}_\mathbf{0}$ in the sense that there are entity-linking specifications in $\mathcal{L}_\mathbf{0}$ whose certain links are different from those of every entity-linking specification in $\exists\mathcal{L}_\mathbf{0}$; in particular, $\exists\mathcal{L}_\mathbf{0}$ is a proper fragment of $\mathcal{L}_\mathbf{0}$ in terms of logical equivalence. Furthermore, we 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 [1969] and is at the core of many commercial systems, including IBM's QualityStage [Alur et al. 2008]. 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) [Richardson and Domingos 2006], 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 [Singla and Domingos 2006] is captured by a fragment of $\exists\mathcal{L}_\mathbf{0}$ (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 concerning algorithmic aspects of $\mathcal{L}_\mathbf{0}$ 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 the existential fragment $\exists\mathcal{L}_\mathbf{1}$ of $\mathcal{L}_\mathbf{1}$ 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 $\exists\mathcal{L}_\mathbf{1}$ 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 [Jonsson and Krokhin 2004]. Since $\exists\mathcal{L}_\mathbf{1}$ is a fragment of $\mathcal{L}_\mathbf{1}$, the same hardness results hold true for $\mathcal{L}_\mathbf{1}$ as well. On the positive side, we identify a large syntactic fragment of $\mathcal{L}_\mathbf{1}$ 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.

Relation to the conference version [Burdick et al. 2015]. This article is the expanded full version of a paper that appeared in the proceedings of the ICDT 2015 conference. There are two main differences between the two versions. First, we included

in the present article complete proofs of all results. Second, we identified and studied the existential fragment $\exists\mathcal{L}_\mathbf{0}$ and its variants. We showed that $\exists\mathcal{L}_\mathbf{0}$ is a proper fragment of $\mathcal{L}_\mathbf{0}$ in terms of expressing certain links, and hence in terms of logical equivalence as well. We also showed that $\exists\mathcal{L}_\mathbf{0}$ under the maximal solutions semantics can capture, in a precise sense, the entity resolution fragment of the HIL language [Hernández et al. 2013]. Finally, via realistic examples, we demonstrated the utility of the more expressive language $\mathcal{L}_\mathbf{0}$ for practical cases of entity linking.

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

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

where $\psi(x, y, \mathbf{u})$ is a (possibly empty) conjunction of atomic formulas over $\mathbf{S}$, where the universally quantified variables $\mathbf{u}$ must occur in ψ, and where α_i is of the form $\exists\mathbf{z_i}\, \phi_i(x, y, \mathbf{u}, \mathbf{z_i})$. Each ϕ_i is a conjunction of source atomic formulas, along with equalities and other built-in or user-defined 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., $\alpha_1, \ldots, \alpha_k$) for why a link $L(x, y)$ may exist (provided ψ holds). 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 $L : X \rightarrow Y$ and $L : Y \rightarrow X$. Functional dependencies encode basic cardinality constraints on the result of entity linking. (See also Hernández et al. [2013] 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.

Definition 2.1. We write $\mathcal{L}_\mathbf{0}$ for the language where each formula is either a matching constraint of the form

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

or an inclusion dependency from a link relation to a source relation, or a functional dependency on a link relation. We write $\exists\mathcal{L}_\mathbf{0}$ for the fragment of $\mathcal{L}_\mathbf{0}$ in which the matching constraints have the form

$$L(x, y) \rightarrow \alpha_1 \vee \ldots \vee \alpha_k,$$

where each formula α_i is of the form $\exists \mathbf{z_i}\ \phi_i(x, y, \mathbf{u}, \mathbf{z_i})$, as before. (In particular, there are no universal quantifiers $\forall \mathbf{u}$ and the conjunction ψ is empty, but the formulas α_i may contain existential quantifiers.)

Later, we also consider extensions to $\mathcal{L}_\mathbf{0}$, such as matching constraints that allow for interdependencies among the links.

Definition 2.2. An *entity-linking specification* in $\mathcal{L}_\mathbf{0}$ is 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 $\mathbf{L}$, at most one matching constraint in $\mathcal{L}_\mathbf{0}$, two inclusion dependencies, and zero, one, or two functional dependencies. An *entity-linking specification* in $\exists\mathcal{L}_\mathbf{0}$ is defined similarly with every matching constraint required to be in $\exists\mathcal{L}_\mathbf{0}$.

Definition 2.3. Let $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ be an entity-linking specification, and let 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 2.4. In this scenario, we link subsidiaries in one database with parent companies in another database. Consider the following source schema $\mathbf{S}$:

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

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 here as a set of facts:

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

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 $\texttt{L}(\mathit{sid}, \mathit{cid})$. 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.

The following is an entity-linking specification in $\mathcal{L}_\mathbf{0}$ that exploits the available attributes and the relationship between the source tables. (We will see shortly a second entity-linking specification in $\exists\mathcal{L}_\mathbf{0}$ that also makes sense for this scenario but has “weaker” guarantees.) The entity-linking specification is $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$, where Σ

consists of a matching constraint:

$$\begin{aligned}\mathtt{L}(sid, cid) \rightarrow \forall sn, loc, cn, hd\ \ (&\mathtt{Subsid}(sid, sn, loc) \wedge \mathtt{Company}(cid, cn, hd)\\ &\rightarrow (sn \sim cn)\\ &\vee\\ &\quad \exists e, n, t(\mathtt{Exec}(e, cid, n, t) \wedge contains(t, sn))),\end{aligned}$$

two inclusion dependencies $\mathtt{L}[sid] \subseteq \mathtt{Subsid}[sid]$, $\mathtt{L}[cid] \subseteq \mathtt{Company}[cid]$, and the functional dependency $\mathtt{L} : sid \rightarrow cid$. 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 lists all possible matching conditions as to 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 a similarity 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 $\mathtt{Subsid}\ (sid, sn, loc) \wedge \mathtt{Company}\ (cid, cn, hd)$ 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 similarity 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 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 previous example illustrates that, in general, we allow matching of entities that are not necessarily of the same type; moreover, a link relation need not be an equivalence relation.

The entity-linking specification in Example 2.4 made use, in an essential way, of universal quantification on the right-hand side of the matching constraint. Next, we consider an entity-linking specification in $\exists\mathcal{L}_\mathbf{0}$ for the same scenario. As we shall see, this specification in $\exists\mathcal{L}_\mathbf{0}$ is not as good as the earlier one in the more expressive language $\mathcal{L}_\mathbf{0}$, because it may produce "incorrect" links.

Example 2.5. Consider the same source and link schemas as in Example 2.4. Let $\mathcal{E}' = (\mathbf{L}, \mathbf{S}, \Sigma')$ be the entity-linking specification in $\exists\mathcal{L}_\mathbf{0}$ that has the same inclusion dependencies and functional dependency as in Example 2.4 but has the following

matching constraint:

$$\begin{aligned}\mathtt{L}(sid, cid) \rightarrow\ & \exists sn, loc, cn, hd\,(\mathtt{Subsid}(sid, sn, loc) \wedge \mathtt{Company}(cid, cn, hd) \wedge (sn \sim cn)) \\ \vee\ & \\ & \exists sn, loc, cn, hd, e, n, t\,(\mathtt{Subsid}(sid, sn, loc) \wedge \mathtt{Company}(cid, cn, hd) \wedge \\ & \qquad \mathtt{Exec}(e, cid, n, t) \wedge \mathit{contains}(t, sn)).\end{aligned}$$

The key syntactic difference from the earlier constraint is that the two matching conditions ($\sim$-based and Exec-based) no longer share a common universally quantified context; instead, each matching condition quantifies existentially the source tuples in the Subsid and Company relations.

To illustrate the difference in semantics between the two specifications, assume that we add an extra tuple:

$$\mathtt{Subsid}(s_1, \text{“Chase Manhattan”}, \text{“New York”})$$

to the earlier source instance. Let I' be the resulting source instance. Intuitively, this source instance contains “errors”: we have two different (conflicting) names for s_1, and for all we know now, s_1 could be a subsidiary of “JP Morgan Chase & Co” instead of “Citigroup Inc.”

It is easy to see that the earlier link instance $J_1 = \{\mathtt{L}(s_1, c_1), \mathtt{L}(s_2, c_1)\}$ is a solution for I' with respect to the newer specification $\mathcal{E}'$. In particular, the right-hand side of the matching constraint in $\mathcal{E}'$ holds when s_1 and c_1 play the roles of *sid* and *cid*. However, J_1 is no longer a solution for I' with respect to the earlier specification $\mathcal{E}$, since the link $\mathtt{L}(s_1, c_1)$ is no longer allowed: for some name variation for s_1, namely, “Chase Manhattan,” none of the matching conditions hold. The earlier specification $\mathcal{E}$ requires that the disjunction of matching conditions holds for *every* subsidiary name associated to s_1. In contrast, the specification $\mathcal{E}'$ has weaker requirements and allows the link $\mathtt{L}(s_1, c_1)$ to exist in a solution, which may be incorrect.

Another important feature of our 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 then considered (e.g., functional dependencies), the links in L may become invalid. As a result, it is often the case that source instances may have no solutions with respect to specifications that include source-to-link rules. (This is a theme that we shall revisit later in various contexts.) In contrast, our notion of entity-linking specification always has solutions. A large part of this article 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 2.6. Assume a class C of solutions and an entity-linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$. Let I be a source instance.

(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 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 3.1. 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 3.2. We revisit Example 2.4. 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 versus repairs. We next show a connection with source-to-link constraints and the framework of repairs [Arenas et al. 1999], 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 this, 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 [Fagin et al. 2005] where the link schema plays the role of a target schema. The constraints in Σ' are a particular case of first-order tgds [Arenas et al. 2013], 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 Arenas et al. [2013], 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* first-order 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. Thus, U itself is not a solution for I with respect to the specification given by the source-to-link constraints Σ' *and* the functional dependencies (and, in general, no such solutions may exist). 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* [Arenas et al. 1999] 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 3.3. *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.

PROOF. Assume first that J is a maximal solution for I w.r.t. $\mathcal{E}$. Since $(J, I) \models \Sigma$, we have that (J, I) satisfies m_L for every link symbol L. This, together with the fact that (J, I) also satisfies the inclusion dependencies in $\mathcal{E}$, implies that the relation L in J is a set of pairs (x, y) such that (1) x occurs in $R[A]$ in I, (2) y occurs in $R'[A']$ in I, and (3) I satisfies the right-hand side of m_L for the given x and y. But, by definition of m'_L and by the properties of the chase, we have that $U = chase_{\mathcal{M}}(I)$ contains all such pairs (x, y). Hence, we obtain that $J \subseteq U$.

Furthermore, it is also the case that $J \models F$, where F is the set of functional dependencies in Σ. We show next there is no J' such that $J \subset J' \subseteq U$ and such that $J' \models F$. This will imply that J is a maximal subinstance of U that is consistent w.r.t. F and, hence, J is a subset repair of U w.r.t. F. Assume that there is J' such that $J \subset J' \subseteq U$ and such that $J' \models F$: we shall derive a contradiction. We have argued earlier that U contains, for every link L, all pairs (x, y) such that conditions (1), (2), and (3) hold. Since $J' \subseteq U$, we obtain that every pair (x, y) in a link relation L of J' satisfies conditions (1), (2), and (3). From (1) and (2), we obtain that (J', I) satisfies $L[X] \subseteq R[A]$ and $L[Y] \subseteq R'[A']$ for every link L. From condition (3), we obtain that $(J', I) \models m_L$ for every link L. Furthermore, we also have that $J' \models F$, by the assumption on J'. Hence, J' is a solution for I w.r.t. $\mathcal{E}$. Since $J \subset J'$, we obtain a contradiction to the fact that J was a maximal solution for I w.r.t. $\mathcal{E}$.

For the converse, assume that J is a subset repair of U w.r.t. F. Thus, J is a maximal subinstance of U that is consistent w.r.t. F. Since $J \subseteq U$, by the same argument we used for J' earlier, we obtain that (J, I) satisfies the inclusion dependencies and the matching constraints in Σ. Since we also have that $J \models F$, we obtain that J is a solution for I w.r.t. $\mathcal{E}$. We show next that there is no J' such that $J \subset J'$ and such that $(J', I) \models \Sigma$. This will imply that J is a maximal solution for I w.r.t. $\mathcal{E}$. Assume that there is J' such that $J \subset J'$ and such that $(J', I) \models \Sigma$. Applying the same argument used in the first paragraph of the proof to show that $(J, I) \models \Sigma$ implies $J \subseteq U$, we obtain that $J' \subseteq U$. Since $J' \models F$, it follows that J' is a subinstance of U that is consistent w.r.t. F. Given that $J \subset J'$, we obtain a contradiction to the fact that J is a maximal subinstance of U that is consistent w.r.t. F. □

Based on the previous proposition and known results about the consistent answers of projection-free queries [Chomicki and Marcinkowski 2005], we immediately obtain the following tractability results.

THEOREM 3.4. *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 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 3.3 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 to $\mathcal{L}_\mathbf{0}$ 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.

We close this section by showing that the language $\mathcal{L}_\mathbf{0}$ has strictly more expressive power for entity linking than its simpler fragment $\exists\mathcal{L}_\mathbf{0}$.

THEOREM 3.5. *There exists an entity-linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ in $\mathcal{L}_\mathbf{0}$ for which there is no entity-linking specification $\mathcal{E}' = (\mathbf{L}, \mathbf{S}, \Sigma')$ in $\exists\mathcal{L}_\mathbf{0}$ such that for every source instance I, the set of certain links for I w.r.t. the class of maximal solutions and $\mathcal{E}'$ is the same as the set of certain links for I w.r.t. the class of maximal solutions and $\mathcal{E}$.*

PROOF. The idea behind the proof is that entity-linking specifications in $\exists\mathcal{L}_\mathbf{0}$ possess a *monotonicity* property that entity-linking specifications in $\mathcal{L}_\mathbf{0}$ need not possess. Specifically, suppose that $\mathcal{E}'$ is an entity-linking specification in $\exists\mathcal{L}_\mathbf{0}$, I is a source instance, and J is a solution for I w.r.t. $\mathcal{E}'$. If I' is a source instance such that $I \subseteq I'$, then J is also a solution for I' w.r.t. to $\mathcal{E}'$. This is so because every matching constraint of $\mathcal{E}'$ is satisfied by the pair (J, I') since the right-hand side of each such constraint is a disjunction of existential positive formulas; hence, it is preserved when we pass from a source instance I to a source instance I' containing I. Similarly, the inclusion dependencies of $\mathcal{E}'$ are satisfied by the pair (J, I') because they are satisfied by the pair (J, I) and $I \subseteq I'$.

Consider the entity-linking specification $\mathcal{E}$ in Example 2.4, which is a specification in $\mathcal{L}_\mathbf{0}$. Toward a contradiction, assume that there is an entity-linking specification $\mathcal{E}'$ in $\exists\mathcal{L}_\mathbf{0}$ such that for every source instance I, the set of certain links for I w.r.t. the class of maximal solutions and $\mathcal{E}'$ is the same as the set of certain links for I w.r.t. the class of maximal solutions and $\mathcal{E}$. Let I be the source instance consisting of the facts $\mathtt{Subsid}(s_1, \text{“Citibank N.A.”}, \text{“New York”})$ and $\mathtt{Company}(c_1, \text{“Citigroup Inc”}, \text{“New York”})$. The only maximal solution for I w.r.t. $\mathcal{E}$ is the link instance $J = \{\mathtt{L}(s_1, c_1)\}$. Hence, $\mathtt{L}(s_1, c_1)$ is the only certain link for I w.r.t. the class of maximal solutions and $\mathcal{E}$, as well as the only certain link for I w.r.t. the class of maximal solutions and $\mathcal{E}'$. Let I' be the source instance obtained from I by adding the fact $\mathtt{Subsid}(s_1, \text{“Chase Manhattan”}, \text{“New York”})$. It is easy to see that the empty relation $\emptyset$ is the only solution for I' w.r.t. $\mathcal{E}$; hence, there are no certain links for I' w.r.t. to the class of maximal solutions and $\mathcal{E}$. In contrast, because of the aforementioned monotonicity property, the link instance $J = \{\mathtt{L}(s_1, c_1)\}$ is a solution for I' w.r.t. $\mathcal{E}'$. Furthermore, we claim it is the only maximal solution for I' w.r.t. $\mathcal{E}'$. This is so because the only two inclusion dependencies that are possible for $\mathcal{E}'$ are $\mathtt{L}[sid] \subseteq \mathtt{Subsid}[sid]$ and $\mathtt{L}[cid] \subseteq \mathtt{Company}[cid]$, and therefore the only possible values that can appear in $\mathtt{L}$ are s_1 and c_1. We conclude that $\mathtt{L}(s_1, c_1)$ is a certain link for I' w.r.t. the class of maximal solutions and $\mathcal{E}'$, which violates the hypothesis about $\mathcal{E}'$. □

COROLLARY 3.6. *There exists an entity-linking specification in $\mathcal{L}_\mathbf{0}$ that is not logically equivalent to any entity-linking specification in $\exists\mathcal{L}_\mathbf{0}$.*

The preceding results reveal that universal quantification in the matching constraints of $\mathcal{L}_\mathbf{0}$ is necessary, in the sense that it cannot be expressed in general by using matching constraints in the syntactically simpler language $\exists\mathcal{L}_\mathbf{0}$.

4. CONNECTION TO THE HIL LANGUAGE

We now make a connection between our declarative framework for entity linking and the HIL language [Hernández et al. 2013], which was recently developed at IBM Research–Almaden and used in the Midas system [Burdick et al. 2011] for entity extraction and integration from public financial data. We note that HIL is a language that, in addition to supporting entity linking, contains constructs that support entity fusion (i.e., merging of linked entities into unified entities). In this section, we will focus our comparison only on the entity-linking fragment of HIL. Concretely, we show that

a core entity-linking fragment of HIL is captured, in a precise sense, by the language $\exists\mathcal{L}_{\mathbf{0}}$ under the semantics given by maximal solutions.

4.1. HIL: Entity-Linking Fragment and Semantics

We start by introducing, via an example, the core entity-linking fragment of HIL, and then formulate its operational semantics.

Entity-linking operations in HIL are expressed via SQL-like statements that specify the type of links being created, the source sets that are used for linking, and the matching rules. Additionally, a cardinality clause enforces whether the links should be many to many, many to one, or one to one. Other constructs, which will not be considered here, include a blocking clause to specify ways to partition the input data for faster processing, and a form of preference constraints to disambiguate among links in certain situations.

As a concrete example, we can use the following HIL statement to link subsidiaries and companies in our running example (see Example 2.4):

```
create link L as
select [sid : S.sid, cid : C.cid]
from Subsid S, Company C, Exec E
match using
  m1 : S.sname ∼ C.cname,
  m2 : E.cid = C.cid and contains (E.title, S.sname)
cardinality N : 1
```

Instead of presenting the semantics of the entity-linking fragment of HIL given in Hernández et al. [2013], we give an equivalent description that is based on translating every HIL statement into a set of constraints. Concretely, the previous HIL statement can be formally translated into a set Π, consisting of the following constraints:

$$(\pi_1)\ \texttt{Subsid}(sid, sn, loc) \wedge \texttt{Company}(cid, cn, hd) \wedge (sn \sim cn) \rightarrow \texttt{L}(sid, cid)$$

$$(\pi_2)\ \texttt{Subsid}(sid, sn, loc) \wedge \texttt{Company}(cid, cn, hd) \wedge \texttt{Exec}(e, cid, n, t) \wedge contains(t, sn) \rightarrow \texttt{L}(sid, cid)$$

$$(i_1)\ \texttt{L}[sid] \subseteq \texttt{Subsid}[sid]$$

$$(i_2)\ \texttt{L}[cid] \subseteq \texttt{Company}[cid]$$

$$(f)\ \texttt{L} : sid \rightarrow cid$$

Thus, the matching rules m1 and m2 in the preceding HIL statement become two *source-to-link tgds* (π_1) and (π_2), which will be used to populate the link relation L from the relevant source relations. Note that, in order to be faithful to the HIL semantics, not all source relations in the from clause need to be used in the left-hand side of a source-to-link tgd. In particular, π_1 does not include an atom for Exec because the matching rule m_1 does not use this relation.

We note that the source-to-link tgds used in this translation are a variation on the notion of source-to-target tgds in data exchange [Fagin et al. 2005] where the link schema plays the role of a target schema and where the left-hand side of the tgd is allowed to have built-in or user-defined predicates (in addition to the source atomic formulas).

When translating well-formed HIL statements, the set of relational atoms in the left-hand side of each source-to-link tgd is formed based on a subset of the source relations in the from clause of the HIL statement. This subset contains, minimally, the

two relational atoms that are needed to form the links (e.g., `Subsid` and `Company` for *sid* and *cid*, respectively) and, additionally, any other relational atom that may be required by the matching rule (e.g., `Exec` in the case of π_2).

Note that the specification Π also contains the two inclusion dependencies (i_1) and (i_2) and the functional dependency (f). The inclusion dependencies reflect how the link attributes in the `select` clause of the HIL statement relate to the source relations in the `from` clause of the same HIL statement. The functional dependency (f) captures the `N:1` cardinality requirement of matching subsidiaries to a single parent company. (Note that in the case of a 1:1 cardinality constraint in the HIL statement, we will need two functional dependencies.) Let us refer to such a specification Π as a *HIL specification*.

The following *chase-and-remove* process describes the operational semantics of a HIL specification. First, we chase with the source-to-link tgds in Π. This means that, given a source instance I, we construct in polynomial time a link instance U by evaluating the left-hand side of the source-to-link tgds and by adding the corresponding link facts. However, some of the links in U may violate the functional dependencies in Π. To address this, HIL removes from U all links involved in the violations. Thus, if two or more links violate a functional dependency, then all of them are removed. We write $U_0 = \textit{chase-and-remove}(\Pi, I)$ to denote the subset U_0 of U returned by HIL. Note that the inclusion dependencies (i_1) and (i_2) do not play a role in the "chase-and-remove" procedure, but they are guaranteed to be satisfied by the result U_0. We carry them around for the purposes of translation to $\exists\mathcal{L}_\mathbf{0}$, which we describe next.

4.2. Translation to $\exists\mathcal{L}_\mathbf{0}$

We now show that the aforementioned entity-linking fragment of HIL, with the operational semantics described in the previous subsection, is captured in a precise sense by the language $\exists\mathcal{L}_\mathbf{0}$ with the semantics given by maximal solutions.

For simplicity of discussion, we assume there is one link relation symbol L. Assume a HIL specification Π, consisting of source-to-link tgds $\phi_i(x, y, \mathbf{z}_i) \rightarrow L(x, y)$, for $1 \leq i \leq n$, together with functional dependencies and inclusion dependencies on the link relations. We define the *corresponding entity-linking specification in* $\exists\mathcal{L}_\mathbf{0}$ to consist of the matching constraint $L(x, y) \rightarrow \exists\mathbf{z_1}\phi_1 \vee \cdots \vee \exists\mathbf{z_n}\phi_n$, together with the same inclusion dependencies and functional dependencies on L as in the specification Π.

THEOREM 4.1. *Let Π be a HIL specification, let $\mathcal{E}$ be the corresponding entity-linking specification in $\exists\mathcal{L}_\mathbf{0}$, and let I be a source instance. Then the set chase-and-remove(Π, I) coincides with the set of certain links for I w.r.t. the class of maximal solutions and $\mathcal{E}$.*

PROOF. First, we invert the entity-linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ by constructing a source-to-link specification $\mathcal{M} = (\mathbf{S}, \mathbf{L}, \Sigma')$ as in Section 3. However, since the matching constraint in Σ has the special form $L(x, y) \rightarrow \exists\mathbf{z_1}\phi_1(x, y, \mathbf{z_1}) \vee \cdots \vee \exists\mathbf{z_n}\phi_n(x, y, \mathbf{z_n})$, it follows that the corresponding source-to-link constraint in Σ' has the form

$$(m'_L)\ \ R(\ldots, x, \ldots) \wedge R'(\ldots, y, \ldots) \wedge\ (\exists\mathbf{z_1}\phi_1(x, y, \mathbf{z_1}) \vee \ldots \vee \exists\mathbf{z_n}\phi_n(x, y, \mathbf{z_n}))\ \rightarrow L(x, y).$$

In this, 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', where we assumed that the inclusion dependencies are $L[X] \subseteq R[A]$ and $L[Y] \subseteq R'[A']$. We now observe that each formula ϕ_i, for $1 \leq i \leq n$, arises from the HIL specification Π. Then each such formula already contains two relational atoms $R(\ldots, x, \ldots)$ and $R'(\ldots, y, \ldots)$ with x and y in the positions of A and A', respectively. It follows that we can eliminate the two relational atoms for R and R' on the left-hand side of the previous source-to-link constraint (m'_L) and obtain an equivalent constraint. We can also eliminate disjunction by further transforming the source-to-link constraint into a *set* of source-to-link tgds

as follows:

$$\phi_1(x, y, \mathbf{z_1}) \rightarrow L(x, y), \ldots, \phi_n(x, y, \mathbf{z_n}) \rightarrow L(x, y).$$

It can now be seen that this is precisely the set of source-to-link tgds in the HIL specification Π. Thus, we showed that the specification $\mathcal{M}$ that is used to "invert" $\mathcal{E}$ via the construction in Section 3 is the same as the original HIL specification from which $\mathcal{E}$ was constructed. We now make use of Proposition 3.3 to conclude the proof of this theorem.

Let I be a source instance. First, by Proposition 3.3, we have that 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, where F is the set of functional dependencies. It follows that the set of certain links for I w.r.t. the class of maximal solutions and $\mathcal{E}$ is the same as the set of links that appear in every subset repair of $U = chase_{\mathcal{M}}(I)$ w.r.t. F. However, it can be seen that the latter set is exactly the set of links in U that are not involved in any violations of F (otherwise they would not appear in every subset repair of U). Hence, the set of certain links for I w.r.t. the class of maximal solutions and $\mathcal{E}$ is the same as the set *chase-and-remove*(Π, I). □

Since by the earlier Theorem 3.5 the language $\exists\mathcal{L}_\mathbf{0}$ has strictly less expressive power for entity linking than the language $\mathcal{L}_\mathbf{0}$, it follows that the language of HIL specifications has strictly less expressive power for entity linking than the language $\mathcal{L}_\mathbf{0}$. In particular, HIL specifications do not take advantage of the universal quantification feature of $\mathcal{L}_\mathbf{0}$, whose benefits were illustrated in Section 2.

5. MAXIMUM-VALUE SOLUTIONS

5.1. Deficiency of Maximal Solutions

We now point out the main deficiency or limitation of the semantics based on maximal solutions, which is that, 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. This limitation also applies to the core entity-linking fragment of HIL that was discussed in the previous section, since we established there that the semantics of a HIL specification is equivalent to the certain links of an $\exists\mathcal{L}_\mathbf{0}$ specification w.r.t. maximal solutions.

To exemplify the limitation of maximal solutions, consider our scenario in Example 2.4. 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 this issue, we will 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 us to increase the number of links that qualify as certain, thus reducing ambiguity.

5.2. The Language $\mathcal{L}_\mathbf{0}(\oplus)$ and Maximum-Value Solutions

We start by introducing 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:

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

Syntactically, everything else is the same as in $\mathcal{L}_\mathbf{0}$. We call the resulting language $\mathcal{L}_\mathbf{0}(\oplus)$. Similarly to the case of $\exists\mathcal{L}_\mathbf{0}$, we denote by $\exists\mathcal{L}_\mathbf{0}(\oplus)$ the existential fragment of $\mathcal{L}_\mathbf{0}(\oplus)$.

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 earlier (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

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

In the previous equation, $\mathbf{u}_0$ 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 Equation (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 5.1. 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 5.2. Recall Example 2.4. By applying Equation (1), the values of the individual links that can be formed between subsidiary ids and company ids, based on the matching constraint, are

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

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 Equation (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 is 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 $\mathrm{Val}(L(a, b))$ is 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 7, we will consider richer languages, where the αs 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 5.3. *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. $\mathcal{E}$.*

PROOF. Let J be a maximum-value solution. If J is not maximal, then it is possible to add another fact t to J such that $J' = J \cup \{t\}$ is a solution. But then the value of J' is more than the value of J, since each additional fact adds value of at least 1. This contradicts the assumption that J is a maximum-value solution. □

The reverse direction for Proposition 5.3 does not hold, as seen in Example 5.2. 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 3.4, which follows from results on data repairs, the proof of the following theorem is of a different nature and makes use of algorithms for matchings in weighted bipartite graphs.

THEOREM 5.4. *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 $\mathcal{E}$.*

In the remainder of this section, we give a proof of Theorem 5.4. We begin with a digression into graph theory and formulate a graph problem that will be used to prove results for entity-linking specifications, including Theorem 5.4.

A *weighted (undirected) graph* is a triple (V, E, w), where V is a set of *nodes*; E is a set of *edges*, each of which is a pair (a, b) of distinct nodes; and w is a function that assigns to each edge a nonnegative integer. A *matching* of a graph is a subset $\mathcal{M}$ of edges such that each node appears in at most one edge in $\mathcal{M}$. A *perfect matching* is a matching in which each node appears in exactly one edge in $\mathcal{M}$. A *maximum-weight matching* is a matching $\mathcal{M}$ such that the sum of the weights of its edges in $\mathcal{M}$ is maximized among all matchings. Similarly, a *maximum-weight perfect matching* is a perfect matching $\mathcal{M}$ such that the sum of the weights of its edges in $\mathcal{M}$ is maximized among all perfect matchings. A *weighted bipartite graph* is a weighted graph whose nodes can be partitioned into two disjoint sets of nodes V_1 and V_2 such that every edge in E connects a node in V_1 and a node in V_2.

There is a long line of research in algorithms for enumerating maximum weight perfect matchings in weighted bipartite graphs, including Murty [1968] and Chegireddy and Hamacher [1987]. The next result, which will be of interest to us in the sequel, has been recently proven in Droschinsky et al. [2014] for complete bipartite graphs. We note, however, that a general version (for graphs that are not necessarily complete) can also be obtained by combining and adapting ideas from the work of Itai et al. [1978] on the enumeration of maximum matchings in bipartite graphs and from the work of Chegireddy and Hamacher [1987] on the enumeration of maximum weight perfect matchings in weighted bipartite graphs.

THEOREM 5.5. *There is a polynomial-delay algorithm for enumerating the maximum-weight matchings in a weighted bipartite graph.*

Given a weighted graph, we say that an edge is *certain* if it is in every maximum-weight matching. We say that an edge is *possible* if it is in some maximum-weight matching. We say that an edge is *ambiguous* if it is possible but not certain. We next give a theorem, which will be very useful in our subsequent proofs.

THEOREM 5.6. *For matchings in weighted graphs, there is a polynomial-time algorithm for computing the set of certain edges and a polynomial-time algorithm for computing the set of ambiguous edges.*

PROOF. Let $G = (V, E, w)$ be a weighted graph. To obtain our polynomial-time algorithms for computing the sets of certain and ambiguous edges, we consider each of the (polynomially many) edges one by one and decide if it is certain or ambiguous.

To decide if the edge (a, b) is certain, do the following. Find a maximum-weight matching of G, and call its total weight W. Then delete the edge (a, b), and find the maximum-weight matching over the remaining edges, and call its total weight W'. We now show that the edge (a, b) is certain if and only if $W' < W$. First, if the edge (a, b) is certain, then every maximum-weight matching that does not include (a, b) has total weight less than W, and so $W' < W$. Conversely, if $W' < W$, then every maximum-weight matching that does not use the edge (a, b) has total weight less than W, so (a, b) is a certain edge.

We now give a polynomial-time algorithm for deciding if an edge is a possible edge. Since there is also a polynomial-time algorithm for deciding if an edge is a certain edge, this implies that there is a polynomial-time algorithm for deciding if an edge is an ambiguous edge. To decide if the edge (a, b) is possible, do the following. Find a maximum-weight matching of G, and call its total weight W. Then delete the nodes a, b from G, and delete every edge that contains either a or b. For the remaining weighted graph, find a maximum-weight matching, and call its total weight W'. Let $W'' = W' + w((a, b))$. It is easy to see that W'' is the maximum weight of those matchings that includes the edge (a, b) (intuitively, we are "forcing" (a, b) into the matching). Therefore, (a, b) is a possible edge if and only if $W'' = W$. □

We can now give the proof of Theorem 5.4.

PROOF OF THEOREM 5.4. For simplicity of description, we assume first that there is only one link, link_1. Later we shall consider the more general case.

Let R_{link_1} be the set of all pairs (a, b) that satisfy the inclusion dependencies for link_1, and that satisfy the right-hand side of the matching constraint for link_1 when the left-hand side is instantiated by $\text{link}_1(a, b)$ (so that (a, b) is a candidate tuple for the link_1 relation). We can think of R_{link_1} as a weighted bipartite graph, where the weight of the edge (a, b) is the value assigned to $\text{link}(a, b)$ according to our conventions.

There are three possibilities, depending on the FDs of link_1:

(a) If link_1 has both FDs, then we make use of Theorem 5.5 to give a polynomial-delay algorithm, and Theorem 5.6 to find the certain and ambiguous links.
(b) If link_1 has only one FD, say, $A \to B$, then for each a in the A column of R_{link_1}, let X_a be the set of all B-values b such that (a, b) is in R_{link_1}, and such that (a, b) has the maximum weight among all pairs (a, b') of R_{link_1} (with A entry a). The maximum-value solutions are exactly those solutions J such that for every a in the A column of R_{link_1} there is exactly one fact $\text{link}_1(a, b)$ in J, and that choice of b is in X_A. It is then straightforward to systematically enumerate in a polynomial-delay algorithm the maximum-value solutions. We have that $\text{link}_1(a, b)$ is certain if and only if b is the unique member of X_a. Further, $\text{link}_1(a, b)$ is ambiguous if and only if $b \in X_a$ and there is another member besides b in X_a.
(c) If link_1 has no FDs, then the only maximum-value solution is R_{link_1} itself. Every member of R_{link_1} is a certain link, and there are no ambiguous links.

This completes the proof when there is only one link. When there are multiple links, we modify the polynomial-delay algorithm as follows. We take a fixed ordering of the links, say, $\text{link}_1, \text{link}_2, \text{link}_3, \ldots$, and use a polynomial-delay algorithm to enumerate all

choices for the link_1 relation in a maximum-value solution. As each such link_1 relation appears, we then do a polynomial-delay algorithm to enumerate all choices for the link_2 relation in a maximum-value solution. For each pair consisting of a choice for the link_1 relation and the link_2 relation, we use a polynomial-delay algorithm to enumerate all choices for the link_3 relation in a maximum-value solution, and so on. The certain links $\text{link}_i(a, b)$ are exactly those we would get if link_i were the only link, and similarly for the ambiguous links. □

6. 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 that 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. We note that Theorem 5.4 goes through when we replace $\mathcal{L}_\mathbf{0}(\oplus)$ by $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$, by the same proof.

Similarly to the case of $\exists\mathcal{L}_\mathbf{0}$ and $\exists\mathcal{L}_\mathbf{0}(\oplus)$, we denote by $\exists\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$ the existential fragment of $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$.

6.1. Comparison to Probabilistic Matching

The first connection we make is to a well-known class of probabilistic matching algorithms that originated with Fellegi and Sunter [1969] and is at the core of many commercial systems including IBM's QualityStage [Alur et al. 2008], 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. Second, 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). Third, a link is returned if it has a high enough aggregated score (higher than a threshold, which also must be learned or tuned).

We show by example that the first two steps in the aforementioned 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 6.1. 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 address attributes (e.g., street name street, etc.). Furthermore, we assume that 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:

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

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 previous 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 nonconflicting 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 previous matching constraint is in a very limited fragment of $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$, where each disjunct is a simple atomic formula with no existential quantification and no conjunction.

6.2. Comparison to Markov Logic Networks for Entity Resolution

We now connect to a richer probabilistic framework, that of Markov Logic Networks (MLNs) [Richardson and Domingos 2006], 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 [Singla and Domingos 2006] is captured, in a precise sense, by the fragment $\exists\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$. Thus, rather surprisingly, a purely probabilistic approach (based on MLNs) can be captured in a deterministic way (via $\exists\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$). 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.

6.2.1. Markov Logic Networks: Preliminaries. We now define a fragment of MLNs that we call "linear MLNs." 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 Singla and Domingos [2006], 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 interdependencies 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 $\sigma_i \rightarrow L(x, y)$ 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 Singla and Domingos [2006]) to $e^{w_1|\sigma_1 \rightarrow L| + \cdots + w_n|\sigma_n \rightarrow L|}$, 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.

6.2.2. Translation to $\exists\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$. Given the linear MLN with formulas $\sigma_i \rightarrow L(x, y)$ with weight w_i, for $1 \leq i \leq n$, we define the *corresponding entity-linking specification in* $\exists\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ to consist of the matching constraint $L(x, y) \rightarrow \exists \mathbf{z_1}\sigma_1 \oplus \cdots \oplus \exists \mathbf{z_n}\sigma_n$, 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 $\exists\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$. 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 6.2. *Let $\mathcal{M}$ be a linear MLN, let $\mathcal{E}$ be the corresponding entity-linking specification in $\exists\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$, 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 that we shall give of maximum-probability worlds in terms of maximum-value solutions. As we shall see, we also use that characterization to prove the analog of Theorem 5.4 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.

We now begin the proof of Theorem 6.2. We first prove the first part. We must show that

$$\text{Max Value Solutions}_{\mathcal{E}}(I) = \tag{2}$$

$$\text{Max Probability Worlds}_{\mathcal{M}}(I) \cap \text{Solutions}_{\mathcal{E}}(I). \tag{3}$$

Recall that the probability assigned to L_0 is proportional to

$$e^{w_1|\sigma_1 \to L| + \cdots + w_n|\sigma_n \to L|}, \tag{4}$$

where the role of L is played by L_0. Now $|\sigma_i \to L| = |\neg\sigma_i| + |\sigma_i \wedge L|$ (this relies on the fact that every variable of $\sigma_i \to L$ is a variable of σ_i). So Formula (4) is equal to

$$e^{w_1|\neg\sigma_1| + w_1|\sigma_1 \wedge L| + \cdots + w_n|\neg\sigma_n| + w_n|\sigma_n \wedge L|}. \tag{5}$$

We see that Formula (5) equals

$$e^{w_1|\neg\sigma_1| + \cdots + w_n|\neg\sigma_n|} e^{w_1|\sigma_1 \wedge L| + \cdots + w_n|\sigma_n \wedge L|}. \tag{6}$$

Now $e^{w_1|\neg\sigma_1| + \cdots + w_n|\neg\sigma_n|}$ is independent of L. So to maximize Formula (6), we must maximize the expression $e^{w_1|\sigma_1 \wedge L| + \cdots + w_n|\sigma_n \wedge L|}$; that is, we must maximize

$$w_1|\sigma_1 \wedge L| + \cdots + w_n|\sigma_n \wedge L|. \tag{7}$$

Assume that $L_0 \in \text{Solutions}_{\mathcal{E}}(I)$. Since $\text{Max Value Solutions}_{\mathcal{E}}(I) \subseteq \text{Solutions}_{\mathcal{E}}(I)$, it follows easily that to prove the theorem, we need only show that $L_0 \in \text{Max Value Solutions}_{\mathcal{E}}(I)$ if and only if $L_0 \in \text{Max Probability Worlds}_{\mathcal{M}}(I)$. The score we assign to L_0 in the entity-linking specification is obtained by holding fixed a tuple (a, b) of L_0, then taking $A_{a,b}$ to be the sum over i of w_i times the number of assignments to σ_i that hold when x is taken to be a, and y is taken to be b, and finally summing $A_{a,b}$ over all (a, b) in L_0. But we can simply reverse the order of summation by holding i fixed and taking B_i to be the sum over all (a, b) in L_0 of w_i times the number of assignments to σ_i that hold when x is taken to be a, and y is taken to be b, and finally summing B_i over all i. But this latter sum is equal to Formula (7), when the role of L is played by L_0.

Hence, we are checking to see whether L_0 maximizes the same quantity, namely Formula (7), to decide if $L_0 \in \text{Max Value Solutions}_{\mathcal{E}}(I)$ and to decide if $L_0 \in \text{Max Probability Worlds}_{\mathcal{M}}(I)$. This proves the first part of the theorem.

To prove the second part of Theorem 6.2, we need another connection between maximum-probability worlds and maximum-value solutions, which we state and prove next.

THEOREM 6.3. *Let $\mathcal{M}$ be the linear MLN given by the set of formulas $\sigma_i \to L(x, y)$ with weight w_i, for $1 \le i \le n$. Let $\mathcal{E}$ be the corresponding entity-linking specification in $\exists\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$, let I be a source instance, and let $N = \{(a, b) : \neg\exists \mathbf{z_i} \sigma_i(a, b)$ holds in I for every $i\}$, where $\mathbf{z_i}$ consists of the variables in σ_i other than x and y. Then the members of Max Probability Worlds$_{\mathcal{M}}(I)$ are precisely those possible worlds of the form $L_1 \cup L_2$, where L_1 is a member of Max Value Solutions$_{\mathcal{E}}(I)$, and L_2 is a subset of N.*

PROOF. We now show that if $L_0 \in \text{Max Probability Worlds}_{\mathcal{M}}(I)$, then there are L_1 and L_2 as in the statement of the theorem. Let $L_1 = L_0 \cap \overline{N}$, where $\overline{N}$ is the set of tuples (a, b) from the domain of I where $(a, b) \notin N$, and let $L_2 = L_0 \cap N$. Then $L_0 = L_1 \cup L_2$. Since clearly $L_2 \subseteq N$, we need only show that $L_1 \in \text{Max Value Solutions}_{\mathcal{E}}(I)$. It is straightforward to see that $|\sigma_i \wedge L|$ takes on the same value when the role of L is played by L_0 as when the role of L is played by L_1. Hence, Formula (7) takes on the

same value for both, and so since L_0 maximizes Formula (7), so does L_1. Therefore, $L_1 \in$ Max Value Solutions$_{\mathcal{E}}(I)$, as desired.

Conversely, assume that $L_0 = L_1 \cup L_2$, where L_1 and L_2 are as in the statement of the theorem. Since $L_1 \in$ Max Value Solutions$_{\mathcal{E}}(I)$, it follows from Theorem 6.2 that we have that $L_1 \in$ Max Probability Worlds$_{\mathcal{M}}(I)$. So similarly to before, it follows that we also have $L_0 \in$ Max Probability Worlds$_{\mathcal{M}}(I)$.

We are now able to prove the second part of Theorem 6.2. Assume first that ℓ is a certain link of $\mathcal{E}$. Let L_0 be a member of Max Probability Worlds$_{\mathcal{M}}(I)$; we must show that $\ell \in L_0$. By Theorem 6.3, we know that L_0 is of the form $L_1 \cup L_2$, where $L_1 \in$ Max Value Solutions$_{\mathcal{E}}(I)$. Since ℓ is a certain link of $\mathcal{E}$, it follows that $\ell \in L_1$, and hence $\ell \in L_0$, as desired.

Conversely, assume that ℓ is a certain link of $\mathcal{M}$. Let L_1 be a member of Max Value Solutions$_{\mathcal{E}}(I)$; we must show that $\ell \in L_1$. Now $L_1 \in$ Max Probability Worlds$_{\mathcal{M}}(I)$, by the first part of Theorem 6.2. Therefore, since ℓ is a certain link of $\mathcal{M}$, it follows that $\ell \in L_1$, as desired.

The next corollary says that the analog of Theorem 5.4 holds for linear MLNs.

COROLLARY 6.4. *Let $\mathcal{M}$ be a linear MLN. Then:*

- *There is a polynomial-delay algorithm that, given a source instance I, enumerates all the maximum-probability worlds for I w.r.t. $\mathcal{M}$.*
- *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-probability worlds and $\mathcal{M}$.*
- *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-probability worlds and $\mathcal{M}$.*

PROOF. For convenience, let us refer to the result of replacing $\mathcal{L}_{\mathbf{0}}(\oplus)$ by $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ in Theorem 5.4 as the $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ *version of Theorem 5.4*. As we noted earlier, the $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ version of Theorem 5.4 holds (by the same proof as Theorem 5.4). For all three parts of the corollary, we shall make use of the $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ version of Theorem 5.4. Let $\mathcal{E}$ be the entity-linking specification in $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ that corresponds to $\mathcal{M}$.

For the first part of the corollary, it is easy to see from the characterization of maximum-probability worlds in Theorem 6.3 that the polynomial-delay algorithm for listing the maximum-value solutions for a source instance I w.r.t. $\mathcal{E}$ can be modified to give a polynomial-delay algorithm for listing the maximum-probability worlds of I w.r.t. $\mathcal{M}$.

The second part of the corollary follows immediately from the second part of Theorem 6.2 and the second part of the $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ version of Theorem 5.4.

We now prove the third part of the corollary. The set of possible links for a source instance I w.r.t. $\mathcal{E}$ is the union of the sets of certain links and ambiguous links. So by the $\mathcal{L}_{\mathbf{0}}(\oplus, \mathbf{w})$ version of Theorem 5.4, there is a polynomial-time algorithm for computing the possible links for I w.r.t. $\mathcal{E}$. But it follows easily from Theorem 6.3 that the possible links for I w.r.t $\mathcal{M}$ are exactly the possible links for I w.r.t. $\mathcal{E}$, along with the members of N. Since also there are polynomial-time algorithms for computing the possible links for I w.r.t. $\mathcal{E}$ and for computing N, it follows that there is a polynomial-time algorithm for computing the possible links for I w.r.t $\mathcal{M}$. Since the set of ambiguous links is the set difference between the possible and certain links, this gives a polynomial-time algorithm for computing the ambiguous links for I w.r.t. $\mathcal{M}$. □

In summary, in Sections 6.1 and 6.2, we showed that, under a suitable extension that allows for weights, our declarative language can capture important classes of probabilistic methods. 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

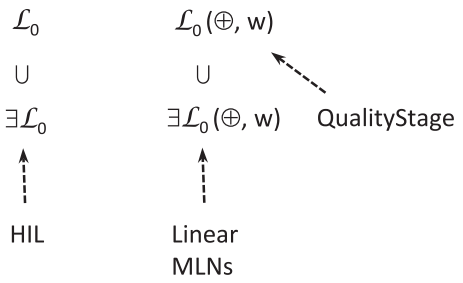

Fig. 1. A hierarchy of languages for entity linking.

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. We also point out that in our study, we have focused extensively on the set of certain links, which is the intersection of the links that appear in maximum-value solutions (or, alternatively, in maximum-probability worlds in the case of linear MLNs). Another semantics that is commonly used (especially in probabilistic approaches) is one where each link is generated together with its score (whether it is a probability or a value in our framework) as long as the score is above a threshold. The score itself gives insight into the degree of certainty (or confidence) for a link, especially when the link may not be a certain link in the aforementioned sense. An interesting future research question is to formally compare or combine the two semantics, one given by the certain links and the other given by scores and thresholds.

6.3. Hierarchy of Languages

We close this section with a pictorial view, shown in Figure 1, summarizing the various languages we discussed so far and their relationships. In the figure, we use the symbol $\subset$ to denote the "proper fragment" relationship between languages and use the dotted arrow to denote when one language is "captured" by another language (typically via a translation). The left-hand side of the figure is concerned with the maximal solutions semantics, while the right-hand side is concerned with the maximum-value solutions semantics and also with weights on the disjuncts.

In more concrete terms, for the maximal solutions semantics, we have that the language $\exists\mathcal{L}_\mathbf{0}$ is a proper fragment of the language $\mathcal{L}_\mathbf{0}$ in the precise sense established by Theorem 3.5. We also have that the HIL entity-linking fragment, which was discussed in Section 4.1, is captured by the $\exists\mathcal{L}_\mathbf{0}$ language in the precise sense established by Theorem 4.1. For the maximum-value semantics, we have that the language $\exists\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$ is a proper fragment of the language $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$. This can be easily verified by using the same proof of Theorem 3.5, along with the fact that every maximum-value solution in $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$ is a maximal solution. We also showed that linear MLNs are captured by $\exists\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$, as established by Theorem 6.2. Finally, we also argued earlier (Section 6.1) that the core functionality of QualityStage can be captured by specifications in $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$ (in fact, by specifications in $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$ that use neither conjunction nor existential quantification).

7. 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* [Bhattacharya and Getoor 2007], that is, the process of creating multiple types of links *together*.

The matching constraint for a link symbol L has the same form

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

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 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 $\mathcal{L}_\mathbf{2}(\oplus, \mathbf{w})$. Similarly to the case of $\exists\mathcal{L}_\mathbf{0}$, $\exists\mathcal{L}_\mathbf{0}(\oplus)$ and $\exists\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$, we write $\exists\mathcal{L}_\mathbf{1}$, $\exists\mathcal{L}_\mathbf{1}(\oplus)$, and $\exists\mathcal{L}_\mathbf{1}(\oplus, \mathbf{w})$ for the existential fragments of $\mathcal{L}_\mathbf{1}$, $\mathcal{L}_\mathbf{1}(\oplus)$, and $\mathcal{L}_\mathbf{1}(\oplus, \mathbf{w})$.

Example 7.1. 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: $\texttt{PaperLink}(pid, ano)$ and $\texttt{VenueLink}(venue, journal)$. 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 the following:

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

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

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., PaperLink[*pid*] $\subseteq$ 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 nonrecursive, 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.

7.1. 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 $\exists\mathcal{L}_\mathbf{1}$, the existential nonrecursive 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 earlier correspondence between $\exists\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$ and MLNs breaks when we go to the richer $\exists\mathcal{L}_\mathbf{1}(\oplus, \mathbf{w})$.

Our first result, for $\exists\mathcal{L}_\mathbf{1}(\oplus)$, 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 7.2. *There is a fixed entity-linking specification $\mathcal{E}$ in $\exists\mathcal{L}_\mathbf{1}(\oplus)$ 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?*

PROOF. We shall make use of the NP-completeness of the 3-colorability decision problem. The 3-colorability problem asks: given a graph $G(V, E)$, does there exist a function f such that for every $v \in V$, we have $f(v) \in \{r, b, g\}$, and for every $(u, v) \in E$, we have $f(u) \neq f(v)$? Without loss of generality, we shall assume that every node in G belongs to some edge.

The entity-linking specification $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ is defined as follows. There are five relations D, E, R, G, and B in the source schema $\mathbf{S}$. There are two links F and C in the link schema $\mathbf{L}$, and Σ consists of the matching constraint,

$$(m)\ \ F(x, y) \rightarrow \exists u \exists v (C(x, u) \wedge C(y, v) \wedge\ D(u, v) \wedge E(x, y)),$$

the functional dependency,

$$(f)\ \ C[1] \rightarrow C[2],$$

on the link relation C, and the following inclusion dependencies:

$$(i_1)\ C[1] \subseteq E[1] \qquad (i_2)\ C[2] \subseteq D[1]$$
$$(i_3)\ F[1] \subseteq E[1] \qquad (i_4)\ F[2] \subseteq E[2].$$

The functional dependency f states that the first attribute of C is a key. In addition, i_1 states that a value in the first attribute of C is a vertex, and i_2 states that a value in the second attribute of C is a color. The last two inclusion dependencies state that the values in the first or second attributes of F are vertices in E. The matching constraint m states that if there is pair in F, then that pair must be an edge in E such that the vertices of that edge are colored with different colors.

Given an instance of the 3-colorability problem, we can translate it, in polynomial time, into a source instance I as follows. First, D is a binary relation that contains all pairs of distinct colors of r, g, and b. In other words, $D = \{(r, g), (g, r), (b, g), (g, b), (r, b),$

$(b, r)\}$. The relation E is the edge relation from the graph G, and R, G, and B are unary relations that each contains a single tuple for r, g, and b, respectively.

We are now ready to show that G is 3-colorable if and only if there is a solution for I w.r.t. $\mathcal{E}$ of value at least $|V| + |E|$.

Suppose G is 3-colorable. Then let f be a 3-coloring function that maps each node of G to one of the colors r, g, or b so that for every edge $(x, y) \in E$, we have $f(x) \neq f(y)$. Define $(x, u) \in C$ if and only if $f(x) = u$ and define $(x, y) \in F$ if and only if $(x, y) \in E$. It is straightforward to see that C satisfies f, i_1, and i_2, and F satisfies i_3 and i_4. Furthermore, since every pair of adjacent nodes has different colors, the matching constraint m is also satisfied. Hence, C and F form a solution for I w.r.t. $\mathcal{E}$. Next, we show that the total value of this solution is at least $|V| + |E|$. Indeed, the value of all links in C is $|V|$, since every node is colored with one color. The value of all links in F is $|E|$, since F is the edge relation E and for every edge, there is only one instantiation for the existential variables of m. Hence, the total value of the solution is $|V| + |E|$.

For the converse, assume that there is a solution of value at least $|V| + |E|$; we will show that G is 3-colorable. First, observe that C can contribute at most $|V|$ to the value of the solution. Next, observe that F, through m, can contribute at most $|E|$ to the value of the solution since every tuple in F must be an edge in E and every value in the first component of C is associated with exactly one value in the second component of C. Since the solution has a value of at least $|V| + |E|$, it must be that C contributes $|V|$ and F contributes $|E|$ to the value of the solution. In other words, the relation C in the solution assigns a unique color (among r, g, b) to every node in G. Furthermore, since every edge in E exists in F in the solution and every pair of colors assigned to nodes of an edge must be different, we must have that the graph G is 3-colorable. □

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 $\exists\mathcal{L}_1(\oplus)$. The main result is that there is an entity-linking specification in $\exists\mathcal{L}_1(\oplus)$ for which no polynomial-time algorithms for recognizing certain and ambiguous links exist, unless NP = coNP.

By Theorem 7.2, there is an entity-linking specification $\mathcal{E}$ in $\exists\mathcal{L}_1(\oplus)$ 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 [2004] 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 intersection of a problem in NP and a problem in coNP; in particular, both NP and coNP are subclasses of DP (see also Papadimitriou [1994]).

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 Jonsson and Krokhin [2004] 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 this result, we are able to establish the intractability of recognizing certain and ambiguous links for entity-linking specifications in $\exists\mathcal{L}_1(\oplus)$.

THEOREM 7.3. *There is a fixed entity-linking specification $\mathcal{E}$ in $\exists\mathcal{L}_1(\oplus)$ 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 $\mathcal{E}$.*

PROOF. Let $\mathcal{E} = (\mathbf{L}, \mathbf{S}, \Sigma)$ be the following entity-linking specification:

- The link schema $\mathbf{L}$ consists of two binary link relations L_1 and L_2.
- The source schema $\mathbf{S}$ consists of two binary relations P and D, and three unary relations U, *Zero*, and *One*.
- The inclusion dependencies for L_1 are $L_1[1] \subseteq P[2]$ and $L_1[2] \subseteq U[1]$.
- The inclusion dependencies for L_2 are $L_2[1] \subseteq P[1]$ and $L_2[2] \subseteq P[2]$.
- There is one functional dependency for L_1, namely, $L_1[1] \rightarrow L_1[2]$.
- There are no functional dependencies for L_2.
- There is no matching constraint for L_1.
- There is one matching constraint for L_2:

$$\begin{aligned} L_2(c, x) \;\rightarrow\; & \exists y, z, v_1, v_2, v_3 (P(c, x) \wedge P(c, y) \wedge P(c, z) \wedge \\ & D(y, z) \wedge L_1(x, v_1) \wedge L_1(y, v_2) \wedge L_1(z, v_3) \wedge \\ & \mathit{One}(v_1) \wedge \mathit{Zero}(v_2) \wedge \mathit{Zero}(v_3)). \end{aligned}$$

Given a positive 3CNF-formula φ, we construct a source instance I_φ as follows:

- $P = \{(c, x) : c \text{ is a clause of } \varphi \text{ and } x \text{ is a variable of } \varphi\}$.
- $D = \{(x, y) : x, y \text{ are variables of } \varphi \text{ such that } x \neq y\}$.
- $U = \{0, 1\}$, $\mathit{Zero} = \{0\}$, $\mathit{One} = \{1\}$.

Suppose that φ has n variables and m clauses. We begin with two observations concerning the maximum-value solutions for I_φ w.r.t. $\mathcal{E}$. Assume that J is a maximum-value solution for I_φ w.r.t. $\mathcal{E}$. First, we claim that the link relation L_1 contains precisely n links. Indeed, the inclusion dependencies for L_1 and the functional dependency for L_1 imply that L_1 cannot have more than n members. Moreover, if L_1 had fewer than n members, then there exists a variable x of φ such that neither $(x, 0)$ nor $(x, 1)$ are member of L_1. But then we can augment L_1 with one of the pairs $(x, 0)$ and $(x, 1)$, and obtain a solution of bigger value, which contradicts the assumption that J is a maximum-value solution. Second, we claim that the value of every solution J for I_φ w.r.t. $\mathcal{E}$ is at most $n + 2m$. To see this, observe that each link in L_1 has value 1, while each link in L_2 has value 2. The latter claim follows from the fact that if $(c, x) \in L_2$, then there are exactly two pairs witnessing the existential quantifiers $\exists y$ and $\exists z$, while the quantifiers $\exists v_1$, $\exists v_2$, and $\exists v_3$ have unique witnesses. Using these two observations, it is easy to verify that the following statements are equivalent:

1. φ has a 1-in-3 satisfying truth assignment.
2. If J is a maximum-value solution for I_φ w.r.t. $\mathcal{E}$, then $\mathrm{Val}(J) = n + 2m$.
3. If J is a maximum-value solution for I_φ w.r.t. $\mathcal{E}$, then $\mathrm{Val}(J) \geq n + 2m$.

Moreover, there is a 1-1 correspondence between 1-in-3 satisfying truth assignments of φ and solutions of I_φ of value $n + 2m$. Specifically, if s is a 1-in-3 satisfying truth assignment s for φ, then we can construct a solution J for I_φ with $\mathrm{Val}(J_s) = n + 2m$ by populating L_1 with the pairs $(w, s(w))$, where w is a variable of φ, and by populating L_2 with the pairs (c, x) such that c is a clause of φ and x is the unique variable of c such that $s(x) = 1$. Vice versa, if J is a solution for I_φ with $\mathrm{Val}(J) = n + 2m$ (hence, a maximum-value solution), then the truth assignment s, where $s(x) = 1$ if and only if $(x, 1) \in L_1$, is a 1-in-3 satisfying truth assignment of φ.

We are now ready to focus on the certain links for I_φ w.r.t. the class of maximum-value solutions and $\mathcal{E}$. The following claim follows easily from the preceding discussion and the definitions.

Claim 1: The following statements are equivalent for φ and for a variable x of φ:

1. There is a 1-in-3 satisfying truth assignment for φ and the variable x is frozen.
2. There is a maximum-value solution J for I_φ with $\mathrm{Val}(J) \geq n + 2m$ and either the link $(x, 0)$ is certain or the link $(x, 1)$ is certain.

Assume now that there a polynomial-time algorithm for recognizing certain links w.r.t. the class of maximum-value solutions and $\mathcal{E}$. The preceding Claim 1 easily implies that the following problem is in NP: given a positive 3CNF-formula φ and a variable x, is it the case that there is a 1-in-3 satisfying truth assignment for φ and the variable x is frozen? However, as stated earlier, Jonsson and Krokhin [2004] showed that this problem is DP-complete. This implies that DP $\subseteq$ NP, which in turn implies that coNP $\subseteq$ NP, and hence NP = coNP. Thus, unless NP = coNP, there is no polynomial-time algorithm for recognizing certain links w.r.t. the class of maximum-value solutions and the entity-linking specification $\mathcal{E}$.

Next, we focus on the ambiguous links for I_φ w.r.t. the class of maximum-value solutions and $\mathcal{E}$. As observed earlier, for every variable x, we have that if J is a maximum-value solution for I_φ w.r.t. $\mathcal{E}$, then the link relation L_1 contains exactly one of the links $(x, 0)$ and $(x, 1)$. Therefore, we have that $(x, 1)$ is a certain link for I_φ w.r.t. $\mathcal{E}$ if and only if $(x, 0)$ is not a possible link for I_φ w.r.t. $\mathcal{E}$. Similarly, $(x, 0)$ is a certain link for I_φ w.r.t. $\mathcal{E}$ if and only if $(x, 1)$ is not a possible link for I_φ w.r.t. $\mathcal{E}$. Consequently, the following statements are equivalent for φ and for a variable x of φ:

1. There is no 1-in-3 satisfying truth assignment for φ or the variable x is not frozen.
2. There is no maximum-value solution J for I_φ with $\mathrm{Val}(J) \geq n + 2m$ or the link $(x, 0)$ is not certain and the link $(x, 1)$ is not certain.
3. There is no maximum-value solution J for I_φ with $\mathrm{Val}(J) \geq n + 2m$ or the link $(x, 1)$ is possible and the link $(x, 0)$ is possible.
4. There is no maximum-value solution J for I_φ with $\mathrm{Val}(J) \geq n + 2m$ or the link $(x, 1)$ is ambiguous and the link $(x, 0)$ is ambiguous.

Assume now that there a polynomial-time algorithm for recognizing ambiguous links w.r.t. the class of maximum-value solutions and $\mathcal{E}$. The preceding discussion implies that the following problem is in coNP: given a positive 3CNF-formula φ and a variable x, is it the case that there is no 1-in-3 satisfying truth assignment for φ or the variable x is not frozen? Since this problem is coDP-complete, it follows that coDP $\subseteq$ coNP, which in turn implies that NP $\subseteq$ coNP, and hence NP = coNP. Thus, unless NP = coNP,

there is no polynomial-time algorithm for recognizing ambiguous links w.r.t. the class of maximum-value solutions and the entity-linking specification $\mathcal{E}$.

This completes the proof of intractability of recognizing certain links and ambiguous links w.r.t. the class of maximum-value solutions and entity-linking specifications in $\exists\mathcal{L}_\mathbf{1}(\oplus)$. □

While these complexity results show intractability for $\exists\mathcal{L}_\mathbf{1}(\oplus)$, and hence for $\mathcal{L}_\mathbf{1}(\oplus)$ as well, the next theorem identifies a fragment of $\mathcal{L}_\mathbf{1}(\oplus)$ for which the same problems become tractable. Moreover, if any of the conditions defining this fragment are removed, then we fall back into intractability. We say that an entity-linking specification is *two-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 7.4. *Assume that the entity-linking specification is two-level hierarchical. Assume also the following three conditions:*

(*1*) *The top-level links have no FDs.*
(*2*) *There are no universal quantifiers in the right-hand side of 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?

Because of its length, we defer the proof of Theorem 7.4 until the next subsection, to avoid breaking the flow. As an immediate application of this theorem, recall the entity-linking specification in $\mathcal{L}_\mathbf{2}$ for `VenueLink` and `PaperLink` in Example 7.1, and the entity-linking specification in $\mathcal{L}_\mathbf{1}$ obtained from it by the modification described in the same Example 7.1. The entity-linking specification in $\mathcal{L}_\mathbf{1}$, where `VenueLink` is the top-level link and `PaperLink` is the bottom-level link, satisfies the assumptions of Theorem 7.4, 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 $\mathcal{L}_\mathbf{2}$ for this example enjoys the desirable properties.

We close this section by considering the weighted versions $\mathcal{L}_\mathbf{1}(\oplus, \mathbf{w})$ and $\mathcal{L}_\mathbf{2}(\oplus, \mathbf{w})$. Theorem 6.2 shows a precise correspondence between a linear MLN and its corresponding entity-linking specification in $\exists\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$. Does this correspondence carry over to $\exists\mathcal{L}_\mathbf{1}(\oplus, \mathbf{w})$ or $\exists\mathcal{L}_\mathbf{2}(\oplus, \mathbf{w})$? 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. The next theorem says that the analog of Theorem 6.2 fails. Thus, the two frameworks, one based on deterministic entity-linking specifications, the other based on probabilistic Markov Logic Networks, diverge when allowing for interdependencies among links.

THEOREM 7.5. *There is an extended linear MLN and a source instance I such that if $\mathcal{E}$ is the corresponding entity-linking specification in $\exists\mathcal{L}_1(\oplus, \mathbf{w})$, then Max Value Solutions$_{\mathcal{E}}(I)$ and Max Probability Worlds$_{\mathcal{M}}(I)$ are disjoint.*

PROOF. Let $\mathcal{M}$ consist only of $L_1(x, y) \rightarrow L_2(x, y)$, with weight 1. Then the corresponding entity-linking specification in $\exists\mathcal{L}_1(\oplus, \mathbf{w})$ has the formula $L_2(x, y) \rightarrow L_1(x, y)$, with weight 1. Assume that L_1 has no FDs, and L_2 has both FDs. Let $I = \{R(0), R(1)\}$, and let the inclusion dependencies say that both components of each of L_1 and L_2 are contained in R. A necessary and sufficient condition for an instance J that satisfies the inclusion dependencies and FDs to be a maximum-probability world is

$$L_1^J \subseteq L_2^J. \tag{8}$$

This is because in this example, the probability is the highest possible if and only if for every assignment to the variables, the member of $\mathcal{M}$ holds, which happens precisely if inclusion (8) holds. But it is straightforward to see that the maximum-value solutions J all have L_2^J with two tuples (a maximum matching) and L_1^J with all four possible tuples consistent with the inclusion dependencies. Thus, no maximum-value solution satisfies inclusion (8), whereas every maximum-probability world does satisfy inclusion (8). This proves the theorem. □

7.2. Proof of Theorem 7.4

For readability, we break this proof into parts.

Finding a single maximum-value solution: We first give a polynomial-time algorithm for finding a single maximum-value solution. Then we shall show how to use the ideas in the algorithm to obtain a polynomial-delay algorithm for enumerating the maximum-value solutions.

It follows from our assumptions that each top-level link, link_2, can have only one matching constraint, of the form

$$\text{link}_2(x, y) \rightarrow \alpha_1(x, y) \oplus \cdots \oplus \alpha_k(x, y), \tag{9}$$

where each $\alpha_i(x, y)$ is existentially quantified and contains at most one bottom-level link occurring in it. In particular, the matching constraint for link_2 has no universal quantifiers.

We shall deal separately for now with each bottom-level link and each top-level link. Let link_1 be a bottom-level link, and let link_2 be a top-level link. Let R_{link_1} be the set of all pairs (a, b) that satisfy the inclusion dependencies for link_1, and that satisfy the right-hand side of the matching constraint for link_1 when the left-hand side is instantiated by $\text{link}_1(a, b)$ (so that (a, b) is a candidate tuple for the link_1 relation). We think of R_{link_1} as a weighted graph, where the weight of the edge (a, b) is the value assigned to $\text{link}_1(a, b)$ according to our conventions. For each pair (a, b) in R_{link_1}, define the *discretionary link$_1$ value of* (a, b) to be the value that would be obtained from the link_1 matching constraint alone if (a, b) were in the link_1 relation of a solution. Let T_{link_2} be the set of candidates for the link_2 relation (because of the inclusion dependencies). For each pair (c, d) in T_{link_2} and each bottom-level link (link_1) and each (a, b) in R_{link_1}, define the *discretionary link$_2$ value of* (c, d) *due to link$_1$*(a, b) to be the incremental value for $\text{link}_2(c, d)$ that would be caused by the presence of (a, b) in the link_1 relation. This is obtained by adding the values that would be caused by disjuncts α_i of matching constraint (9) that contain $\text{link}_1(w, z)$ for some variables w,z, when (w, z) is instantiated by (a, b). For each pair (c, d) in T_{link_2}, define the *nondiscretionary link$_2$ value of* (c, d) to be 1 if there is no matching constraint for link_2, and otherwise the incremental weight for $\text{link}_2(c, d)$ that would be caused by the disjuncts α_i of matching constraint (9) that do not contain any bottom-level link. Note that the total value of a solution J

is the sum of (1) the sum over all bottom-level links, link_1, and all (a, b) in the link_1 relation of J of the discretionary link_1 value of (a, b); (2) the sum over all bottom-level links, link_1, and top-level links, link_2, and all (a, b) in the link_1 relation of J and all (c, d) in the link_2 relation of J of the discretionary link_2 value (c, d) due to $\text{link}_1(a, b)$; and (3) the sum over all top-level links, link_2, and all (c, d) in the link_2 relation of J of the nondiscretionary link_2 value of (c, d). Since every solution J has the same value for (3), our goal in finding a maximum-value solution is to maximize the sum of (1) and (2).

Let us do bookkeeping by allocating to each (a, b) in R_{link_1} not only the discretionary link_1 value of (a, b) but also the discretionary link_2 value of (c, d) due to $\text{link}_1(a, b)$ for every top-level link, link_2, and every (c, d) in T_{link_2}. Thus, for each bottom-level link, link_1, each (a, b) in R_{link_1}, define $f_{\text{link}_1}(a, b)$ to be the result of adding the discretionary link_1 value of (a, b) and the sum over all top-level links, link_2, and all (c, d) in T_{link_2} of the discretionary link_2 value of (c, d) due to $\text{link}_1(a, b)$. Intuitively, $f_{\text{link}_1}(a, b)$ is the incremental value of a maximum-value solution caused by the presence of (a, b) in the link_1 relation.

It follows that the maximum-value solutions are exactly those solutions J where for each bottom-level link, link_1, the link_1 relation of J is chosen to maximize the sum of $f_{\text{link}_1}(a, b)$ over all (a, b) in the link_1 relation of J, and where the link_2 relation of J is chosen to consist of all (c, d) such that either the nondiscretionary link_2 value of (c, d) is positive or there is a bottom-level link, link_1 and there is (a, b) in the link_1 relation of J such that the discretionary link_2 value of (c, d) due to $\text{link}_1(a, b)$ is positive. This is because there is no functional dependency for link_2.

So we need only describe how to obtain a link_1 relation that maximizes the sum of $f_{\text{link}_1}(a, b)$ over all (a, b) in link_1. This is just as in the proof of Theorem 5.4, where there are three possibilities, depending on whether link_1 has two FDs, one FD, or no FDs. This completes the description of the polynomial-time algorithm to compute a single maximum-value solution.

A polynomial-delay algorithm: We now discuss how to use the ideas in the algorithm to obtain a polynomial-delay algorithm for enumerating the maximum-value solutions. For simplicity of description, we assume first that there is only one top-level link, link_2, and only one bottom-level link, link_1. Later we shall consider the more general case.

We make use of the polynomial-delay algorithm as described in the proof of Theorem 5.4 for enumerating the link_1 relation in maximum-value solutions, and as each such link_1 relation is generated, we then generate the corresponding link_2 relation, as described in the polynomial-time algorithm we gave. This completes the proof that there is a polynomial-delay algorithm for enumerating the maximum-value solutions when there is only one top-level link and only one bottom-level link. When there are possibly multiple top-level links and/or bottom-level links, we modify the polynomial-delay algorithm very similarly to the modification used in the proof of Theorem 5.4.

Finding the certain and ambiguous links: We now show that there are polynomial-time algorithms for computing the certain links and the ambiguous links.

To determine the certain and ambiguous links, we first determine, for each bottom-level link, link_1, the certain and ambiguous links for link_1 just as we do in the proof of Theorem 5.4.

We now find the certain and ambiguous top-level links. It is convenient to find the possible links instead of the ambiguous links, and we can then obtain the ambiguous links as those that are possible but not certain.

Assume that the matching constraint for the top-level link, link_2, is matching constraint (9) in the first part of the proof, and we are trying to decide if $\text{link}_2(c, d)$ is a

certain or possible link, where (c, d) is playing the role of (x, y) in matching constraint (9). There are two cases.

Case 1: There is i such that α_i has no bottom-level link occurring in it (and so α_i refers only to base relations) and $\alpha_i(c, d)$ holds for the base relations. Then $\text{link}_2(c, d)$ is a certain link.

Case 2: There is no such i as described in Case 1. Let I_{link_1} be the set of all indices i such that α_i contains an occurrence of the bottom-level link, link_1. For each i in I_{link_1}, we now define a set $B^{(c,d)}_{\text{link}_1,i}$ that consists of some formulas of the form $\text{link}_1(a, b)$. Assume that the occurrence of link_1 in α_i is $\text{link}_1(x_1, x_2)$. Recall that the left-hand side of matching constraint (9) is $\text{link}_2(x, y)$. For each mapping μ that maps x to c and maps y to d, and that maps the existentially quantified variables in α_i to values in the domain determined by the inclusion dependencies, we do the following. For every base relation atom $Q(z_1, \ldots, z_k)$ that appears in α_i, we check whether $Q(\mu(z_1), \ldots, \mu(z_k))$ holds. If this is true for every base relation atom $Q(z_1, \ldots, z_k)$ that appears in α_i, then we put $\text{link}_1(\mu(x_1), \mu(x_2))$ into $B^{(c,d)}_{\text{link}_1,i}$. Intuitively, $B^{(c,d)}_{\text{link}_1,i}$ is the set of all possible formulas $\text{link}_1(a, b)$ that can make α_i hold, when the left-hand side of matching constraint (9) is instantiated as $\text{link}_2(c, d)$. Let $B^{(c,d)}_{\text{link}_1}$ be the union of $B^{(c,d)}_{\text{link}_1,i}$ over all i in I_{link_1}. Intuitively, $B^{(c,d)}_{\text{link}_1}$ is the set of all formulas $\text{link}_1(a, b)$ that can make the right-hand side of matching constraint (9) hold in a maximum-value solution, when the left-hand side of matching constraint (9) is instantiated as $\text{link}_2(c, d)$. Let $B^{(c.d)}$ be the union of $B^{(c,d)}_{\text{link}_1}$ over all bottom-level links, link_1. Intuitively, $B^{(c.d)}$ is the set of all formulas $\text{link}_1(a, b)$ for bottom-level links, link_1, that can make the right-hand side of matching constraint (9) hold in a maximum-value solution, when the left-hand side of matching constraint (9) is instantiated as $\text{link}_2(c, d)$. Now there is only a polynomial number of choices for μ as earlier, where the exponent is the number of existentially quantified variables in α_i. Hence, there is a polynomial-time algorithm to construct $B^{(c.d)}$.

It is not hard to see that $\text{link}_2(c, d)$ is a possible link precisely if either we are in Case 1 or we are in Case 2 and there is a member of $B^{(c.d)}$ that is a possible bottom-value link (i.e., there is a bottom-level link, link_1, such that the member of $B^{(c.d)}$ is $\text{link}_1(a, b)$, and such that $\text{link}_1(a, b)$ is a possible link). From this characterization of the possible links for link_2, and from the fact that we have a polynomial-time algorithm to construct $B^{(c.d)}$, it follows that there is a polynomial-time algorithm for finding possible links for the top-level links.

We now show that $\text{link}_2(c, d)$ is a certain link precisely if either (1) we are in Case 1, or (2) we are in Case 2 and for every maximum-value solution S for the bottom-level links alone, there is a member of $B^{(c.d)}$ that is a fact of S (note the order of quantification). To see this, assume first that (1) or (2) holds; we must show that $\text{link}_2(c, d)$ is a certain link. Clearly if (1) holds, then $\text{link}_2(c, d)$ is a certain link. Assume now that (2) holds. We now use the fact that the maximum-value solutions for the bottom-level links and the top-level links together are precisely those obtained from a maximum-value solution for the bottom-level links and then putting everything possible into the top-level links (i.e., by putting (c, d) into the link_2 relation if the right-hand side of matching constraint (9) holds when c and d play the role of x and y). It therefore follows that if (2) holds, then $\text{link}_2(c, d)$ is a certain link.

Assume now that $\text{link}_2(c, d)$ is a certain link; we must show that (1) or (2) holds. Assume that (1) and (2) fail. Since (1) fails, we are in Case 2, and since (2) fails, there is a maximum-value solution S for the bottom-level links such that no member of $B^{(c.d)}$ is a fact of S. We use S to create a maximum-value solution for all links together as before. In this solution, we see that the right-hand side of matching constraint (9) fails,

so $\text{link}_2(c, d)$ is not in this maximum-value solution, which contradicts our assumption that $\text{link}_2(c, d)$ is a certain link.

The only problematic issue in giving a polynomial-time algorithm for finding certain links for link_2 is whether there is a polynomial-time algorithm for deciding whether, for every maximum-value solution S for the bottom-level links, there is a member of $B^{(c,d)}$ that is a fact of S. We now show that this question is equivalent to the question of asking whether there is a bottom-level link, link_1, such that for every maximum-value solution S_{link_1} for link_1, there is a member of $B^{(c,d)}_{\text{link}_1}$ that is a fact of S_{link_1}. It is straightforward to see that the latter implies the former. The former also implies the latter, because if the latter fails, then for each bottom-level link, link_1, let S_{link_1} be a maximum-value solution for link_1 that does not contain any member of $B^{(c,d)}_{\text{link}_1}$. Then the union of S_{link_1} over all bottom-level links, link_1, is a maximum-value solution S for the bottom-level links that does not contain any member of $B^{(c,d)}$.

So we need only show that there is a polynomial-time algorithm that, for a fixed bottom-level link, link_1, decides whether for every maximum-value solution S_{link_1} for link_1, there is a member of $B^{(c,d)}_{\text{link}_1}$ that is a fact of S_{link_1}. To decide this, we let R'_{link_1} be the result of removing all of the members of $B^{(c,d)}_{\text{link}_1}$ from R_{link_1}, and as before taking the weight of each remaining edge (a, b) as $f_{\text{link}_1}(a, b)$. We then compute a maximum weighted matching of R'_{link_1}, and call its weight M'. Let M be the weight of a maximum weighted matching of R_{link_1}. It is easy to see that $M' < M$ if and only if for every maximum-value solution S_{link_1} for link_1, there is a member of $B^{(c,d)}_{\text{link}_1}$ that is a fact of S_{link_1}.

NP-hardness if assumption (1) of the theorem fails: We now give an example where the top-level links have FDs, but all of the other assumptions hold, and where it is NP-hard to decide if there is a solution of at least a given value.

We shall make use of the NP-hardness of perfect 3-dimensional matching (one of Karp's original NP-complete problems). This problem asks the following. Assume that we are given a ternary relation P with the same number n of distinct elements in each column. Is there a subrelation Q of P that is a perfect matching, in the sense that Q contains n tuples, no two of which agree on any attribute (coordinate)? It is easy to see that this is equivalent to asking that the projection of Q on the first two attributes is a perfect matching for the projection of P on the first two attributes, and similarly for the last two attributes.

Here is our set of matching constraints where we will show that it is NP-hard to decide if there is a solution of at least a given value:

$$\text{link}_2(y, z) \rightarrow \exists x(\text{link}_1(x, y) \wedge P(x, y, z))$$
$$\text{link}_1(x, y) \rightarrow R(x, y).$$

We assume that both FDs hold for both link_1 and link_2. Note that the value for $\text{link}_2(y, z)$ is at most 1 because of the FDs for link_1, and the value for $\text{link}_1(x, y)$ is at most 1. We take the inclusion dependencies that say that the first argument of link_1 is in the first column of P, the second argument of link_1 and the first argument of link_2 are in the second column of P, and the second argument of link_2 is in the third column of P.

Given a ternary relation P that we want a perfect 3-dimensional matching for, where each attribute has the same number n of distinct elements, we take R to be the binary relation that is the projection of P onto the first two attributes. Note that we have the matching constraint for link_1 as it is rather than $\text{link}_1(x, y) \rightarrow \exists z P(x, y, z)$, so that the value for $\text{link}_1(x, y)$ is at most 1. So because of the FDs, the total value is at most $2n$.

We now show that there is a solution with value at least $2n$ if and only if P has a perfect 3-dimensional matching. If P has a perfect 3-dimensional matching Q, then let link_1 be the projection of Q on the first two attributes, and let link_2 be the projection of Q on the last two attributes. Then the value for this solution is $2n$.

Conversely, if the value is at least $2n$ (and hence $2n$, since $2n$ is the maximum possible value), then let $Q = \{(x, y, z) : \text{link}_1(x, y) \text{ and } \text{link}_2(y, z)\}$. Let P_{12} be the projection of P onto its first two attributes, and let P_{23} be the projection of P onto its last two attributes. Similarly, let Q_{12} be the projection of Q onto its first two attributes, and let Q_{23} be the projection of Q onto its last two attributes. Since the total value is $2n$, the value for each of link_1 and link_2 is n, so link_1 is a perfect matching for P_{12}, and link_2 is a perfect matching for P_{23}. We now show that $Q_{12} = \text{link}_1$ (a similar argument shows that $Q_{23} = \text{link}_2$). By definition of Q, we know that $Q_{12} \subseteq \text{link}_1$. Assume now that $\text{link}_1(x, y)$ is a link; we must show that $Q_{12}(x, y)$ holds. Since link_2 is a perfect matching for P_{12}, there is z such that $\text{link}_2(y, z)$ is a link. Since $\text{link}_1(x, y)$ and $\text{link}_2(y, z)$ are links, it follows by definition of Q that $Q(x, y, z)$ holds, and so $Q_{12}(x, y)$ holds, as desired. Hence, Q_{12} is a perfect matching for P_{12}, and Q_{23} is a perfect matching for P_{23}. Further, $Q \subseteq P$, as we now show. Assume that $Q(x, y, z)$ holds, and so $\text{link}_1(x, y)$ and $\text{link}_2(y, z)$ are links. Since $\text{link}_2(y, z)$ is a link, by the matching constraint for link_2, we know that there is x' such that $\text{link}_1(x', y)$ is a link and $P(x', y, z)$ holds. Since $\text{link}_1(x, y)$ and $\text{link}_1(x', y)$ are links, it follows by the FDs on link_1 that $x' = x$; hence, $P(x, y, z)$ holds. It follows that Q is a perfect 3-dimensional matching for P.

NP-hardness if assumption (2) of the theorem fails: We now give an example where there are universal quantifiers for the top-level links, and where it is NP-hard to decide if there is a solution of at least a given value. There is a single matching constraint:

$$\begin{aligned}\text{link}_2(u, w) \rightarrow (\forall c(C(c) \rightarrow \\ \exists v \exists t(T(t) \wedge R_+(c, v) \wedge \text{link}_1(v, t)) \\ \oplus \\ \exists v \exists t(F(t) \wedge R_-(c, v) \wedge \text{link}_1(v, t))).\end{aligned}$$

We shall make use of the NP-hardness of SAT. Assume that we are given an SAT problem. Assume for convenience that the clauses and (positive) propositional atoms are numbered. Define the relation C to consist of all indices of clauses, and the relation A to consist of all indices of atoms. Define the relation R_+ to consist of those tuples (c, v) where the atom v appears positively in the clause c, and define the relation R_- to consist of those tuples (c, v) where the atom v appears negatively in the clause c. Define the relation T to consist of the single tuple (1) ("true"), the relation F to consist of the single tuple (0) ("false"), the relation TF to consist of the two tuples (0) and (1), and the relation D to consist of the single tuple $(0, 0)$. We have the FD that says that the first attribute of link_1 functionally determines the second attribute of link_1 (so that, intuitively, each atom is assigned at most one truth value). We have the inclusion dependencies that say that each projection of link_2 onto one variable is a subset of the corresponding column of D. We also have the inclusion dependencies that say that the first argument of link_1 is a subset of the only column of A, and the second argument of link_1 is a subset of the only column of TF.

It is not hard to see that every maximum-value solution assigns a truth value to each atom (i.e., more formally, that in a maximum-value solution, the projection of link_1 on the first column consists of the indices of all propositional atoms). Intuitively, it does not hurt to assign values to all of the atoms. Let N be the number of atoms. We then have that if the propositional formula is not satisfiable, then the value of every maximum-value solution is N (since every link_1 link is assigned value 1, and there is

no link_2 link). However, if the propositional formula is satisfiable, then the maximum-value solution has a value strictly greater than N, because of the additional link_2 value. So there is a solution with value at least $N + 1$ if and only if the propositional formula is satisfied.

If the reader is concerned that u and w do not appear in the right-hand side of the matching constraint, we can add a new disjunct $D(u, w)$. Or we can simply add $D(u, w)$ as a conjunct to each of the disjuncts.

NP-hardness if assumption (3) of the theorem fails: This follows from the proof of Theorem 7.2.

This concludes the proof of Theorem 7.4.

8. RELATED WORK

As mentioned in the introduction, there has been extensive work on entity resolution; for overviews, see the surveys of Fan and Geerts [2012] and Ganti and Sarma [2013] and the tutorial of Getoor and Machanavajjhala [2012]. We have also made connections to existing probabilistic approaches for entity resolution, and to the HIL language [Hernández et al. 2013]. We now comment briefly on other declarative approaches to entity resolution.

Our framework is more declarative than prior approaches, such as Dedupalog [Arasu et al. 2009], in the following precise sense. In our framework, a user declares the constraints for the links, and then the semantics is based on instances that are *guaranteed* to satisfy the constraints. In particular, the solutions (both maximal and maximum value) always satisfy the constraints, and the certain links are the links that appear in all such solutions. Furthermore, in the case of $\mathcal{L}_\mathbf{0}$ (and its variations $\mathcal{L}_\mathbf{0}(\oplus)$ and $\mathcal{L}_\mathbf{0}(\oplus, \mathbf{w})$), the set of certain links itself forms a solution; in other words, the certain links, when taken as an instance, satisfy the given constraints. Dedupalog is also based on a high-level specification given as Datalog-style constraints. However, by design, the result of Dedupalog may violate some of the constraints (the *soft* constraints). In effect, the Dedupalog specification is only a guideline for the implementation, which is an algorithm that attempts to minimize the number of violations of soft constraints and, as a result, makes its own choices of which links will survive and which will not. Thus, a significant part of the semantics of Dedupalog is hidden in the algorithm.

An early argument in favor of using link-centric constraints to specify links in a declarative manner appeared in Alexe et al. [2013], but no formal language, semantics, or algorithms were given there. The language LinQL [Hassanzadeh et al. 2009] 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 Fan [2008] to enforce equality on attribute values based on matching conditions. In effect, MDs are source-to-link constraints that may lead to modifications of the source relations. MDs have been given operational semantics in Bertossi et al. [2013] 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.

Blocking is a technique that is widely used in practice to avoid unnecessary comparisons between records (see Christen [2012] for a survey). Blocking amounts to preventing records (or entities) from being compared to each other unless they fall in the same block, where a block is typically defined by a key based on one or more of the available attributes. For example, if the blocking key is given by the first three digits of the zip code, then two records will be compared only if the first three digits of the zip codes are the same. It should be pointed out that such blocking conditions can be easily incorporated in our framework as conjunctions of equality predicates that guard all the disjuncts in a matching constraint. More precisely, we can write a disjunctive

matching constraint of the form

$$L(x, y) \rightarrow \forall \mathbf{u}(\psi(x, y, \mathbf{u}) \rightarrow \exists \mathbf{z_1}(B_1 \wedge \phi_1) \vee \ldots \vee \exists \mathbf{z_k}(B_1 \wedge \phi_k)),$$

where B_1 is a conjunction of equality predicates that represents a blocking condition. In effect, this prevents any links from being generated unless the condition B_1 is satisfied. We also note that multiple blocking conditions $B_1, B_2, \ldots$ can be incorporated by repeating the previous disjunction pattern for each B_i. This captures a "union" of blocking conditions, which is also widely used in practice.

9. 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.

REFERENCES

N. Alur, A. K. Jha, B. Rosen, and T. Skov. 2008. IBM WebSphere QualityStage Methodologies, Standardization, and Matching. Redbooks. http://www.redbooks.ibm.com/redbooks/pdfs/sg247546.pdf.

B. Alexe, D. Burdick, M. A. Hernández, G. Koutrika, R. Krishnamurthy, L. Popa, I. R. Stanoi, and R. Wisnesky. 2013. 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*. 36–55.

A. Arasu, C. Re, and D. Suciu. 2009. Large-scale deduplication with constraints using dedupalog. In *ICDE*. 952–963.

M. Arenas, P. Barceló, R. Fagin, and L. Libkin. 2013. Solutions and query rewriting in data exchange. *Inf. Comp.* 228 (2013), 28–51.

M. Arenas, L. E. Bertossi, and J. Chomicki. 1999. Consistent query answers in inconsistent databases. In *PODS*. 68–79.

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

I. Bhattacharya and L. Getoor. 2007. Collective entity resolution in relational data. *TKDD* 1, 1 (2007).

D. Burdick, R. Fagin, Ph. G. Kolaitis, L. Popa, and W.-C. Tan. 2015. A declarative framework for linking entities. In *18th International Conference on Database Theory (ICDT'15)*. 25–43.

D. Burdick, M. A. Hernández, H. Ho, G. Koutrika, R. Krishnamurthy, L. Popa, I. R. Stanoi, S. Vaithyanathan, and S. Das. 2011. Extracting, linking and integrating data from public sources: A financial case study. *IEEE Data Eng. Bull.* 34, 3 (2011), 60–67.

C. R. Chegireddy and H. W. Hamacher. 1987. Algorithms for finding k-best perfect matchings. *Discrete Appl. Math.* 18, 2 (1987), 155–165.

J. Chomicki and J. Marcinkowski. 2005. Minimal-change integrity maintenance using tuple deletions. *Inf. Comp.* 197 (2005), 90–121.

P. Christen. 2012. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Trans. Knowl. Data Eng.* 24, 9 (2012), 1537–1555.

X. Dong, A. Y. Halevy, and J. Madhavan. 2005. Reference reconciliation in complex information spaces. In *SIGMOD*. 85–96.

A. Droschinsky, B. Heinemann, N. Kriege, and P. Mutzel. 2014. Enumeration of maximum common subtree isomorphisms with polynomial-delay. In *Proceedings of Algorithms and Computation - 25th International Symposium, (ISAAC'14)*. 81–93.

A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. 2007. Duplicate record detection: A survey. *IEEE TKDE* 19, 1 (2007), 1–16.

R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci. (TCS)* 336, 1 (2005), 89–124.

W. Fan. 2008. Dependencies revisited for improving data quality. In *PODS*. 159–170.

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

I. P. Fellegi and A. B. Sunter. 1969. A theory for record linkage. *J. Am. Statistical Assoc.* 64, 328 (1969), 1183–1210.

H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. 2001. Declarative data cleaning: Language, model, and algorithms. In *VLDB*. 371–380.

V. Ganti and A. Das Sarma. 2013. *Data Cleaning: A Practical Perspective*. Morgan & Claypool Publishers.

L. Getoor and A. Machanavajjhala. 2012. Entity resolution: Theory, practice & open challenges. *PVLDB* 5, 12 (2012), 2018–2019.

O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang. 2009. A framework for semantic link discovery over relational data. In *CIKM*. 1027–1036.

M. A. Hernández, G. Koutrika, R. Krishnamurthy, L. Popa, and R. Wisnesky. 2013. HIL: A high-level scripting language for entity integration. In *EDBT*. 549–560.

M. A. Hernández and S. J. Stolfo. 1995. The merge/purge problem for large databases. In *SIGMOD*. 127–138.

A. Itai, M. Rodeh, and S. L. Tanimoto. 1978. Some matching problems for bipartite graphs. *J. ACM* 25, 4 (1978), 517–525.

D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. 1988. On generating all maximal independent sets. *Inf. Process. Lett.* 27, 3 (1988), 119–123.

P. Jonsson and A. A. Krokhin. 2004. Recognizing frozen variables in constraint Satisfaction Problems. *Theor. Comput. Sci. (TCS)* 329, 1–3 (2004), 93–113.

N. Koudas, S. Sarawagi, and D. Srivastava. 2006. Record linkage: Similarity measures and algorithms. In *SIGMOD*. 802–803.

K. G. Murty. 1968. An algorithm for ranking all the assignments in order of increasing cost. *Oper. Res.* 16, 3 (1968), 682–687.

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

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learn.* 62, 1–2 (2006), 107–136.

P. Singla and P. Domingos. 2006. Entity resolution with Markov logic. In *ICDM*. 572–582.

Received August 2015; revised December 2015; accepted February 2016