



ELSEVIER

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs



A theoretical framework for knowledge-based entity resolution



Klaus-Dieter Schewe^{a,1}, Qing Wang^{b,*}

^a Software Competence Center Hagenberg and Johannes-Kepler-University Linz, Austria

^b Research School of Computer Science, The Australian National University, ACT 0200, Australia

ARTICLE INFO

Article history:

Received 10 June 2012

Received in revised form 1 October 2013

Accepted 1 October 2013

Available online 27 June 2014

Communicated by W. Fan

Keywords:

Entity resolution

Data matching

Record linkage

Query evaluation

Query containment

Knowledge representation

Knowledge optimization

ABSTRACT

Entity resolution is the process of determining whether a collection of entity representations refer to the same entity in the real world. In this paper we introduce a theoretical framework that supports knowledge-based entity resolution. From a logical point of view, the expressive power of the framework is equivalent to a decidable fragment of first-order logic including conjunction, disjunction and a certain form of negation. Although the framework is expressive for representing knowledge about entity resolution in a collective way, the questions that arise are: (1) how efficiently can knowledge patterns be processed; (2) how effectively can redundancy among knowledge patterns be eliminated. In answering these questions, we first study the evaluation problem for knowledge patterns. Our results show that this problem is NP-complete w.r.t. combined complexity but in PTIME w.r.t. data complexity. This nice property leads us to investigate the containment problem for knowledge patterns, which turns out to be NP-complete. We further develop a notion of optimality for knowledge patterns and a mechanism of optimizing a knowledge model (i.e. a finite set of knowledge patterns). We prove that the optimality decision problem for knowledge patterns is still NP-complete.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Entity resolution is one of the major impediments affecting data quality provided by information systems. The difficulty of this problem has been widely acknowledged by research communities [10,16,21] and industry practitioners [1,37,38]. State-of-the-art approaches to entity resolution favor similarity-based methods [14]. Numerous classification techniques have been developed under a variety of perspectives such as probabilistic [21,26], cost-based [42], ruled-based [15], supervised [23], active learning [5,30,39], and collective classifications [10,16]. A common rationale behind similarity-based methods is that, the more similar two entities are, the more likely they refer to the same real-world object. However, since entities that look similar may refer to different objects, and conversely entities that look different may refer to the same objects, similarity-based methods are far from perfect. Such problems become more evident when the information about entities is inadequate. Imagine that for two entities e_1 and e_2 it is only known that they have the same name, how can we decide whether or not e_1 and e_2 refer to the same real-world object? [Example 1.1](#) shows that, as resolving entities based on similarity is

* Corresponding author.

E-mail addresses: kd.schewe@scch.at, kd.schewe@faw.at (K.-D. Schewe), qing.wang@anu.edu.au (Q. Wang).

¹ The research reported in this paper was supported by the European Fund for Regional Development (DAABAA_00121) as well as the State of Upper Austria for the project *Vertical Model Integration* within the program "Regionale Wettbewerbsfähigkeit Oberösterreich 2007–2013" (Wi-219205).

| AID | NAME | AFFILIATION | EMAIL |
|-----|-----------|---|--------------------------------|
| 1 | Q. Wang | PBRF Office, University of Otago | qing.wang@otago.ac.nz |
| 2 | Qing Wang | Dept. of Information Science, University of Otago | qwang@infoscience.otago.ac.nz |
| 3 | Qing Wang | RSCS, Australian National University | qing.wang@anu.edu.au |
| 4 | Q. Wang | CAU Kiel, Germany | wang@is.informatik.uni-kiel.de |
| 5 | Q. Wang | Dept. of Information Systems, Massey University | q.q.wang@massey.ac.nz |

Fig. 1. Sample records in AUTHOR.

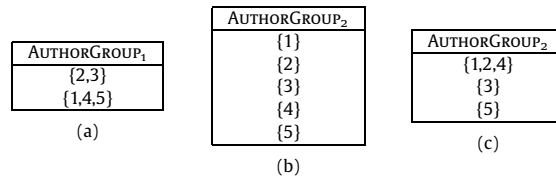


Fig. 2. Identifying authors.

hardly ever completely accurate, we should care for the possibility to revise decisions on entity resolution whenever more knowledge becomes available.

Example 1.1. Consider the relation AUTHOR in Fig. 1 and the question “which of these authors refer to the same person in the real world”. To answer this question, traditional similarity-based methods would use certain techniques to measure the similarity between authors in AUTHOR and then group them based on their similarity. For instance, authors may be grouped based on the similarity of their names as shown in Fig. 2(a). Since authors with similar names could be different persons, and on the other hand authors with different names could be the same person, we do not actually know whether the result presented in Fig. 2(a) is accurate.

Suppose that over time we gradually acquire the following knowledge from other sources:

- (K1) *Qing Wang worked at both the PBRF Office and the Department of Information Science, University of Otago;*
 (K2) *Q. Wang studied at the CAU Kiel, Germany before joining the Department of Information Science, University of Otago.*

By K1, we know that the authors 1 and 2 refer to the same person. Similarly, by K2 we know that the authors 2 and 4 refer to the same person. Hence, the result presented in Fig. 2(a) is incorrect. Considering that the main reason for this problem is that Qing Wang is a very common name used in Asian countries, we revise the previous name-similarity-based approach by excluding all authors named “Qing Wang” or “Q. Wang”. This change would yield one distinct group for each author named “Qing Wang” or “Q. Wang” by default, as shown in Fig. 2(b). These authors can only be grouped together if more specific knowledge about their entity resolution become available. In this case, by using the knowledge patterns that capture K1 and K2 (they will be presented as P_{A1} and P_{A2} in Example 2.1), the authors 1, 2 and 4 can be grouped together, while the authors 3 and 5 are still left in different groups if no more knowledge is available yet. Fig. 2(c) presents this result.

Although we can use knowledge about entity resolution to improve the results of traditional similarity-based methods, knowledge artifacts acquired from different sources at different times are often not consistent. For instance, we may acquire the following K3 later on, which is however conflicting with K1 and K2.

- (K3) *Q. Wang at the PBRF Office of the University of Otago has never studied or worked in Germany.*

Hence, questions concerning inconsistencies of knowledge artifacts naturally arise, such as “how can we efficiently detect the inconsistency among K1, K2 and K3?” and “can we reverse the decision on resolving the authors 2 and 4 if K2 is incorrect?”. To answer such questions, we first need to represent knowledge about entity resolution in a structural and efficient way, and then build automatic reasoning tools for checking consistency of knowledge artifacts.

It is also worth noting that knowledge artifacts are often acquired at different levels of abstraction. For instance, the following K4 is more general than K3.

- (K4) *Nobody at the PBRF Office of the University of Otago has ever studied or worked in Germany.*

The aim of this paper is to develop a theoretical framework for knowledge-based entity resolution. Such a framework can incorporate traditional similarity-based methods into knowledge patterns for improving the quality of entity resolution. As the first step, we are concerned with finding a suitable knowledge representation formalism for entity resolution with attractive computational properties. In particular, we are interested in the following issues:

- What is a knowledge representation formalism that can provide sufficient expressive power for specifying knowledge about entity resolution at flexible levels of abstraction? Can such a knowledge representation formalism also support collective entity resolution?
- How efficiently knowledge about entity resolution can be extracted using such a knowledge representation formalism?
- How can we identify and eliminate redundancy of knowledge representation? How efficient is it?
- How can we characterize optimality of knowledge representation? What is the mechanism for finding an optimal representation? How efficient is it?

The framework we propose exploits Datalog-style rules for describing when two entities should be considered to be identical in a default setting (e.g., authors with the same name and affiliation, see [Example 2.1](#)). Such rules may change over time, as we learn more knowledge about entity resolution. Therefore, a key element in our framework is to support both positive instances to be included and negative instances to be excluded, the combination of which empowers us to improve the accuracy of entity resolution for many real-life applications.

Contributions. Our first contribution is to show that a decidable fragment of first-order logic, including conjunction, disjunction and a certain form of negation, is well suited for representing *knowledge patterns* of entity resolution. Applying such knowledge patterns to resolve entities can be viewed as a restricted program in Datalog with negation under the inflationary semantics [2]. In the sense of collective entity resolution discussed in [10], our proposed formalism has the ability to capture knowledge of entity resolution for entities of different types, and resolve them in an iterative process. Furthermore, such knowledge patterns have the following main features:

- capable of expressing knowledge at flexible levels of abstraction, ranging from a generic perspective expressed as a pattern rule to more specific perspectives expressed as instantiations of a pattern rule by applying tuples in the corresponding pattern relation,
- capable of specifying knowledge about entity resolution conditionally (e.g., two entities can be resolved as one object under certain conditions),
- capable of specifying exceptions of knowledge about entity resolution (e.g., two entities can be resolved as one object except for certain conditions).

Inevitably, the design of the framework involves a tradeoff between the expressive power of the formalism and its computational properties. This leads us to further investigate important properties of the formalism such as the complexity of the evaluation, containment and optimization problems for knowledge patterns.

In order to manage knowledge-based entity resolution efficiently, knowledge patterns need to be processed as fast as possible, and consequently redundant information in knowledge patterns needs to be removed as much as possible. For these reasons, we first study the evaluation and containment problems for knowledge patterns. We show that the evaluation problem for knowledge patterns is NP-complete w.r.t. combined complexity. Nevertheless, this problem is in PTIME w.r.t. data complexity, even though the logical formalization of a knowledge pattern involves conjunction, disjunction and also a restricted form of negation. Based on the Homomorphism Theorem for the containment of conjunctive queries and theorem for the containment of unions of conjunctive queries, we provide a characterization for the containment of knowledge patterns, and show that the containment problem for knowledge patterns is NP-complete. Similar to the query containment problem that has important applications in query optimization, the result of the containment problem for knowledge patterns enables us to effectively discover redundancy among knowledge patterns and thus optimize knowledge models.

Our third contribution is to characterize an optimal representation of knowledge models and to further develop a mechanism of finding such an optimal representation. The optimization of a knowledge model plays a vital role in the evaluation, i.e., it can transform knowledge patterns to equivalent patterns that can be evaluated more efficiently. In this paper we propose a three-stage mechanism of finding such an optimal representation: i) *normalization*: normalizing knowledge patterns into ones having exactly one inclusion query and possibly many exclusion queries; ii) *elimination*: eliminating redundancy of different patterns such that the total number of inclusion queries is minimized; iii) *composition*: composing patterns in a way such that the total number of exclusion queries is also minimized. We prove that every knowledge model obtained by applying the first two stages on a given knowledge model is positively optimal with the minimal number of inclusion queries, and every knowledge model obtained by applying the three stages on a given knowledge model is optimal in the sense that the number of exclusion queries in a positively optimal model is also minimal. In analyzing the complexity of the optimization problem, we show a reduction from the clique cover problem. It turns out that the decision problem of finding a positively optimal model is NP-complete and the decision problem of finding an optimal model is also NP-complete.

Organization. The remainder of the paper is structured as follows. In Section 2 we present our framework along with a motivating example. The definitions of knowledge model, knowledge pattern and queries that are associated with knowledge patterns will be provided. Then we characterize the minimal representation of a knowledge pattern in Section 3. After that, we discuss the containment problem of knowledge patterns in Section 4 and the optimization problem of knowledge models in Section 5. In Section 5, we develop a mechanism of finding an optimal representation of a knowledge model. Section 6 analyzes complexity of the evaluation, containment and optimization problems for knowledge patterns. Section 7 discusses related work, and we briefly conclude the paper in Section 8. For clarity the usage of symbols is summarized in [Fig. 3](#).

| Symbol | Meaning | Symbol | Meaning |
|---------------|-------------------------------------|---|---|
| D | domain | $t_1 \wedge t_2$ | intersection of t_1 and t_2 |
| D_0 | identity domain | $t_1 \vee t_2$ | union of t_1 and t_2 |
| \mathcal{Y} | database schema | $t_1 \sqsubseteq t_2$ ($r_1 \sqsubseteq r_2$) | t_2 subsumes t_1 (r_2 subsumes r_1) |
| \mathcal{S} | extended database schema | $t_1 \sqsubseteq_{\uparrow} t_2$ | t_2 upward-subsumes t_1 |
| P | pattern | $t_1 \sqsubseteq_{\downarrow} t_2$ | t_2 downward-subsumes t_1 |
| \mathcal{P} | set of patterns | φ | query |
| R | relation name | φ^+ | inclusion query |
| \mathcal{R} | set of equivalence relation names | φ^- | exclusion query |
| ℓ | pattern rule | Σ_P^+ | set of inclusion queries of P |
| r | pattern relation | Σ_P^- | set of exclusion queries of P |
| r^+ | set of tuples with $A^* = +$ in r | Σ_P | set of all queries of P |
| r^- | set of tuples with $A^* = -$ in r | $\varphi(I)$ | interpretation of φ over I |
| $attr(r)$ | set of attributes of r | $P(I)$ (or $\llbracket P \rrbracket$) | interpretation of P over I |
| I | database instance | M | knowledge model |
| $I(R)$ | relation of R in I | μ^+ | inflationary fixpoint operator |
| A | attribute | S | set |
| A^* | sign attribute | $ S $ | cardinality of S |
| λ | arbitrary value from a domain | $no(P)$ | set of normalized patterns of P |
| t | tuple | \mathcal{G} | graph |
| $t.A$ | value of A in t | \mathcal{C} | compatibility graph |
| | | \mathcal{M} | mergeability graph |

Fig. 3. A list of symbols.

2. Formal framework

In this section we present our framework for knowledge-based entity resolution, and elaborate the key ideas using a real-world example. Let us fix a family $\{D_i\}_{i \in I}$ of possibly infinite domains that are pairwise disjoint, an identity domain $D_0 \in \{D_i\}_{i \in I}$ containing a countably infinite number of entity identifiers, and a (relational) database schema \mathcal{Y} over $\{D_i\}_{i \in I}$ consisting of a finite, non-empty set of relation names. Each relation name is associated with a finite, non-empty set $\{A_1, \dots, A_n\}$ of attributes, each having a domain $D \in in\{D_i\}_{i \in I}$. An *extended database schema* \mathcal{S} over $\{D_i\}_{i \in I}$ is a pair $(\mathcal{Y}, \mathcal{R})$, where \mathcal{Y} is a database schema over $\{D_i\}_{i \in I}$, \mathcal{R} is a finite, non-empty set of equivalence relation names over D_0 , and $\mathcal{Y} \cap \mathcal{R} = \emptyset$. Each equivalence relation name $R \in \mathcal{R}$ corresponds to at least one relation $R' \in \mathcal{Y}$ such that D_0 is the domain of some attributes in R' . A *database instance* over \mathcal{S} is a set of relations, and each relation over $R' \in \mathcal{Y}$ has a finite, non-empty set of tuples.

Definition 2.1. Let $\mathcal{S} = (\mathcal{Y}, \mathcal{R})$ be an extended database schema. A (*knowledge*) *pattern* P over \mathcal{S} is a pair $\langle \ell, r \rangle$ consisting of

- a *pattern rule* ℓ that has the form $R(x, y) \leftarrow \varphi(x_1, \dots, x_n, x, y)$, in which φ is a conjunction of atoms over \mathcal{S} , $R \in \mathcal{R}$ is an equivalence relation name, and x and y are variables over D_0 , and
- a *pattern relation* r of the arity $n + 1$ with n attributes A_1, \dots, A_n that are in 1-1 correspondence to the variables x_1, \dots, x_n in φ , expressed as $\iota(x_i) = A_i$ ($i = 1, \dots, n$), plus a *sign attribute* A^* with domain $\{+, -\}$.

Two types of atoms may occur in φ : a *relation atom* $R(x_1, \dots, x_n)$ for $R \in \mathcal{Y} \cup \mathcal{R}$ and an *equality atom* $x = y$. Each pattern rule ℓ must be generic (i.e., containing no constants) in the sense of the genericity principle for database queries [4]. Each pattern relation has exactly one sign attribute A^* whose values are either $+$ or $-$. For other non-sign attributes, a pattern relation r may contain constants from the domains of its attributes, variables that do not occur in φ , or a special symbol λ indicating that any appropriate value may occur in its place (i.e., constants, variables or λ).

Intuitively, given a knowledge pattern $P = \langle \ell, r \rangle$, ℓ can be viewed as the default rule for describing the knowledge pattern, tuples with $A^* = +$ are *positively* substantiating ℓ , yielding *inclusion* cases, and tuples with $A^* = -$ are *negatively* substantiating ℓ , yielding *exclusion* cases.

Example 2.1. Consider a publication management system with the relation schemas $AUTHOR = \{AID, NAME, AFFILIATION, EMAIL\}$ (as in Example 1.1), $PUBLICATION = \{PID, TITLE, YEAR, AID\}$, and $R_{aut} = \{AID, AID\}$. We can develop the following knowledge patterns for resolving authors.

- The pattern $P_{A1} = \langle \ell_1, r_1 \rangle$ specifies that two authors are identical if they have the same name, but the name is neither “Q. Wang” nor “Qing Wang”. The first tuple in r_1 is positively substantiating the pattern rule, which applies to every author because all non-sign attributes of r_1 contain λ , while the second and third tuples are negative describing two exceptions.

$$\ell_1 = R_{aut}(x, y) \leftarrow AUTHOR(x, x_1, x_2, x_3) \wedge AUTHOR(y, y_1, y_2, y_3) \wedge x_1 = y_1$$

$$r_1 =$$

| A_{x_1} | A_{x_2} | A_{x_3} | A_{y_1} | A_{y_2} | A_{y_3} | A^* |
|-----------|-----------|-----------|-----------|-----------|-----------|-------|
| λ | λ | λ | λ | λ | λ | + |
| Q. Wang | λ | λ | λ | λ | λ | – |
| Qing Wang | λ | λ | λ | λ | λ | – |

- The pattern $P_{A2} = \langle \ell_2, r_2 \rangle$ specifies that an author named “Q. Wang” and affiliated with “PBRF Office, University of Otago” or with “CAU Kiel, Germany” is the same person as an author named “Qing Wang” affiliated with “Dept. of Information Science, University of Otago”. The tuples in r_2 are both positive, substantiating the default rule.

$$\ell_2 = R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, x_1, x_2, x_3) \wedge \text{AUTHOR}(y, y_1, y_2, y_3)$$

$$r_2 =$$

| A_{x_1} | A_{x_2} | A_{x_3} | A_{y_1} |
|-----------|----------------------------------|-----------|-----------|
| Q. Wang | PBRF Office, University of Otago | λ | Qing Wang |
| Q. Wang | CAU Kiel, Germany | λ | Qing Wang |

| A_{y_2} | A_{y_3} | A^* |
|---|-----------|-------|
| Dept. of Information Science, University of Otago | λ | + |
| Dept. of Information Science, University of Otago | λ | + |

- The pattern $P_{A3} = \langle \ell_3, r_3 \rangle$ specifies that two authors are identical if they have the same name and affiliation, but their affiliation is not “PBRF Office, University of Otago”. The first tuple in r_3 is positively substantiating the pattern rule, which applies to every author, while the second tuple is negative, describing an exception.

$$\ell_3 = R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, z_1, x_1, x_2) \wedge \text{AUTHOR}(y, z_1, y_1, y_2) \wedge x_1 = y_1$$

$$r_3 =$$

| A_{z_1} | A_{x_1} | A_{x_2} | A_{y_1} | A_{y_2} | A^* |
|-----------|----------------------------------|-----------|-----------|-----------|-------|
| λ | λ | λ | λ | λ | + |
| λ | PBRF Office, University of Otago | λ | λ | λ | – |

- The pattern $P_{A4} = \langle \ell_4, r_4 \rangle$ specifies that two authors are identical if they have the same name and both co-authored with another author who is not Qing Wang with “q.q.wang@massey.ac.nz” (assume that Qing Wang with “q.q.wang@massey.ac.nz” has co-authored with two different persons whose names are exactly the same). Again, the first tuple in r_4 is positively substantiating the pattern rule, which applies to every author, while the second tuple is negative, describing an exception.

$$\ell_4 = R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, z_5, x_2, x_3) \wedge \text{PUBLICATION}(z_1, x_1, z_4, x) \wedge \text{PUBLICATION}(z_1, x_1, z_4, z) \wedge \text{AUTHOR}(y, z_5, y_2, y_3) \wedge \text{PUBLICATION}(z_2, y_1, z_3, y) \wedge \text{PUBLICATION}(z_2, y_1, z_3, z') \wedge R_{aut}(z, z') \wedge \text{AUTHOR}(z, z_6, z_7, z_8)$$

$$r_4 =$$

| ... | A_{z_5} | A_{z_6} | A_{z_7} | A_{z_8} | A^* |
|-----------|-----------|-----------|-----------|-----------------------|-------|
| λ | λ | λ | λ | λ | + |
| λ | λ | Qing Wang | λ | q.q.wang@massey.ac.nz | – |

- The pattern $P_{A5} = \langle \ell_5, r_5 \rangle$ specifies that two authors are identical if they have the same email address. The tuple in r_5 is positively substantiating the pattern rule, which applies to every author.

$$\ell_5 = R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, x_1, x_2, z) \wedge \text{AUTHOR}(y, y_1, y_2, z)$$

$$r_5 =$$

| A_z | A_{x_1} | A_{x_2} | A_{y_1} | A_{y_2} | A^* |
|-----------|-----------|-----------|-----------|-----------|-------|
| λ | λ | λ | λ | λ | + |

Each knowledge pattern uses a pattern rule and a pattern relation in a combined way such that variables of the pattern rule are bound to attributes of the pattern relation, leading to a finite set of queries. More specifically, given a pattern $P = \langle \ell, r \rangle$, the set Σ_P of queries can be obtained by substituting the occurrences of bound variables in ℓ with their associated attribute values in r if the attribute values are not λ and adding an existential quantifier for the other (bounded) variables. One tuple t in the relation r “generates” exactly one query by replacing each bound variable x of ℓ with $t.\iota(x)$, and removing $t.\iota(x)$ from the existential quantification if $t.\iota(x)$ is a constant. We distinguish two kinds of queries according to values of the attribute A^* . A query is an *inclusion query* if $t.A^* = +$ for the tuple t that generates it; otherwise it is an *exclusion query*.

Example 2.2. The pattern $P_{A1} = \langle \ell_1, r_1 \rangle$ in Example 2.1 has the following queries:

$$\begin{aligned}\varphi^+ &= \{R_{aut}(x, y) \mid \exists x_1, x_2, x_3, y_1, y_2, y_3. \text{AUTHOR}(x, x_1, x_2, x_3) \wedge \text{AUTHOR}(y, y_1, y_2, y_3) \wedge x_1 = y_1\} \\ \varphi_1^- &= \{R_{aut}(x, y) \mid \exists x_2, x_3, y_1, y_2, y_3. \text{AUTHOR}(x, \text{“Q. Wang”}, x_2, x_3) \\ &\quad \wedge \text{AUTHOR}(y, y_1, y_2, y_3) \wedge \text{“Q. Wang”} = y_1\} \\ \varphi_2^- &= \{R_{aut}(x, y) \mid \exists x_2, x_3, y_1, y_2, y_3. \text{AUTHOR}(x, \text{“Qing Wang”}, x_2, x_3) \\ &\quad \wedge \text{AUTHOR}(y, y_1, y_2, y_3) \wedge \text{“Qing Wang”} = y_1\}\end{aligned}$$

φ^+ is an inclusion query generated by the first tuple of r_1 , while φ_1^- and φ_2^- are exclusion queries respectively generated by the second and third tuples of r_1 .

Let Σ_P^+ and Σ_P^- be the set of inclusion queries and the set of exclusion queries of the knowledge pattern P , respectively, and $\Sigma_P = \Sigma_P^+ \cup \Sigma_P^-$. Suppose that P over an extended database schema \mathcal{S} has the form $R(x, y) \leftarrow \varphi(x_1, \dots, x_n, x, y)$ and I is an instance over \mathcal{S} . Then the *interpretation* of a query $\varphi(x, y) \in \Sigma_P$ over I is

$$\{R(v(x), v(y)) \mid v \text{ is a valuation over variables of } \varphi, \text{ and } \varphi \text{ is true in } I \text{ under } v\},$$

and the *interpretation* of the pattern P over I is

$$P(I) = \bigcup_{\varphi^+ \in \Sigma_P^+} \varphi^+(I) - \bigcup_{\varphi^- \in \Sigma_P^-} \varphi^-(I).$$

Alternatively, we may express P as one combined query $\bigvee \Sigma_P^+ \wedge (\neg \bigvee \Sigma_P^-)$, and $P(I)$ as $\llbracket P \rrbracket$ when I can be omitted without ambiguity.

A *knowledge model* M over \mathcal{S} consists of a finite, non-empty set \mathcal{P} of patterns over \mathcal{S} . In view of a knowledge model M as a program consisting of rules that have one-to-one correspondence with its patterns, the semantics of applying M over a database instance can be interpreted in the same way as a Datalog program with negation under the inflationary semantics [2]. Let $J_0 = I$ and $I(R)$ be the relation of R in I . Then we have μ^+ as an *inflationary fixpoint operator* such that $\mu^+(M)$ over I defines each equivalence relation $R \in \mathcal{R}$ that is the limit of the sequence $\{J_n(R)\}_{n \geq 0}$ with

$$\begin{aligned}J_0(R) &= I(R) = \emptyset \\ J_n(R) &= J_{n-1}(R) \cup \bigcup_{P \in \mathcal{P}_R} P(J_{n-1}), \quad n > 0\end{aligned}$$

where \mathcal{P}_R is the set of all patterns in M that are associated with R , and $P(J_{n-1})$ is the interpretation of P over J_{n-1} , i.e., denoting the result of applying the pattern P on the instance J_{n-1} over $\mathcal{S} = \mathcal{Y} \cup \mathcal{R}$ whose restriction to \mathcal{Y} is I (i.e., $J_{n-1}(\mathcal{Y}) = I$) and restriction to $R \in \mathcal{R}$ is $J_{n-1}(R)$.

The above definition ensures that the sequence $\{J_n(R)\}_{n \geq 0}$ is increasing: $J_{i-1}(R) \subseteq J_i(R)$ for each $i > 0$. Tuples in each equivalence relation are created in a cumulative rather than destructive way, until each equivalence relation reaches a fixpoint. Since for each database instance there are finitely many tuples that can be added, the sequence $\{J_n(R)\}_{n \geq 0}$ converges in all cases. It is well-known that Datalog programs with negation under the inflationary semantics can terminate in time polynomial in the size of the database [2]. Since a knowledge model is a special kind of Datalog program in which negative facts inferred by exclusion queries are always guarded by positive facts inferred by inclusion queries of the same pattern as illustrated in Example 2.3, the inflationary semantics of a knowledge model can also ensure termination of the computation in time polynomial in the size of the underlying database.

Given a knowledge pattern $P = \langle \ell, r \rangle$, we use r^+ (resp. r^-) to refer to the set of tuples in r with $A^* = +$ (resp. $A^* = -$). If the arity of r is $n + 1$, the number of tuples in r^+ is m and the number of tuples in r^- is k , then the number of Datalog rules [2] required in a Datalog program that is equivalent to the pattern P is $m \times n^k$ in the worst case.

Example 2.3. Consider the following pattern $P_{A8} = \langle \ell_8, r_8 \rangle$.

$$\ell_8 = R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, z_1, z_2, x_1) \wedge \text{AUTHOR}(y, z_1, z_2, y_1)$$

| | A_{z_1} | A_{z_2} | A_{x_1} | A_{y_1} | A^* |
|---------|-----------|-----------|-----------|--------------|-------|
| $r_8 =$ | λ | λ | λ | λ | + |
| | Q. Wang | ANU | λ | λ | - |
| | Qing Wang | λ | λ | qw@gmail.com | - |

A Datalog program that is equivalent to P_{A8} consists of the following Datalog rules [2]:

- (1) $R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, z_1, z_2, x_1), \text{AUTHOR}(y, z_1, z_2, y_1), \neg z_1 = \text{“Q. Wang”}, \neg z_1 = \text{“Qing Wang”};$
- (2) $R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, z_1, z_2, x_1), \text{AUTHOR}(y, z_1, z_2, y_1), \neg z_1 = \text{“Q. Wang”}, \neg y_1 = \text{“qw@gmail.com”};$

- (3) $R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, z_1, z_2, x_1), \text{AUTHOR}(y, z_1, z_2, y_1), \neg z_2 = \text{"ANU"}, \neg z_1 = \text{"Qing Wang"};$
 (4) $R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, z_1, z_2, x_1), \text{AUTHOR}(y, z_1, z_2, y_1), \neg z_2 = \text{"ANU"}, \neg y_1 = \text{"qw@gmail.com"}.$

Since the arity of r_8 is 5, the number of tuples in r_8^+ is 1, and the number of tuples in r_8^- is 2, the number of Datalog rules that correspond to P_{A8} is $m \times k^n = 1 \times 2^4 = 8$ in the worst case. Nevertheless, we only need four Datalog rules as shown above. This is because each tuple in r_8^- has constants in only two attributes. In other words, we may ignore any attributes that have λ when constructing an equivalent set of Datalog rules from a knowledge pattern.

Hence, although each knowledge pattern can be equivalently represented by a Datalog program consisting of a set of Datalog rules, such a Datalog program is often tedious in representation and less efficient in implementation.

Note that, for each knowledge model, we restrict equivalence relations in \mathcal{R} to be the only intensional predicate symbols appearing in the left hand side of a pattern rule. Nevertheless, the equivalence relations can still appear in the right hand side.

Remark 2.1. Every knowledge model must contain patterns that reflect the reflexivity, symmetry and transitivity properties of equivalence relations in \mathcal{R} .

Our framework also supports collective entity resolution. A knowledge model can be constituted by a number of knowledge patterns, each of which resolves entities of certain type (e.g., the patterns in [Example 2.1](#) are all associated with R_{aut} for resolving authors). The final results in the equivalence relations of different types are generated by applying all their relevant patterns over the database in a cumulative way until no more tuples can be added.

Example 2.4. Now we extend [Example 2.1](#) to enable collective entity resolution of authors and publications. Suppose that $R_{pub} = \{\text{PID}, \text{PID}\}$, and we replace the pattern P_{A4} by the pattern $P'_{A4} = \langle \ell'_4, r'_4 \rangle$ below.

$$\ell'_4 = R_{aut}(x, y) \leftarrow \text{AUTHOR}(x, z_5, z_2, x_3) \wedge \text{PUBLICATION}(z_1, x_1, z_4, x) \wedge \text{PUBLICATION}(z'_1, x'_1, z'_4, z) \\ \wedge \text{AUTHOR}(y, z_5, y_2, y_3) \wedge \text{PUBLICATION}(z_2, y_1, z_3, y) \wedge \text{PUBLICATION}(z'_2, y'_1, z'_3, z') \\ \wedge R_{aut}(z, z') \wedge \text{AUTHOR}(z, z_6, z_7, z_8) \wedge R_{pub}(z_1, z'_1) \wedge R_{pub}(z_2, z'_2)$$

$$r'_4 = \begin{array}{c} \begin{array}{|c|c|c|c|c|c|} \hline \dots & A_{z_5} & A_{z_6} & A_{z_7} & A_{z_8} & A^* \\ \hline \lambda & \lambda & \lambda & \lambda & \lambda & + \\ \hline \lambda & \lambda & \text{Qing Wang} & \lambda & \text{q.q.wang@massey.ac.nz} & - \\ \hline \end{array} \end{array}$$

Note that, R_{aut} occurs in the both sides of the pattern rule ℓ'_4 . In doing so, recursion can be incorporated into the process of entity resolution. In addition to P'_{A4} , we also add the following pattern for publications.

- The pattern $P_{P1} = \langle \ell_6, r_6 \rangle$ specifies that two publications are identical if they have the same title and publication year. The first tuple in r_6 is positive substantiating the default rule, which applies to every publication in PUBLICATION , while the second tuple is negative describing an exception to the rule.

$$\ell_6 = R_{pub}(x, y) \leftarrow \text{PUBLICATION}(x, z_1, z_2, x_1) \wedge \text{PUBLICATION}(y, z_1, z_2, y_1)$$

$$r_6 = \begin{array}{c} \begin{array}{|c|c|c|c|c|} \hline A_{z_1} & A_{z_2} & A_{x_1} & A_{y_1} & A^* \\ \hline \lambda & \lambda & \lambda & \lambda & + \\ \hline \text{The Prince of Darkness} & \lambda & \lambda & \lambda & - \\ \hline \end{array} \end{array}$$

Putting the patterns presented in [Example 2.1](#) and [Example 2.4](#) together, the resulting knowledge model $\{P_{A1}, P_{A2}, P_{A3}, P'_{A4}, P_{A5}, P_{P1}\}$ supports collective entity resolution of authors and publications, e.g., resolved entities of PUBLICATION by applying P_{P1} may lead to resolving more entities of AUTHOR by applying P'_{A4} .

Hence, a knowledge model in our framework is capable to resolve different entities in a collective way [10], by specifying patterns that capture the interrelationship of different kinds of entities.

Remark 2.2. For each knowledge model, because of the inflationary semantics defined previously, applying its patterns in any order over a given database instance would lead to the same intermediate result at each step. Since equivalence relations yielded by the knowledge model are the fixpoints of applying their patterns, they are always unique and irrelevant to the order of applying the patterns.

Our framework can also support similarity-based methods by viewing them as black boxes and representing their similarity results in the form of similarity relations in a database.

Example 2.5. Suppose that $SIM = \{NAME1, NAME2\}$ is added into the database schema described in Example 2.1. A relation over SIM contains similar names of authors, e.g., “Klaus-Dieter Schewe” and “Klaus D. Schewe”. Then we can have the following pattern P_{A7} to resolve some authors whose names are similar and if they are also associated with the same affiliation.

- The pattern $P_{A7} = \langle \ell_7, r_7 \rangle$ describes that two authors are identical if they have similar names and the same affiliation, but neither of their names is similar to “Q. Wang”.

$$\ell_7 = R_{aut}(x, y) \leftarrow AUTHOR(x, x_1, x_2, x_3) \wedge AUTHOR(y, y_1, y_2, y_3) \wedge sim(x_1, y_1) \wedge x_2 = y_2$$

| | A_{x_1} | A_{x_2} | A_{x_3} | A_{y_1} | A_{y_2} | A_{y_3} | A^* |
|---------|-----------|-----------|-----------|-----------|-----------|-----------|-------|
| $r_7 =$ | λ | λ | λ | λ | λ | λ | + |
| | Q. Wang | λ | λ | λ | λ | λ | – |

Such similarity relations are reflexive and symmetric, but not necessarily transitive. In this sense, they are different from the equality relation. Nevertheless, they are predefined by some chosen similarity methods and treated as part of the database. Hence, representing similarities of attribute values by these similarity relations does not affect our results presented in Sections 4, 5 and 6.

Remark 2.3. The results on the containment problem of knowledge patterns in Section 4, the results on the optimization problem of knowledge models in Section 5 and the complexity results in Section 6 are the same with or without any similarity relations in a database.

Matching dependencies were introduced in [19,20] for specifying matching records in a database. Nevertheless, they are not expressive enough to specify the pattern P_{A7} above. Although the inclusion query generated by the first tuple in r_7 can be expressed as the following matching dependency:

$$AUTHOR[NAME] \approx AUTHOR[NAME] \rightarrow AUTHOR[AID] \doteq AUTHOR[AID],$$

the exclusion query generated by the second tuple in r_7 cannot be expressed by any matching dependency. One of the reasons is because matching dependencies do not provide an ability to specify exceptions as discussed in [19]. Furthermore, the exclusion query generated by the second tuple in r_7 also involves the constant “Q. Wang” from the domain of the attribute $NAME$, which cannot be captured by merely using matching dependencies at the schema level.

3. Minimality of patterns

To improve efficiency of evaluating patterns, we first need to minimize patterns. Since each pattern $P = \langle \ell, r \rangle$ is constituted by a pattern rule ℓ and a pattern relation r , the minimal representation of a pattern has two aspects: 1) finding an equivalent and minimized rule to ℓ ; 2) finding an equivalent and minimized relation to r .

Given two queries φ_1 and φ_2 over the same schema S , φ_1 is *contained* in φ_2 (denoted as $\varphi_1 \subseteq \varphi_2$) if, for every database instance I of S , $\varphi_1(I) \subseteq \varphi_2(I)$, where $\varphi(I)$ denotes the interpretation of φ in I . φ_1 and φ_2 are *equivalent* (denoted as $\varphi_1 \equiv \varphi_2$) if $\varphi_1 \subseteq \varphi_2$ and $\varphi_2 \subseteq \varphi_1$. Since each pattern rule $\ell = R(x, y) \leftarrow \varphi(x_1, \dots, x_n, x, y)$ is essentially a conjunctive query $\varphi_\ell = \{R(x, y) \mid \exists x_1, \dots, x_n. \varphi(x_1, \dots, x_n, x, y)\}$, we say that, given two pattern rules ℓ_1 and ℓ_2 , ℓ_1 is *contained* in ℓ_2 (denoted as $\ell_1 \subseteq \ell_2$) if $\varphi_{\ell_1} \subseteq \varphi_{\ell_2}$, and ℓ_1 and ℓ_2 are *equivalent* (denoted as $\ell_1 \equiv \ell_2$) if $\varphi_{\ell_1} \subseteq \varphi_{\ell_2}$ and $\varphi_{\ell_2} \subseteq \varphi_{\ell_1}$. Then finding the minimization of a pattern rule can thus be handled in the same way of tableau query minimization [2]. We omit further discussion on the first aspect.

For the second aspect, i.e., redundant tuples in a pattern relation, we start with the following example.

Example 3.1. Suppose that the relation r_{m1} below associates with the pattern rule ℓ_3 of P_{A3} in Example 2.1. Then it is easy to see that the inclusion query generated by t_1 contains the inclusion query generated by t_2 , and the exclusion query generated by t_3 contains the exclusion query generated by t_4 . Hence, t_2 and t_4 are redundant in r_{m1} .

| | A_{z_1} | A_{x_1} | A_{x_2} | A_{y_1} | A_{y_2} | A^* | |
|------------|-----------|---------------------|-----------|-----------------------|-----------|-------|-------|
| $r_{m1} =$ | λ | λ | λ | λ | λ | + | t_1 |
| | λ | University of Otago | λ | q.q.wang@massey.ac.nz | λ | + | t_2 |
| | λ | λ | λ | q.q.wang@massey.ac.nz | λ | – | t_3 |
| | Qing Wang | λ | λ | q.q.wang@massey.ac.nz | λ | – | t_4 |

If removing t_2 and t_4 from r_{m1} , we can obtain the relation r_{m2} below. The knowledge captured by $\langle \ell_3, r_{m2} \rangle$ remains the same as captured by $\langle \ell_3, r_{m1} \rangle$.

$$r_{m2} = \begin{array}{|c|c|c|c|c|c|} \hline A_{z_1} & A_{x_1} & A_{x_2} & A_{y_1} & A_{y_2} & A^* \\ \hline \lambda & \lambda & \lambda & \lambda & \lambda & + \\ \hline \lambda & \lambda & \lambda & \text{q.q.wang@massey.ac.nz} & \lambda & - \\ \hline \end{array}$$

In order to formally define redundant tuples in a pattern relation, we need the notion of subsumption under the assumption that constants and λ are partially ordered, i.e., $a \leq \lambda$ for any constant a . We use $\text{attr}(r)$ to denote the set of attributes of a relation r and $t.A$ to denote the value of attribute A in a tuple t .

Definition 3.1. Let t_1 and t_2 be two tuples in a pattern relation r , and let $\mathcal{A} = \text{attr}(r) - \{A^*\}$ be the set of attributes other than A^* in r , i.e., non-sign attributes. Then

- t_1 *subsumes* t_2 (denoted as $t_2 \sqsubseteq t_1$) if $t_2.A \leq t_1.A$ holds for each attribute $A \in \mathcal{A}$, and $t_1.A^* = t_2.A^*$;
- t_1 *upward-subsumes* t_2 (denoted as $t_2 \sqsubseteq_{\uparrow} t_1$) if $t_2.A \leq t_1.A$ holds for each attribute $A \in \mathcal{A}$, $t_1.A^* = -$ and $t_2.A^* = +$;
- t_1 *downward-subsumes* t_2 (denoted as $t_2 \sqsubseteq_{\downarrow} t_1$) if $t_2.A \leq t_1.A$ holds for each attribute $A \in \mathcal{A}$, $t_1.A^* = +$ and $t_2.A^* = -$.

Let r_1 and r_2 be two pattern relations with the same attributes. Then r_1 is said to *subsume* r_2 , denoted as $r_2 \sqsubseteq r_1$, if for each $t_2 \in r_2$ there exists a $t_1 \in r_1$ such that $t_2 \sqsubseteq t_1$ holds.

Example 3.2. Consider the following two pattern relations r_{m3} and r_{m4} (we omit their associated pattern rules because they are irrelevant here), $t_1 \sqsubseteq_{\uparrow} t_2$ holds in r_{m3} while $t_2 \sqsubseteq_{\downarrow} t_1$ holds in r_{m4} .

$$r_{m3} = \begin{array}{|c|c|c|c|} \hline A_{x_1} & A_{x_2} & A_{x_3} & A^* \\ \hline a & a & a & + \\ \hline \lambda & \lambda & \lambda & - \\ \hline \end{array} \begin{array}{l} t_1 \\ t_2 \end{array} \quad r_{m4} = \begin{array}{|c|c|c|c|} \hline A_{x_1} & A_{x_2} & A_{x_3} & A^* \\ \hline \lambda & \lambda & \lambda & + \\ \hline a & a & a & - \\ \hline \end{array} \begin{array}{l} t_1 \\ t_2 \end{array}$$

A pattern relation without redundant tuples should satisfy the following minimality property.

Definition 3.2. Let r be a pattern relation. Then r satisfies the *minimality property* if it satisfies the condition $\bigwedge_{t_i \neq t_j \wedge t_i \in r \wedge t_j \in r} t_i \not\sqsubseteq t_j \wedge t_i \not\sqsubseteq_{\uparrow} t_j$.

Each tuple in a pattern relation represents a piece of knowledge about entity resolution. These pieces of knowledge are often provided by different people at different times, and may be represented at different levels of abstraction, e.g., K_3 and K_4 in [Example 1.1](#). It would thus lead to redundancy existing in a knowledge pattern and further cause inefficiency for maintenance. Hence, finding a minimal but equivalent representation for each pattern relation is important for improving efficiency of managing and using a knowledge pattern, making the framework more practically useful. The condition of minimality property specified in [Definition 3.2](#) can be “efficiently” checked by comparing pairs of tuples, which (with exploitation of the partial order) can be done in $O(n \log n)$ time.

Example 3.3. Consider the pattern relations r_{m1} and r_{m2} in [Example 3.1](#) again.

- r_{m1} does not satisfy the minimality property because we have $t_2 \sqsubseteq t_1$, $t_4 \sqsubseteq t_3$ and $t_2 \sqsubseteq_{\uparrow} t_3$.
- r_{m2} satisfies the minimality property.

A pattern relation r is *well-defined* if r is minimal and contains at least one tuple t with $t.A^* = +$. This is because we require that every pattern only yields positive results (i.e., two entities refer to the same real-world entity) rather than negative results (i.e., two entities do not refer to the same real-world entity). The following proposition describes relationships between queries generated by tuples of a well-defined pattern relation. The proof follows easily from the definition of minimality property.

Proposition 3.1. Let $P = \langle \ell, r \rangle$ be a pattern and r be minimal. Then,

- $\varphi_i \not\sqsubseteq \varphi_j$ holds for any two different φ_i and φ_j from Σ_P^+ ;
- $\varphi_i \not\sqsubseteq \varphi_j$ holds for any two different φ_i and φ_j from Σ_P^- ;
- $\varphi_i \not\sqsubseteq \varphi_j$ holds for any $\varphi_i \in \Sigma_P^+$ and $\varphi_j \in \Sigma_P^-$.

Proof. We first prove the statements (a) and (b). Since r is minimal, by [Definition 3.2](#), we know that $t_i \not\sqsubseteq t_j$ holds for any two different tuples t_i and t_j in r . Whenever $t_i.A^* = t_j.A^* = +$, the inclusion queries φ_i and φ_j generated by t_i and t_j can thus satisfy $\varphi_i \not\sqsubseteq \varphi_j$. Whenever $t_i.A^* = t_j.A^* = -$, the exclusion queries φ_i and φ_j generated by t_i and t_j can also satisfy $\varphi_i \not\sqsubseteq \varphi_j$.

The proof for the statement (c) can be done in a similar way. Because r is minimal, by Definition 3.2, we know that $t_i \sqsubseteq \uparrow t_j$ does not hold for any two tuples t_i and t_j of r with $t_i.A = +$ and $t_j.A = -$. Hence, by Definition 3.1, we know that $\varphi_i \not\subseteq \varphi_j$ holds for the inclusion query φ_i generated by t_i and the exclusion query φ_j generated by t_j . \square

In the sequel patterns are assumed to have well-defined pattern relations, unless otherwise stated.

4. The containment problem

In this section we discuss the containment problem of knowledge patterns because the problem is important for identifying redundancy among knowledge patterns and optimizing knowledge models. Given two knowledge patterns P_1 and P_2 over the same schema \mathcal{S} , P_1 is *contained* in P_2 , denoted as $P_1 \subseteq P_2$, if, for each database instance I of \mathcal{S} , $P_1(I) \subseteq P_2(I)$ holds, and similarly P_1 and P_2 are *equivalent*, denoted as $P_1 \equiv P_2$, if $P_1 \subseteq P_2$ and $P_2 \subseteq P_1$ both hold. The *containment problem* of knowledge patterns is to determine whether or not $P_1 \subseteq P_2$ holds for all instances of \mathcal{S} .

We use $\text{sym}(\varphi)$ to denote the set of all variables and constants occurring in a query φ . Let φ_1 and φ_2 be queries over the same schema. A *homomorphism* from φ_1 to φ_2 is a function $\theta : \text{sym}(\varphi_1) \mapsto \text{sym}(\varphi_2)$ such that: (1) $\theta(a) = a$ for every constant $a \in \text{sym}(\varphi_1)$, (2) $\theta(x) \in \text{sym}(\varphi_2)$ for every variable $x \in \text{sym}(\varphi_1)$, (3) for every relation atom $R(x_1, \dots, x_n)$ of φ_1 , $R(\theta(x_1), \dots, \theta(x_n))$ is a relation atom of φ_2 , and (4) for every equality atom $x = y$ of φ_1 , $\theta(x) = \theta(y)$ is an equality atom of φ_2 .

We first discuss the containment problems for two subclasses of knowledge patterns, and show how these problems relate to the query containment problems considered in database theory.

Case I: Patterns that have exactly one inclusion query and no exclusion queries can be equivalently viewed as conjunctive queries. The Homomorphism theorem from provides a characterization for the containment of conjunctive queries.

Theorem 4.1. (See [12].) *Let φ and ϕ be conjunctive queries over the same schema. Then $\varphi \subseteq \phi$ iff there exists a homomorphism from ϕ to φ .*

Case II: Patterns that have one or more inclusion queries and no exclusion queries can be equivalently viewed as unions of conjunctive queries. The following theorem is known for the containment of unions of conjunctive queries [28].

Theorem 4.2. (See [28].) *Let $\phi_1 = q_1 \cup \dots \cup q_n$ and $\phi_2 = q'_1 \cup \dots \cup q'_m$ be two unions of conjunctive queries of the same arity over the same schema. Then $\phi_1 \subseteq \phi_2$ iff for every $i \leq n$, there is $j \leq m$ such that $q_i \subseteq q'_j$.*

Our theorem on characterizing the containment of knowledge patterns is built upon the Homomorphism theorem for conjunctive queries [12], the theorem for unions of conjunctive queries [28], Proposition 3.1 and the following two lemmata.

Lemma 4.1. *Let $\{\varphi_1, \dots, \varphi_n\}$ be a finite set of queries associated with a knowledge pattern P satisfying the condition $\varphi_k \subsetneq \varphi_1$ for $k = 2, \dots, n$. Then $\bigvee_{2 \leq k \leq n} \varphi_k \subsetneq \varphi_1$ holds.*

Proof. By the condition $\varphi_k \subsetneq \varphi_1$ for $k = 2, \dots, n$, we know that $\bigvee_{2 \leq k \leq n} \varphi_k \subseteq \varphi_1$ holds. To prove that $\bigvee_{2 \leq k \leq n} \varphi_k$ is also a proper subset of φ_1 , we choose an arbitrary query φ_j ($j \in [2, n]$) from $\{\varphi_2, \dots, \varphi_n\}$. Since φ_j is a conjunctive query and $\varphi_j \subsetneq \varphi_1$ holds, then according to the Homomorphism Theorem [2], there must exist a homomorphism θ from φ_1 to φ_j , and two different variables x_1 and x_2 of φ_1 such that $\theta(x_1) = y$ and $\theta(x_2) = y$ hold for a variable y of φ_j . It means that, in order get $\varphi_1 \equiv \bigvee_{2 \leq k \leq n} \varphi_k$, we at least require that the results of $\bigvee_{2 \leq k \neq j \leq n} \varphi_k$ contain the results of $(\varphi_1 \wedge x_1 \neq x_2)$. However, the domain of x_1 and x_2 has an infinite number of constants and each query φ_k ($2 \leq k \neq j \leq n$) can be assigned with at most a pair of different constants on x_1 and x_2 . Hence, it is impossible to find such a finite set $\{\varphi_k | 2 \leq k \neq j \leq n\}$ of queries to satisfy $\bigvee_{2 \leq k \neq j \leq n} \varphi_k \equiv (\varphi_1 \wedge x_1 \neq x_2)$. Consequently, $\bigvee_{2 \leq k \leq n} \varphi_k \subsetneq \varphi_1$ is proven. \square

Lemma 4.2. *Let $\{\varphi_1, \dots, \varphi_n\}$ be a finite set of queries associated with a knowledge pattern P satisfying the condition $\varphi_1 \not\subseteq \varphi_k$ for $k = 2, \dots, n$. Then $\varphi_1 \not\subseteq \bigvee_{2 \leq k \leq n} \varphi_k$ holds.*

Proof. From a set-theoretic point of view, each φ_k ($k = 2, \dots, n$) can be expressed as the disjoint union of two parts $\varphi_{(1,k)} = \varphi_k \wedge \varphi_1$ and $\varphi_{(2,k)} = \varphi_k \wedge (\neg \varphi_1)$.

- (1) Since $\varphi_{(1,k)} \subsetneq \varphi_1$ ($k = 2, \dots, n$), according to Lemma 4.1, we have $\varphi_1 \not\subseteq \bigvee_{2 \leq k \leq n} \varphi_{(1,k)}$.
- (2) Since $\varphi_1 \wedge (\neg \varphi_{(2,k)}) \equiv \varphi_1$ ($k = 2, \dots, n$), we have $\varphi_1 \wedge (\neg \bigvee_{2 \leq k \leq n} \varphi_{(2,k)}) \equiv \varphi_1$.

Hence, we have $\varphi_1 \wedge (\neg \bigvee_{2 \leq k \leq n} \varphi_k) \equiv \varphi_1 \wedge (\neg \bigvee_{2 \leq k \leq n} \varphi_{(1,k)})$. Because $\varphi_1 \not\subseteq \bigvee_{2 \leq k \leq n} \varphi_{(1,k)}$ holds, $\varphi_1 \not\subseteq \bigvee_{2 \leq k \leq n} \varphi_k$ is proven. \square

In fact, [Lemmata 4.1 and 4.2](#) can be easily generalized to any finite set of conjunctive queries that associate with the same output relation.

The following theorem characterizes containment of two knowledge patterns in terms of the connections between inclusion queries and exclusion queries of these knowledge patterns.

Theorem 4.3. *Let $P_1 = \langle \ell_1, r_1 \rangle$ and $P_2 = \langle \ell_2, r_2 \rangle$ be two knowledge patterns associated with the same equivalence relation over the same schema \mathcal{S} . Then $P_1 \subseteq P_2$ iff the following condition is satisfied:*

- for each inclusion query $\varphi_1 \in \Sigma_{P_1}^+$, there must exist an inclusion query $\phi_1 \in \Sigma_{P_2}^+$ such that
 - (a) $\varphi_1 \subseteq \phi_1$ and
 - (b) for each exclusion query $\varphi_2 \in \Sigma_{P_2}^-$, there must exist an exclusion query $\varphi_2 \in \Sigma_{P_1}^-$ such that

$$\phi_2 \wedge \varphi_1 \subseteq \varphi_2 \wedge \varphi_1.$$

Proof. Let us start with the if part. By the part (a) of the condition, we know that, for each inclusion query φ_1 of P_1 , there must exist an inclusion query ϕ_1 of P_2 that contains φ_1 . Furthermore, the part (b) of the condition guarantees that if any tuples in the result of φ_1 are eliminated by an exclusion query φ_2 from P_2 then the tuples must also be eliminated by an exclusion query φ_2 from P_1 . Therefore, $P_1 \subseteq P_2$ holds.

For the only if part, the proof is built upon [Proposition 3.1](#) and [Lemma 4.2](#). We proceed it in three steps:

1. We first explain why, for each inclusion query φ_1 of P_1 , there must exist an inclusion query ϕ_1 of P_2 such that $\varphi_1 \subseteq \phi_1$ holds. To get a contradiction, we assume that there does not exist any inclusion query ϕ_1 of P_2 satisfying $\varphi_1 \subseteq \phi_1$. By the definition of a knowledge pattern and the given condition $P_1 \subseteq P_2$, we have $(\bigvee \Sigma_{P_1}^+ \wedge (\neg \bigvee \Sigma_{P_1}^-)) \subseteq (\bigvee \Sigma_{P_2}^+ \wedge (\neg \bigvee \Sigma_{P_2}^-))$. Since $\varphi_1 \subseteq \bigvee \Sigma_{P_1}^+$, we know that $(\varphi_1 \wedge (\neg \bigvee \Sigma_{P_1}^-)) \subseteq (\bigvee \Sigma_{P_2}^+ \wedge (\neg \bigvee \Sigma_{P_2}^-))$ holds. It then implies that the following equation should also hold:

$$\begin{aligned} \varphi_1 &\subseteq \left(\bigvee \Sigma_{P_2}^+ \wedge \left(\neg \bigvee \Sigma_{P_2}^- \right) \right) \vee \left(\bigvee \Sigma_{P_1}^- \wedge \varphi_1 \right) \\ &\subseteq \bigvee \Sigma_{P_2}^+ \vee \bigvee \Sigma_{P_1}^-. \end{aligned}$$

However, because the pattern relation of P_1 is minimal, by [Proposition 3.1](#), $\varphi_1 \not\subseteq \varphi_2$ holds for every exclusion query $\varphi_2 \in \Sigma_{P_1}^-$. Furthermore, according to our assumption, we know that $\varphi_1 \not\subseteq \phi_1$ holds for every inclusion query $\phi_1 \in \Sigma_{P_2}^+$. Since the numbers of queries in $\Sigma_{P_2}^+$ and $\Sigma_{P_1}^-$ are both finite, by using [Lemma 4.2](#), we can get $\varphi_1 \not\subseteq \bigvee \Sigma_{P_2}^+ \vee \bigvee \Sigma_{P_1}^-$, which contradicts with the above equation. Hence, we have proven that, for each inclusion query $\varphi_1 \in \Sigma_{P_1}^+$, there must exist an inclusion query $\phi_1 \in \Sigma_{P_2}^+$ such that $\varphi_1 \subseteq \phi_1$.

2. The second step is to discuss why the part (b) of the condition is necessary. That is, for each exclusion query φ_2 of P_2 , we need to show that there must exist an exclusion query φ_2 of P_1 such that $\phi_2 \wedge \varphi_1 \subseteq \varphi_2 \wedge \varphi_1$ holds. There are two sub-steps.

- We first prove that each exclusion query φ_2 of P_2 needs to satisfy $\phi_2 \wedge \varphi_1 \subseteq \bigvee \Sigma_{P_1}^-$. That is, any results of φ_1 that are eliminated by such an exclusion query φ_2 should also be eliminated by one or more exclusion queries of P_1 . Assume that there exists an exclusion query φ_2 of P_2 that does not satisfy $\phi_2 \wedge \varphi_1 \subseteq \bigvee \Sigma_{P_1}^-$. By $\phi_2 \wedge \varphi_1 \not\subseteq \bigvee \Sigma_{P_1}^-$ and the part (a) of the condition $\varphi_1 \subseteq \phi_1$, there must exist some results of φ_1 that are eliminated from P_2 by φ_2 but still included in P_1 . Thus, $P_1 \not\subseteq P_2$ and there is a contradiction.
- Second, we prove that there must exist an exclusion query φ_2 of P_1 such that $\phi_2 \wedge \varphi_1 \subseteq \varphi_2 \wedge \varphi_1$ holds for each exclusion query φ_2 of P_2 . Assume that there does not exist such a φ_2 satisfying $\phi_2 \wedge \varphi_1 \subseteq \varphi_2 \wedge \varphi_1$. By using [Lemma 4.2](#), we would have

$$\phi_2 \wedge \varphi_1 \not\subseteq \bigvee_{\varphi_2 \in \Sigma_{P_1}^-} (\varphi_2 \wedge \varphi_1).$$

Since $\bigvee_{\varphi_2 \in \Sigma_{P_1}^-} (\varphi_2 \wedge \varphi_1) \equiv \bigvee \Sigma_{P_1}^- \wedge \varphi_1$, we then have

$$\begin{aligned} \phi_2 \wedge \varphi_1 &\not\subseteq \bigvee \Sigma_{P_1}^- \wedge \varphi_1 \\ &\not\subseteq \bigvee \Sigma_{P_1}^-. \end{aligned}$$

This contradicts with our result in the first sub-step. Hence, for each exclusion query $\varphi_2 \in \Sigma_{P_2}^-$, there must exist an exclusion query $\varphi_2 \in \Sigma_{P_1}^-$ such that $\phi_2 \wedge \varphi_1 \subseteq \varphi_2 \wedge \varphi_1$ holds.

3. Our last step is to prove that each inclusion query φ_1 of P_1 needs to satisfy the requirements in the first two steps. Since the pattern relation of P_1 is minimal, by Proposition 3.1, $\varphi_1 \not\subseteq \varphi_2$ holds for any exclusion query $\varphi_2 \in \Sigma_{P_1}^-$ and $\varphi_1 \not\subseteq \varphi_1'$ holds for any inclusion query $\varphi_1' \in (\Sigma_{P_1}^+ - \{\varphi_1\})$. Thus, by Lemma 4.2, $\varphi_1 \not\subseteq \bigvee \Sigma_{P_1}^- \vee \bigvee (\Sigma_{P_1}^+ - \{\varphi_1\})$ holds. It means that each inclusion query φ_1 of P_1 contributes some results into the final results of the knowledge pattern P_1 , which cannot be replaced by any other inclusion queries in $\Sigma_{P_1}^+$. Hence, the requirements in the previous two steps need to be satisfied by each $\varphi_1 \in \Sigma_{P_1}^+$. \square

Example 4.1. Consider the following patterns $P_1 = \langle \ell_1, r_1 \rangle$ and $P_2 = \langle \ell_2, r_2 \rangle$.

$$\ell_1 = R(x, y) \leftarrow R_1(x, z_1) \wedge R_2(z_2, y, z_3)$$

| A_{z_1} | A_{z_2} | A_{z_3} | A^* | |
|-----------|-----------|-----------|-------|-------|
| a | a | λ | $+$ | t_1 |
| b | b | λ | $+$ | t_2 |
| λ | λ | b | $-$ | t_3 |

$$\ell_2 = R(x, y) \leftarrow R_1(x, z_1) \wedge R_2(z_1, y, z_3)$$

| A_{z_1} | A_{z_3} | A^* | |
|-----------|-----------|-------|-------|
| λ | λ | $+$ | t_1 |
| λ | b | $-$ | t_2 |

To check whether $P_1 \subseteq P_2$ holds, by Theorem 4.3, we need to check whether the following containments between conjunctive queries hold, i.e., $P_1 \subseteq P_2$ holds iff (1)–(4) hold,

- (1) $\varphi_1^{t_1} \subseteq \varphi_2^{t_1}$
- (2) $\varphi_1^{t_3} \wedge \varphi_1^{t_1} \subseteq \varphi_2^{t_2} \wedge \varphi_1^{t_1}$
- (3) $\varphi_1^{t_2} \subseteq \varphi_2^{t_1}$
- (4) $\varphi_1^{t_3} \wedge \varphi_1^{t_1} \subseteq \varphi_2^{t_2} \wedge \varphi_1^{t_1}$

where

- $\varphi_1^{t_1} = \exists z_3. R_1(x, a) \wedge R_2(a, y, z_3)$;
- $\varphi_1^{t_2} = \exists z_3. R_1(x, b) \wedge R_2(b, y, z_3)$;
- $\varphi_1^{t_3} = \exists z_1, z_2. R_1(x, z_1) \wedge R_2(z_2, y, b)$;
- $\varphi_2^{t_1} = \exists z_1, z_3. R_1(x, z_1) \wedge R_2(z_1, y, z_3)$;
- $\varphi_2^{t_2} = \exists z_1. R_1(x, z_1) \wedge R_2(z_1, y, b)$.

Note that, the pattern rules of equivalent knowledge patterns may not necessarily be equivalent. Similarly, given two knowledge patterns $P = \langle \ell, r \rangle$ and $P' = \langle \ell', r' \rangle$ with $P \subseteq P'$, we may not necessarily have $\ell \subseteq \ell'$, e.g., $P_1 \subseteq P_2$ but $\ell_2 \not\subseteq \ell_1$ in Example 4.1. Nevertheless, it can be proven that, if $P \subseteq P'$, then either $\ell \subseteq \ell'$ or $\ell' \subseteq \ell$ holds.

Example 4.2. The knowledge pattern $P_1 = \langle \ell_1, r_1 \rangle$ in Example 4.1 can be transformed into $P_1'' = \langle \ell_2, r_1' \rangle$ where $P_1 \equiv P_1''$ and r_1' is shown as below.

| A_{z_1} | A_{z_3} | A^* |
|-----------|-----------|-------|
| a | λ | $+$ |
| b | λ | $+$ |
| λ | b | $-$ |

5. The optimization problem

We investigate the optimization problem of knowledge models in this section. The formal definition of optimal model is provided in Section 5.1 and, given a knowledge model, our three-stage approach of finding an optimal model is presented in Section 5.2. We prove the correctness of our approach in Theorems 5.1 and 5.2.

5.1. Optimal model

In general, the optimality of a knowledge model can be defined in many different ways. Here we develop the notion of optimality based on the observation that eliminating redundancy among knowledge patterns will improve computational efficiency.

Example 5.1. Consider a knowledge model $M_1 = \{P_1, P_3, P_4, P_5, P_6\}$, where $P_i = \langle \ell_1, r_i \rangle$ for $i = 1, 3, 4, 5, 6$.

$$\ell_1 = R(x, y) \leftarrow R_1(x, z_1) \wedge R_2(z_2, y, z_3)$$

| r_1 | | | |
|-----------|-----------|-----------|-------|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* |
| a | a | λ | $+$ |
| b | b | λ | $+$ |
| λ | λ | b | $-$ |

| r_3 | | | |
|-----------|-----------|-----------|-------|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* |
| λ | a | λ | $+$ |
| λ | a | b | $-$ |

| r_4 | | | |
|-----------|-----------|-----------|-------|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* |
| λ | λ | b | $+$ |

| r_5 | | | |
|-----------|-----------|-----------|-------|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* |
| λ | b | λ | $+$ |
| a | b | λ | $-$ |
| c | b | λ | $-$ |
| d | b | λ | $-$ |

| r_6 | | | |
|-----------|-----------|-----------|-------|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* |
| λ | λ | a | $+$ |
| a | b | a | $-$ |
| d | λ | a | $-$ |
| c | λ | a | $-$ |

Case 1: Although $P_1 \not\subseteq P_3$ and $P_3 \not\subseteq P_1$, there is some redundancy between the inclusion queries of P_1 and P_3 . The tuple t_1 can be removed from r_1 of P_1 , while the knowledge captured by the modified pattern and P_3 together remains the same, i.e., $\llbracket P_1 \rrbracket \cup \llbracket P_3 \rrbracket = \llbracket \langle \ell_1, r_1 - \{t_1\} \rangle \rrbracket \cup \llbracket P_3 \rrbracket$.

Case 2: Although $P_1 \not\subseteq P_4$ and $P_4 \not\subseteq P_1$, there is some redundancy between an exclusion query of P_1 and the inclusion query of P_4 . We can merge P_1 and P_4 into one pattern $\langle \ell_1, r_1 \cup r_4 - \{t_3\} \rangle$ without affecting the captured knowledge, i.e., $\llbracket P_1 \rrbracket \cup \llbracket P_4 \rrbracket = \llbracket \langle \ell_1, r_1 \cup r_4 - \{t_3\} \rangle \rrbracket$.

Case 3: Although $P_5 \not\subseteq P_6$ and $P_6 \not\subseteq P_5$, P_5 and P_6 can be composed into one pattern $\langle \ell_1, r_{56} \rangle$ to remove the redundancy between their exclusion queries. In doing so, the total number of exclusion queries is decreased from 6 (i.e., 3 in P_5 and 3 in P_6) to 3, and $\llbracket P_5 \rrbracket \cup \llbracket P_6 \rrbracket = \llbracket \langle \ell_1, r_{56} \rangle \rrbracket$ still holds.

| r_{56} | | | |
|-----------|-----------|-----------|-------|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* |
| λ | λ | a | $+$ |
| λ | b | λ | $+$ |
| a | b | λ | $-$ |
| c | λ | λ | $-$ |
| d | λ | λ | $-$ |

Since redundancy may occur between (inclusion and exclusion) queries of two different patterns, we will consider the optimality of a knowledge model with respect to the number of inclusion and exclusion queries associated with its knowledge patterns. Nevertheless, for knowledge patterns their inclusion and exclusion queries are not equally expensive to evaluate. The following example illustrates that an inclusion query needs to be evaluated over the underlying database while an exclusion query is evaluated over the results of its associated inclusion queries in the same knowledge pattern (each exclusion query is indeed contained by its associated inclusion queries as will be discussed in the normalization stage in Section 5.2).

Example 5.2. Consider the pattern $\langle \ell_1, r_{56} \rangle$ presented in Example 5.1. The evaluation of this pattern over a database instance I is conducted as follows:

- Evaluate the inclusion queries $q_1 := \exists z_1, z_2. R_1(x, z_1) \wedge R_2(z_2, y, a)$ and $q_2 := \exists z_1, z_3. R_1(x, z_1) \wedge R_2(b, y, z_3)$ over I
- Evaluate the following exclusion queries over $q_1(I) \cup q_2(I)$:
 - $\exists z_3. R_1(x, a) \wedge R_2(b, y, z_3)$;
 - $\exists z_2, z_3. R_1(x, c) \wedge R_2(z_2, y, z_3)$;
 - $\exists z_2, z_3. R_1(x, d) \wedge R_2(z_2, y, z_3)$.

It is well-known [14] that entity resolution tasks is often an *imbalanced problem*, i.e., the number of non-matches far exceeds the number of matches. This holds even after some form of indexing has been applied. This imbalance problem thus implies that there could be a huge performance difference between evaluating an inclusion query over the whole database to find matches and evaluating an exclusion query over the results of inclusion queries to find non-matches. Based on this, we consider that an optimal knowledge model M must be a positively optimal model (i.e., contains the minimal number of inclusion queries), in which the number of exclusion queries is also minimal among other positively optimal models of M .

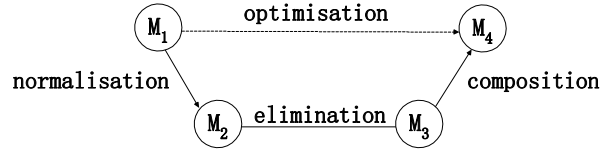


Fig. 4. Optimizing a knowledge model.

Definition 5.1. Let M_1 and M_2 be two knowledge models over $S = (\mathcal{Y}, \mathcal{R})$. Then M_1 and M_2 are *equivalent* (denoted as $M_1 \equiv M_2$) if, for each set $\mathcal{P}_R^i \subseteq M_i$ ($i = 1, 2$) of patterns associated with the same equivalent relation name $R \in \mathcal{R}$, $\bigcup_{P_1 \in \mathcal{P}_R^1} P_1(I) = \bigcup_{P_2 \in \mathcal{P}_R^2} P_2(I)$ holds for every database instance I over S . Furthermore,

- M_2 is a *positively optimized model* of M_1 if $M_1 \equiv M_2$, and $\sum_{\langle \ell_2, r_2 \rangle \in M_2} |r_2^+| \leq \sum_{\langle \ell_1, r_1 \rangle \in M_1} |r_1^+|$;
- M_2 is a *negatively optimized model* of M_1 if $M_1 \equiv M_2$, and $\sum_{\langle \ell_2, r_2 \rangle \in M_2} |r_2^-| \leq \sum_{\langle \ell_1, r_1 \rangle \in M_1} |r_1^-|$;
- M_2 is a *positively optimal model* of M_1 if
 - M_2 is a positively optimized model of M_1 , and
 - there is no positively optimized model M_3 of M_2 with $\sum_{\langle \ell_2, r_2 \rangle \in M_2} |r_2^+| > \sum_{\langle \ell_3, r_3 \rangle \in M_3} |r_3^+|$;
- M_2 is an *optimal model* of M_1 if
 - M_2 is a positively optimal model of M_1 , and
 - there is no both positively and negatively optimized model M_3 of M_2 with $\sum_{\langle \ell_2, r_2 \rangle \in M_2} |r_2^-| > \sum_{\langle \ell_3, r_3 \rangle \in M_3} |r_3^-|$.

The *optimization problem* of a knowledge model M is to find an optimal model of M .

5.2. Finding optimal representation

In the following we develop a mechanism of finding optimal models. Fig. 4 illustrates that, given a knowledge model M_1 , we can obtain its optimal model M_4 by applying the normalization, elimination and composition processes in order.

- The *normalization* stage is to decompose each pattern into a number of smaller, normalized patterns so that redundancy among parts of patterns can be identified at the elimination stage later on, e.g., Case 1 in Example 5.1.
- The *elimination* stage is to reconcile normalized patterns and then to eliminate redundant patterns based on checking pattern containment, e.g., Case 2 in Example 5.1.
- The *composition* stage is to compose normalized and reconciled patterns into larger patterns such that the redundancy among their exclusion queries is removed, e.g., Case 3 in Example 5.1.

Before diving into the details of the optimization issues, we define the notions of intersection and union for tuples and pattern relations. Let r_1 and r_2 be two pattern relations that have the same attributes, $t_1 \in r_1$ and $t_2 \in r_2$. Then the *intersection* $t_1 \wedge t_2$ is a tuple t if either $t_1.A \leq t_2.A$ or $t_2.A \leq t_1.A$ holds for every non-sign attribute A such that

- $t.A^* = t_1.A^*$,
- $t.A = t_1.A$ if $t_1.A \leq t_2.A$ holds for a non-sign attribute A , and
- $t.A = t_2.A$ if $t_2.A \leq t_1.A$ holds for a non-sign attribute A ;

otherwise, $t_1 \wedge t_2$ is undefined. The *union* $t_1 \vee t_2$ is a tuple t such that

- $t.A^* = t_1.A^*$,
- $t.A = t_2.A$ if $t_1.A \leq t_2.A$ holds for a non-sign attribute A ,
- $t.A = t_1.A$ if $t_2.A \leq t_1.A$ holds for a non-sign attribute A , and
- $t.A = \lambda$ for a non-sign attribute A , otherwise.

Furthermore, the *intersection* of r_1 and r_2 is a pattern relation $r_1 \wedge r_2 = \{t_1 \wedge t_2 | t_1 \in r_1, t_2 \in r_2 \text{ and } t_1 \wedge t_2 \text{ is defined}\}$.

5.2.1. Normalization

The normalization process transforms each pattern P in a knowledge model M into a set $\text{no}(P)$ of normalized patterns, while the knowledge captured by M remains unchanged. For convenience, we use $\text{no}(M)$ to denote $\bigcup_{P \in M} \text{no}(P)$.

Definition 5.2. Let $P = \langle \ell, r \rangle$ be a pattern. Then P is *normalized* if $|r^+| = 1$ and $\bigwedge_{t_2 \in r^-} t_2 \sqsubseteq t_1$ for $t_1 \in r^+$.

Example 5.3. Consider the following four patterns that have the same pattern rule:

$$\ell_1 = R(x, y) \leftarrow R_1(x, z_1) \wedge R_2(z_2, y, z_3).$$

Input: A pattern $P = \langle \ell, r \rangle$

Output: A set $\text{NO}(P)$ of normalized patterns

1. Decompose r into $\{r_1, \dots, r_n\}$ where $n = |r^+|$, and each r_k ($k \in [1, n]$) has one distinct tuple of r^+ and all tuples of r^- .
2. For each r_k ($k \in [1, n]$), do the following:
 - Purging every $t_2 \in r_k^-$ w.r.t. to the tuple $t_1 \in r_k^+$, i.e., $t_2 := t_2 \wedge t_1$.
3. Return $\text{NO}(P) = \{\langle \ell, r_1 \rangle, \dots, \langle \ell, r_n \rangle\}$.

Algorithm 1: Normalizing patterns.

The patterns $P_{11}^* = \langle \ell_1, r_{11}^* \rangle$ and $P_{12}^* = \langle \ell_1, r_{12}^* \rangle$ are not normalized because they do not satisfy the condition $\bigwedge_{t_2 \in r^-} t_2 \sqsubseteq_{\downarrow} t_1$ for $t_1 \in r^+$. Nevertheless, P_{11}^* and P_{12}^* can be normalized into $P_{11} = \langle \ell_1, r_{11} \rangle$ and $P_{12} = \langle \ell_1, r_{12} \rangle$, respectively.

| | r_{11}^* | <table style="width: 100%; border-collapse: collapse;"> <tr><th style="padding: 2px;">A_{z_1}</th><th style="padding: 2px;">A_{z_2}</th><th style="padding: 2px;">A_{z_3}</th><th style="padding: 2px;">A^*</th></tr> <tr><td style="padding: 2px;">a</td><td style="padding: 2px;">a</td><td style="padding: 2px;">λ</td><td style="padding: 2px;">$+$</td></tr> <tr><td style="padding: 2px;">λ</td><td style="padding: 2px;">λ</td><td style="padding: 2px;">b</td><td style="padding: 2px;">$-$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | a | a | λ | $+$ | λ | λ | b | $-$ | <table style="width: 100%; border-collapse: collapse;"> <tr><th style="padding: 2px;">A_{z_1}</th><th style="padding: 2px;">A_{z_2}</th><th style="padding: 2px;">A_{z_3}</th><th style="padding: 2px;">A^*</th></tr> <tr><td style="padding: 2px;">b</td><td style="padding: 2px;">b</td><td style="padding: 2px;">λ</td><td style="padding: 2px;">$+$</td></tr> <tr><td style="padding: 2px;">λ</td><td style="padding: 2px;">λ</td><td style="padding: 2px;">b</td><td style="padding: 2px;">$-$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | b | b | λ | $+$ | λ | λ | b | $-$ | |
|-----------|--------------|--|-----------|-----------|-----------|-------|-----|-----|-----------|-----|-----------|-----------|-----|-----|--|-----------|-----------|-----------|-------|-----|-----|-----------|-----|-----------|-----------|-----|-----|--|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | a | λ | $+$ | | | | | | | | | | | | | | | | | | | | | | | | | |
| λ | λ | b | $-$ | | | | | | | | | | | | | | | | | | | | | | | | | |
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | | | | | | |
| b | b | λ | $+$ | | | | | | | | | | | | | | | | | | | | | | | | | |
| λ | λ | b | $-$ | | | | | | | | | | | | | | | | | | | | | | | | | |
| | \Downarrow | <table style="width: 100%; border-collapse: collapse;"> <tr><th style="padding: 2px;">A_{z_1}</th><th style="padding: 2px;">A_{z_2}</th><th style="padding: 2px;">A_{z_3}</th><th style="padding: 2px;">A^*</th></tr> <tr><td style="padding: 2px;">a</td><td style="padding: 2px;">a</td><td style="padding: 2px;">λ</td><td style="padding: 2px;">$+$</td></tr> <tr><td style="padding: 2px;">a</td><td style="padding: 2px;">a</td><td style="padding: 2px;">b</td><td style="padding: 2px;">$-$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | a | a | λ | $+$ | a | a | b | $-$ | <table style="width: 100%; border-collapse: collapse;"> <tr><th style="padding: 2px;">A_{z_1}</th><th style="padding: 2px;">A_{z_2}</th><th style="padding: 2px;">A_{z_3}</th><th style="padding: 2px;">A^*</th></tr> <tr><td style="padding: 2px;">b</td><td style="padding: 2px;">b</td><td style="padding: 2px;">λ</td><td style="padding: 2px;">$+$</td></tr> <tr><td style="padding: 2px;">b</td><td style="padding: 2px;">b</td><td style="padding: 2px;">b</td><td style="padding: 2px;">$-$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | b | b | λ | $+$ | b | b | b | $-$ | |
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | a | λ | $+$ | | | | | | | | | | | | | | | | | | | | | | | | | |
| a | a | b | $-$ | | | | | | | | | | | | | | | | | | | | | | | | | |
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | | | | | | |
| b | b | λ | $+$ | | | | | | | | | | | | | | | | | | | | | | | | | |
| b | b | b | $-$ | | | | | | | | | | | | | | | | | | | | | | | | | |
| | r_{11} | | r_{12} | | | | | | | | | | | | | | | | | | | | | | | | | |

For each normalized pattern, all of its exclusion queries are strictly contained by its inclusion query. In other words, each exclusion query of a normalized pattern is not trivial because it can always exclude some matches inferred by using the inclusion query of the same pattern.

The following lemma says that Algorithm 1 normalizes a pattern without losing any knowledge about entity resolution.

Lemma 5.1. *By applying Algorithm 1, each pattern in $\text{NO}(P)$ is normalized, and $\llbracket P \rrbracket = \bigcup_{P_k \in \text{NO}(P)} \llbracket P_k \rrbracket$ holds.*

Proof. In Algorithm 1, Step 1 gives us a set $\{r_1, \dots, r_n\}$ of pattern relations, each of which has exactly one distinct tuple with $A^* = +$, and $\llbracket P \rrbracket = \bigcup_{1 \leq k \leq n} \llbracket \langle \ell, r_k \rangle \rrbracket$ holds. Step 2 purges tuples with $A^* = -$ with respect to the tuple with $A^* = +$ in each r_k such that $\bigwedge_{t_2 \in r_k^-} t_2 \sqsubseteq_{\downarrow} t_1$ holds for $t_1 \in r_k^+$. This means that each pattern $\langle \ell, r_k \rangle$ is normalized and $\llbracket P \rrbracket = \bigcup_{1 \leq k \leq n} \llbracket \langle \ell, r_k \rangle \rrbracket$ still holds. Step 3 returns a set $\text{NO}(P) = \{\langle \ell, r_1 \rangle, \dots, \langle \ell, r_n \rangle\}$ of normalised patterns. We have $\llbracket P \rrbracket = \bigcup_{P_k \in \text{NO}(P)} \llbracket P_k \rrbracket$. \square

The following lemma describes a nice property of normalized patterns.

Lemma 5.2. *Let P be a normalized pattern. Then $\bigvee \Sigma_P^- \subsetneq \bigvee \Sigma_P^+$ holds.*

Proof. By the definition of normalized pattern, we know that P has exactly one inclusion query, i.e., $\Sigma_P^+ = \{\varphi^+\}$. Since each tuple with $A^* = -$ in the pattern relation r of P is downward-subsumed by the tuple with $A^* = +$ in r , $\varphi^- \subsetneq \varphi^+$ holds for every exclusion query $\varphi^- \in \Sigma_P^-$. By Lemma 4.1, we have $\bigvee \Sigma_P^- \subsetneq \varphi^+$, and thus $\bigvee \Sigma_P^- \subsetneq \bigvee \Sigma_P^+$ holds. \square

Example 5.4. Consider the patterns P_1 in Example 5.1, P_{11}^* , P_{12}^* , P_{11} and P_{12} in Example 5.3. By applying Algorithm 1 on P_1 , P_1 is transformed into P_{11}^* and P_{12}^* in Step 1, and then P_{11}^* and P_{12}^* are transformed into P_{11} and P_{12} , respectively, in Step 2. Hence, the knowledge model M_1 in Example 5.1 can be normalized into the knowledge model $M_2 = \text{NO}(M_1)$ with $\text{NO}(M_1) = \{P_{11}, P_{12}, P_3, P_4, P_5, P_6\}$.

It is also easy to verify that $\llbracket P_1 \rrbracket = \llbracket P_{11} \rrbracket \cup \llbracket P_{12} \rrbracket$ holds. Although $\bigvee \Sigma_{P_1}^- \subseteq \bigvee \Sigma_{P_1}^+$ does not hold for P_1 , but we have $\bigvee \Sigma_{P_{11}}^- \subsetneq \bigvee \Sigma_{P_{11}}^+$ and $\bigvee \Sigma_{P_{12}}^- \subsetneq \bigvee \Sigma_{P_{12}}^+$ for the normalized patterns P_{11} and P_{12} .

Remark 5.1. The normalization process is important for eliminating redundancy among inclusion queries. In Example 5.1, we know that $P_1 \not\subseteq P_3$. Nevertheless, by normalizing P_1 into P_{11} and P_{12} , we will be able to identify $P_{11} \subseteq P_3$ and thus eliminate P_{11} at the elimination stage.

5.2.2. Elimination

At the elimination stage, the task is to remove redundancy among normalized patterns in a way that can minimize the total number of inclusion queries in a knowledge model. The main difficulty is to identify exclusion queries that are redundant with respect to inclusion queries of another pattern, which cannot be solved by simply checking pattern containment. Hence, we proceed the elimination process in two steps.

Input: A set \mathcal{P} of normalized patterns with the same pattern rule

Output: A set \mathcal{P}' of normalized patterns with the same pattern rule

1. Start with $\mathcal{P}' = \mathcal{P}$;
2. Do the following as long as there are changes to \mathcal{P}' :
 - (1) Set $W = \emptyset$;
 - (2) Choose $P = \langle \ell, r_2 \rangle$ from $\mathcal{P}' - W$ and do the following until $\mathcal{P}' = W$:
 - Check whether $t_1 \sqsubseteq_{\downarrow} t_2$ for $t_2 \in r_2^+$ and every $t_1 \in \bigcup_{\langle \ell, r_1 \rangle \in \mathcal{P}'} r_1^-$;
 - If $t_1 \sqsubseteq_{\downarrow} t_2$ holds, then
 - replace $\{t_1\}$ with $\{t_1 \wedge t_3 \mid t_3 \in r_2^-, \text{ and } t_1 \wedge t_3 \text{ is defined}\}$ if $r_2^- \neq \emptyset$;
 - replace $\{t_1\}$ with \emptyset , i.e., remove t_1 from r_1^- otherwise.
 - Set $W = W \cup \{P\}$;
3. Return \mathcal{P}' .

Algorithm 2: Reconciling patterns.

- **Step 1:** Normalized patterns are *reconciled* by minimizing their exclusion queries but without increasing the knowledge captured by the knowledge model as a whole.
- **Step 2:** Redundancy between normalized and reconciled patterns are checked based on pattern containment, and then redundant patterns are eliminated.

Definition 5.3. Let M be a knowledge model. Then a normalized pattern $P \in \text{NO}(M)$ is reconciled in $\text{NO}(M)$ if there does not exist any normalized pattern P' such that $P \subseteq P'$, $P \neq P'$ and $\bigcup_{P_1 \in \text{NO}(M)} \llbracket P_1 \rrbracket = \bigcup_{P_2 \in (\text{NO}(M) - \{P\} \cup \{P'\})} \llbracket P_2 \rrbracket$.

We present [Algorithm 2](#) for reconciling patterns as described in the first step of the elimination process. Each tuple t_1 with $A^* = -$ is iteratively checked against the tuple t_2 with $A^* = +$ in a chosen pattern, and purged when the condition $t_1 \sqsubseteq_{\downarrow} t_2$ is satisfied. This procedure continues until no more changes on tuples with $A^* = -$ can be made. Note that [Algorithm 2](#) considers only normalized patterns with the same pattern rule. This is because each pattern rule ℓ corresponds to a conjunction query φ_{ℓ} , and by the Homomorphism Theorem [12] a pattern $\langle \ell, r \rangle$ can always be transformed into an equivalent pattern $\langle \ell', r' \rangle$ with $\ell \equiv \ell'$. Hence, for simplicity, we assume that all patterns associated with equivalent pattern rules are transformed into the same pattern rule.

The following lemma says that [Algorithm 2](#) reconciles every pattern in a given set \mathcal{P} of normalized patterns with the same pattern rule.

Lemma 5.3. Let \mathcal{P} be a set of normalized patterns with the same pattern rule. By applying [Algorithm 2](#) over \mathcal{P} , each pattern in the result \mathcal{P}' is reconciled.

Proof. We first prove that $\bigcup_{P \in \mathcal{P}} \llbracket P \rrbracket = \bigcup_{P' \in \mathcal{P}'} \llbracket P' \rrbracket$, i.e., the knowledge models corresponding \mathcal{P} and \mathcal{P}' capture the same knowledge about entity resolution. In Step 2.2 of [Algorithm 2](#), for all tuples t_1 that are downward-subsumed by the tuple t_2 with $A^* = +$ in the pattern relation r_2 of a chosen pattern P_2 , they are purged with respect to all the tuples t_3 with $A^* = -$ in r_2 . By [Lemma 5.2](#), each t_3 is also downward-subsumed by t_2 . Hence, for each replacement of $\{t_1\}$ in Step 2.2, it reduces t_1 but the reduced part is always covered by t_2 , and there is no change on the overall knowledge captured by the knowledge models.

Now we show that for each $P' \in \mathcal{P}'$, there does not exist another normalized pattern P_0 such that $P' \subseteq P_0$, $P' \neq P_0$ and $\bigcup_{P_1 \in \text{NO}(M)} \llbracket P_1 \rrbracket = \bigcup_{P_2 \in (\text{NO}(M) - \{P'\} \cup \{P_0\})} \llbracket P_2 \rrbracket$. Suppose that such pattern P_0 exists. Then by [Lemma 4.1](#), [Lemma 4.2](#) and $P' \subseteq P_0$, we know that $\ell' = \ell_0$, $r'^+ = r_0^+$ and $r_0^- \sqsubseteq r'^-$ where $P_0 = \langle \ell_0, r_0 \rangle$ and $P' = \langle \ell', r' \rangle$. Then there are two cases for each $t_1 \in r'^-$.

- There does not exist $t_2 \in r_0^-$ with $t_2 \sqsubseteq t_1$: By [Lemma 4.2](#) and $\bigcup_{P_1 \in \text{NO}(M)} \llbracket P_1 \rrbracket = \bigcup_{P_2 \in (\text{NO}(M) - \{P'\} \cup \{P_0\})} \llbracket P_2 \rrbracket$, there must exist a pattern $P_b \in \text{NO}(M)$ whose pattern relation has a tuple t' with $t_1 \sqsubseteq_{\downarrow} t'$. Then by Step 2.2 of [Algorithm 2](#), we know this is impossible.
- There exists $t_2 \in r_0^-$ with $t_2 \sqsubseteq t_1$: By [Lemma 4.1](#) and $\bigcup_{P_1 \in \text{NO}(M)} \llbracket P_1 \rrbracket = \bigcup_{P_2 \in (\text{NO}(M) - \{P'\} \cup \{P_0\})} \llbracket P_2 \rrbracket$, there must exist a pattern $P_b \in \text{NO}(M)$ whose pattern relation has a tuple t' with $t_1 \sqsubseteq_{\downarrow} t'$ and thus $t_2 \sqsubseteq_{\downarrow} t'$, and a tuple t'' with $t' = t_2$. Then by Step 2.2 of [Algorithm 2](#), we know t_1 can be reduced to t_2 .

Hence, we know that there does not exist a normalized pattern P_0 with $P' \subseteq P_0$, $P' \neq P_0$ and $\bigcup_{P_1 \in \text{NO}(M)} \llbracket P_1 \rrbracket = \bigcup_{P_2 \in (\text{NO}(M) - \{P'\} \cup \{P_0\})} \llbracket P_2 \rrbracket$. $P_0 = P'$. \square

The following lemma describes a nice property of reconciled and normalized patterns.

Lemma 5.4. Let P_1 and P_2 be two reconciled and normalized patterns in the same knowledge model. If P_1 and P_2 satisfy $\bigvee \Sigma_{P_1}^- \subseteq \bigvee \Sigma_{P_2}^+$, then $\bigvee \Sigma_{P_1}^- \equiv \bigvee \Sigma_{P_1}^- \wedge \bigvee \Sigma_{P_2}^-$ holds.

Proof. By $\bigvee \Sigma_{P_1}^- \subseteq \bigvee \Sigma_{P_2}^+$, we know that $\bigvee \Sigma_{P_1}^- \wedge \neg \bigvee \Sigma_{P_2}^- \subseteq \bigvee \Sigma_{P_2}^+$. Since P_1 and P_2 are two reconciled and normalized patterns in the same knowledge model, $\bigvee \Sigma_{P_1}^- \subseteq \bigvee \Sigma_{P_1}^- \wedge \bigvee \Sigma_{P_2}^-$ must hold. Because $\bigvee \Sigma_{P_1}^- \wedge \bigvee \Sigma_{P_2}^- \subseteq \bigvee \Sigma_{P_1}^-$ also holds, the proof is complete. \square

In the second step of the elimination process, given a set \mathcal{P}' of reconciled and normalized patterns, we check their redundancy. That is, if two patterns P_i and $P_j \in \mathcal{P}'$ satisfy $P_i \subseteq P_j$, then P_i is said to be redundant w.r.t. P_j , and needs to be eliminated from \mathcal{P}' .

The following theorem says that applying the normalization and elimination processes on a knowledge model M can yield a positively optimal model of M .

Theorem 5.1. Let M_1 and M_3 be two knowledge models. Then M_3 is a positively optimal model of M_1 if M_3 is obtained by applying the elimination process on $\text{no}(M_1)$.

Proof. We first prove that M_3 is a positively optimized model of M_1 . When reconciling patterns in $\text{no}(M_1)$, by Definition 5.3, the knowledge captured by all patterns remains the same. Furthermore, the second step of the elimination process only removes redundant patterns, which reduces the number of patterns and thus also the total number of inclusion queries, but makes no changes on the overall knowledge captured. Thus, together with Lemma 5.1 and Definition 5.2, we prove that $M_1 \equiv M_3$ and $\sum_{(\ell_3, r_3) \in M_3} |r_3^+| \leq \sum_{(\ell_1, r_1) \in M_1} |r_1^+|$ must hold.

Now we show that there is no positively optimized model M' of M_3 with $\sum_{(\ell_3, r_3) \in M_3} |r_3^+| > \sum_{(\ell', r') \in M'} |r'^+|$. To get a contradiction, we assume that there exists such a model M' containing only normalized patterns. Because of $M' \equiv M_3$, we have $\llbracket P \rrbracket \subseteq \bigcup_{P' \in M'} \llbracket P' \rrbracket$ for each $P \in M_3$. As a result, $\varphi_P^+ \subseteq \bigvee_{P' \in M'} \varphi_{P'}^+ \vee \bigvee \Sigma_P^-$ holds where φ_P^+ (resp. $\varphi_{P'}^+$) is the inclusion query of pattern P (resp. P'). By Lemma 5.2, we have $\varphi_P^+ \not\subseteq \bigvee \Sigma_P^-$. If $\varphi_P^+ \not\subseteq \varphi_{P'}^+$ for every $P' \in M'$, then by Lemma 4.2 we would have $\varphi_P^+ \not\subseteq \bigvee_{P' \in M'} \varphi_{P'}^+ \vee \bigvee \Sigma_P^-$. Hence, $\varphi_P^+ \subseteq \varphi_{P'}^+$ must hold for every $P \in M_3$ and some $P' \in M'$. Similarly, we can prove that $\varphi_{P'}^+ \subseteq \varphi_P^+$ must hold for every $P' \in M'$ and some $P \in M_3$. Since $\sum_{(\ell_3, r_3) \in M_3} |r_3^+| > \sum_{(\ell', r') \in M'} |r'^+|$, and M' and M_3 are normalized, there must exist two patterns P_1 and P_2 in M_3 whose inclusion queries $\varphi_{P_1}^+$ and $\varphi_{P_2}^+$ satisfy $\varphi_{P_1}^+ \subseteq \varphi_{P_2}^+$. However, by Lemma 5.4 and the removal of redundant patterns in the second step of the elimination process, we know that, for any two patterns P_1 and P_2 in M_3 with the inclusion queries $\varphi_{P_1}^+$ and $\varphi_{P_2}^+$, respectively, $\varphi_{P_1}^+ \not\subseteq \varphi_{P_2}^+$ must hold. This leads to a contradiction. \square

Example 5.5. Consider the knowledge model M_2 in Example 5.4 and its patterns $P_{11} = \langle \ell_1, r_{11} \rangle$, $P_{12} = \langle \ell_1, r_{12} \rangle$ and $P_3 = \langle \ell_1, r_3 \rangle$. The first step of the elimination process reconciles these patterns into $P'_{11} = \langle \ell_1, r'_{11} \rangle$, $P'_{12} = \langle \ell_1, r'_{12} \rangle$ and $P'_3 = \langle \ell_1, r'_3 \rangle$, as shown below.

| $r_{11} =$ | <table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 10px;">A_{z_1}</th><th style="padding: 2px 10px;">A_{z_2}</th><th style="padding: 2px 10px;">A_{z_3}</th><th style="padding: 2px 10px;">A^*</th></tr> <tr><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">$+$</td></tr> <tr><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">$-$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | a | a | λ | $+$ | a | a | b | $-$ | \Rightarrow | <table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 10px;">A_{z_1}</th><th style="padding: 2px 10px;">A_{z_2}</th><th style="padding: 2px 10px;">A_{z_3}</th><th style="padding: 2px 10px;">A^*</th></tr> <tr><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">$+$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | a | a | λ | $+$ |
|------------|--|-----------|-----------|-----------|-------|-----------|-----|-----------|-----|-----------|-----|-----|-----|---------------|--|-----------|-----------|-----------|-------|-----------|-----|-----------|-----|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | |
| a | a | λ | $+$ | | | | | | | | | | | | | | | | | | | | |
| a | a | b | $-$ | | | | | | | | | | | | | | | | | | | | |
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | |
| a | a | λ | $+$ | | | | | | | | | | | | | | | | | | | | |
| $r_{12} =$ | <table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 10px;">A_{z_1}</th><th style="padding: 2px 10px;">A_{z_2}</th><th style="padding: 2px 10px;">A_{z_3}</th><th style="padding: 2px 10px;">A^*</th></tr> <tr><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">$+$</td></tr> <tr><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">$-$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | b | b | λ | $+$ | b | b | b | $-$ | \Rightarrow | <table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 10px;">A_{z_1}</th><th style="padding: 2px 10px;">A_{z_2}</th><th style="padding: 2px 10px;">A_{z_3}</th><th style="padding: 2px 10px;">A^*</th></tr> <tr><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">$+$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | b | b | λ | $+$ |
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | |
| b | b | λ | $+$ | | | | | | | | | | | | | | | | | | | | |
| b | b | b | $-$ | | | | | | | | | | | | | | | | | | | | |
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | |
| b | b | λ | $+$ | | | | | | | | | | | | | | | | | | | | |
| $r_3 =$ | <table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 10px;">A_{z_1}</th><th style="padding: 2px 10px;">A_{z_2}</th><th style="padding: 2px 10px;">A_{z_3}</th><th style="padding: 2px 10px;">A^*</th></tr> <tr><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">$+$</td></tr> <tr><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">b</td><td style="padding: 2px 10px;">$-$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | λ | a | λ | $+$ | λ | a | b | $-$ | \Rightarrow | <table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 10px;">A_{z_1}</th><th style="padding: 2px 10px;">A_{z_2}</th><th style="padding: 2px 10px;">A_{z_3}</th><th style="padding: 2px 10px;">A^*</th></tr> <tr><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">a</td><td style="padding: 2px 10px;">λ</td><td style="padding: 2px 10px;">$+$</td></tr> </table> | A_{z_1} | A_{z_2} | A_{z_3} | A^* | λ | a | λ | $+$ |
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | |
| λ | a | λ | $+$ | | | | | | | | | | | | | | | | | | | | |
| λ | a | b | $-$ | | | | | | | | | | | | | | | | | | | | |
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | | | | | | | | | | | | | | | | | | | | |
| λ | a | λ | $+$ | | | | | | | | | | | | | | | | | | | | |

This is because the exclusion queries of P_{11} , P_{12} and P_3 are redundant with respect to the inclusion query of P_4 . Since $r_4^- = \emptyset$ but all tuples in r_{11}^- , r_{12}^- and r_3^- are downward-subsumed by the tuple $(\lambda, \lambda, b, +)$ in r_4^+ , by Algorithm 2, the tuples in r_{11}^- , r_{12}^- and r_3^- are removed. This gives us the pattern relations r'_{11} , r'_{12} and r'_3 . Then P'_{11} is eliminated at the second step of the elimination process since it is redundant with P'_3 , i.e., $P'_{11} \subseteq P'_3$. Hence, we would have $M_3 = \{P'_{12}, P'_3, P_4, P_5, P_6\}$ after applying the elimination process over M_2 .

Remark 5.2. The above example illustrates why the reconciliation of patterns is important for eliminating redundant patterns. Without the reconciliation step, it is difficult to observe the redundancy between P_{11} and P_4 , and between P_3 and P_4 . Consequently, P'_{11} would not be identified as being redundant with respect to P'_3 , and be eliminated from the knowledge model.

5.2.3. Composition

By [Theorem 5.1](#) we know that the total number of inclusion queries in a knowledge model is minimized after the normalization and elimination stages. Nonetheless, it is still possible to further reduce the total number of exclusion queries associated with these inclusion queries. Therefore, the task at the composition stage is to minimize the total number of exclusion queries in a positively optimal model. To accomplish this task, we take three steps at the composition stage:

- **Step 1:** Check the compatibility of patterns, and prove that patterns can be composed into one pattern iff they are pairwise compatible.
- **Step 2:** Check the mergeability of tuples in pairwise compatible patterns, and prove that tuples can be merged into one tuple iff they are pairwise mergeable;
- **Step 3:** Decide the way of composing patterns such that the total number of exclusion queries in a knowledge model is minimal but the knowledge captured by the knowledge model remains unchanged.

Definition 5.4. Let P_1 and P_2 be two normalized patterns that have the same pattern rule. Then P_1 and P_2 are *compatible* if the following two conditions are both satisfied:

- $(r_1^- \wedge r_2^+) \sqsubseteq r_2^-$, and
- $(r_2^- \wedge r_1^+) \sqsubseteq r_1^-$.

Example 5.6. Consider the patterns $P_5 = \langle \ell_1, r_5 \rangle$ and $P_6 = \langle \ell_1, r_6 \rangle$ in [Example 5.1](#). We have,

$$r_5^- \wedge r_6^+ = \begin{array}{|c|c|c|c|} \hline A_{z_1} & A_{z_2} & A_{z_3} & A^* \\ \hline a & b & a & - \\ \hline d & b & a & - \\ \hline c & b & a & - \\ \hline \end{array} = r_6^- \wedge r_5^+$$

Since both $(r_5^- \wedge r_6^+) \sqsubseteq r_6^-$ and $(r_6^- \wedge r_5^+) \sqsubseteq r_5^-$ hold, P_5 and P_6 are compatible. Now consider the patterns $P_4 = \langle \ell_1, r_4 \rangle$ and $P_5 = \langle \ell_1, r_5 \rangle$ in [Example 5.1](#). We have $r_4^- \wedge r_5^+ = \emptyset$, and

$$r_5^- \wedge r_4^+ = \begin{array}{|c|c|c|c|} \hline A_{z_1} & A_{z_2} & A_{z_3} & A^* \\ \hline a & b & b & - \\ \hline d & b & b & - \\ \hline c & b & b & - \\ \hline \end{array}$$

Because of $(r_5^- \wedge r_4^+) \not\sqsubseteq r_4^-$, P_4 and P_5 are not compatible.

Lemma 5.5. Let \mathcal{P} be a set of normalized and reconciled patterns that have the same pattern rule. Then the patterns in \mathcal{P} can be combined into one pattern P satisfying $\bigcup_{P' \in \mathcal{P}} \llbracket P' \rrbracket = \llbracket P \rrbracket$ iff they are pairwise compatible.

Proof. We first consider the if part. By [Definition 5.4](#), we know that any two compatible patterns $P_i = \langle \ell, r_i \rangle$ and $P_j = \langle \ell, r_j \rangle$ in \mathcal{P} can be combined into a single pattern $P_{ij} = \langle \ell, r_{ij} \rangle$ with $r_{ij} = r_i \cup r_j$, which satisfies $\llbracket P_i \rrbracket \cup \llbracket P_j \rrbracket = \llbracket P_{ij} \rrbracket$. Then we can further combine P_{ij} with a pattern P_k from $\mathcal{P} - \{P_i, P_j\}$ into $P_{ijk} = \langle \ell, r_{ijk} \rangle$ with $r_{ijk} = r_{ij} \cup r_k$, which satisfies $\llbracket P_{ij} \rrbracket \cup \llbracket P_k \rrbracket = \llbracket P_{ijk} \rrbracket$. This is because by [Definition 5.4](#) and the given condition that P_k is compatible with both P_i and P_j , we have $(r_k^- \wedge r_{ij}^+) \sqsubseteq r_{ij}^-$ and $(r_{ij}^- \wedge r_k^+) \sqsubseteq r_k^-$. Inductively applying this combination process with the other patterns in \mathcal{P} , all patterns in \mathcal{P} can be combined into one pattern P satisfying $\bigcup_{P' \in \mathcal{P}} \llbracket P' \rrbracket = \llbracket P \rrbracket$.

To show the only if part, assume that two patterns $P_i = \langle \ell, r_i \rangle$ and $P_j = \langle \ell, r_j \rangle$ in \mathcal{P} can be composed into one pattern $P = \langle \ell, r \rangle$ satisfying $\llbracket P_i \rrbracket \cup \llbracket P_j \rrbracket = \llbracket P \rrbracket$. Since the knowledge captured by a pattern P can be expressed as $(\bigvee \Sigma_P^- \wedge \neg \bigvee \Sigma_P^+)$, the following equation holds:

$$\llbracket P \rrbracket = \left(\bigvee \Sigma_{P_i}^+ \wedge \neg \bigvee \Sigma_{P_i}^- \right) \vee \left(\bigvee \Sigma_{P_j}^+ \wedge \neg \bigvee \Sigma_{P_j}^- \right). \quad (1)$$

After applying the logical equivalence rules $\varphi \vee (\varphi_1 \wedge \varphi_2) \Leftrightarrow (\varphi \vee \varphi_1) \wedge (\varphi \vee \varphi_2)$, $\neg \varphi_1 \wedge \neg \varphi_2 \Leftrightarrow \neg(\varphi_1 \vee \varphi_2)$ and $\neg \varphi_1 \vee \neg \varphi_2 \Leftrightarrow \neg(\varphi_1 \wedge \varphi_2)$, we can convert the right hand side of the above expression into:

$$\left(\bigvee \Sigma_{P_i}^+ \vee \bigvee \Sigma_{P_j}^+ \right) \wedge \neg \left(\left(\bigvee \Sigma_{P_i}^- \wedge \bigvee \Sigma_{P_j}^- \right) \vee \left(\bigvee \Sigma_{P_i}^- \wedge \neg \bigvee \Sigma_{P_j}^+ \right) \vee \left(\neg \bigvee \Sigma_{P_i}^+ \wedge \bigvee \Sigma_{P_j}^- \right) \right). \quad (2)$$

To enable that a single pattern P can express the same knowledge as captured by P_i and P_j , $\neg \bigvee \Sigma_{P_j}^+$ in the part $(\bigvee \Sigma_{P_i}^- \wedge \neg \bigvee \Sigma_{P_j}^+)$ and $\neg \bigvee \Sigma_{P_i}^+$ in the part $(\neg \bigvee \Sigma_{P_i}^+ \wedge \bigvee \Sigma_{P_j}^-)$ must be removed. That is, the following two conditions must be satisfied:

- either (1.a) $(\bigvee \Sigma_{P_i}^- \subseteq \bigvee \Sigma_{P_j}^+)$ or (1.b) $(\bigvee \Sigma_{P_i}^- \cap \bigvee \Sigma_{P_j}^+) = \emptyset$ holds, and
- either (2.a) $(\bigvee \Sigma_{P_j}^- \subseteq \bigvee \Sigma_{P_i}^+)$ or (2.b) $(\bigvee \Sigma_{P_j}^- \cap \bigvee \Sigma_{P_i}^+) = \emptyset$ holds.

Because P_i and P_j are normalized and reconciled, (1) when Conditions 1.a and 2.a hold, $r_i^- = r_j^-$, $(r_i^- \wedge r_j^+) = r_j^-$ and $(r_j^- \wedge r_i^+) = r_i^-$; (2) when Conditions 1.b and 2.a hold, r_j^- is empty, and $(r_i^- \wedge r_j^+) = (r_j^- \wedge r_i^+) = \emptyset$; (3) when Conditions 1.a and 2.b hold, r_i^- is empty, and $(r_i^- \wedge r_j^+) = (r_j^- \wedge r_i^+) = \emptyset$; (4) when Conditions 1.b and 2.b hold, we also have $(r_i^- \wedge r_j^+) = (r_j^- \wedge r_i^+) = \emptyset$. Hence, the conditions $(r_i^- \wedge r_j^+) \sqsubseteq r_j^-$ and $(r_j^- \wedge r_i^+) \sqsubseteq r_i^-$ can be always satisfied, and any two patterns P_i and P_j in $\{P\}$ are compatible. \square

Definition 5.5. Let $\{(\ell, r_1), \dots, (\ell, r_n)\}$ be a set of pairwise compatible patterns and $r = \bigcup_{1 \leq i \leq n} r_i$. Two tuples t_1 and t_2 in r^- are *mergeable* if, for every r_k^+ ($1 \leq k \leq n$), the following equation holds:

$$\exists t^- \in r_k^- . t^+ \wedge (t_1 \vee t_2) \sqsubseteq t^- . \quad (3)$$

Example 5.7. Consider the compatible patterns $P_5 = (\ell_1, r_5)$ and $P_6 = (\ell_1, r_6)$ discussed in Example 5.6.

| r_5 | | | | |
|-----------|-----------|-----------|-------|----------|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | |
| λ | b | λ | $+$ | t_{50} |
| a | b | λ | $-$ | t_{51} |
| c | b | λ | $-$ | t_{52} |
| d | b | λ | $-$ | t_{53} |

| r_6 | | | | |
|-----------|-----------|-----------|-------|----------|
| A_{z_1} | A_{z_2} | A_{z_3} | A^* | |
| λ | λ | a | $+$ | t_{60} |
| a | b | a | $-$ | t_{61} |
| d | λ | a | $-$ | t_{62} |
| c | λ | a | $-$ | t_{63} |

In accordance with the definition for mergeable tuples, we have three pairs of mergeable tuples:

- t_{51} and t_{61} are mergeable by $t_{50} \wedge (t_{51} \vee t_{61}) \sqsubseteq t_{51}$ and $t_{60} \wedge (t_{51} \vee t_{61}) \sqsubseteq t_{61}$;
- t_{52} and t_{63} are mergeable by $t_{50} \wedge (t_{52} \vee t_{63}) \sqsubseteq t_{52}$ and $t_{60} \wedge (t_{52} \vee t_{63}) \sqsubseteq t_{63}$;
- t_{53} and t_{62} are mergeable by $t_{50} \wedge (t_{53} \vee t_{62}) \sqsubseteq t_{53}$ and $t_{60} \wedge (t_{53} \vee t_{62}) \sqsubseteq t_{62}$.

In accordance with Definition 5.5, the mergeability of two tuples depends on a chosen set of pairwise compatible patterns. In Example 5.7, the tuples t_{52} and t_{63} are mergeable with respect to $\{P_5, P_6\}$. Suppose that we have $P_7 = (\ell_1, r_7)$ and $r_7 = \{(c, c, c, +)\}$, then the tuples t_{52} and t_{63} are not mergeable with respect to $\{P_5, P_6, P_7\}$. The following proposition can be easily proven using the well-definedness property of patterns.

Proposition 5.1. Let \mathcal{P} be a set of pairwise compatible patterns, and $P \in \mathcal{P}$ with $P = (\ell, r)$. Then any two tuples t_1 and t_2 in r^- are not mergeable in \mathcal{P} .

Proof. Suppose that t_1 and t_2 are mergeable, then the condition $\exists t^- \in r^- . t^+ \wedge (t_1 \vee t_2) \sqsubseteq t^-$ for $t^+ \in r^+$ must be satisfied. Since t_1, t_2 and t^- are all in r^- , and P is normalized, this implies that P is not well-defined, contradicting with our assumption stated in Section 3. \square

Lemma 5.6. Let $\{(\ell, r_1), \dots, (\ell, r_n)\}$ be a set of pairwise compatible patterns and $P = (\ell, r)$ with $r = \bigcup_{1 \leq k \leq n} r_k$. Then a subset $\{t_1^-, \dots, t_m^-\} \subseteq r^-$ of tuples can be combined into one tuple t^- satisfying $\llbracket P \rrbracket = \llbracket \langle \ell, r - \{t_1^-, \dots, t_m^-\} \cup \{t^-\} \rangle \rrbracket$ iff t_1^-, \dots, t_m^- are pairwise mergeable.

Proof. We first show the if part. Let us start with $P_{12} = \{(\ell, r_1), (\ell, r_2)\}$, and suppose that $t_1^- \in r_1^-$ and $t_2^- \in r_2^-$ are mergeable. By Definition 5.5, we can construct $t_{12}^- = t_1^- \vee t_2^-$ satisfying $\llbracket P_{12} \rrbracket = \llbracket \langle \ell, r - \{t_1^-, t_2^-\} \cup \{t_{12}^-\} \rangle \rrbracket$. Now we consider $P_{123} = P_{12} \cup \{(\ell, r_3)\}$ and continue to combine t_{12}^- with the next tuple $t_3^- \in r_3^-$ into $t_{123}^- = t_{12}^- \vee t_3^-$. We need to show that $\llbracket P_{123} \rrbracket = \llbracket \langle \ell, r - \{t_1^-, t_2^-, t_3^-\} \cup \{t_{123}^-\} \rangle \rrbracket$ still holds. By the condition in Definition 5.5, we know that if $\exists t^- \in r_k^- . t^+ \wedge (t_1 \vee t_2) \sqsubseteq t^-$ holds, then $r_k^+ \wedge (t_1 \vee t_2) \sqsubseteq r_k^-$ must also hold (we may alternatively view $t_1 \vee t_2$ as a relation with one tuple here). Therefore, on the one side, we can obtain $r_3^+ \wedge t_{12}^- \sqsubseteq r_3^-$ because $t_{12}^- = (t_1^- \vee t_2^-) \vee (t_2^- \vee t_1^-)$, $r_3^+ \wedge (t_1^- \vee t_2^-) \sqsubseteq r_3^-$, and $r_3^+ \wedge (t_2^- \vee t_1^-) \sqsubseteq r_3^-$. On the other side, we can obtain $r_1^+ \wedge t_{123}^- \sqsubseteq r_1^-$ and $r_2^+ \wedge t_{123}^- \sqsubseteq r_2^-$ because $r_1^+ \wedge t_{12}^- \sqsubseteq r_1^-$, $r_1^+ \wedge t_{13}^- \sqsubseteq r_1^-$, $r_2^+ \wedge t_{12}^- \sqsubseteq r_2^-$, $r_2^+ \wedge t_{23}^- \sqsubseteq r_2^-$. Since $t_{123}^- = t_1^- \vee t_2^- \vee t_3^-$, we have $t_i^- \sqsubseteq t_{123}^-$ for $i = 1, 2, 3$. Hence, we can conclude that $\llbracket P_{123} \rrbracket = \llbracket \langle \ell, r - \{t_1^-, t_2^-, t_3^-\} \cup \{t_{123}^-\} \rangle \rrbracket$ holds. The similar proof steps can be iteratively done for combining t_{123}^- with t_4^- into t_{1234}^- and so on until t^- is constructed from $\{t_1^-, \dots, t_m^-\}$, which satisfies $\llbracket P \rrbracket = \llbracket \langle \ell, r - \{t_1^-, \dots, t_m^-\} \cup \{t^-\} \rangle \rrbracket$.

Now we show the only if part. To get a contradiction, we assume that there are two tuples t_i^- and t_j^- ($1 \leq i \neq j \leq m$) that are not mergeable. Then the condition in Definition 5.5 must be violated. By Lemma 4.2 we know that there exists a

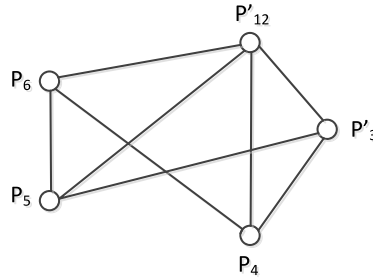


Fig. 5. Graph for compatible patterns.

pattern $\langle \ell, r_k \rangle$ with $r_k^+ \wedge (t_1^- \vee t_j^-) \not\sqsubseteq r_k^-$. By the definition of \vee , we would have $\{r_k^+ \wedge (t_1^- \vee \dots \vee t_i^- \vee \dots \vee t_j^- \vee \dots \vee t_m^-)\} \not\sqsubseteq r_k^-$. However, by Definition 5.4 and the given condition that all patterns are pairwise compatible, we also have $\{r_k^+ \wedge t_q^-\} \sqsubseteq r_k^-$ for $q = 1, \dots, m$. Thus, $\llbracket P \rrbracket = \llbracket \langle \ell, r - \{t_1^-, \dots, t_m^-\} \cup \{(t_1^- \vee \dots \vee t_i^- \vee \dots \vee t_j^- \vee \dots \vee t_m^-)\} \rangle \rrbracket$ cannot hold. By Definition 2.1 and \vee , we know that there does not exist any other tuple t^- that can subsume every tuple in $\{t_1^-, \dots, t_m^-\}$ and meanwhile is subsumed by $t_1^- \vee \dots \vee t_i^- \vee \dots \vee t_j^- \vee \dots \vee t_m^-$. This contradicts with the given condition. \square

Given a set \mathcal{P} of patterns that have been processed through the normalization and elimination stages, by Lemma 5.5 and Lemma 5.6, the composition problem of \mathcal{P} at the composition stage is to find a partition $\{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ of \mathcal{P} such that patterns in each \mathcal{P}_i ($1 \leq i \leq n$) are pairwise compatible, and a partition $\{\mathcal{T}_{i1}, \dots, \mathcal{T}_{im}\}$ for each $\bigcup_{\langle \ell, r_i \rangle \in \mathcal{P}_i} r_i^-$ such that tuples in each \mathcal{T}_{ij} ($1 \leq j \leq m$) are pairwise mergeable and $\sum_{i=1}^n \sum_{j=1}^m |T_{ij}|$ is minimal. The task at the composition process is thus to solve the composition problem of a set of normalized and reconciled patterns. In doing so, we can obtain an optimal model in which the number of patterns is n , the total number of inclusion queries is $|\mathcal{P}|$ and the total number of exclusion queries is $\sum_{i=1}^n \sum_{j=1}^m |T_{ij}|$.

Example 5.8. Consider the knowledge model $M_3 = \{P'_{12}, P'_3, P_4, P_5, P_6\}$ in Example 5.5. We use the graph in Fig. 5 to express the compatibility of patterns in M_3 , i.e., add an edge between two patterns iff they are compatible. Note that, P_5 is not compatible with P_4 , and P_6 is not compatible with P'_3 .

Based on the mergeability of tuples in the pairwise compatible patterns as discussed in Example 5.7, we can partition the patterns in M_3 into $\{P'_{12}, P'_3, P_4\}$ and $\{P_5, P_6\}$, where P'_{12}, P'_3 and P_4 are pairwise compatible, and P_5 and P_6 are pairwise compatible. As a result, the patterns P'_{12}, P'_3 and P_4 can be composed into one pattern $P_{1234} = \langle \ell_1, r_{1234} \rangle$, and P_5 and P_6 can be composed into one pattern $P_{56} = \langle \ell_1, r_{56} \rangle$, which are shown below. Hence, given the knowledge model M_3 , after the composition stage, we can obtain the knowledge model $M_4 = \{P_{1234}, P_{56}\}$, which is an optimal model of M_1 presented in Example 5.1.

| | | | | |
|--------------|-----------|-----------|-----------|-------|
| $r_{1234} =$ | A_{z_1} | A_{z_2} | A_{z_3} | A^* |
| | b | b | λ | $+$ |
| | λ | a | λ | $+$ |
| | λ | λ | b | $+$ |

| | | | | |
|------------|-----------|-----------|-----------|-------|
| $r_{56} =$ | A_{z_1} | A_{z_2} | A_{z_3} | A^* |
| | λ | b | λ | $+$ |
| | λ | λ | a | $+$ |
| | a | b | λ | $-$ |
| | c | λ | λ | $-$ |
| d | λ | λ | $-$ | |

Theorem 5.2. Let M_1 and M_4 be two knowledge models, and M_4 is obtained by applying the normalization, elimination and composition processes on M_1 . Then M_4 is an optimal model of M_1 .

Proof. Suppose that the knowledge model M_3 is obtained by applying the normalization and elimination processes on M_1 , and applying the composition process on M_3 gives us the knowledge model M_4 . Then by Definition 5.1, we need to prove two things:

- applying the composition process on M_3 can yield a both positively and negatively optimized model M_4 of M_3 , and
- there is no both positively and negatively optimized model M of M_3 with $\sum_{\langle \ell, r_4 \rangle \in M_4} |r_4^-| > \sum_{\langle \ell, r \rangle \in M} |r^-|$.

We first prove the first one. By Theorem 5.1, we know that M_3 must be a positively optimal model of M_1 . It means that M_4 cannot have a smaller number of inclusion queries than M_3 . Then because the composition process only merges exclusion queries, but does not make any changes on the number of inclusion queries and does not change the knowledge captured by the model, M_4 must have an equal number of inclusion queries as M_3 , an equal or smaller number of exclusion queries than M_3 , and $M_4 \equiv M_3$. Hence, M_4 is a both positively and negatively optimized model of M_3 .

Now we prove the second one. Suppose that the model M exists. Then by Lemma 5.5 we know that patterns can only be composed if they are pairwise compatible, and by Lemma 5.6 we know that tuples in pairwise compatible patterns can only be merged if they are pairwise mergeable. Hence, all tuples in each $T_{ij} (1 \leq i \leq n, 1 \leq j \leq m)$ can be merged into one tuple of M_3 , and the total number of exclusion queries in the model M_3 is $\sum_{i=1}^n \sum_{j=1}^m |T_{ij}|$. By the given condition $\sum_{(\ell, r_4) \in M_4} |r_4^-| > \sum_{(\ell, r) \in M} |r^-|$, we know that $\sum_{i=1}^n \sum_{j=1}^m |T_{ij}|$ is not minimal when the compatibility and mergeability conditions are satisfied. This contradicts with the definition of the composition process. Hence, it is impossible to have $\sum_{(\ell, r_4) \in M_4} |r_4^-| > \sum_{(\ell, r) \in M} |r^-|$ and M does not exist. The proof is complete. \square

6. Complexity analysis

We analyze the complexity of the evaluation, containment and optimization problems for knowledge patterns in this section.

Generally, time- or space-complexity refers to a function $f(n)$ expressing the number of steps or the number of space units needed for the computation, where n is the size of the input. In analogy to the query evaluation problem in relational databases [41], we distinguish two kinds of complexity measures – data complexity and combined complexity² – in evaluating knowledge patterns on a database instance. The evaluation problem for knowledge patterns w.r.t. *combined complexity* is, given a knowledge pattern P and a database instance I , to find $P(I)$, where both the knowledge pattern P and the database instance I are part of the input. The evaluation problem for knowledge patterns w.r.t. *data complexity* is, given a database instance I , to find $P(I)$ for some fixed knowledge pattern P , where we only take the database instance I as the input. We can also state the evaluation problem for knowledge patterns w.r.t. data complexity and combined complexity as a decision problem:

- (1) Data complexity: Let P be some fixed knowledge pattern.

INSTANCE: Given a database instance I .
QUESTION: Is $P(I)$ non-empty, i.e., $P(I) \neq \emptyset$?

- (2) Combined complexity:

INSTANCE: Given a database instance I and a knowledge pattern P .
QUESTION: Is $P(I)$ non-empty, i.e., $P(I) \neq \emptyset$?

Theorem 6.1. *The evaluation problem for knowledge patterns is in PTIME w.r.t. data complexity, and is NP-complete w.r.t. combined complexity.*

Proof. We first show that the data complexity of the evaluation problem for knowledge patterns is in PTIME. For any fixed knowledge pattern $P = \langle \ell, r \rangle$, it is evaluated based on the evaluation of its inclusion queries in Σ_P^+ and its exclusion queries in Σ_P^- . The number $|\Sigma_P^+| + |\Sigma_P^-|$ is a constant, and each inclusion or exclusion query is a conjunctive query. Since it is well-known that the data complexity of evaluating conjunctive queries is in LOGSPACE [41], the data complexity of evaluating knowledge patterns is also in LOGSPACE.

Then we show that the combined complexity of the evaluation problem for knowledge patterns is NP-complete.

- NP membership: For a given knowledge pattern $P = \langle \ell, r \rangle$ with $|\Sigma_P^+| = m$ and $|\Sigma_P^-| = n$ and a given database instance I , we guess $P(I) \neq \emptyset$ and there are a number $\{t_1, \dots, t_n\}$ of tuples in $P(I)$. For this, we need to verify for each tuple t_k ($1 \leq k \leq n$) whether $t_k \in \varphi_i(I)$ for some $\varphi_i \in \Sigma_P^+$ and $t_k \notin \varphi_j(I)$ for every $\varphi_j \in \Sigma_P^-$. In the worst case, we need to verify whether t_k is in the result of an inclusion or exclusion query $m + n$ times. Since both inclusion queries in Σ_P^+ and exclusion queries in Σ_P^- are conjunctive queries, and the combined complexity of evaluating conjunctive queries is known to be NP-complete [12], we can verify whether each t_k in polynomial time. Hence, we can also verify whether $P(I) \neq \emptyset$ in polynomial time.
- NP hardness: We prove that the evaluation problem for conjunctive queries can be reduced to the evaluation problem for knowledge patterns in polynomial time. For this, we need to show that, given any conjunctive query $\varphi(x_1, \dots, x_n)$, a knowledge pattern P can be constructed from $\varphi(x_1, \dots, x_n)$ such that if the answer to $P(I)$ is YES, then the answer to $\varphi(x_1, \dots, x_n)$ over I is also YES, and if the answer to $P(I)$ is NO, then the answer to $\varphi(x_1, \dots, x_n)$ over I is also NO. For this, we construct such a knowledge pattern P as a pair $\langle \ell, r \rangle$, where $\ell = R(y_1, y'_1) \leftarrow \varphi(x_1, \dots, x_n) \wedge R'(y_1, \dots, y_m) \wedge R'(y'_1, \dots, y'_m)$ for an equivalence relation $R \in \mathcal{R}$ and a relation $R' \in \mathcal{Y}$ with y_1 and y'_1 over D_O , and r is as follows:

² We omit the discussion on expression complexity because it is rarely different from combined expression [41].

$$r = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline A_{x_1} & \dots & A_{x_n} & A_{y_2} & \dots & A_{y_m} & A_{y'_2} & \dots & A_{y'_m} & A^* \\ \hline \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & \lambda & + \\ \hline \end{array}$$

Since $R'(y_1, \dots, y_m) \wedge R'(y'_1, \dots, y'_m)$ can always return a non-empty result in a database instance (as we defined previously, each relation over $R' \in \mathcal{T}$ is not empty), the above reduction shows that an arbitrary conjunctive query can always correspond to a knowledge pattern as discussed above, which has exactly one inclusion query but no any exclusion queries. Because the combined complexity of evaluating conjunctive queries is known to be NP-complete [12], the evaluation problem for knowledge patterns is also NP-hard w.r.t. combined complexity. \square

In the following we discuss the complexity of the containment and optimization problems for knowledge patterns in terms of standard complexity theory [22].

Theorem 6.2. *The containment problem for knowledge patterns is NP-complete.*

Proof. We first show that the problem is in NP. Let $P_1 = \langle \ell_1, r_1 \rangle$ and $P_2 = \langle \ell_2, r_2 \rangle$ be two knowledge patterns, where $\Sigma_{P_1}^+$ (resp. $\Sigma_{P_2}^+$) has m_1 (resp. m_2) inclusion queries and $\Sigma_{P_1}^-$ (resp. $\Sigma_{P_2}^-$) has n_1 (resp. n_2) exclusion queries. Suppose that we want to determine whether or not $P_1 \subseteq P_2$ holds. Following from Theorem 4.3, we guess m_1 pairs (φ_k, ϕ_b) for each $\varphi_k \in \Sigma_{P_1}^+$ and some $\phi_b \in \Sigma_{P_2}^+$, and for each pair (φ_k, ϕ_b) we further guess n_2 pairs (ϕ'_j, φ'_i) for each $\phi'_j \in \Sigma_{P_2}^-$ and some $\varphi'_i \in \Sigma_{P_1}^-$. Then we verify that,

- for each pair (φ_k, ϕ_b) whether $\varphi_k \subseteq \phi_b$ holds, and
- for each pair (ϕ'_j, φ'_i) whether $\phi'_j \wedge \phi_b \subseteq \varphi'_i \wedge \phi_b$ holds.

Because the above containments between conjunctive queries can be verified in polynomial time [12], the containment problem for knowledge patterns is thus in NP.

Now we prove the NP hardness by showing that the containment problem for conjunctive queries can be reduced to the containment problem for knowledge patterns in polynomial time. That is, given two conjunctive queries φ_1 and φ_2 of the same arity, we construct two knowledge patterns P_1 and P_2 from φ_1 and φ_2 , respectively, such that if the answer to $P_1(I) \subseteq P_2(I)$ is YES, then the answer to $\varphi_1(I) \subseteq \varphi_2(I)$ is also YES, and if the answer to $P_1(I) \subseteq P_2(I)$ is NO, then the answer to $\varphi_1(I) \subseteq \varphi_2(I)$ is also NO. For this, we can construct the knowledge pattern P_1 (resp. P_2) from φ_1 (resp. φ_2) as described in the proof for Theorem 6.1. Again, since $R'(y_1, \dots, y_m) \wedge R'(y'_1, \dots, y'_m)$ used in the construction of knowledge patterns can be chosen from the ones that have non-empty results, $P_1(I) \subseteq P_2(I)$ holds iff $\varphi_1(I) \subseteq \varphi_2(I)$ holds. Because determining the containment between two conjunctive queries is known to be NP-complete [12], determining the containment problem for knowledge patterns is thus NP-hard. \square

Theorem 6.3. *Given a knowledge model M , the problem of finding a positively optimal model of M is NP-complete.*

Proof. We first show that finding a positively optimal model of M is NP-complete. Finding a positively optimal model of M involves two stages: normalization and elimination. At the normalization stage, normalizing patterns in the knowledge model M can be computed in polynomial time. At the elimination stage, reconciling these normalized knowledge patterns can also be computed in polynomial time; however, eliminating redundancy is a process of determining the containment of patterns, which by Theorem 6.2 is NP-complete. Hence, finding a positively optimal model of M is in NP. Then by Theorem 4.1, we know that the containment problem of conjunctive queries is NP-hard. Viewing that each conjunctive query is a normalized and reconciled pattern without tuples for exclusion queries, the containment problem of conjunctive queries is PTIME-reducible to the problem of finding a positively optimal model. More specifically, given two conjunctive queries φ_1 and φ_2 of the same arity, we can first construct two knowledge patterns P_1 and P_2 from φ_1 and φ_2 , respectively, in the same way as the two knowledge patterns P_1 and P_2 are constructed in Theorem 6.2. After this, we construct a knowledge model $M = \{P_1, P_2\}$, and check if we can find a positively optimal model M' of M . If $M' = \{P_1\}$ (resp. $M' = \{P_2\}$), then φ_1 (resp. φ_2) must contain φ_2 (resp. φ_1). If $M' = \{P_1, P_2\}$, then $\varphi_1 \not\subseteq \varphi_2$ and $\varphi_2 \not\subseteq \varphi_1$. Hence, finding a positively optimal model of M is also NP-hard. \square

To study the complexity of finding an optimal model for a given knowledge model, we recast the problem as a decision problem in terms of patterns that are obtained from M after the normalization and elimination stages:

- *Instance:* a knowledge model M and a positive integer k .
- For the set \mathcal{P} of patterns that are obtained from M after the normalization and elimination stages, is there a partition $\{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ of \mathcal{P} such that patterns in each \mathcal{P}_i ($1 \leq i \leq n$) are pairwise compatible, and a partition $\{\mathcal{T}_1, \dots, \mathcal{T}_m\}$ for each $\bigcup_{(\ell, r_i) \in \mathcal{P}_i} r_i^-$ such that tuples in each $\{\mathcal{T}_{ij} \mid 1 \leq j \leq m\}$ are pairwise mergeable and $\sum_{i=1}^n \sum_{j=1}^m |T_{ij}|$ is less than k ?

Theorem 6.4. *Given a knowledge model M , the problem of finding an optimal model of M is NP-complete.*

Proof. Since by [Theorem 6.3](#) finding a positively optimal model of M is NP-complete, by [Definition 5.1](#) (i.e., an optimal model of M must be a positively optimal model of M), finding an optimal model of M is NP-hard. Then we only need to prove that finding an optimal model of M is in NP.

We first show that this decision problem can be described in terms of graphs. A *clique* is a graph with every vertex connected to every other vertex. For a graph $\mathcal{G} = (V, E)$ we call $\mathcal{G}_1, \dots, \mathcal{G}_n$ a *clique cover* iff for all $i = 1, \dots, n$ $\mathcal{G}_i = (V_i, E_i)$ is a clique with pairwise disjoint $V_i \subseteq V$ satisfying $V = \bigcup_{i=1}^n V_i$ and $E_i \subseteq E$. Given a graph \mathcal{G} and an integer $k > 0$, the *clique cover problem* is to decide if a clique cover $\mathcal{G}_1, \dots, \mathcal{G}_k$ with k cliques exists. It is known that the clique cover problem is NP-complete [\[22\]](#). We construct two (undirected) graphs as follows.

- (1) **Compatibility graph:** Given a set \mathcal{P} of patterns that have been processed through the normalization and elimination stages, we construct the *compatibility graph* $\mathcal{C} = (V, E)$ for \mathcal{P} by adding a vertex $v \in V$ for each pattern in \mathcal{P} , and adding an edge $(v, v') \in E$ between two vertices v and v' if the patterns represented by these two vertices are compatible. By [Definition 5.4](#) for compatible patterns, we know that the construction of a compatibility graph can be done in polynomial time. A clique in the compatibility graph is a subset of patterns in \mathcal{P} that are pairwise compatible.
- (2) **Mergeability graph:** Given a set \mathcal{P} of patterns that have been processed through the normalization and elimination stages, we construct the *mergeability graph* $\mathcal{M} = (W, F, L)$ for \mathcal{P} by
 - adding a vertex $w \in W$ for each tuple in $\bigcup_{(\ell, r) \in \mathcal{P}} r^-$, i.e., each vertex $v \in V$ defines a set of vertices $W_v \subseteq W$ and $W = \bigcup_{v \in V} W_v$, and the vertices in W_v correspond to the negative tuples of the pattern associated with v ;
 - adding an edge $f \in F$ with $f = (w_1, w_2)$ between two vertices w_1, w_2 with $w_i \in W_{v_i}$ (i.e., w_i and v_i correspond to the same pattern) iff the tuples represented by the vertices w_1 and w_2 are mergeable with respect to the patterns represented by v_1 and v_2 , i.e., as the mergeability depends on a given set of pairwise compatible patterns, we consider only two such patterns here – the corresponding patterns of these tuples represented by w_1 and w_2 must be compatible;
 - associating with each edge $f = (w_1, w_2)$ in F a label $V_f \in L$, where $V_f \subseteq V$ is a set of vertices satisfying the following conditions:
 - if $w_1 \in W_{v_1}, w_2 \in W_{v_2}$, then $v_1, v_2 \in V_f$ holds, and
 - if $v_k \in V$ corresponds to the pattern (ℓ_k, r_k) , and t_1, t_2 (corresponding to w_1, w_2) satisfy [Eq. \(3\)](#) described in [Definition 5.5](#) with respect to r_k , then $v_k \in V_f$ holds.
 By [Definition 5.4](#) for compatible patterns and [Definition 5.5](#) for mergeable tuples, we know that the construction of a mergeability graph can also be done in polynomial time.

The intuition behind associating a label with each edge in the mergeability graph is to capture the mergeability of two tuples for more than two patterns involved. This is based on the observation that the mergeability condition for tuples t_1, t_2 requires a condition for each r_k , which only depends on t_1, t_2 and r_k . Thus, whenever a subset $V' \subseteq V_f$ defines a clique, the tuples t_1, t_2 are mergeable with respect to the union of all patterns associated with V' . Moreover, the above construction of the compatibility graph \mathcal{C} and the mergeability graph \mathcal{M} ensures that the following condition relating \mathcal{C} with \mathcal{M} must hold: for every edge $(w_1, w_2) \in F$ with $w_1 \in W_{v_1}, w_2 \in W_{v_2}$ we have an edge $(v_1, v_2) \in E$.

Now we rephrase the problem in terms of the compatibility graph \mathcal{C} and the mergeability graph \mathcal{M} as follows:

Let \mathcal{C}, \mathcal{M} and a positive integer k be given. Does there exist a clique cover $\mathcal{C}_1, \dots, \mathcal{C}_n$ of \mathcal{C} such that for the restrictions \mathcal{M}_i ($i = 1, \dots, n$) of \mathcal{M} with respect to $\mathcal{C}_i = (V_i, E_i)$ (i.e. $\mathcal{M}_i = (W_i, F_i)$ with $W_i = \bigcup_{v \in V_i} W_v$ and $F_i = \{(w_1, w_2) \in F \mid w_1, w_2 \in W_i\}$), we obtain clique covers $\mathcal{M}_{i1}, \dots, \mathcal{M}_{ik_i}$ such that for all $i = 1, \dots, n$, all $j = 1, \dots, k_i$ and all edges f in \mathcal{M}_{ij} we have $V_i \subseteq V_f$ (where V_i is the set of vertices of \mathcal{C}_i) and $k = \sum_{i=1}^n k_i$?

The condition that relates the clique covers in \mathcal{C} and \mathcal{M} is justified as follows: the cliques \mathcal{C}_i correspond to pairwise compatible knowledge patterns, so the tuples in these patterns can be merged by set union in accordance with [Lemma 5.5](#). Then a subset of (negative) tuples in this union can be merged into a single tuple iff they are pairwise mergeable in accordance with [Lemma 5.6](#) (i.e. a clique in \mathcal{M}) with respect to the whole union (i.e. $V_i \subseteq V_f$).

To show that this problem is in NP, we guess non-deterministically a set of subgraphs $\mathcal{M}_1, \dots, \mathcal{M}_k$ of \mathcal{M} and simultaneously a set of subgraphs $\mathcal{C}_1, \dots, \mathcal{C}_\ell$ of \mathcal{C} . Then it is well known that we can check in polynomial time, if $\mathcal{M}_1, \dots, \mathcal{M}_k$ is a clique cover of \mathcal{M} and likewise, if $\mathcal{C}_1, \dots, \mathcal{C}_\ell$ is a clique cover of \mathcal{C} . Then based on the construction of the two graphs, we know that for each edge (w_1, w_2) of \mathcal{M}_i there is an edge (v_1, v_2) in \mathcal{C} with $w_1 \in W_{v_1}, w_2 \in W_{v_2}$, and thus each of the cliques \mathcal{M}_i determines a clique \mathcal{C}'_i in \mathcal{C} . Now we have to check that for each $i = 1, \dots, k$ there is some $j = j(i) \in \{1, \dots, \ell\}$ such that \mathcal{C}'_i is a subgraph of \mathcal{C}_j . Since each vertex in \mathcal{C}'_i represents a distinct pattern, and each vertex in \mathcal{C}_j also represents a distinct pattern, and a vertex in \mathcal{C}'_i must be matched to a vertex in \mathcal{C}_j , which represent the same pattern, so this checking process can be done in polynomial time. It remains to check that for each $i = 1, \dots, k$ and each edge f in \mathcal{M}_i we also have that all vertices of $\mathcal{C}_{j(i)}$ belong to V_f , which can be done in polynomial time.

Taking into account all the complexity results at the normalization, elimination and composition stages, we can conclude that the decision problem for finding an optimal model of M is NP-complete. \square

Remark 6.1. For the problems that are in NP, by Fagin's theorem [18] it should be possible to show that the problems can be formulated by an existential second-order formula. For the above decision problem defined in terms of the compatibility graph \mathcal{C} and the mergeability graph \mathcal{M} , this formula would be $\exists \mathcal{M}_1, \dots, \mathcal{M}_k \exists \mathcal{C}_1, \dots, \mathcal{C}_\ell. \Phi$, where $\mathcal{M}_1, \dots, \mathcal{M}_k$ of \mathcal{M} and $\mathcal{C}_1, \dots, \mathcal{C}_\ell$ of \mathcal{C} are the clique covers of \mathcal{M} and \mathcal{C} , respectively (as in the proof above), and Φ is a first-order formula expressing all the conditions that were used in the proof such as checking that we have clique covers, that cliques \mathcal{M}_i determine unique cliques \mathcal{C}'_i , that \mathcal{C}'_i is a subgraph of $\mathcal{C}_{j(i)}$, and that all vertices of $\mathcal{C}_{j(i)}$ belong to V_f for all edges f in \mathcal{M}_i .

7. Related work

A large number of research efforts have been directed at investigating the entity resolution problem under different representations such as de-duplication [9,17], record linkage [21], object identification [27,35] and data matching [14]. The predominant research approaches are similarity-based, which classify entities as matching or non-matching pairs using various techniques, e.g., probabilistic [21,26], ruled-based [15], supervised [23], and active learning [30,39,5] approaches. Compared with studies that largely focus on the syntactic similarity of entities, the collective entity resolution techniques proposed by [10,16] also take into account the relationships between entities so as to resolve entities of different types in a collective way. The paper by O. Benjelloun et al. [8] formalizes the generic entity resolution problem by treating the functions for comparing and merging records as black-boxes, and analyzes important properties that enable efficient entity resolution algorithms. A book by P. Christen [14] provides a comprehensive survey for various research works in this field.

Despite the increasing importance for identifying entities by sharing knowledge and working collaboratively within a wide range of communities, there is very little research looking into knowledge representation and reasoning of entity resolution. In [36] entity resolution was studied using Markov logic that combines first-order logic and probabilistic models by attaching weights to first-order logic formulas. Recently, some research efforts have been directed to incorporating constraints into similarity-based methods [6,13,19,20,33], such as matching dependencies [19,20]. The work by Z. Bahmani et al. [7] also developed a declarative approach to compute clean database instances as determined by matching dependencies. Different from these existing approaches, we present a theoretical framework that uses defaults and exceptions to leverage knowledge about entity resolution. This framework can represent, manage and reason about domain specific knowledge with the following characteristics: (1) knowledge can be represented at flexible levels of abstraction, (2) exceptions to a default rule can be expressed using a certain form of negation in a knowledge pattern, and (3) it has an ability to match entities that are not similar and also separate entities that are similar. Compared with matching dependencies [19,20], in addition to the above characteristics, our framework has also the advantage of supporting collective entity resolution. In [6] a Datalog-style language was developed to allow users to specify hard and soft constraints. The use of such hard constraints may cause conflicts, and reasoning about such conflicts is often computationally expensive. Moreover, the expressive power of the Datalog-style language in [6] is limited. It cannot capture exceptions to domain-independent constraints, although exceptions commonly exist in many entity resolution applications. We should also emphasize that our work in this paper does not specify how to merge entities so as to lead to clean database instances as studied in [8]. Recent works on crowd-sourcing entity resolution, such as CrowdER [43], also attempted to bring human insight into the entity resolution process by requiring users to verify matching pairs of entities in crowd-sourcing platforms (e.g., Amazon Mechanical Turk and Crowdflower). However, these approaches are often expensive and difficult to scale unless trading-off accuracy by only requiring to check the most likely matching pairs. People in the crowd often have little or no domain expertise, and thus cannot accurately decide matching pairs in many cases.

The potential of our knowledge-based approach is to *complement* traditional similarity approaches rather than complete with them. In doing so, the accuracy of entity resolution can be continually improved over time, beyond the limits of traditional similarity techniques. The work in this paper is still in the early stages towards formally understanding the entity resolution problem from a knowledge management perspective. In our preliminary work [32], we investigated knowledge patterns that can revolve entities of a single type, and discussed decidability and complexity of the containment problem for such knowledge patterns. In [31] we developed knowledge-aware identity services in the service-oriented environment along with a small experimental study on the Scopus data set and an institutional repository that verifies the effectiveness of using knowledge for entity solution in practice. In this paper, we extend knowledge patterns in [32] into a more general framework in which entities of different types can be collectively resolved. In addition to this, the optimization problem of knowledge models is studied. Studying this problem is particularly important for understanding how identity knowledge can be managed in an effective and efficient way.

Since a knowledge pattern is essentially represented by a fragment of first-order logic, the optimization problem discussed in our work is thus related to the previous studies on the query containment and optimization problems in database theory [2]. It is well-known that query containment is decidable for conjunctive queries [12,3] and union of conjunctive queries [29], but not decidable for first-order queries and Datalog queries [11,34]. The papers [24,25,40,44] studied the containment problem for conjunctive queries with inequality, and the complexity of the containment problem for conjunctive queries with inequality is proved to be Π_2^P -complete [40]. Due to a certain form of negation permitted in knowledge patterns of our formalism for obtaining sufficient expressive power, both the containment and the optimization problems become challenging. This drives us to study these important formal properties associated with the proposed framework.

8. Conclusion

In this paper we have presented an expressive formalism for knowledge representation of entity resolution. To understand how efficiently knowledge patterns can be processed, and how effectively redundancy among knowledge patterns can be removed, we studied the evaluation and containment problems for knowledge patterns. Our result shows that the evaluation problem for knowledge patterns is NP-complete w.r.t. combined complexity but in PTIME w.r.t. data complexity, and the containment problem for knowledge patterns is NP-complete. We further investigated the optimization problem for knowledge patterns that is useful for transforming a given knowledge model into an equivalent model that can be evaluated more efficiently. It turns out that finding a positively optimal model is NP-complete, and finding an optimal model that also minimizes the number of exclusion queries in positively optimal models is still NP-complete. We plan to further investigate approximate solutions for finding an optimal representation of a knowledge model in future work.

Our framework can support traditional similarity-based methods by defining similarity relations in database schemas. The results on the evaluation, containment and optimisation problems for knowledge patterns still hold with this extension, i.e., all the relevant proofs remain valid with the addition of similarity relations. In addition to combining with similarity-based techniques, there are also other possible lines of extensions to our work, such as to incorporate probabilities and contexts into this knowledge-based framework. This would allow us to reason about probabilities of entity resolution in connection to contexts of interest—a promising direction for future research on entity resolution.

References

- [1] Fischer identity as a service. Architecture overview for client organizations. White paper, Fischer International Identity, 2009.
- [2] S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995.
- [3] A. Aho, Y. Sagiv, J. Ullman, Equivalences among relational expressions, *SIAM J. Comput.* 8 (1979) 218.
- [4] A. Aho, J. Ullman, Universality of data retrieval languages, in: *Proceedings of Principles of Programming Languages*, ACM, 1979, pp. 110–119.
- [5] A. Arasu, M. Götz, R. Kaushik, On active learning of record matching packages, in: *Proceedings of the 2010 International Conference on Management of Data*, ACM, 2010, pp. 783–794.
- [6] A. Arasu, C. Ré, D. Suciu, Large-scale deduplication with constraints using dedupalog, in: *ICDE'09*, IEEE, 2009, pp. 952–963.
- [7] Z. Bahmani, L. Bertossi, S. Kolahi, L. Lakshmanan, Declarative entity resolution via matching dependencies and answer set programs, in: *Proc. KR*, 2012, pp. 380–390.
- [8] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. Whang, J. Widom, Swoosh: a generic approach to entity resolution, *VLDB J.* 18 (1) (2009) 255–276.
- [9] I. Bhattacharya, L. Getoor, Deduplication and group detection using links, in: *Proceedings of the 2004 ACM SIGKDD Workshop on Link Analysis and Group Detection*, 2004.
- [10] I. Bhattacharya, L. Getoor, Collective entity resolution in relational data, *ACM Trans. Knowl. Discov. Data* 1 (1) (2007) 5.
- [11] D. Calvanese, G. De Giacomo, M.Y. Vardi, Decidable containment of recursive queries, *Theoret. Comput. Sci.* 336 (May 2005) 33–56.
- [12] A.K. Chandra, P.M. Merlin, Optimal implementation of conjunctive queries in relational data bases, in: *STOC*, 1977, pp. 77–90.
- [13] S. Chaudhuri, A. Das Sarma, V. Ganti, R. Kaushik, Leveraging aggregate constraints for deduplication, in: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ACM, 2007, pp. 437–448.
- [14] P. Christen, *Data Matching*, Springer, 2012.
- [15] W. Cohen, Data integration using similarity joins and a word-based information representation language, *ACM Trans. Inform. Syst.* 18 (3) (2000) 288–321.
- [16] X. Dong, A. Halevy, J. Madhavan, Reference reconciliation in complex information spaces, in: *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ACM, 2005, pp. 85–96.
- [17] A.K. Elmagarmid, P.G. Ipeirotis, V.S. Verykios, Duplicate record detection: a survey, *IEEE Trans. Knowl. Data Eng.* 19 (2007) 1–16.
- [18] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, in: *SIAM-AMS Proceedings*, vol. 7, 1974, pp. 43–73.
- [19] W. Fan, H. Gao, X. Jia, J. Li, S. Ma, Dynamic constraints for record matching, *VLDB J.* 20 (4) (2011) 495–520.
- [20] W. Fan, X. Jia, J. Li, S. Ma, Reasoning about record matching rules, *Proc. VLDB Endow.* 2 (1) (2009) 407–418.
- [21] I. Fellegi, A. Sunter, A theory for record linkage, *J. Amer. Statist. Assoc.* 64 (328) (1969) 1183–1210.
- [22] M. Garey, D. Johnson, *Computers and Intractability. A Guide to the Theory of NP Completeness*, WH Freeman and Company, New York, 1979.
- [23] J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2006.
- [24] A. Klug, On conjunctive queries containing inequalities, *J. ACM* 35 (January 1988) 146–160.
- [25] M. Leclere, M. Mugnier, Some algorithmic improvements for the containment problem of conjunctive queries with negation, in: *ICDT*, 2007, 2007, pp. 404–418.
- [26] H. Newcombe, J. Kennedy, Record linkage: making maximum use of the discriminating power of identifying information, *Commun. ACM* 5 (11) (1962) 563–566.
- [27] H. Pasula, B. Marthi, B. Milch, S. Russell, I. Shpitser, Identity uncertainty and citation matching, in: *NIPS*, 2003.
- [28] Y. Sagiv, M. Yannakakis, Equivalences among relational expressions with the union and difference operators, *J. ACM* 27 (4) (1980) 633–655.
- [29] Y. Sagiv, M. Yannakakis, Equivalences among relational expressions with the union and difference operators, *J. ACM* 27 (October 1980) 633–655.
- [30] S. Sarawagi, A. Bhamidipaty, Interactive deduplication using active learning, in: *KDD*, 2002, pp. 269–278.
- [31] K.-D. Schewe, Q. Wang, Knowledge-aware identity services, *Knowl. Inf. Syst.* (2012) 1–23.
- [32] K.-D. Schewe, Q. Wang, On the decidability and complexity of identity knowledge representation, in: *DASFAA*, Springer, 2012, pp. 288–302.
- [33] W. Shen, X. Li, A. Doan, Constraint-based entity matching, in: *Proceedings of the 20th National Conference on Artificial Intelligence*, AAAI'05, vol. 2, 2005, pp. 862–867.
- [34] O. Shmueli, Equivalence of datalog queries is undecidable, *J. Log. Program.* 15 (February 1993) 231–241.
- [35] P. Singla, P. Domingos, Object identification with attribute-mediated dependences, in: *KDD*, 2005, pp. 297–308.
- [36] P. Singla, P. Domingos, Entity resolution with Markov logic, in: *ICDM*, 2006, pp. 572–582.
- [37] S. Slone, The open group identity management work area. Identity management. White paper, The Open Group, 2004.
- [38] A. Swan, Author identification web page, <http://repinf.pbworks.com/Author-identification>.
- [39] S. Tejada, C.A. Knoblock, S. Minton, Learning object identification rules for information integration, *Inf. Syst.* 26 (2001) 2001.
- [40] J. Ullman, Information integration using logical views, *Theoret. Comput. Sci.* 239 (May 2000) 189–210.

- [41] M. Vardi, The complexity of relational query languages, in: *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, ACM, 1982, pp. 137–146.
- [42] V. Verykios, G. Moustakides, M. Efkery, A Bayesian decision model for cost optimal record matching, *VLDB J.* 12 (1) (2003) 28–40.
- [43] J. Wang, T. Kraska, M. Franklin, J. Feng, Crowder: crowdsourcing entity resolution, *VLDB Endow.* 5 (11) (2012) 1483–1494.
- [44] F. Wei, G. Lausen, Containment of conjunctive queries with safe negation, in: *Proceedings of the 9th International Conference on Database Theory*, ICDT '03, Springer-Verlag, 2002, pp. 346–360.