Knowledge-Based Matching of n-ary Tuples

Pierre Monnin, Miguel Couceiro, Amedeo Napoli, Adrien Coulet

Knowledge-Based Matching of n-ary Tuples[⋆]

Pierre Monnin[0000−0002−2017−8426], Miguel Couceiro[0000−0003−2316−7623], Amedeo Napoli, and Adrien Coulet[0000−0002−1466−062X]

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
{pierre.monnin, miguel.couceiro, amedeo.napoli, adrien.coulet}@loria.fr

Abstract. An increasing number of data and knowledge sources are accessible by human and software agents in the expanding Semantic Web. Sources may differ in granularity or completeness, and thus be complementary. Consequently, they should be reconciled in order to unlock the full potential of their conjoint knowledge. In particular, units should be matched within and across sources, and their level of relatedness should be classified into equivalent, more specific, or similar. This task is challenging since knowledge units can be heterogeneously represented in sources (*e.g.*, in terms of vocabularies). In this paper, we focus on matching n-ary tuples in a knowledge base with a rule-based methodology. To alleviate heterogeneity issues, we rely on domain knowledge expressed by ontologies. We tested our method on the biomedical domain of pharmacogenomics by searching alignments among 50,435 n-ary tuples from four different real-world sources. Results highlight noteworthy agreements and particularities within and across sources.

Keywords: Alignment · Matching · n-ary Tuple · Order · Ontology

1 Introduction

In the Semantic Web [4], data or knowledge sources often describe similar units but may differ in quality, completeness, granularity, and vocabularies. Unlocking the full potential of the knowledge that these sources conjointly express requires matching equivalent, more specific, or similar knowledge units within and across sources. This matching process results in alignments that enable the reconciliation of these sources, *i.e.*, the harmonization of their content [7]. Such a reconciliation then provides a consolidated view of a domain that is useful in many applications, *e.g.*, in knowledge fusion and fact-checking.

Here, we illustrate the interest of such a matching process to reconcile knowledge within the biomedical domain of pharmacogenomics (PGx), which studies the influence of genetic factors on drug response phenotypes. PGx knowledge originates from distinct sources: reference databases such as PharmGKB, biomedical literature, or the mining of Electronic Health Records of hospitals.

[⋆] Supported by the *PractiKPharma* project, founded by the French National Research Agency (ANR) under Grant ANR15-CE23-0028, by the IDEX "Lorraine Université d'Excellence" (15-IDEX-0004), and by the *Snowball* Inria Associate Team.

Knowledge represented in these sources may differ in levels of validation, completeness, and granularity. Consequently, reconciling these sources would provide a consolidated view on the knowledge of this domain, certainly beneficial in precision medicine, which aims at tailoring drug treatments to patients to reduce adverse effects and maximize drug efficacy [5, 6]. PGx knowledge consists of n-ary relationships, here represented as tuples relating sets of drugs, sets of genomic variations, and sets of phenotypes. Such an n-ary tuple states that a patient being treated with the specified sets of drugs, while having the specified genomic variations will be more likely to experience the given phenotypes, *e.g.*, adverse effects. For example, Figure 1 depicts the tuple `pgt_1`, which states that patients treated with warfarin may experience cardiovascular diseases because of variations in the CYP2C9 gene. If a source contained the same tuple but with the genetic factor unknown, then it should be identified as less specific than `pgt_1`. Conversely, if a source contained the same tuple but with myocardial infarction as phenotype, then it should be identified as more specific than `pgt_1`.

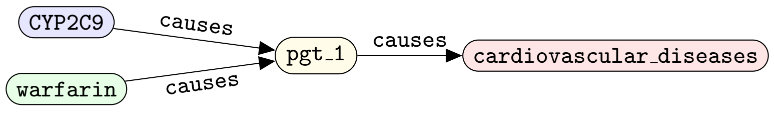


Fig. 1. Representation of a PGx relationship between gene `CYP2C9`, drug `warfarin` and phenotype `cardiovascular_diseases`. It can be seen as an n-ary tuple `pgt_1` = ({`warfarin`}, {`CYP2C9`}, {`cardiovascular_diseases`}). This tuple is reified through the individual `pgt_1`, connecting its components through the `causes` predicate.

Motivated by this application, we propose a general and mathematically well-founded methodology to match n-ary tuples. Precisely, given two n-ary tuples, we aim at deciding on their relatedness among five levels such as being equivalent or more specific. We suppose that such tuples are represented within a knowledge base that is expressed using Semantic Web standards. In such standards, only binary predicates exist, which requires the reification of n-ary tuples to represent them: tuples are individualized and linked to their components by predicates (see Figure 1) [12]. In these knowledge bases, entities can also be associated with ontologies, *i.e.*, formal representations of a domain [9]. Ontologies consist of classes and predicates, partially ordered by the subsumption relation, denoted by $\sqsubseteq$. This relation states that a class (respectively a predicate) is more specific than another.

The process of matching n-ary tuples appears naturally in the scope of ontology matching [7], *i.e.*, finding equivalences or subsumptions between classes, predicates, or instances of two ontologies. Here, we match individuals representing reified n-ary tuples, which is somewhat related to instance matching and the extraction of *linkkeys* [2]. However, we allow ourselves to state that a tuple is more specific than another, which is unusual in instance matching but common when matching classes or predicates with systems such as PARIS [14] and AMIE [8]. Besides, to the best of our knowledge, works available in the literature do not deal with the complex task of matching n-ary tuples with potentially unknown arguments formed by sets of individuals. See Appendix A for further details.

In our approach, we assume that the tuples to match have the same arity, the same indices for their arguments, and that they are reified with the same predicates and classes. Arguments are formed by sets of individuals (no literal values) and may be unknown. This matching task thus reduces to comparing each argument of the tuples and aggregating these comparisons to establish their level of relatedness. We achieve this process by defining five general rules, designed to satisfy some desired properties such as transitivity and symmetry. To tackle the heterogeneity in the representation of tuples, we enrich this structure-based comparison with domain knowledge, *e.g.*, the hierarchy of ontology classes and links between individuals.

This paper is organized as follows. In Section 2, we formalize the problem of matching n-ary tuples. To tackle it, we propose two preorders in Section 3 to compare sets of individuals by considering domain knowledge: links between individuals, instantiations, and subsumptions. These preorders are used in Section 4 to define matching rules that establish the level of relatedness between two n-ary tuples. These rules are applied to PGx knowledge in Section 5. We discuss our results in Section 6 and present some directions of future work in Section 7. Appendices are available online (https://arxiv.org/abs/2002.08103).

2 Problem Setting

We aim at matching n-ary tuples represented within a knowledge base $\mathcal{K}$, *i.e.*, we aim at determining the relatedness level of two tuples t_1 and t_2 (*e.g.*, whether they are equivalent, more specific, or similar). $\mathcal{K}$ is represented in the formalism of Description Logics (DL) [3] and thus consists of a TBox and an ABox.

Precisely, we consider a set $\mathcal{T}$ of n-ary tuples to match. This set is formed by tuples whose matching makes sense in a given application. For example, in our use-case, $\mathcal{T}$ consists of all PGx tuples from the considered sources. All tuples in $\mathcal{T}$ have the same arity n, and their arguments are sets of individuals of $\mathcal{K}$. Such a tuple t can be formally represented as $t = (\pi_1(t), \ldots, \pi_n(t))$, where $\pi_i : \mathcal{T} \to 2^{\Delta}$ is a mapping that associates each tuple t to its i-th argument $\pi_i(t)$, which is a set of individuals included in the domain of interpretation Δ. The index set is the same for all tuples in $\mathcal{T}$. Tuples come from potentially noisy sources and some arguments may be missing. As $\mathcal{K}$ verifies the Open World Assumption, such arguments that are not explicitly specified as empty, can only be considered unknown and they are set to Δ to express the fact that all individuals may apply. To illustrate, `pgt_1` in Figure 1 could be seen as a ternary tuple `pgt_1` $= (\{$`warfarin`$\}, \{$`CYP2C9`$\}, \{$`cardiovascular_diseases`$\})$, where arguments respectively represent the sets of involved drugs, genetic factors, and phenotypes.

In view of our formalism, matching two n-ary tuples t_1 and t_2 comes down to comparing their arguments $\pi_i(t_1)$ and $\pi_i(t_2)$ for each $i \in \{1, \dots, n\}$. For instance, if $\pi_i(t_1) = \pi_i(t_2)$ for all i, then t_1 and t_2 are representing the same knowledge unit, highlighting an agreement between their sources. In the next section, we propose other tests between arguments that are based on domain knowledge.

3 Ontology-Based Preorders

As previously illustrated, the matching of two n-ary tuples t_1 and t_2 relies on the comparison of each of their arguments $\pi_i(t_1)$ and $\pi_i(t_2)$, which are sets of individuals. Such a comparison can be achieved by testing their inclusion or equality. Thus, if $\pi_i(t_1) \subseteq \pi_i(t_2)$, then $\pi_i(t_1)$ can be considered as more specific than $\pi_i(t_2)$. It is noteworthy that testing inclusion or equality implicitly considers `owl:sameAs` links that indicate identical individuals. For example, the comparison of $\{\mathtt{e}_1\}$ with $\{\mathtt{e}_2\}$ while knowing that $\mathtt{owl{:}sameAs}(\mathtt{e}_1, \mathtt{e}_2)$ results in an equality. However, additional domain knowledge can be considered to help tackle the heterogeneous representation of tuples. For instance, some individuals can be *part of* others. Individuals may also instantiate different ontological classes, which are themselves comparable through subsumption. To consider this domain knowledge in the matching process, we propose two preorders, *i.e.*, reflexive and transitive binary relations.

3.1 Preorder $\preccurlyeq^{\mathtt{p}}$ Based on Links Between Individuals

Several links may associate individuals in $\pi_i(t_j)$ with other individuals in $\mathcal{K}$. Some links involve a transitive and reflexive predicate (*i.e.*, a preorder). Then, for each such predicate $\mathtt{p}$, we define a preorder $\preccurlyeq^{\mathtt{p}}$ parameterized by $\mathtt{p}$ as follows[1]:

$$\pi_i(t_1) \preccurlyeq^{\mathtt{p}} \pi_i(t_2) \Leftrightarrow \forall e_1 \in \pi_i(t_1),\ \exists e_2 \in \pi_i(t_2),\ \mathcal{K} \models \mathtt{p}(e_1, e_2) \qquad (1)$$

Note that, from the reflexivity of $\mathtt{p}$ and the use of quantifiers $\forall$ and $\exists$, $\pi_i(t_1) \subseteq \pi_i(t_2)$ implies $\pi_i(t_1) \preccurlyeq^{\mathtt{p}} \pi_i(t_2)$. The equivalence relation $\sim^{\mathtt{p}}$ associated with $\preccurlyeq^{\mathtt{p}}$ is defined as usual by:

$$\pi_i(t_1) \sim^{\mathtt{p}} \pi_i(t_2) \Leftrightarrow \pi_i(t_1) \preccurlyeq^{\mathtt{p}} \pi_i(t_2) \text{ and } \pi_i(t_2) \preccurlyeq^{\mathtt{p}} \pi_i(t_1) \qquad (2)$$

3.2 Preorder $\preccurlyeq^{\mathcal{O}}$ Based on Instantiation and Subsumption

The second preorder we propose takes into account classes of an ontology $\mathcal{O}$ ordered by subsumption and instantiated by individuals in $\pi_i(t_j)$. We denote by classes$(\mathcal{O})$ the set of all classes of $\mathcal{O}$. As it is standard in DL, $\top$ denotes the largest class in $\mathcal{O}$. Given an individual e, we denote by ci$(\mathcal{O}, e)$ the set of classes of $\mathcal{O}$ instantiated by e and distinct from $\top$, *i.e.*,

$$\mathrm{ci}(\mathcal{O}, e) = \{C \in \mathrm{classes}(\mathcal{O}) \setminus \{\top\} \mid \mathcal{K} \models C(e)\}\,.$$

[1] See Appendix B and Appendix C for the proof and examples.

Note that $\mathrm{ci}(\mathcal{O}, e)$ may be empty. We explicitly exclude $\top$ from $\mathrm{ci}(\mathcal{O}, e)$ since $\mathcal{K}$ may be incomplete. Indeed, individuals may lack instantiations of specific classes but instantiate $\top$ by default. Thus, $\top$ is excluded to prevent $\preccurlyeq^{\mathcal{O}}$ from inadequately considering these individuals more general than individuals instantiating classes other than $\top$[2].

Given $\mathcal{C} = \{C_1,\ C_2,\ \ldots,\ C_k\} \subseteq \mathrm{classes}(\mathcal{O})$, we denote by $\mathrm{msc}(\mathcal{C})$ the set of the most specific classes of $\mathcal{C}$, *i.e.*, $\mathrm{msc}(\mathcal{C}) = \{C \in \mathcal{C} \mid \nexists D \in \mathcal{C},\ D \sqsubset C\}$[3]. Similarly, we denote by $\mathrm{msci}(\mathcal{O}, e)$ the set of the most specific classes of $\mathcal{O}$, except $\top$, instantiated by an individual e, *i.e.*, $\mathrm{msci}(\mathcal{O}, e) = \mathrm{msc}(\mathrm{ci}(\mathcal{O}, e))$.

Given an ontology $\mathcal{O}$, we define the preorder $\preccurlyeq^{\mathcal{O}}$ based on set inclusion and subsumption as follows[4]:

$$\pi_i(t_1) \preccurlyeq^{\mathcal{O}} \pi_i(t_2) \Leftrightarrow \forall e_1 \in \pi_i(t_1),\ \Big[\underbrace{e_1 \in \pi_i(t_2)}_{(3a)}\Big] \bigvee \Big[\mathrm{msci}(\mathcal{O}, e_1) \neq \emptyset \wedge \\ \underbrace{\forall C_1 \in \mathrm{msci}(\mathcal{O}, e_1),\ \exists e_2 \in \pi_i(t_2),\ \exists C_2 \in \mathrm{msci}(\mathcal{O}, e_2),\ C_1 \sqsubseteq C_2}_{(3b)}\Big] \quad (3)$$

Clearly, if $\pi_i(t_1)$ is more specific than $\pi_i(t_2)$ and $e_1 \in \pi_i(t_1)$, then (3a) $e_1 \in \pi_i(t_2)$, or (3b) all the most specific classes instantiated by e_1 are subsumed by at least one of the most specific classes instantiated by individuals in $\pi_i(t_2)$. Thus individuals in $\pi_i(t_2)$ can be seen as "more general" than those in $\pi_i(t_1)$. As before, $\preccurlyeq^{\mathcal{O}}$ induces the equivalence relation $\sim^{\mathcal{O}}$ defined by:

$$\pi_i(t_1) \sim^{\mathcal{O}} \pi_i(t_2) \Leftrightarrow \pi_i(t_1) \preccurlyeq^{\mathcal{O}} \pi_i(t_2) \text{ and } \pi_i(t_2) \preccurlyeq^{\mathcal{O}} \pi_i(t_1) \quad (4)$$

The preorder $\preccurlyeq^{\mathcal{O}}$ can be seen as parameterized by the ontology $\mathcal{O}$, allowing to consider different parts of the TBox of $\mathcal{K}$ for each argument $\pi_i(t_j)$, if needed.

4 Using Preorders to Define Matching Rules

Let $t_1, t_2 \in \mathcal{T}$ be two n-ary tuples to match. We assume that each argument $i \in \{1, \ldots, n\}$ is endowed with a preorder $\preccurlyeq_i \in \left\{\subseteq, \preccurlyeq^{\mathrm{P}}, \preccurlyeq^{\mathcal{O}}\right\}$ that enables the comparison of $\pi_i(t_1)$ and $\pi_i(t_2)$. We can define rules that aggregate such comparisons for all $i \in \{1, \ldots, n\}$ and establish the relatedness level of t_1 and t_2. Hence, our matching approach comes down to applying these rules to every ordered pair (t_1, t_2) of n-ary tuples from $\mathcal{T}$.

Here, we propose the following five relatedness levels: $=$, $\sim$, $\preccurlyeq$, $\lesseqgtr$, and $\propto$, from the strongest to the weakest. Accordingly, we propose five matching rules of the form $B \Rightarrow H$, where B expresses the conditions of the rule, testing equalities, equivalences, or inequalities between arguments of t_1 and t_2. Classically, these

[2] See Appendix D for a detailed example.
[3] $D \sqsubset C$ means that $D \sqsubseteq C$ and $D \not\equiv C$.
[4] See Appendix E and Appendix F for the proof and examples.

conditions can be combined using conjunctions or disjunctions, respectively denoted by $\wedge$ and $\vee$. If B holds, H expresses the relatedness between t_1 and t_2 to add to $\mathcal{K}$. Rules are applied from Rule 1 to Rule 5. Once conditions in B hold for a rule, H is added to $\mathcal{K}$ and the following rules are discarded, meaning that at most one relatedness level is added to $\mathcal{K}$ for each pair of tuples. When no rule can be applied, t_1 and t_2 are considered incomparable and nothing is added to $\mathcal{K}$. The first four rules are the following:

Rule 1. $\forall i \in \{1, \ldots, n\}\,,\ \pi_i(t_1) = \pi_i(t_2) \Rightarrow t_1 = t_2$

Rule 2. $\forall i \in \{1, \ldots, n\}\,,\ \pi_i(t_1) \sim_i \pi_i(t_2) \Rightarrow t_1 \sim t_2$

Rule 3. $\forall i \in \{1, \ldots, n\}\,,\ \pi_i(t_1) \preccurlyeq_i \pi_i(t_2) \Rightarrow t_1 \preccurlyeq t_2$

Rule 4. $\forall i \in \{1, \ldots, n\}\,,\ [\ (\pi_i(t_1) = \pi_i(t_2)) \vee (\pi_i(t_2) \neq \varDelta \wedge \pi_i(t_1) \preccurlyeq_i \pi_i(t_2))\ \vee$ $(\pi_i(t_1) \neq \varDelta \wedge \pi_i(t_2) \preccurlyeq_i \pi_i(t_1))\ ] \Rightarrow t_1 \lesseqgtr t_2$

Rule 1 states that t_1 and t_2 are identical ($=$) whenever t_1 and t_2 coincide on each argument. Rule 2 states that t_1 and t_2 are equivalent ($\sim$) whenever each argument $i \in \{1, \ldots, n\}$ of t_1 is equivalent to the same argument of t_2. Rule 3 states that t_1 is more specific than t_2 ($\preccurlyeq$) whenever each argument $i \in \{1, \ldots, n\}$ of t_1 is more specific than the same argument of t_2 w.r.t. $\preccurlyeq_i$. Rule 4 states that t_1 and t_2 have comparable arguments ($\lesseqgtr$) whenever they have the same specified arguments (*i.e.*, different from $\varDelta$), and these arguments are comparable w.r.t. $\preccurlyeq_i$. Rules 1 to 3 satisfy the transitivity property. Additionally, Rules 1, 2, and 4 satisfy the symmetry property.

In Rules 1 to 4, comparisons are made argument-wise. However, other relatedness cases may require to aggregate over arguments. For example, we may want to compare all individuals involved in two tuples, regardless of their arguments. Alternatively, we may want to consider two tuples as weakly related if their arguments have a specified proportion of comparable individuals. To this aim, we propose Rule 5. Let $\mathbb{I} = \{I_1, \ldots, I_m\}$ be a partition of $\{1, \ldots, n\}$, defined by the user at the beginning of the matching process. We define the aggregated argument I_k of t_j as the union of all specified $\pi_i(t_j)$ (*i.e.*, different from $\varDelta$) for $i \in I_k$. Formally,

$$\pi_{I_k}(t_j) = \bigcup_{\substack{i \in I_k \\ \pi_i(t_j) \neq \varDelta}} \pi_i(t_j).$$

We assume that each aggregated argument $I_k \in \mathbb{I}$ is endowed with a preorder $\preccurlyeq_{I_k} \in \left\{\subseteq, \preccurlyeq^{\mathtt{P}}, \preccurlyeq^{\mathcal{O}}\right\}$. We denote by $\mathrm{SSD}(\pi_{I_k}(t_1), \pi_{I_k}(t_2))$ the semantic set difference between $\pi_{I_k}(t_1)$ and $\pi_{I_k}(t_2)$, *i.e.*,

$$\mathrm{SSD}(\pi_{I_k}(t_1), \pi_{I_k}(t_2)) = \{e_1 \mid e_1 \in \pi_{I_k}(t_1) \text{ and } \{e_1\} \not\preccurlyeq_{I_k} \pi_{I_k}(t_2)\}\,.$$

Intuitively, it is the set of elements in $\pi_{I_k}(t_1)$ preventing it from being more specific than $\pi_{I_k}(t_2)$ w.r.t. $\preccurlyeq_{I_k}$. We define the operator $\propto_{I_k}$ as follows:

$$\pi_{I_k}(t_1) \propto_{I_k} \pi_{I_k}(t_2) = \begin{cases} 1 \text{ if } \pi_{I_k}(t_1) \preccurlyeq_{I_k} \pi_{I_k}(t_2) \text{ or } \pi_{I_k}(t_2) \preccurlyeq_{I_k} \pi_{I_k}(t_1) \\ 1 - \frac{|\mathrm{SSD}(\pi_{I_k}(t_1), \pi_{I_k}(t_2)) \,\cup\, \mathrm{SSD}(\pi_{I_k}(t_2), \pi_{I_k}(t_1))|}{|\pi_{I_k}(t_1) \,\cup\, \pi_{I_k}(t_2)|} \text{ otherwise} \end{cases}$$

This operator returns a number measuring the similarity between $\pi_{I_k}(t_1)$ and $\pi_{I_k}(t_2)$. This number is equal to 1 if the two aggregated arguments are comparable. Otherwise, it is equal to 1 minus the proportion of incomparable elements. We denote by $\mathbb{I}_{\neq \Delta}(t_1, t_2) = \{I_k \mid I_k \in \mathbb{I} \text{ and } \pi_{I_k}(t_1) \neq \Delta \text{ and } \pi_{I_k}(t_2) \neq \Delta\}$ the set of aggregated arguments that are specified for both t_1 and t_2 (*i.e.*, different from Δ). Then, Rule 5 is defined as follows:

Rule 5. Let $\mathbb{I} = \{I_1, \ldots, I_m\}$ be a partition of $\{1, \ldots, n\}$, and let $\gamma_{\neq \Delta}$, γ_S, and γ_C be three parameters, all fixed at the beginning of the matching process.

$$\Big(|\mathbb{I}_{\neq \Delta}(t_1, t_2)| \ \geq \gamma_{\neq \Delta}\Big) \bigwedge \Big(\big[\forall I_k \in \mathbb{I}_{\neq \Delta}(t_1, t_2),\ \pi_{I_k}(t_1) \propto_{I_k} \pi_{I_k}(t_2) \geq \gamma_S\big] \vee \\ \big[\big(\sum_{I_k \in \mathbb{I}_{\neq \Delta}(t_1, t_2)} \mathbb{1}\left(\pi_{I_k}(t_1) \propto_{I_k} \pi_{I_k}(t_2) = 1\right)\big) \geq \gamma_C\big]\Big) \Rightarrow t_1 \propto t_2$$

Rule 5 is applicable if at least $\gamma_{\neq \Delta}$ aggregated arguments are specified for both t_1 and t_2. Then, t_1 and t_2 are weakly related ($\propto$) whenever all these specified aggregated arguments have a similarity of at least γ_S or when at least γ_C of them are comparable. Notice that $\propto$ is symmetric.

5 Application to Pharmacogenomic Knowledge

Our methodology was motivated by the problem of matching pharmacogenomic (PGx) tuples. Accordingly, we tested this methodology on PGxLOD[5] [10], a knowledge base represented in the $\mathcal{ALHI}$ Description Logic [3]. In PGxLOD, 50,435 PGx tuples were integrated from four different sources: (i) 3,650 tuples from structured data of PharmGKB, (ii) 10,240 tuples from textual portions of PharmGKB called clinical annotations, (iii) 36,535 tuples from biomedical literature, and (iv) 10 tuples from results found in EHR studies. We obtained the matching results summarized in Table 1 and discussed in Section 6. Details about formalization, code and parameters are given in Appendix G.

6 Discussion

In Table 1, we observe only a few inter-source links, which may be caused by missing mappings between the vocabularies used in sources. Indeed, our matching process requires these mappings to compare individuals represented with different vocabularies. This result underlines the relevance of enriching the knowledge base with ontology-to-ontology mappings. We also notice that Rule 5 generates more links than the other rules, which emphasizes the importance of weaker relatedness levels to align sources and overcome their heterogeneity. Some results were expected and therefore seem to validate our approach. For example, some tuples from the literature appear more general than those of PharmGKB (with

[5] https://pgxlod.loria.fr

Table 1. Number of links resulting from each rule. Links are generated between tuples of distinct sources or within the same source. PGKB stands for "PharmGKB", sd for "structured data", and ca for "clinical annotations". As Rules 1, 2, 4, and 5 satisfy symmetry, links from t_1 to t_2 as well as from t_2 to t_1 are counted. Similarly, as Rules 1 to 3 satisfy transitivity, transitivity-induced links are counted. Regarding `skos:broadMatch` links, rows represent origins and columns represent destinations.

		PGKB (sd)	PGKB (ca)	Literature	EHRs
Links from Rule 1 Encoded by `owl:sameAs`	PGKB (sd)	166	0	0	0
	PGKB (ca)	0	10,134	0	0
	Literature	0	0	122,646	0
	EHRs	0	0	0	0
Links from Rule 2 Encoded by `skos:closeMatch`	PGKB (sd)	0	5	0	0
	PGKB (ca)	5	1,366	0	0
	Literature	0	0	16,692	0
	EHRs	0	0	0	0
Links from Rule 3 Encoded by `skos:broadMatch`	PGKB (sd)	87	3	15	0
	PGKB (ca)	9,325	605	42	0
	Literature	0	0	75,138	0
	EHRs	0	0	0	0
Links from Rule 4 Encoded by `skos:relatedMatch`	PGKB (sd)	20	0	0	0
	PGKB (ca)	0	110	0	0
	Literature	0	0	18,050	0
	EHRs	0	0	0	0
Links from Rule 5 Encoded by `skos:related`	PGKB (sd)	100,596	287,670	414	2
	PGKB (ca)	287,670	706,270	1,103	19
	Literature	414	1,103	1,082,074	15
	EHRs	2	19	15	0

15 and 42 `skos:broadMatch` links). These links are a foreseen consequence of the completion process of PharmGKB. Indeed, curators achieve this completion after a literature review, inevitably leading to tuples more specific or equivalent to the ones in reviewed articles. Interestingly, our methodology could ease such a review by pointing out articles describing similar tuples. Clinical annotations of PharmGKB are in several cases more specific than structured data (9,325 `skos:broadMatch` links). This is also expected as structured data are a broad-level summary of more complex phenotypes detailed in clinical annotations.

Regarding our method, using rules is somehow off the current machine learning trend [1, 11, 13]. However, writing simple and well-founded rules constitutes a valid first step before applying machine learning approaches. Indeed, such explicit rules enable generating a "silver" standard for matching, which may be useful to either train or evaluate supervised approaches. Rules are readable and may thus be analyzed and confirmed by domain experts, and provide a basis of explanation for the matching results. Additionally, our rules are simple enough to be generally true and useful in other domains. By relying on instantiated classes and links between individuals, we illustrate how domain knowledge and reasoning mechanisms can serve a structure-based matching. In future works, conditions under which preorders $\preccurlyeq^{\mathsf{P}}$ and $\preccurlyeq^{\mathcal{O}}$ could be merged into one unique preorder deserve a deeper study. See Appendix H for further discussion.

7 Conclusion

In this paper, we proposed a rule-based approach to establish the relatedness level of n-ary tuples among five proposed levels. It relies on rules and preorders that leverage domain knowledge and reasoning capabilities. We applied our methodology to the real-world use case of matching pharmacogenomic relationships, and obtained insightful results. In the future, we intend to compare and integrate our purely symbolic approach with ML methodologies.

References

1. Alam, M., et al.: Reconciling event-based knowledge through RDF2VEC. In: Proceedings of HybridSemStats@ISWC 2017. CEUR Workshop Proceedings (2017)
2. Atencia, M., David, J., Euzenat, J.: Data interlinking through robust linkkey extraction. In: Proceedings of ECAI 2014. Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 15–20 (2014)
3. Baader, F., et al. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
4. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The Semantic Web. Scientific american **284**(5), 28–37 (2001)
5. Caudle, K.E., et al.: Incorporation of pharmacogenomics into routine clinical practice: the clinical pharmacogenetics implementation consortium (cpic) guideline development process. Current Drug Metabolism **15**(2), 209–217 (2014)
6. Coulet, A., Smaïl-Tabbone, M.: Mining electronic health records to validate knowledge in pharmacogenomics. ERCIM News **2016**(104) (2016)
7. Euzenat, J., Shvaiko, P.: Ontology Matching, Second Edition. Springer (2013)
8. Galárraga, L.A., Preda, N., Suchanek, F.M.: Mining rules to align knowledge bases. In: Proceedings of AKBC@CIKM 2013. pp. 43–48. ACM (2013)
9. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge acquisition **5**(2), 199–220 (1993)
10. Monnin, P., et al.: PGxO and PGxLOD: a reconciliation of pharmacogenomic knowledge of various provenances, enabling further comparison. BMC Bioinformatics **20-S**(4), 139 (Apr 2019)
11. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. Proceedings of the IEEE **104**(1), 11–33 (2016)
12. Noy, N., Rector, A., Hayes, P., Welty, C.: Defining N-ary Relations on the Semantic Web. W3C working group note **12**(4) (2006)
13. Ristoski, P., Paulheim, H.: Rdf2vec: RDF graph embeddings for data mining. In: The Semantic Web - ISWC 2016 - Proceedings, Part I. pp. 498–514 (2016)
14. Suchanek, F.M., Abiteboul, S., Senellart, P.: PARIS: probabilistic alignment of relations, instances, and schema. PVLDB **5**(3), 157–168 (2011)

A Details about Related Works

In ontology matching [7], existing works use different features to suggest alignments. For example, some methods rely on the syntax of units label (*e.g.*, string matching). However, labels may not always be available. Structure-based techniques can alleviate such limitations. They rely either on the internal structure of a unit (*i.e.*, predicates used to link a unit to literals) or the relational structure of a unit (*i.e.*, its links with other units). Two examples of frequently considered relational structures are the hierarchy of classes and `partOf` links in an ontology. As our matching approach compares tuples based on the individuals involved in their arguments and their associated domain knowledge, it relies on the relational structure of reified tuples.

Linkkeys are defined by Atencia *et al.* [2] as a structure-based method to align individuals. A linkkey consists of a pair of classes and a set of pairs of properties from two ontologies. Instances of these classes that share common values for all properties in the linkkey are regarded as identical. Alternatively, PARIS is a holistic method proposed by Suchanek *et al.* [14] to align individuals, classes, and predicates. In this framework, alignments for each type of unit fertilize the others: they are performed repeatedly until convergence. Rules in PARIS rely on the internal and relational structures of units and the functionality of predicates. Similarly, Galarraga *et al.* [8] mine specific rules to align ontologies, using the AMIE system. This system relies on the Partial Completeness Assumption, which is also built upon the functionality of predicates.

B Proof that $\preccurlyeq^{\mathtt{p}}$ is a preorder

Proof. From the fact that p is reflexive, it immediately follows that $\preccurlyeq^{\mathtt{p}}$ is reflexive. Indeed, for every $E \subseteq \Delta$, $E \preccurlyeq^{\mathtt{p}} E$ since for every $e \in E$, $\mathtt{p}(e, e)$.

To prove that $\preccurlyeq^{\mathtt{p}}$ is a preorder, it remains to show that $\preccurlyeq^{\mathtt{p}}$ is transitive. Consider $E_1, E_2, E_3 \subseteq \Delta$ such that:

$$E_1 \preccurlyeq^{\mathtt{p}} E_2 \text{ and } E_2 \preccurlyeq^{\mathtt{p}} E_3.$$

In other words, $\forall e_1 \in E_1,\ \exists e_2 \in E_2,\ \mathcal{K} \models \mathtt{p}(e_1, e_2)$ and $\forall e_2 \in E_2,\ \exists e_3 \in E_3,\ \mathcal{K} \models \mathtt{p}(e_2, e_3)$. By the transitivity of p, we then have that

$$\forall e_1 \in E_1,\ \exists e_3 \in E_3,\ \mathcal{K} \models \mathtt{p}(e_1, e_3),$$

i.e., $E_1 \preccurlyeq^{\mathtt{p}} E_3$. This shows that $\preccurlyeq^{\mathtt{p}}$ is transitive, and the proof is complete. □

C Example of Use of $\preccurlyeq^{\mathtt{p}}$

Example 1. `partOf` is transitive and reflexive. Thus, this predicate is a suitable candidate for the $\preccurlyeq^{\mathtt{p}}$ preorder. Consider three individuals e_1, e_2, e_3 such that $\mathcal{K} \models \mathtt{partOf}(e_3, e_1)$. Then it follows that:

– $\{e_1\} \preccurlyeq^{\texttt{partOf}} \{e_1, e_2\}$, from set inclusion.
– $\{e_3, e_2\} \preccurlyeq^{\texttt{partOf}} \{e_1, e_2\}$.
– $\{e_3\} \preccurlyeq^{\texttt{partOf}} \{e_1, e_2\}$.
– $\{e_3, e_1\} \sim^{\texttt{partOf}} \{e_1\}$. As e_3 is a part of e_1, having both e_3 and e_1 in the same set can be seen as a redundancy. Such a case may arise in $\mathcal{K}$ due to source heterogeneity. This redundancy is adequately identified by this equivalence result.

D Details about Excluding $\top$ in $\mathrm{ci}(\mathcal{O}, e)$

Example 2. Consider two PGx tuples pgt_1 and pgt_2 that involve the same drug and genetic factor. Regarding the phenotype, pgt_1 is linked with an individual representing *headache* that does not instantiate the class *Headache* in $\mathcal{O}$ (*e.g.*, MeSH) but instantiates $\top$ by default. pgt_2 is linked with an individual *pain* that instantiates $Pain$, with $Headache \sqsubseteq Pain$. Intuitively, the knowledge expressed by pgt_1 is more specific than pgt_2. However, by considering instantiated classes and knowing that $Pain \sqsubseteq \top$, $\preccurlyeq^{\mathcal{O}}$ would inadequately conclude that pgt_1 is more general than pgt_2. By excluding $\top$ from $\mathrm{ci}(\mathcal{O}, e)$, the tuples are incomparable, which avoids this unwanted behavior.

E Proof that $\preccurlyeq^{\mathcal{O}}$ is a preorder

Recall that, for every $E_1, E_2 \subseteq \mathit{\Delta}$,

$$E_1 \preccurlyeq^{\mathcal{O}} E_2 \Leftrightarrow \forall e_1 \in E_1,\ \Big[\underbrace{e_1 \in E_2}_{(5a)}\Big] \bigvee \Big[\mathrm{msci}(\mathcal{O}, e_1) \neq \emptyset \wedge \underbrace{\forall C_1 \in \mathrm{msci}(\mathcal{O}, e_1),\ \exists e_2 \in E_2,\ \exists C_2 \in \mathrm{msci}(\mathcal{O}, e_2),\ C_1 \sqsubseteq C_2}_{(5b)}\Big] \quad (5)$$

Proof. The reflexivity of $\preccurlyeq^{\mathcal{O}}$ follows immediately from (5a). To see that it is also transitive, consider distinct $E_1, E_2, E_3 \subseteq \mathit{\Delta}$ such that $E_1 \preccurlyeq^{\mathcal{O}} E_2$ and $E_2 \preccurlyeq^{\mathcal{O}} E_3$. We need to prove that $E_1 \preccurlyeq^{\mathcal{O}} E_3$, that is,

$$\forall e_1 \in E_1,\ \Big[e_1 \in E_3\Big] \bigvee \Big[\mathrm{msci}(\mathcal{O}, e_1) \neq \emptyset \wedge \forall C_1 \in \mathrm{msci}(\mathcal{O}, e_1),\ \exists e_3 \in E_3,\ \exists C_3 \in \mathrm{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3\Big]$$

So let $e_1 \in E_1$. If $e_1 \in E_2$, then it follows from $E_2 \preccurlyeq^{\mathcal{O}} E_3$ that

$$\Big[e_1 \in E_3\Big] \bigvee \Big[\mathrm{msci}(\mathcal{O}, e_1) \neq \emptyset \wedge \forall C_1 \in \mathrm{msci}(\mathcal{O}, e_1),\ \exists e_3 \in E_3,\ \exists C_3 \in \mathrm{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3\Big], \quad (6)$$

and we are done. Otherwise,

$$\text{msci}(\mathcal{O}, e_1) \neq \emptyset \ \wedge \ \forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_2 \in E_2,\ \exists C_2 \in \text{msci}(\mathcal{O}, e_2),\ C_1 \sqsubseteq C_2.$$

As $E_2 \ \preccurlyeq^{\mathcal{O}} E_3$, we have two possible cases for each $e_2 \in E_2$:

- $e_2 \in E_3$ and for each $C_1 \in \text{msci}(\mathcal{O}, e_1)$ we also have:
$$\exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3,\ \text{or}$$
- $\exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_2 \sqsubseteq C_3$. Since the subsumption relation is transitive, $C_1 \sqsubseteq C_3$, and
$$\exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3$$

From these two cases, it follows that for each $e_1 \in E_1$ such that

$$\text{msci}(\mathcal{O}, e_1) \neq \emptyset \ \wedge \ \forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_2 \in E_2,\ \exists C_2 \in \text{msci}(\mathcal{O}, e_2),\ C_1 \sqsubseteq C_2,$$

we have that
$$\exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3. \tag{7}$$

From Equations (6) and (7), it then follows that:

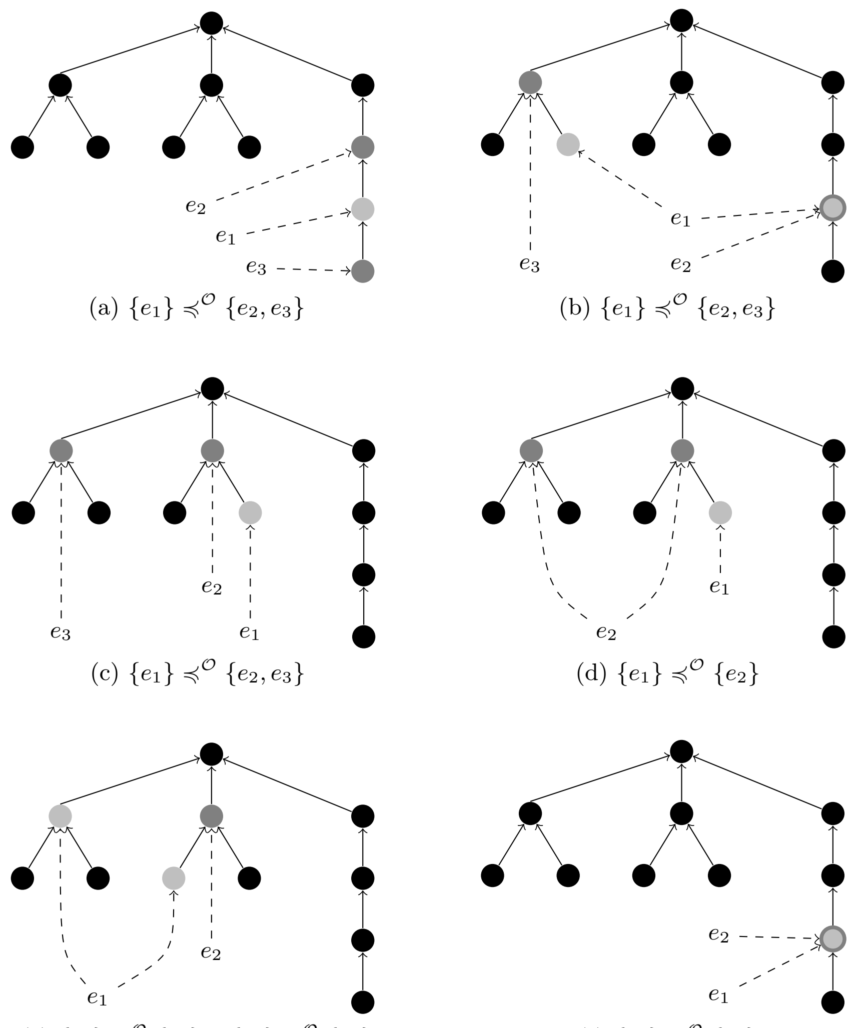
$$\forall e_1 \in E_1,\ \Big[e_1 \in E_3\Big] \bigvee \Big[\text{msci}(\mathcal{O}, e_1) \neq \emptyset \ \wedge \\ \forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3\Big],$$

thus showing that $E_1 \preccurlyeq^{\mathcal{O}} E_3$. As the latter holds for every $E_1, E_2, E_3 \subseteq \varDelta$, $\preccurlyeq^{\mathcal{O}}$ is transitive. □

F Examples of Use of $\preccurlyeq^{\mathcal{O}}$

Example 3. Figure 2 depicts six examples for the application of $\preccurlyeq^{\mathcal{O}}$:

(a) $\{e_1\}$ is more specific than $\{e_2, e_3\}$ even if e_3 instantiates a more specific class than e_1, because of the more general individual e_2.
(b) $\{e_1\}$ is more specific than $\{e_2, e_3\}$ since classes in $\text{msci}(\mathcal{O}, e_1)$ are either the same than those in $\text{msci}(\mathcal{O}, e_2)$ or more specific than those in $\text{msci}(\mathcal{O}, e_3)$.
(c) $\{e_1\}$ is more specific than $\{e_2, e_3\}$ since the class in $\text{msci}(\mathcal{O}, e_1)$ is more specific than the one in $\text{msci}(\mathcal{O}, e_2)$. There is no need to compare it with the class in $\text{msci}(\mathcal{O}, e_3)$.
(d) This example, similar to (c), illustrates the occurrence of the same behavior regardless of classes being instantiated by a single or by several individuals.
(e) $\{e_1\}$ and $\{e_2\}$ cannot be compared. Unlike the two latter examples, here, e_1 instantiates a class that is more specific than the class instantiated by e_2, but also a class that is not comparable.
(f) $\{e_1\}$ and $\{e_2\}$ are equivalent by instantiating the same most specific class.



(a) $\{e_1\} \preccurlyeq^{\mathcal{O}} \{e_2, e_3\}$ (b) $\{e_1\} \preccurlyeq^{\mathcal{O}} \{e_2, e_3\}$

(c) $\{e_1\} \preccurlyeq^{\mathcal{O}} \{e_2, e_3\}$ (d) $\{e_1\} \preccurlyeq^{\mathcal{O}} \{e_2\}$

(e) $\{e_1\} \not\preccurlyeq^{\mathcal{O}} \{e_2\}$; $\{e_2\} \not\preccurlyeq^{\mathcal{O}} \{e_1\}$ (f) $\{e_1\} \sim^{\mathcal{O}} \{e_2\}$

Fig. 2. Examples of use cases of the preorder $\preccurlyeq^{\mathcal{O}}$. Circles represent ontology classes. Solid arrows depict class subsumptions and dashed arrows depict class instantiations by individuals e_1, e_2, and e_3. The light gray color identifies classes in $\mathrm{msci}(\mathcal{O}, e_1)$. The dark gray color identifies classes in $\mathrm{msci}(\mathcal{O}, e_2)$ and $\mathrm{msci}(\mathcal{O}, e_3)$.

G Application to Pharmacogenomic Knowledge (details)

We experimented our methodology with PGxLOD[6] [10], a PGx knowledge base represented in the $\mathcal{ALHI}$ Description Logic [3].

In PGxLOD, PGx tuples are represented using classes and predicates of the PGxO ontology[7]. PGx tuples are n-ary, and thus, they are reified as instances of the `PharmacogenomicRelationship` class. All the individuals involved in PGx tuples instantiate the `Drug`, `GeneticFactor`, or `Phenotype` classes. They are linked with reified PGx tuples by predicates qualifying their association to tuples. These predicates are organized in a hierarchy defined by subsumption axioms, such as `causes` $\sqsubseteq$ `influences` $\sqsubseteq$ `isAssociatedWith`. It is noteworthy that, in PGxLOD, `partOf` links indicate that instances of `GeneticFactor` compose others such instances. For example, a genomic variation may be part of a gene. Similarly, instances of `Phenotype` may have dependencies, expressed with `dependsOn` links. These dependencies enable representing complex phenotypes that refer to other phenotypes or drugs. For example *warfarin-caused hemorrhage* is a phenotype linked with `dependsOn` to *hemorrhage* and *warfarin*. The TBox of PGxLOD contains, alongside PGxO, three other ontologies: individuals representing drugs may instantiate classes from ATC or ChEBI, and individuals representing phenotypes may instantiate classes from MeSH. Table 2 provides global statistics about PGxLOD.

Table 2. Statistics about PGxLOD. # denotes "number of". Instances linked by `owl:sameAs` are counted separately. `partOf` links are counted without transitivity inference. All PGx tuples were programmatically extracted from their sources, except the ten tuples from EHRs that were manually added as a proof of concept.

Class	# instances	Predicate	# links
`Drug`	47,584	`partOf`	16,697
`GeneticFactor`	464,302	`dependsOn`	23,976
`Phenotype`	61,330		
`PharmacogenomicRelationship`	50,435		
↳ From PharmGKB (structured data)	3,650		
↳ From PharmGKB (clinical annotations)	10,240		
↳ From biomedical literature	36,535		
↳ From EHRs	10		

To apply the matching rules on PGx tuples, we specified their arguments. Each argument of a tuple is the set of individuals with a specific type (`Drug`, `GeneticFactor`, or `Phenotype`) that are linked with a specific predicate to the tuple. For example, given pgt a PGx tuple, $\pi_{\texttt{Phenotype},\texttt{causes}}(pgt)$ contains all the

[6] https://pgxlod.loria.fr
[7] https://pgxo.loria.fr

phenotypes caused by pgt. Hence, as there are 3 types of individuals and 38 predicates, PGx tuples have $3 \times 38 = 114$ arguments. Once arguments of tuples are specified, their associated preorders can be defined. Based on the available data and knowledge in PGxLOD, it makes sense to use the $\preccurlyeq^{\mathtt{partOf}}$ preorder for arguments involving instances of `GeneticFactor`. Similarly, we use $\preccurlyeq^{\mathcal{O}_{\mathtt{Drug}}}$ and $\preccurlyeq^{\mathcal{O}_{\mathtt{Phenotype}}}$ as preorders for arguments respectively involving instances of `Drug` and `Phenotype`, where $\mathcal{O}_{\mathtt{Drug}}$ is the concatenation of ATC and ChEBI, and $\mathcal{O}_{\mathtt{Phenotype}}$ is the MeSH ontology.

Finally, to apply Rule 5, a natural three-way partition of arguments appears based on the three types of involved individuals. Therefore, discarding predicates, we gather all drugs, genetic factors, and phenotypes involved in a tuple in three aggregated arguments. To benefit from dependencies of complex phenotypes, we choose to add them to the aggregated arguments corresponding to their type. For example, in *warfarin-caused hemorrhage*, *hemorrhage* is added to the aggregated argument representing phenotypes and *warfarin* is added to the one representing drugs. We arbitrarily set $\gamma_{\neq \Delta} = 3$, $\gamma_S = 0.8$, and $\gamma_C = 2$. These values mean that two PGx tuples pgt_1 and pgt_2 will be matched by Rule 5 if their three aggregated arguments are specified (*i.e.*, different from Δ). Additionally, each of the three aggregated arguments of pgt_1 must have at least 80% of comparable individuals with the same aggregated argument of pgt_2, or at least two aggregated arguments of pgt_1 must be comparable with the same aggregated arguments of pgt_2.

To illustrate the interest of this formalization as well as reasoning mechanisms from Description Logics, let pgt_1 and pgt_2 be two PGx tuples. pgt_1 causes a phenotype ph_1 and is associated with a phenotype ph_2, and pgt_2 is associated with both phenotypes. Thus, by applying reasoning mechanisms along the hierarchy of predicates, it follows that:

$$\pi_{\mathtt{Phenotype},\mathtt{causes}}(pgt_1) = \{ph_1\}; \ \pi_{\mathtt{Phenotype},\mathtt{isAssociatedWith}}(pgt_1) = \{ph_1, \ ph_2\}$$

$$\pi_{\mathtt{Phenotype},\mathtt{causes}}(pgt_2) = \Delta; \ \pi_{\mathtt{Phenotype},\mathtt{isAssociatedWith}}(pgt_2) = \{ph_1, \ ph_2\}$$

By definition of $\preccurlyeq^{\mathcal{O}_{\mathtt{Phenotype}}}$, $\pi_{\mathtt{Phenotype},\mathtt{causes}}(pgt_1) \preccurlyeq^{\mathcal{O}_{\mathtt{Phenotype}}} \pi_{\mathtt{Phenotype},\mathtt{causes}}(pgt_2)$ as well as $\pi_{\mathtt{Phenotype},\mathtt{isAssociatedWith}}(pgt_1) \preccurlyeq^{\mathcal{O}_{\mathtt{Phenotype}}} \pi_{\mathtt{Phenotype},\mathtt{isAssociatedWith}}(pgt_2)$. Therefore, by applying Rule 3, pgt_1 is more specific than pgt_2. This makes sense as the predicate connecting ph_1 with pgt_1 is more specific with than the one used with pgt_2.

We implemented our matching methodology in C++ with multithreading. Our code is available on GitHub[8]. Our program interacts with the knowledge base thanks to SPARQL queries. Previously, we indicated that an unspecified argument of an n-ary tuple is set to Δ. Accordingly, when a SPARQL query returns $\emptyset$ for an argument of a tuple, it is interpreted as if it is returning Δ. On PGxLOD, the matching rules led to perform $\binom{50{,}435}{2} = 1{,}271{,}819{,}395$ comparisons in approximately 54 hours using 4 cores and 15 GB of RAM.

[8] https://github.com/pmonnin/tcn3r

H Discussion (details)

By looking more closely at the results in Table 1, we notice that all `owl:sameAs` links are intra-source and thus indicate duplicates. This is expected in the case of the literature since several articles could mention the same tuple. The 5 `skos:closeMatch` links between tuples from structured data and clinical annotations of PharmGKB highlight expected agreements between these two related sources. However, linked tuples are expressed with different individuals instantiating the same ontology classes, preventing their reconciliation with `owl:sameAs`.

The results of Rule 4 underline that sources may contain tuples with comparable arguments. Source owners can benefit from such results by considering adding a tuple formed by the most specific arguments of the matched tuples. We notice that only Rule 5 generates links between the tuples from EHRs and other sources. As tuples from EHRs are manually represented, there are only a few of them, minimizing the chance of overlap with other sources. Additionally, phenotypes involved in tuples from EHRs are very specific, making their comparison with phenotypes from biomedical literature or PharmGKB difficult.

Regarding our method, we believe our rules are simple and abstract enough to be useful in other domains. Additionally, their readability facilitates their review by experts, for example, to define the arguments and preorders to use in another application domain. Finally, it is noteworthy that the $\preccurlyeq^{\mathcal{O}}$ preorder may result in many equivalences if the ontology $\mathcal{O}$ is not granular enough in terms of width and depth. Such equivalences may make sense, depending on the application domain and the ontology. If not, the two other preorders (*i.e.*, $\subseteq$ and $\preccurlyeq^{\mathtt{P}}$) could be used. It is for now up to experts to choose the correct preorder for each argument. However, in future works, we could investigate metrics about domain knowledge that may guide their choice.