A framework supporting imprecise queries and data

Giacomo Bergami
Self-Conducted Project
bergamigiacomo@gmail.com

Abstract
This technical report provides some lightweight introduction and some generic use case scenarios motivating the definition of a database supporting uncertainties in both queries and data. This technical report is only providing the logical framework, which implementation is going to be provided in the final paper.

2012 ACM Subject Classification Information systems → Graph-based database models

Keywords and phrases approximate graph matching, inconsistency detection, knowledge base expansion, probabilistic reasoning

This work is licensed under a Creative Commons “Attribution-NonCommercial-ShareAlike 4.0 International” licence.

1 Introduction

In industry [11,30] as well as in academia [9,20,22], current knowledge base and linked data research focuses on automatic graph knowledge base construction and supporting efficient graph queries (SPARQL or Cypher) over graph databases (RDF or Property Graph stores). In addition to these common use cases for graph databases, relational databases might be also be represented as graphs [6], and therefore graph queries might be also supported by relational databases [29]. Such knowledge bases have been grown in popularity in several domain experts and decision makers such as governments and industrial stakeholders [1]. they are interested in querying graph data without necessarily knowing both the data schema and its representation [33]. As a result, the graph query generated from their full text information need might have a significantly different schema from the one represented in the graph database and, still, the query answering system needs to provide neither a few nor empty answers [33]. In addition to this, full text imputation comes with also with user typos, thus resulting into entities that are hard (if not impossible) to match in the exact graph querying scenario [1]. In addition to that, automatic query generation tasks [34] might refuse to generate queries from full text if no exact linking was available [34]. An example of such

```sql
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX res: <http://dbpedia.org/resource/>
SELECT DISTINCT ?uri WHERE {
    res:Martin_Luther_King,.Jr. text:"the_man_convicted_of_killing_King"
    ↦ ?x.
    ?x dbo:birthPlace ?uri.
}
```

Listing 1 QALD-6 SPARQL query generated from the full-text “In which town was the man convicted of killing Martin Luther King born?”
A framework supporting imprecise queries and data

Figure 1 Answer relevant from the query in Listing 1 as retrieved from DBPedia. Please note that this answer contains disconnected information and approximately matches the query.

scenario is provided by the QALD-6 hybrid question answering dataset, and an example of such (SPARQL) query is provided in Listing 1. As we can see, the query generation from full text might fail to link the text to the relationships represented within the knowledge base and, in the case the user inputs typos, it might also fail to link the entities to the ones in the knowledge base. As a result, no exact match interpretation of this generated query can return the expected result in Figure 1 at the time of the writing, no graph database system supports approximate graph matching for either imprecise data or imprecise queries. In both approximate query answering and in answering a query having a different schema of the integrated knowledge base, the problem reduces in considering the query as the final hub schema [13], over which aligning the data source to the query also corresponds to a graph matching phase.

Furthermore, Modern Knowledge Base Management Systems allow the ingestion of various data representations, such as relational, graph and full-text data [2,24]. Current KBMS ingest data sources that might be reliable as well as contain contradictory pieces of information. For the former specimen we might consider encyclopaedic data [24], business data [26], medical journals and clinical data [31], while for the latter we can include on-line social network tweets [17] or even medical diagnoses [7]. Such inconsistency might be detected by linking hierarchical datasets such as ontologies [23], taxonomies [32] and semantic networks [11,13,28] to the original data sources [18].

Despite there are several databases and logical data models, there is currently no system that is able to query a database where errors may be contained in both the query and the database instance itself. This paper is going to show how such challenge could be faced by generalizing well known approaches from the state of the art, as well as proposing novel contributions.

2 Logical Framework

This section provides which are the required constituent tools to implement a system matching all the aforementioned requirements. In this section, we give for granted that all the data sources have been integrated, so multiple instances of the same entity are grouped together within one same cluster: this step is fairly common in current literature [3,5], and therefore this preliminary step won’t be detailed by the present technical report. Then, we need

\[
\text{Assassination of Martin Luther King, Jr.}
\]

\[
\text{James Earl Ray}
\]

\[
\text{Alton, Illinois}
\]

\[
\text{Martin Luther King, Jr.}
\]

\[
\text{preps}
\]

\[
\text{deathCause}
\]

\[
''\ast \text{James Earl Ray according to a criminal case} \ast \text{Lloyd Jowers and "others, including unspecified governmental agencies" according to a later civil case}''
\]
first to expand the knowledge base using some ad-hoc logical rules, that are going to draw inconsistencies. This grounding step needs to happen before query time (§2.1). After observing that querying data can be represented as a unification/grounding step of the query, we show how we can extend this approach to approximate graph matching, thus allowing to return approximated results when the right or expected information is not available within the knowledge base (§2.2). As a side effect, this is going to provide a first hypothesis rank based on the query similarity. Last, we’re going to rank all the hypotheses in a probabilistic fashion using the possible world enumeration over the finite possible worlds retrievable from the knowledge base (§2.3).

2.1 Grounding and Knowledge Base Expansion

Preliminaries and Notation. In our scenario, the database associated to the event knowledge base is the union of all the possible data sources \( d_i \), represented as a set of tuples and describing different biased point of views potentially mutually inconsistent, that is \( \mathbb{DB}_w = \bigcup_{i \in D} d_i \). Our database is agnostic on the actual data provenance information. The database is also associated to a schema \( \mathcal{DB} = (\mathcal{DB}, \Sigma_{DB}) \), where \( \mathcal{DB} \) is a finite set of relation symbols (atoms) and \( \Sigma_{DB} \) is a finite set of constraints (or queries) over the atoms in \( \mathcal{DB} \). Any hard constraint value in \( \lambda \in \Sigma_{DB} \) has an associated weight \( w(\lambda) \rightarrow +\infty \), and \([0, +\infty)\) if it is a soft constraint/rule. Each tuple \( e \in \mathbb{DB}_w \) is associated to a confidence extraction weight \( w(e) \) and is associated to a type \( T \in \mathcal{DB} \). The database induces a finite domain of values \( D = (\bigcup_{e \in \mathbb{DB}} \{ e \}) \backslash \{ \text{NULL} \} \) , thus obtaining a set of all the entities depicted in the relational database.

A knowledge base expansion over a relational database \( \mathbb{DB} \) over a set of rules \( \Sigma_{DB} \) is defined as the iterated application of the rules \( \lambda \in \Sigma_{DB} \) over \( \mathbb{DB} \) until no further knowledge is provided. Let us assume that \( \mathbb{DB} \) also includes the hierarchical datasets describing the entities \( D \) in their transitive and reflexive closure; such process can be described as a fixed point over \( \mathbb{DB} \) via an unification algorithm. In particular, An unification function \( \theta \) is a function mapping a set of variables into a value in \( D \). An unification algorithm \( U(\mathbb{DB}, \lambda) \) for a rule \( \lambda \) in the form of \( \forall^h \exists^k \bigwedge_i B_i \Rightarrow H \) over the database \( \mathbb{DB} \) returns a set of unifications \( \Theta \) such that each unification \( \theta \in \Theta \) grounds an atom \( B_i \) into an event of the database, that is \( \theta B_i \in \mathbb{DB} \). A knowledge expansion step (or “grounding”) uses the unification algorithm to add new events that are generated from the rules’ heads as follows:

\[
ES_{\Sigma_{DB}}(\mathbb{DB}) := \mathbb{DB} \cup \{ \theta H \mid \forall^h \exists^k \bigwedge_i B_i \Rightarrow H \in \Sigma_{DB}, \theta \in U(\mathbb{DB}, \lambda) \}
\]

Therefore, the knowledge expansion algorithm is the least fix point of the \( KE_{\Sigma_{DB}} \), that is:

\[
KE_{\Sigma_{DB}}(\mathbb{DB}) := \bigcup_{n \geq 0} ES^n_{\Sigma_{DB}}(\mathbb{DB}) = \mathbb{GB}
\]

where \( \mathbb{GB} \) is the grounded knowledge base. This definition might be trivially extended to jointly perform contemporaneously the formulae grounding from \( \Sigma \) and the factor graph generation over the finite model \( \mathbb{DB} \), thus describing the knowledge expansion process.

Grounding for inconsistency detection. At this stage, let us suppose that the query \( \forall^h \exists^k \bigwedge_i B_i \Rightarrow H \) could be skolemized so that everything could be expressed only with the universal quantifier \([12]\), and then the possibly bounded universal quantifiers could be grounded with the actual \( \mathbb{GB} \) values as a set of queries \( Q = \{ q_j := B_j \Rightarrow H \}_{j \leq s, s \in \mathbb{N}} \). For the moment, we’ll consider \( q \) as just one of these \( q_j \in Q \) queries, and we’ll generalize the
A framework supporting imprecise queries and data

As we may observe, the actual knowledge base expansion part provides the actual reasoning part, where logical rules are actually applied and new knowledge is inferred. As a consequence of our observations in paraconsistent logic, the probabilistic reasoning will occur only after the actual logical inference as a way to rank all the possible words that could be inferred from one single query (§2.3). Consequently, the query answering system could directly work on the expanded knowledge base, and collect all the atoms matching with the query. As a result, all the possible worlds of interest that need to be ranked are only the possible subsets \( \langle e_1, \ldots, e_m \rangle \) of \( GB \) matched by the query \( q \) for returning the expected result \( \gamma \); more formally, there exists an unification \( \theta \) such that \( \theta q = \land_{1 \leq l \leq m} e_l \Rightarrow \gamma \). We will refer to such subsets of the grounded knowledge base as candidate hypothesis. \( \gamma \) is then valid if and only if its expansion at any stage \( n \geq 0 \) generates no contradiction, \( \bot \); therefore, it should be trivially true that \( \forall \gamma.\gamma \neq \bot \). As a consequence, we need to also consider the queries in \( Q \) as rules required for the expansion step, so that we can ensure that the query answering system does not potentially generate further inconsistencies. Given that we cannot forecast all the possible \( Q \), we will perform the knowledge base expansion for \( Q \) after the one required for \( \Sigma_{DB} \). In order to track down which expanded elements lead to a contradiction, we define a logical provenance function \( \kappa \). In order to do so, \( \kappa \) is initially defined over the non-grounded initial DB elements as:

\[
\kappa^0(u) = \begin{cases} \{\{u\}\} & u \in DB \\ \emptyset & \text{oth.} \end{cases}
\]

Still, we need to extend \( \kappa^0 \) for the stepwise expansion, thus requiring the following new knowledge expansion step function:

\[
ES'_{\Sigma_{DB}}(\langle DB, \kappa \rangle) = \left\{ ES_{\Sigma_{DB}}(\emptyset) \cup \bigcup_{1 \leq i \leq n} \kappa(\theta B_i) \land \bigwedge_{1 \leq i \leq n} B_i \Rightarrow H = \lambda \in \Sigma_{DB} \theta \in U(DB, \lambda) \right\}
\]

As a consequence, the resulting knowledge expansion algorithm could be defined via \( ES' \) as follows:

\[
KE'_{\Sigma_{DB}}(\langle DB, \kappa^0 \rangle) := \bigcup_{n \geq 0} ES^n_{\Sigma_{DB}}(\langle DB, \kappa^0 \rangle) = \langle GB, \kappa^f \rangle
\]

This fixed point produces a pair \( \langle GB, \kappa^f \rangle \), where \( \kappa^f \) is the desired final function for all the possible expansion steps. The function \( \kappa^f \) is also able to detect which elements are sufficient to draw a contradiction, thus obtaining the overall minimal inconsistent subsets as in \cite{16} by calling \( \kappa^f(\bot) \). In particular, we can check if any possible subset of \( GB \) is inconsistent using the following predicate using \( \kappa^f \):

\[
\text{Cons}(\vec{e}) := \neg \exists e' \in \kappa^f(\bot).e' \subseteq \vec{e}
\]

Therefore, the set of the valid events generating hypotheses for candidate \( \gamma \neq \bot \) from an answered query \( q \) is defined as all the possible subsets of the grounded knowledge base over \( KE'_{\Sigma_{DB} \cup \Sigma_{A}(Q, DB)}(\langle GB, \kappa^f \rangle) \) that do not contain minimal inconsistent subsets. Given that approximate graph matching requires aligning multiple relationships into possible one single relationship \cite{15}, hen we need to infer such expansion or reduction rules from an aligning task \( A \) as extended in \cite{2} generating the correspondences for matching the set of queries in \( Q \) (source schema/ontology) to the knowledge base schema \( DB \) (target schema/ontology).
With an abuse of notation introduced not to further complicate the report’s notation, we denote the output of this further expansion step as \( \langle \mathbb{GB}, \kappa f \rangle \). Therefore, the set of the consistent unifications for the query \( q \) is given as follows:

\[
U_{GB}(q) = \{ \land_{1 \leq i \leq m} e_i \Rightarrow \gamma \mid \theta \in U(\mathbb{GB}, q) \cdot \theta q = \land_{1 \leq i \leq m} e_i \Rightarrow \gamma \land \text{Cons}(e_1, \ldots, e_m) \}
\]

\( U_{GB}(q) \) does not return set of events generating contradictions by construction. The requirement and assumption that \( \gamma \neq \bot \) does not break the generality of the querying approach: in fact, the user could still query which are all the (minimal) inconsistent hypotheses by invoking \( \kappa f(\bot) \), thus allowing to get all the possible MIS in a tractable time, that is linearly to the number of the expansion steps required to ground the knowledge base which, by previous assumptions, is polynomial over the size of the data. At the moment, we won’t consider any paraconsistent reasoning to tolerate inconsistent or incomplete queries with respect to the query’s completeness; we’ll focus more on these aspects in the next section, where approximated queries will be introduced. On the other hand, this definition also guarantees that each generated candidate has associated some non-empty evidence supporting it, that is it is always associated to a non-empty set of hypotheses. We can show that in the following lemma:

We can then define a function \( A \) generating all the candidates \( \gamma_k \) and a function \( H_{GB}(q, \gamma_k) \) generating all the consistent candidates for \( \gamma_k \):

\[
A^{\delta}(\mathbb{GB}, q) := \{ \gamma \mid H \Rightarrow \gamma \in U_{GB}(q) \} \quad H^{\delta}_{GB}(q, \gamma) := \tilde{\Gamma}(\{ H \mid H \Rightarrow \gamma \in U_{GB}(q) \})
\]

where \( \tilde{\Gamma} \) is an hypothesis summarization function using a similarity function \( \sim \) to merge similar results. At the moment we shall consider \( \sim \) as the equivalence similarity function \( \equiv \) for which \( \forall S. \tilde{\Gamma}(S) = S \). When the algorithm \( \mathcal{U} \) used for generating the expansions from the consistent unification, then we shall write \( A \) (or \( H \)) instead of \( A^{\delta} \) (or \( H^{\delta} \)). A dot \( \cdot \) replaces \( \mathcal{U} \) when we want to express that any unification algorithm \( \mathcal{U} \) could be used instead (e.g., \( A \)). After providing those definitions, we can now guarantee that if \( H \) returns no hypotheses associated to the target \( \gamma \) for query \( q \), then it means that either no exact and consistent subset of matched elements from the grounded knowledge base is possible.

\[\blacktriangleright\text{Lemma 1.}\]

\[
H^{\delta}_{GB}(q, \gamma) = \emptyset \iff \forall \theta \in U(\mathbb{GB}, q) \cdot \theta q = \theta B \Rightarrow \gamma \land \exists e \in \mathbb{GB} \land \exists e \in \kappa f(\bot), e \subseteq \mathbb{GB}
\]

\[\text{Proof.}\]

We can rewrite the following assertion by equivalence as follows:

\[
H^{\delta}_{GB}(q, \gamma) = \emptyset \iff \neg \exists H.H \Rightarrow \gamma \in U_{GB}(q) \iff \forall H.H \Rightarrow \gamma \notin U_{GB}(q)
\]

The lemma is then proved by expansion of the definition of \( \mathcal{U} \); in fact, \( \mathcal{U} \) does not contain a rule \( H \Rightarrow \gamma \) if and only if there exists any possible unification \( \theta \) over \( \mathbb{GB} \) of \( q \) that produces that grounded rule, or if all the possible unifications \( \theta \) produce a head which is inconsistent, e.g., \( \neg \text{Cons}(H) \).

\[\blacktriangleright\]

This proof should be further enforced when we will replace the exact unification algorithm with an approximated one in §\ref{approx}.

Finally, the set of all the answer associated to all the possible worlds as in Table ? could be represented by the following justification function:

\[
\forall \gamma \in A^{\delta}(\mathbb{GB}, q). \quad J^{\delta}_{GB,q}(\gamma) := H^{\delta}_{GB}(q, \gamma)
\]

Even in this case, we shall write \( J \) instead of \( J^{\delta} \) when \( \mathcal{U} \) is clear from the context.
2.2 Approximate Graph matching

As previously discussed, this phase is the one substituting the traditional exact query matching in relational databases and providing the approximation for the query answering scenario. We want also to remark why the graph query answering and approximate matching problem is the most general possible solution for a query answering scenario, thus also including the full-text query answering.

**Problem statement and generalization.** Let us suppose that \( q \) is a full-text query: an graph database and domain expert might "compile" the information need expressed by \( q \) as a domain-specific gold query \( q^G \), perfectly matching its informative needs. On the other hand, the real-world domain expert is does not generally know how to query a graph \([10]\) (or, as in this case, a relational \([19]\)) database, and therefore uses an automated compiler \( C: D \rightarrow \wp(\mathcal{L}) \) to translate each full text query \( d \in D \) into multiple possible queries \([19]\). Each generated query \( q_i \in \mathcal{L} \) also comes with an interpretation error \( \varepsilon(q_i) \in [0,1] \). Let us suppose to have a graph similarity function \( \sim \) as the one initially discussed for graph matching in the related work section (i.e., where both structural and semantical features might be combined). We are now ideally able to measure \( \varepsilon(q_i) \) as \( 1 - (q^G \sim q_i) \), that is how similar is the expected graph query to the one that has been generated by the graph query compiler \( C \). On the other hand, \( q^G \) is not generally available, and therefore the only way to assess this error is to evaluate \( 1 - (g(q) \sim q_i) \) instead, where \( g(q) \) is the graph representation of the full text query \([2]\). Consequently, the \( \sim \) (or its opposite, \( \varepsilon \)) function should be already used by \( C \) while generating the queries from the full-text, so that the quality of the query generation output could be assessed.

The generality of this compiler framework for the full-text interpretation scenario could be assessed by looking at all the different possible implementations of \( C \) depending of the querying scenario of interest:

1. when \( q \) is a query that needs to be aligned towards different possible knowledge base schemas, then each different query \( q_i \) represents \( q \) for a specific knowledge base schema;
2. when \( q \) is a skolemized FOL query belonging to the AE fragment, then \( q_i \) represents each possible combination of variables grounded for the bounded universal quantifiers.
3. when \( q \) is a combinatio of all the three aforementioned query scenarios, the associated compiler is a specific combination of the aforementioned query compilers.

In this generalization step, we can then generalize \( \mathcal{A}, \mathcal{H} \) and \( \mathcal{J} \) by simply adopting a new generalization algorithm \( U_C (compiler generalization) \), which is defined as follows:

\[
U_C (\mathsf{GB}, q) = \bigcup_{q_i \in C(q)} U(\mathsf{GB}, q)
\]

**Result generation.** By extending the definition of graph query languages, approximate graph query answering systems such as \([8]\) also provide a set of subgraphs (or morphisms) as the result of the graph matching phase. Therefore, given a graph database/knowledge graph \( \mathbb{D} \), \( q_i(\mathbb{D}) \) returns a set of graphs \( \{ \tilde{g}_j \}_{j \leq n, n \in \mathbb{N}} \), where the same \( \sim \) function – also used by the compiler – is used to define the query result ranking function. The requirement that \( \sim \) is kept the same throughout the whole approximate querying process is crucial, as it guarantees that we’re always judging and ranking the data in the same way, thus allowing better optimization tasks.

Within the query generation and decomposition task for approximate graph matching outlined in \([8]\), in addition to structural clusters we might want to adopt semantic clusters as
well: that is always possible via the unique metric \( \sim \) that is used throughout the process. After the graph combination phase that could be carried out in a multi-way join fashion as outlined in [6], we might consequently obtain graph result clusters via the combination of the intermediate subgraph clusters, thus already subsets of different solutions. These representations of graph collections in such clusters can be then further summarized as one single graph using graph summarization algorithms [3,5]. In particular, we require a clustering and summarization function \( \hat{\Gamma} \) that, taken as an input a collection of graphs, it will indirectly use the similarity function \( \sim \) for the clustering and then summarizes the clusters using an UDF function. Given \( \nu_{\text{UDF}} \) the graph summarization function [4], we can then define \( \hat{\Gamma} \) as follows:

\[
\hat{\Gamma}(G) = \{ \nu_{\text{UDF}}(\{ g' \in G \mid g \sim g' > \theta_r \}) \mid g \in G \}
\]

where \( \theta_r \) is a similarity threshold value identifying if the hypotheses are similar or not (see §2.2). Given that the unification algorithms guarantee the consistency of the result by returning only consistent results, nothing is guaranteed that the cluster of similar graphs are indeed also (at least partially) consistent between each other. In order to provide such guarantee, we should impose that all the similarity functions \( \sim \) provide such guarantee:

\[
g \sim g' = 1 \Rightarrow \text{Cons}(g \cup g')
\]

Please note that it is not possible to evaluate the opposite direction of the logical implication, because we might also have two graphs \( g \) and \( g' \) that are consistent but not necessarily similar or identical.

Given that the \( \nu \) operation presented in [4] preserves the original graph \( g \in G \) information, we can always track each summarized graph to all the graphs providing such summarization and define the following graph provenance function:

\[
\forall s \in \hat{\mathcal{J}}_{\text{GB},q}(\gamma). \quad \hat{\text{map}}_{\text{GB},q}(s) := \max_{G' \subseteq \hat{\mathcal{J}}_{\text{GB},q}(\gamma)} \hat{\Gamma}(G') = \{ s \}
\]

As a last requirement, \( \sim \) shall also consider inconsistency detection via \( \Sigma_{DB} \), such that all the possible subgraphs of \( \bigcup \hat{\text{map}}_{\text{GB},q}(s) \) are not in \( \kappa^f(\bot) \), thus requiring that the summarization of similar graph does not lead to an inconsistent graph (i.e., hypothesis).

### 2.3 Hypothesis World Generation

As stated in the introduction, our decision support system does not want to provide the correct answer to a query, but only ranking all the provided results by data reliability and query matching. Therefore, we don’t want to evaluate the probability associated to each candidate answer \( \gamma_k \) in comparison with all the other remaining ones, but the probability of \( \gamma_k \) assuming that it might be a correct answer to our problem. More formally, we require instead that each \( \gamma_k \) has probability \( P(\gamma_k) = 1 \) if and only if all both the candidate hypotheses contain no data uncertainties and all the candidate hypotheses perfectly match with the query. For each \( \gamma_k \) we are only interested on the possible worlds that are generated by the query evaluation of \( q \) and the associated grounding process. All the other remaining worlds must be considered as irrelevant w.r.t. the candidate, and therefore with zero probability.

In order to assess such probability values, we can choose to model our grounded knowledge base as a Markov Logic Networks model using factor graphs. Each \( F_i \), either an atom \( A \) or a grounded formula \( \theta H \leftarrow \wedge \theta B_i \), has an associated exponential factor \( \phi_i \in \Phi \), which domain \( X_i \) is the set of the grounded atoms appearing in \( F_i \). A factor graph over a set of factors
A framework supporting imprecise queries and data

\( \Phi \) for a grounded knowledge base GB induces the following joint probability distribution of stochastically independent worlds:

\[
P(\bar{X} = \bar{x}) = \frac{1}{Z} \prod_{\phi_i \in \Phi} \phi(X_i)^{n_i(X_i)}
\]

where \( Z \) is the normalization over the allowed possible worlds \( W \) in the universe \( U \) such that \( \sum_{W \in U} P(\bar{X} = \bar{W}) = 1 \), and \( n_i(X_i) \) is the number of true groundings \( X_i \) of each \( F_i \) associated to \( \phi_i \). Each possible world \( W \) is a truth assignment over the atoms in \( \bar{X} \), while \( U \) is defined as a finite partition of a sample space represented by the union of all the possible worlds.

The following lemma shows that such graph model includes as irrelevant the worlds containing contradictions:

Lemma 2. Given a factor graph described by the set of factors \( \Phi \) describing both the database atoms and the grounded rules, all the worlds containing a contradiction have zero probability.

Proof. Without any loss of generality, let us suppose that our factor graph contains only two grounded atoms \( A \) and \( B \) and a grounded hard rule with infinite weight stating that such atoms are contradictory, and therefore \( F = A \land B \rightarrow \bot \); therefore, we have three factors \( \phi_i(\bar{x}_i) \), both for the atoms and for the grounded formula. Our model is \( P(A, B) = \phi_x(A)\phi_x(B)\phi_F(A, B) / Z \), where the normalization factor \( Z \) is defined as \( e^{\theta_A + \theta_B} + e^{\theta_B} (1 + \theta_A + \theta_B) \). For the only contradictory world where both \( A \) and \( B \) hold we have that \( P(1, 1) = (1 + e^{\theta_B} (1 + e^{\theta_A} + e^{\theta_B}))^{-1} \), such that for \( \theta_B \rightarrow +\infty \) has a zero probability.

In order to meet the requirement set up in this subsection’s first paragraph, we define an universe \( U_{\gamma_k} \) for each possible candidate \( \gamma_k \), where each worlds is to be identified within the graph \( \bigcup \mathcal{J}_{GB,q}(\gamma_k) \): given that the resulting graph could be a summarized version of \( \bigcup \mathcal{J}_{GB,q}(\gamma_k) \), each candidate hypothesis should be evaluated starting from \( \bigcup \mathcal{J}_{map} s(\gamma_k) \) for each \( s \in \mathcal{J}_{GB,q}(\gamma_k) \). Let us remember that, by construction, each candidate hypothesis never contains inconsistencies. Given that we are generally not interested in all the possible sub-worlds containing \( \bigcup \mathcal{J}_{map} s(\gamma_k) \) and that it is more interesting to see which is the biggest relevant and consistent world containing it, we shall define the maximum relevant world where \( s \) holds as:

\[
[s]_{\gamma_k} := \bigcup \mathcal{J}_{map} s(\gamma_k)
\]

Therefore, we have that the universe where \( \gamma_k \) holds with relevance to the query is \( U_{\gamma_k} := \{ [s]_{\gamma_k} | s \in \mathcal{J}_{GB,q}(\gamma_k) \} \). Finally, we can express the probability of the candidate \( \gamma \) subordinate to the data extraction errors by using the law of total probability as in [27], that is \( P(\gamma_k; w) := \sum_{W \in U_{\gamma_k}} P(W \in U_{\gamma_k} | w(W)) P(W) \): \( P(W) \) is the usual joint probability distribution induced by the factor graph considering the data extraction errors, and the likelihood \( P(W \in U_{\gamma_k} | w(W)) \) can be represented by a scoring function combining the query \( q_i \) similarities that generated the results as well as the similarity of each matched subgraph with respect to the query. For efficiency reasons, such score \( w \) shall be evaluated during the \( \bar{\Gamma} \) operation.

---

### References

1. H. Arnaout and S. Elbassuoni. Effective searching of RDF knowledge graphs. *J. Web Sem.*, 48:66–84, 2018.
2. G. Bergami. *A new Nested Graph Model for Data Integration*. PhD thesis, Alma Mater Studiorum – University of Bologna, 2018.

3. G. Bergami, F. Bertini, and D. Montesi. On approximate nesting of multiple social network graphs: a preliminary study. In *Proceedings of the 23rd International Database Applications & Engineering Symposium, IDEAS 2019, Athens, Greece, June 10–12, 2019.*, pages 40:1–40:5, 2019.

4. G. Bergami, A. Petermann, and D. Montesi. Thosp: an algorithm for nesting property graphs. In A. Arora, A. Bhattacharya, G. H. L. Fletcher, J. Larriba-Pey, S. Roy, and R. West, editors, *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Houston, TX, USA, June 10, 2018*, pages 8:1–8:10. ACM, 2018.

5. Š. Čebirić, F. Goasdoué, H. Kondylakis, D. Kotzinos, I. Manolescu, G. Troullinou, and M. Zneika. Summarizing semantic graphs: a survey. *The VLDB Journal*, Dec 2018.

6. Y. Chen, B. Guo, and X. Huang. $\delta$-transitive closures and triangle consistency checking: a new way to evaluate graph pattern queries in large graph databases. *The Journal of Supercomputing*, Feb 2019.

7. D. Costa and M. A. Martins. *Measuring Inconsistency in Information*, chapter Inconsistency Measures in Hybrid Logics, pages 169–194. College Publications, 2018.

8. R. De Virgilio, A. Maccioni, and R. Torlone. Approximate querying of rdf graphs via path alignment. *Distributed and Parallel Databases*, 33(4):555–581, Dec 2015.

9. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP ’11)*, Edinburgh, Scotland, UK, July 27–31 2011.

10. A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’14, New York, NY, USA - August 24 - 27, 2014*, pages 1156–1165, 2014.

11. Google. Freebase data dumps. [https://developers.google.com/freebase/data](https://developers.google.com/freebase/data).

12. J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

13. M. Hartung, A. Groß, and E. Rahm. Composition methods for link discovery. In V. Markl, G. Saake, K. Sattler, G. Hackenbroich, B. Mitschang, T. Härdter, and V. Köppen, editors, *Datenbanksysteme für Business, Technologie und Web (BTW), 15. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 11.-15.3.2013 in Magdeburg, Germany. Proceedings*, volume 214 of LNI, pages 261–277. GI, 2013.

14. J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, Jan. 2013.

15. S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Trans. Knowl. Data Eng.*, 30(5):824–837, 2018.

16. A. Hunter and S. Konieczny. Measuring inconsistency through minimal inconsistent sets. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, KR’08, pages 358–366. AAAI Press, 2008.

17. D. M. J. Lazer, M. A. Baum, Y. Benkler, A. J. Berinsky, K. M. Greenhill, F. Menczer, M. J. Metzger, B. Nyhan, G. Pennycook, D. Rothschild, M. Schudson, S. A. Sloman, C. R. Sunstein, E. A. Thorson, D. J. Watts, and J. L. Zittrain. The science of fake news. *Science*, 359(6380):1094–1096, 2018.

18. P. Lependu and D. Dou. Using ontology databases for scalable query answering, inconsistency detection, and data integration. *J. Intell. Inf. Syst.*, 37(2):217–244, Oct. 2011.

19. F. Li and H. V. Jagadish. Constructing an interactive natural language interface for relational databases. *PVLDB*, 8(1):73–84, 2014.
A framework supporting imprecise queries and data

F. Mahdisoltani, J. Biega, and F. M. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings.

A. Martelli and U. Montanari. An efficient unification algorithm. ACM Trans. Program. Lang. Syst., 4(2):258–282, Apr. 1982.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Groeves, and J. Welling. Never-ending learning. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15), 2015.

U. N. L. of Medicine. Mesh (https://www.nlm.nih.gov/mesh/).

I. J. of Research and Development. This is Watson, volume 56(3/4). IBM Co., May/July 2012.

J. Picado, A. Termehchy, A. Fern, and P. Ataei. Schema independent relational learning. In Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD ’17, pages 929–944, 2017.

D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. Özcan. Athena: An ontology-driven system for natural language querying over relational data stores. Proc. VLDB Endow., 9(12):1209–1220, Aug. 2016.

N. D. Singpurwalla and J. M. Booker. Membership functions and probability measures of fuzzy sets. Journal of the American Statistical Association, 99(467):867–877, 2004.

R. Speer, J. Chin, and C. Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., pages 4444–4451, 2017.

W. Sun, K. Srinivas, A. Fokoue, A. Kementsietsidis, G. Hu, and G. Xie. Sqlgraph: An efficient relational-based property graph store. In SIGMOD, 2015.

R. Trivedi, B. Sisman, X. L. Dong, C. Faloutsos, J. Ma, and H. Zha. Linkbed: Multi-graph representation learning with entity linkage. In I. Gurevych and Y. Miyao, editors, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, pages 252–262. Association for Computational Linguistics, 2018.

Y. Wang, L. Wang, M. Rastegar-Mojarad, S. Moon, F. Shen, N. Afzal, S. Liu, Y. Zeng, S. Mehrabi, S. Sohn, and H. Liu. Clinical information extraction applications: A literature review. Journal of Biomedical Informatics, 77:34 – 49, 2018.

W.H.O. Icd-11 (https://icd.who.int/).

L. Yan, R. Ma, D. Li, and J. Cheng. RDF approximate queries based on semantic similarity. Computing, 99(5):481–491, 2017.

W. Zheng, J. X. Yu, L. Zou, and H. Cheng. Question answering over knowledge graphs: Question understanding via template decomposition. PVLDB, 11(11):1373–1386, 2018.