Distinguishing Answers in Conceptual Graph Knowledge Bases

Nicolas Moreau, Michel Leclère, and Madalina Croitoru

LIRMM, Univ. Montpellier 2, CNRS
161, rue Ada
34392 Montpellier, France
{moreau,leclere,croitoru}@lirmm.fr

Abstract. In knowledge bases (KB), the open world assumption and the ability to express variables may lead to an answer redundancy problem. This problem occurs when the returned answers are comparable. In this paper, we define a framework to distinguish amongst answers. Our method is based on adding contextual knowledge extracted from the KB. The construction of such descriptions allows clarification of the notion of redundancy between answers, based not only on the images of the requested pattern but also on the whole KB. We propose a definition for the set of answers to be computed from a query, which ensures both properties of non-redundancy and completeness. While all answers of this set can be distinguished from others with a description, an open question remains concerning what is a good description to return to an end-user. We introduce the notion of smart answer and give an algorithm that computes a set of smart answers based on a vertex neighborhood distance.

1 Motivation

In the semantic web age, a large number of applications strongly rely on the building and processing of knowledge bases (KB) for different domains (multimedia, information management, semantic portals, e-learning, etc.). The formal languages used for representation will encounter obvious scaling problems and therefore rely on implicit or explicit graph based representations (see for example Topic Maps, RDF, Conceptual Graphs, etc.) As a direct consequence, querying such systems will have to be done through graph based mechanisms and, accordingly, optimization techniques implemented [1].

In ICCS'08 [2], we identified the *semantic database context* in which a set of answers has to be computed. For this case, there are two kinds of answer graphs: answers as subgraphs of the knowledge base graph that are used for browsing the KB or for applying rules; and answers as graphs, independent of the KB, corresponding to the classical use of a querying system[1]. With this latter kind

[1] For example, in SPARQL, as blank node labels can be renamed in results, see [1] 2.4.

S. Rudolph, F. Dau, and S.O. Kuznetsov (Eds.): ICCS 2009, LNAI 5662, pp. 233–246, 2009.

of answer, an important problem concerns detecting redundant answers. Unlike classical databases, redundancies are not limited to duplicate tuples. Indeed, the presence of unspecified entities (generic markers in CG or blank nodes in RDF) and also a type hierarchy leads us to consider that an answer which is more general than another as redundant. In [2], we studied this problem and proposed to return irredundant forms of the most specific answers to a query.

An important problem arising in this context is ultimately related to the nature of a KB vs. a database. With a classical database, one can assume that a query designer knows the database schema and is thus able to build a query corresponding to its needs. With a KB, the schema is extended to an ontology and the different assertions are not supposed to instantiate a specific frame (the knowledge is semi-structured). Consequently, it is difficult for a query designer to specify the content of the searched knowledge: he/she wants to obtain some information about a specific pattern of knowledge. The suppression of redundant answers only by comparing answer graphs independently of the KB results in the problem not being considered. A better way to address the redundancy problem consists of completing the answer graphs with their neighborhood to obtain more detailed answers in order to return some relevant knowledge to the end-user in order to get insight (restitution, reflet) into the diversity of the knowledge in the KB. Moreover, users seem to prefer answers in their context (e.g. paragraph) rather than the exact answer[3].

In this paper we focus on answers given in a graph based form. Our motivation stems from the homogeneity of preserving the same format between the KB, query and answer. Moreover, this will allow the reuse of answers as a KB for different future answers (see for example nested queries). Note that while this paper focuses solely on the problem of distinguishing graph based answers, the same research problem will arise and results will be obtained when of answers are represented as a tuple.

2 Contribution and Related Work

Figure 1 shows a query, a conceptual graph formalized KB (see section 3.1) and all five answers to the query in the KB (from A_1 to A_5, in gray in the picture).

The problem that arises in this scenario is how to define relevant answers when they are independent of the KB (*i.e.* a set of answer graphs and not a set of answer subgraphs of the KB). For instance, answers A_1 and A_5 are equivalent (*"there is a human who owns an animal"*), and knowledge expressed by these answers is expressed by all of the others. The open world assumption makes it impossible to state that all humans or animals represent distinct elements of the described world. Therefore it seems preferable to only return answers that bring more knowledge, i.e. in our example that *"Mary owns a cat"* and *"a human owns a dog"*. But another relevant answer could be that *"there is a human knowing Mary who owns a cat"*.

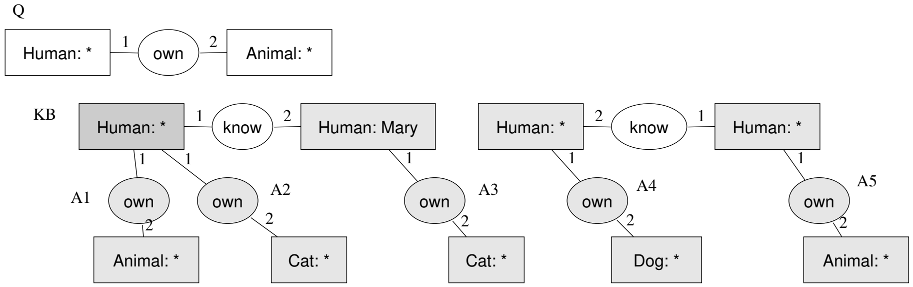


Fig. 1. Query, KB and highlighted answers

The contribution of the paper is to refine our preceding notion of redundancy between answers, to take the knowledge of the KB into account, through the notion of a *fair extension* of an answer, a graph that specifically describes an answer. Thus, an answer is irredundant if there is such a "fair" extension. A special case of this problem (when the answer is a concept node) corresponds to the problem of Generation of Referring Expressions (GRE) studied in the conceptual graph context in [4], that we extend for our purposes. Based on the extended notion of redundancy, we define two answer subset properties of *non-redundancy* (there is no redundant answer in the subset) and *completeness* (each answer is redundant to an answer of the subset). We then explain how to compute all the non-redundant and complete subsets of answers, based on a *redundancy graph*. We then discuss what is a good set of referring graphs to be returned to the user, and give an algorithm that fulfills these good properties.

As previously mentioned, a special case of answer identification was studied in [4]. The RDF query language SPARQL offers a way to describe answers (by the DESCRIBE primitive) but it does not address the specific problem of distinguishing one answer from another according to semantically sound syntactic criteria. The problem of answer redundancy in the Semantic Web context has been studied in [5], but the redundancy stated in this paper concerns the union of all answers, and corresponds, in the CG field, to the classical notion of irredundancy (see section 3.1), as opposed to our redundancy between answers. In an article about OWL-QL [6] the notion of server terseness is defined, which is the ability of a server to always produce a response collection that contains no redundant answers (i.e. there is no answer that subsumes another one). This corresponds to our previous notion of redundancy defined in [2].

In the next section, preliminary notions about conceptual graphs and our query framework are given. Section 4 deals with the extended notion of redundancy between answers and its application to a subset of answers. In section 5 we discuss the relevancy of the set of referring graphs returned to the user. We conclude our work in the last section.

3 Preliminary Notions

3.1 Simple Graphs

The conceptual graph formalism we use in this paper has been developed at LIRMM over the last 15 years [7]. The main difference with respect to the initial general model of Sowa [8] is that only representation primitives allowing graph-based reasoning are accepted.

Simple graphs (SGs) are built upon a *support*, which is a structure $S = (T_C,\ T_R,\ I,\ \sigma)$, where T_C is the set of concept types, T_R is the set of relations with any arity (arity is the number of arguments of the relation). T_C and T_R are partially ordered sets. The partial order represents a specialization relation ($t' \leq t$ is read as "t' is a specialization of t"). I is a set of individual markers. The mapping σ assigns a signature to each relation specifying its arity and the maximal type for each of its arguments.

SGs are labeled bipartite graphs denoted $G = (C_G, R_G, E_G, l_G)$ where C_G and R_G are the concept and relation node sets respectively, E_G is the set of edges and l_G is the mapping that labels nodes and edges. Concept nodes are labeled by a couple $(t : m)$, where t is a concept type and m is a marker. If the node represents an unspecified entity, its marker is the generic marker, denoted $*$, and the node is called a *generic* node, otherwise its marker is an element of I, and the node is called an *individual* node. Relation nodes are labeled by a relation r and, if n is the arity of r, it is incidental to n totally ordered edges.

A specialization/generalization relation corresponding to a deduction notion is defined over SGs and can be easily characterized by a graph homomorphism called *projection*. When there is a projection π from G to H, H is considered to be more specialized than G, denoted $H \leq G$. More specifically, a projection π from G to H is a mapping from C_G to C_H and from R_G to R_H, which preserves edges (if there is an edge numbered i between r and c in G then there is an edge numbered i between $\pi(r)$ and $\pi(c)$ in H) and may specialize labels (by observing type orders and allowing substitution of a generic marker by an individual one).

In the following, we use the notion of *bicolored SG* that was first introduced in [9]. A bicolored SG is an SG $H = \langle H_0, H_1 \rangle$ in which a color on $\{0, 1\}$ is assigned to each node of H, in such a way that the subgraph generated by 0-colored nodes, denoted H_0 and called the core of H, is a subSG. H_1, which is the sub-SG defined by 1-colored vertices and 0-colored concepts that are in relation with at least one 1-colored concept, is called the description.

3.2 Query Framework

The chosen context is a base composed of assertions of entity existences and relations over these entities, called *facts*, and stored in a single graph (not necessarily connected) named the *knowledge base*. This graph is assumed to be *normalized* if it does not contain two individual nodes with the same marker i. A normal form is easily computed by merging duplicate individual nodes of the graph. On the other hand, we do not require the KB graph to be in *irredundant form*: a graph G is in irredundant form iff there is no a projection from G in one of these strict subgraphs; otherwise this graph is said to be in *redundant form*. Indeed, computation of the irredundant form of a graph is expensive as the base can be large [10] and, moreover, there is not any local criterion for computation of the irredundant form and thus no incremental method (started at each updating of the base) can be expected. Such a KB graph B is simply queried by specifying a SG Q called the *query*. There is no constraint on the query (normalization or irredundancy). The answers to the query are found by computing the set $\Pi(Q, B)$ of projections from Q to B. The primary notion of answer consists of returning the set of subgraphs of B image of Q by a projection in $\Pi(Q, B)$.

Definition 1 (Answer set). *The set of answers of a query Q in a base B, denoted $Q(B)$, is $\{\pi(Q) \mid \pi \in \Pi(Q, B)\}$*[2]*.*

The first research question we address in this paper is whether this set of answers contains redundant answers? In fact, three kinds of redundancies can arise:

1. **Duplication:** two answers are identical. There is a duplication when two projections define the same subgraph. This problem can be solved easily by only keeping one of the duplicate subgraphs (this is done in the answer set $Q(B)$). An example of duplication is given in fig. 3(b) taken as the KB and fig. 3(c) taken as the query: there are two projections from the query whose images are the whole KB.
2. **Inclusion:** An answer is contained in another one. There is an inclusion when an answer is in a redundant form. Then its subgraph, in irredundant form, is also an answer. In the previous example (fig. 3(b) and 3(c)), there are two projections that define included answers.
3. **Redundancy:** An answer is more general or equivalent than another one. There is a redundancy when two answers are comparable (thus the knowledge expressed by one is also expressed by another); as an example answers A_1 and A_2 of figure 1.

Inclusion will be studied in section 4.4. The true redundancy problem becomes crucial since the answers are no longer KB subgraphs. Indeed, two answers can appear redundant when they are not really redundant. In [2], we define the notion of redundancy based only on the comparability of the answer graphs. This approach was motivated by the following argument: as the returned set of answers is independent of the KB, the subset of the *more specific irredundant answers*, denoted R_{min}, is sufficient to bring the entire range of answers. Moreover, R_{min} is minimal in terms of vertex number. In the following section, we characterize the true redundancy.

4 Dealing with True Redundancy

In [2], the *completeness* criterion (the knowledge expressed by each initial answer is expressed by one of the answers contained in the returned subset of answers)

[2] "Answer set", denoted $Q(B)$, and "answer" notions correspond respectively in our previous work [2] to notions of "answers by image subgraphs", denoted $R_{IP}(Q, B)$, and "images of proof". Names have been changed for simplification.

ensures that no knowledge is lost when a redundant answer is deleted. But this redundancy is based only on the comparability of answer subgraphs (that are semantically close as they are specializations of the query).

4.1 The True Redundancy

The true redundancy has to take the knowledge brought by the neighbor vertices of the answer into account. With this aim, we introduced the notion of extended answer, which is an answer supplemented with some knowledge extracted from its neighborhood.

Definition 2 (Extension of an answer). *Let B be a KB graph and Q a query graph. An extension of an answer A from $Q(B)$ is a bicolored graph $E = \langle E_0, E_1 \rangle$, where E_0 is isomorphic to A and such that there is a projection π from E to B with $\pi(E_0) = A$.*

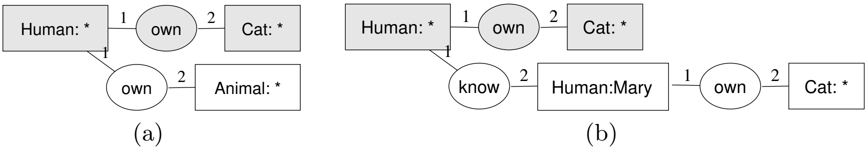


Fig. 2. Several extensions of answer A_2 of fig. 1

Fig. 2 represents several extensions of answer A_2 of fig. 1. Note that the extensions are not necessarily isomorphic copies of subgraphs of the KB.

With the previous definition of redundancy (section 3.2), if two answers have two incomparable extensions, they are two distinct answers and thus have to be considered as non-redundant answers, the one w.r.t. the other one. However, these two answers must not be distinguished in an "artificial" way: i.e. if the answers are distinguished by selectively adding knowledge to the extension of only certain answers but not to all of those which possess this knowledge in their neighborhood.

Definition 3 (Fairness property). *An extension E of an answer A from a set $Q(B)$ is fair iff there is no extension E' of another answer A' from $Q(B)$ such that there is a projection π from E to E' with $\pi(E_0) = E'_0$ and $\pi(E_1) = E'_1$.*

The extension of figure 2(b) is a fair extension of the answer A_2 of figure 1, contrary to the extension of figure 2(a). One can now give a definition of the true redundancy:

Definition 4 (True redundancy). *An answer A from a set of answers $Q(B)$ is truly redundant if there is no fair extension of this answer.*

In the example of fig. 1, answer A_1 is redundant. The search for a fair extension may seem to be a difficult task, considering that the number of extensions that can be generated for an answer is infinite. However, these extensions are semantically bounded by a more specific one that corresponds to an isomorphic copy of the base, $\langle A, B \setminus A\rangle$[3]. Naturally, the more one adds knowledge to the extension (the more it is specific), the more one potentially distinguishes this answer. So A is irredundant only if $\langle A, B \setminus A\rangle$ is a fair extension.

Theorem 1. *There is a fair extension E of an answer A iff $\langle A, B \setminus A\rangle$ is a fair extension.*

Proof. By contraposition. Suppose that there is a fair extension $E = \langle E_0, E_1\rangle$ of A and that $\langle A, B \setminus A\rangle$ is not a fair extension of A. $\langle A, B \setminus A\rangle$ is an extension of A and is isomorphic (without considering the colors) to B. As E is an extension of A, there is a projection π from E to B such that $\pi(E_0) = A$. As $\langle A, B \setminus A\rangle$ is not a fair extension of A, there is a projection π' from $\langle A, B \setminus A\rangle$ to B such that $\pi(R_0) \neq A$. So there is a projection $\pi'' = \pi \circ \pi'$ from E to B such that $\pi''(E_0) \neq A$. Thus E is not a fair extension of A. □

Corollary 1. *An answer A of $Q(B)$ is truly irredundant iff $\langle A, B \setminus A\rangle$ is a fair extension.*

4.2 The GRE Problem

The problem of finding a fair extension of an answer is strongly related to the problem of generation of referring expressions (GRE) known in the natural language processing field, which aims to describe an object in a scene such that the description only refers to this object. The GRE problem was formalized in the CG framework in [4], where the scene is an SG and the object to identify is a concept of the SG. A *referring graph* of a concept v is a subgraph of the KB containing this concept, and a *distinguishing graph* is a referring graph whose v is a fixed point for all projections of the graph in the KB. Our problem of answer redundancy can be seen as an extension of this formalization.

Definition 5 (Referring graph). *A referring graph of a subgraph G' of a graph G is a subgraph R of G that contains G' as a subgraph. To distinguish G' from the rest of the referring graph, we denote it as a bicolored graph $R = \langle R_0, R_1\rangle$ where $R_0 = G'$ and $R_1 = R \setminus G'$.*

Definition 6 (Distinguishing graph). *A referring graph R of a subgraph G' of a graph G is distinguishing if, for each projection π from R to G, $\pi(R_0) = G'$.*

The referring graph of figure 2(b) is a distinguishing graph of the answer A_2 of figure 1, contrary of the referring graph of figure 2(a). We can now link the true redundancy notion with the existence of a distinguishing graph for a given answer. The next properties strengthen the notion of true redundancy by showing its independence in the query and thus in the other answers.

Property 1. Let A be an answer of $Q(B)$. There is a fair extension of A iff there is a distinguishing graph of A w.r.t. B.

[3] For clarity, we sometimes denote subgraphs of KB as cores or descriptions of bicolored graph instead of their isomorphic copies.

Proof. Let $E = \langle E_0, E_1 \rangle$ be a fair extension of A. Then $\pi(E)$ is a distinguishing graph of A w.r.t. B. On the other hand, a distinguishing graph of A is also a fair extension of A. □

Corollary 2. *An answer A of $Q(B)$ is irredundant iff $R = \langle A, B \setminus A \rangle$ is a distinguishing graph of A w.r.t. to B.*

Thus, determining whether an answer is irredundant can be done by testing the distinguishness of the referring graph built from KB. However, the cost of such a test is exponential in the size of the KB.

4.3 Redundancy and Subset of Answers

Since the redundancy of an answer has been defined, it could be considered that for building a set of answers without redundancies one could simply remove redundant answers. However, the redundancy defined in the preceding section hides the fact that they are two types of redundancy. This is shown in fig. 3 (a query and three different KBs) :

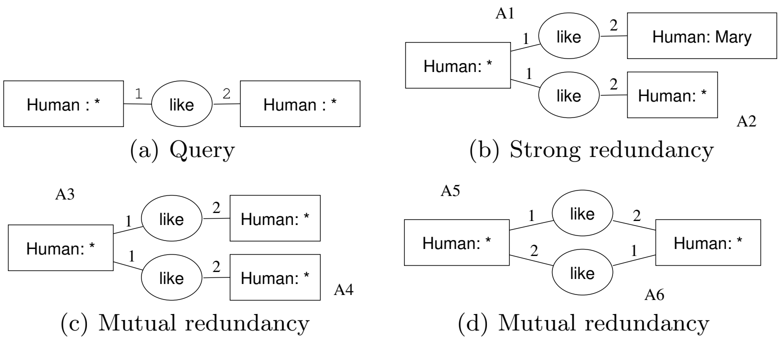
The first case will arise when an answer is completely redundant with respect to another answer (fig. 3(b)). This means that A_2 is redundant w.r.t. A_1 but the reverse does not hold. In this case, we say that A_2 is strongly redundant w.r.t. A_1. Thus, we only return the answer A_1.

The second case arises when there are a set of answers which are redundant amongst themselves (fig. 3(c) and 3(d)). In these two examples, an answer is redundant with respect to the others and vice-versa. In this case, we say that the answers are mutually redundant and we have to choose an answer in this set.

Note that the answer redundancy problem still holds in KBs in irredundant forms, as in the example of fig. 3(d).

Strong and mutual redundancies are based on the redundancy of an answer *w.r.t. another one*:

Definition 7 (Redundancy relation). *An answer A is redundant to an answer A' iff for each referring graph R^A of A, there is a projection π from R^A to a referring graph $R^{A'}$ of A' with $\pi(R_0^A) = \pi(R_0^{A'}) = A'$.*



(a) Query

(b) Strong redundancy

(c) Mutual redundancy

(d) Mutual redundancy

Fig. 3. Different cases of redundant answers

One can test the redundancy relation with the KB referring graph:

Property 2. An answer A is redundant to an answer A' iff there is a projection π from $\langle A, B \setminus A\rangle$ to a referring graph $R^{A'}$ of A' with $\pi(A) = A'$.

Therefore an answer A is redundant to an answer A', the redundancy relation between A' and A defines the two previously seen redundancy cases:

Definition 8 (Redundancies between answers). *Given A and $A' \in Q(B)$, such that A is redundant to A':*

- *A is strongly redundant to A' if A' is not redundant to A ;*
- *A and A' are mutually redundant if A' is redundant to A ;*

Based on the notion of true redundancy, we can refine our notions of *non-redundancy* and *completeness* of subsets of answers that we defined in [2] :

Definition 9 (Non-redundant subset of answers). *A subset of answers $\mathcal{A}$ of $Q(B)$ is non-redundant if there are no two answers A and $A' \in \mathcal{A}$ such that $A \neq A'$ and A is redundant to A'.*

Definition 10 (Completeness). *A subset of answers $\mathcal{A}$ of $Q(B)$ is complete if for each answer A of $Q(B)$ there is an answer A' of $\mathcal{A}$ such that A is redundant to A'.*

We define a *graph of redundancies* based on the notions of strong and mutual redundancies. All types of redundancies of definition 8 can be viewed as relations on $Q(B)^2$, for example (A, A') belongs to the strong redundancy relation if A is strongly redundant to A'. Thus we can characterize properties of these relations. Particularly, mutual redundancy defines equivalent classes over the set of answers, and strong redundancy links all answers of an equivalent class to all answers of another equivalent class. We construct the redundancy graph such that each vertex is an equivalent class, and is linked by the strong redundancy:

Definition 11 (Graph of redundancy). *The graph of redundancy $G = (V, E)$ of $Q(B)$ is a directed graph, where vertices represent equivalent classes of the mutual redundancy relation and where there is an edge between v_1 and v_2 if all answers of the class represented by v_1 are strongly redundant to all answers of the class represented by v_2.*

Fig. 4(a) represents the redundancy graph of query and KB of fig. 1, whereas fig. 4(b) represents the redundancy graph of query of fig. 3(a) and KB defined by the union of KBs of fig. 3(b), 3(c) and 3(d).

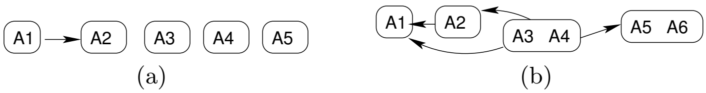


Fig. 4. Redundancy graph of the previous examples

To construct a complete and non-redundant set of answers, one has to only choose a single answer per equivalent class (to avoid mutual redundancy), only from equivalent classes that are sinks (to avoid strong redundancy), and for all of them (to be complete). This is why there is more than one non-redundant complete subset in the second example (fig. 4(b)): there is an equivalent class that is a sink and contains more than one element ($\{A_5, A_6\}$).

Theorem 2. *A subset of answers $\mathcal{A}$ of $Q(B)$ is complete and non-redundant iff for each sink in the redundancy graph there is a single answer of $\mathcal{A}$ that is represented by this sink.*

Proof. Given a non-redundant and complete subset of answers $\mathcal{A}$. Non-redundancy means that there is a single answer for each equivalent class represented in $\mathcal{A}$ and that there are no two answers A_i and A_j such that there is a path from A_i to A_j. Completeness ensures that for all answers A_i of $Q(B)$ (particularly answers belonging to a sink) there is an answer A_j such that A_i is redundant to A_j. So $\mathcal{A}$ has to contain an answer of each sink. When combining completeness and non-redundancy, only one answer of each sink is taken.
• Given a subset of answers $\mathcal{A}$ that is composed of an element of each sink of the redundancy graph. Given two answers of $\mathcal{A}$. These answers are not mutually redundant because there are no two answers of the same equivalent class. These answers are not strongly redundant because each answer comes from a sink. Thus $\mathcal{A}$ is non-redundant. For each answer A_i of $Q(B)$, either there is an answer A_j of the same equivalent class in $\mathcal{A}$ (thus A_j is redundant to A_i), or there is an answer A_k in $\mathcal{A}$ that comes from a sink such that there is a path from the equivalent class of A_i to the sink, and thus A_i is redundant to A_k. $\mathcal{A}$ is complete.

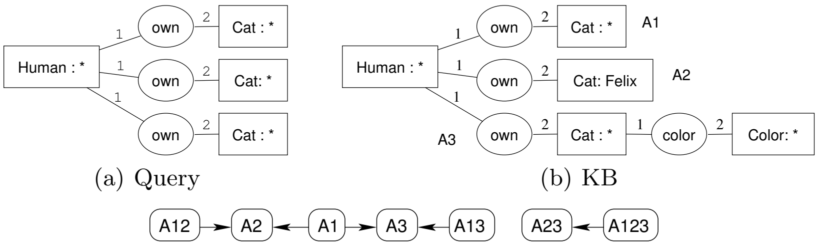
The construction of a redundancy graph combines the problem of finding all projections of a graph into another graph (to compute the set of answers) and the problem of computation of the irredundant form of a graph. Indeed, as stated by property 2, computation of all redundancy links of all answers used to construct the graph is based on all projections of the most specific referring graph of each answer (i.e. a bicolored isomorphic copy of the whole KB). Therefore, a way to compute all redundancy links is to compute all projections from the KB into itself, and check images of a each answer by all projections.

4.4 Inclusion of Answers

As mentioned in section 3.2, the inclusion of answers is one of the problems that can occur. If treated as a kind of duplication, included answers are just deleted from the answer set before computation of a non-redundant complete subset. But this approach is not the best one. We think that the inclusion problem should be treated after the redundancy problem.

In the example of fig. 5, there are seven answers: three that contain only one relation (A_1, A_2, A_3), three that contain two relations (A_{12}[4], A_{13}, A_{23}), and one

[4] Answer A_{ij} represents the answer that is the union of answers A_i and A_j.



(a) Query

(b) KB

(c) Redundancy graph

Fig. 5. A query and a KB that produce included answers, and their redundancy graph

with three relations (A_{123}). Considering all answers, only answers A_2, A_3 and A_{23} are irredundant (see fig. 5(c)), but it is not suitable to keep A_2 and A_3. Otherwise, if only non-included answers are kept, this leads to keeping answers that were redundant to deleted answers (here A_{123}, redundant to A_{23}), which is not good.

So the best way is to deal with redundancy first (by computing a subset that is non-redundant and complete) and to delete included answers after that. In the previous example, this strategy led to the subset $\{A_{23}\}$.

4.5 Redundancy at a Considered Distance

In the query framework section, it was stated that computation of the irredundant form of the KB is a difficult problem. But we also see that computation of the redundancy graph also requires finding all projections of the KB into itself. Thus, we propose to restrict referring graphs of an answer to a portion of the base that is "near" this answer. This is also due to the fact that a referring graph that contains too much knowledge, even if it distinguishes an answer, does not help the user much. So we propose to bound referring graphs by a distance k, and to only consider vertices that are in the distance field of one of the vertex of an answer, which forms the *k-neighborhood graph* of the answer:

Definition 12 (k-neighborhood graph). *Given a KB B, a subgraph S of B, and a step k ($k \geq 0$), the k-neighborhood graph, denoted $N^k(S)$, is defined recursively by:*

- *$N^0(S) = S$*
- *$N^{n+1}(S)$ is composed of $N^n(S)$ expanded by every relation r not in $N^n(S)$ and which is linked to a concept of $N^n(S)$, and by all concepts linked to r.*

Recursion stops when $n = k$ or $N^n(S) = N^{n+1}(S)$ and returns $N^n(S)$.

It seems obvious that all definitions put forward previously should now take this constraint into account. Bounding referring graphs can be seen as a restriction of the KB, which is now considered as the union of the k-neighborhood graph of each answer, for a given distance.

Definition 13 (Truncated KB). *Given a KB B, a query Q, and a distance $k \geq 0$, the truncated KB at distance k is $B_k = \bigcup_{A \in Q(B)} N^k(A)$*

Now we can apply all previous definitions to the truncated KB. For example, an answer A of $Q(B)$ is irredundant considering the distance k iff $R = \langle A, B_k \setminus A \rangle$ is a distinguishing graph of A w.r.t. B_k (see corollary 2).

5 Building Smart Answers

In the previous section, redundancy and completeness were only studied from a theoretical standpoint. A more user-aspect oriented standpoint was introduced in the section 4.5. Even if the user gets a subset of answers that is non-redundant and complete, he/she has no way to distinguish answers, i.e. the only assumption that can be made is that there is, for each answer, a way to distinguish them from the others. Otherwise, property 1 states that the most specific referring graph is one of them, but it usually contains too much knowledge.

So what are the properties of a really good set of referring graphs returned to the user? To keep the notions of non-redundancy and completeness, only answers of such a subset of answers should have a referring graph. As all of these answers can be distinguished, all referring graphs should be a distinguishing graph. Finally, to not introduce unnecessary redundancies in the set of referring graphs, there should be only one distinguishing graph by referred answer. A set of referring graphs that fulfills these properties is called a *set of smart answers*:

Definition 14 (Set of smart answers). *A set of smart answers $\mathcal{S}$ of a query Q on a KB B is a set of distinguishing graphs of all answers of a non-redundant complete set $\mathcal{A}$ of $Q(B)$ such as $|\mathcal{S}| = |\mathcal{A}|$.*

We propose the Bounded Smart Answers (BSA) algorithm (see algo. 1) that takes the answers and a distance as parameters, and returns a set of smart answers of truncated KB at distance k such that each extension is the minimal k-neighborhood graph that distinguishes the answer[5].

Theorem 3. *Algorithm $BSA(Q(B), k)$ produces a set of smart answers of Q on the truncated KB B_k.*

Proof. In DAK, all referring graphs are constructed with the same distance. Therefore, thanks to the distance conservation property of the projection, that if there is a bicolored projection from $N^k(A_i)$ to $N^k(A_j)$, there is a projection from $N^k(A_i)$ to B_k such that image of the core of $N^k(A_i)$ is equal to A_j (i.e. $N^k(A_i)$ is a referring graph of A_j in B_k). For each answer A that belongs to an equivalent class that is not a sink, A will not be returned by DAK because of the second "foreach" of DAK. The third foreach of DAK ensures that for each equivalent class that is a sink, the algorithm will only add one (the first taken) answer that belongs to this class once the good distance is reached. Thus the union of all answers returned by DAK is a non-redundant complete subset of answers of B_k. As BSA returns only one graph per answer of the computed non-redundant complete subset, BSA returns a set of smart answers of B_k. □

[5] Note that comparisons in DAK (algo. 2) are between bicolored graphs, that is $B \leq B'$ iff there is a π from B' to B such that $\pi(B'_0) = B_0$.

```
Data: Answers $Q(B)$, a distance $k$
Result: A set of smart answers of truncated KB $B_k$
begin
    $i \leftarrow 0$
    $D \leftarrow \emptyset$ // distinguished answers
    $S \leftarrow \emptyset$ // smart answers
    while $i \leq k$ or $D \neq Q(B)$ do
        $D' \leftarrow$ DISTING_AT_K$(Q(B), D, i)$
        $D \leftarrow D \cup D'$
        foreach $A \in D'$ do
            $S \leftarrow S \cup \{\langle A, N^i(A) \setminus A \rangle\}$
    return $S$
end
```

Algorithm 1. BOUNDED SMART ANSWERS (BSA)

```
Data: Answers $Q(B)$, distinguished answers $D$, a distance $k$
Result: A set $D'$ of distinguished answers at distance $k$
begin
    $D' \leftarrow \emptyset$
    foreach $A \in Q(B) \setminus D$ do
        disting $\leftarrow$ true
        foreach $A_2 \in Q(B) \setminus \{A\}$ do
            if $N^k(A) \geq N^k(A_2)$ and $N^k(A_2) \ngeq N^k(A)$ then
                disting $\leftarrow$ false
        foreach $A_3 \in D'$ do
            if $N^k(A) \geq N^k(A_3)$ then
                disting $\leftarrow$ false
        if $disting$ = true then
            $D' \leftarrow D' \cup \{A\}$
    return $D'$
end
```

Algorithm 2. DISTING AT K (DAK)

6 Conclusion

We extended our previously defined notion of answer redundancy. Our new framework now considers answers but also descriptions (called referring graphs) that can distinguish an answer amongst others. These descriptions could be adapted to query languages of the semantic web, e.g. by giving a formal definition of the SPARQL DESCRIBE query form. Therefore this new redundancy is now linked to the whole KB. Based on this new redundancy, we refined two other previous definitions concerning subsets of answers: *non-redundancy* property (there is no redundant answer in the subset) and *completeness* (each answer is redundant to an answer of the subset). We proposed a way to construct all non-redundant and complete subsets of answers using a *redundancy graph*. We introduced the notion of smart answer and gave an algorithm that computes a set of smart answers based on a vertex neighborhood distance.

Answer redundancy arises because of open world assumption and undefined objects (generic concepts). A deeper redundancy still exists, that is not related to the formalized world (the KB), but rather to the "real world" (described by the KB). For example, it is possible that non-redundant answers "a big cat" and "a white cat" refer to a single cat of the "real world", which is big and white. The study of this new kind of redundancy can provide a foundation for using aggregation operators (e.g. number of results to a query) in graph based KB query languages.

References

1. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. Technical report, W3C (2008)
2. Leclère, M., Moreau, N.: Query-answering cg knowledge bases. In: Eklund, P., Haemmerlé, O. (eds.) ICCS 2008. LNCS, vol. 5113, pp. 147–160. Springer, Heidelberg (2008)
3. Lin, J.J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B., Karger, D.R.: What makes a good answer? the role of context in question answering. In: INTERACT (2003)
4. Croitoru, M., van Deemter, K.: A Conceptual Graph approach to the Generation of Referring Expressions. In: Proceedings of the 20th International Joint Conference on Artificial Intelligenc, IJCAI 2007 (2007)
5. Gutierrez, C., Hurtado, C., Mendelzon, A.O.: Foundations of semantic web databases. In: PODS 2004: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 95–106. ACM Press, New York (2004)
6. Fikes, R., Hayes, P., Horrocks, I.: OWL-QL, a language for deductive query answering on the Semantic Web. Web Semantics: Science, Services and Agents on the World Wide Web 2(1), 19–29 (2004)
7. Chein, M., Mugnier, M.-L.: Conceptual Graphs: Fundamental Notions. Revue d'Intelligence Artificielle 6(4), 365–406 (1992)
8. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading (1984)
9. Baget, J.-F., Mugnier, M.-L.: The SG family: Extensions of simple conceptual graphs. In: IJCAI, pp. 205–212 (2001)
10. Mugnier, M.L.: On generalization/specialization for conceptual graphs. Journal of Experimental & Theoretical Artificial Intelligence 7(3), 325–344 (1995)