Abstract

We present a novel methodology for manipulating sources in a knowledge integration scenario. First we define and exploit an appropriate data model for representing and querying the data sources. Second we propose a structured integration methodology which combines views on the sources without having to align their background knowledge. Our work provides a novel approach to integration via “materialized views”, it frees the system from having to align the sources itself — as in traditional integration systems — and it benefits from graph mechanisms that are used in a formal manner.

1 Introduction

A knowledge integration system combines knowledge from a number of distributed autonomous data sources and presents them to the user in a uniform manner. The user is then able to pose a query to the system and get answers that none of the isolated sources are able to answer by themselves. The system frees the user from having to locate the sources relevant to its queries, interact with each particular sources and manually combine the answers obtained locally. For example, consider two data sources representing information about cars: one dealing with the make and the price, the other one dealing with the pollution level. We are interested in finding the least polluting car made in Europe available for under £10000. To get our answer, we can retrieve all the cars that cost under £10000 and are made in Europe from the first data source, then feed this result to the second data source to retrieve the cars with the smallest pollution factor. If we build an integrated system containing all the information present in the two data sources, we will simply query the integrated system without worrying which data source contains what information. One of the main problems that arises in the integration process is the semantic mismatch between sources. In the previous example, if the data sources were organized in two different conceptual ways, it is difficult to merge the information they contain.

The knowledge integration problem has attracted significant attention both in the AI community (mostly from knowledge representation) and database systems community. Integration approaches have taken two different directions in the two communities. This is due to the different formalisms being employed, the methodologies, and the distinct emphasis on reasoning techniques. From the formalism perspective, databases and ontologies were the two main data
structures employed. These are represented in different manners: relational models [1], graph-based models [14], description logic formalisms [4], frames [13], etc. In the database world, the formalisms employed benefited from years of research. Ontologies were a new field to be explored. Naturally, techniques used for databases have been adapted for ontologies. That is, commonalities exist between the two worlds. The ontology world seems to adopt a more ad-hoc approach to integration, while the database world tends to structure this process more thoroughly. From the reasoning perspective a new focus on the reasoning capabilities of the integrating system has been seen in the past years. This is mainly due to the AI community, while in the database world, the focus seems to be on the query containment capabilities of different methodologies.

Approaches attempting to solve these problems can be grouped into two different categories: multi-agent systems and global information systems (based on databases or based on ontologies). Intelligent Information Integration has been a popular application for multi agent systems. Due to their approach towards seeking interoperability, multi-agent systems are already performing a integration task. Some of the approaches include: InfoSleuth [2], KRAFT [20] and [18]. Despite their advantages (dealing with dynamically changing environments, software reuse based design) multi agent systems also encounter several problems [18]: increased testing costs, failure due to a non-collaborative agent, memory structuring required etc. Also included in this category, peer to peer integration systems [3] encounter several ontology alignment challenges.

Global information systems' architecture consist of one global domain against which a number of wrapped sources are integrated. Because the sources are heterogeneous – adopting different models for storing data – the global schema has to provide an appropriate abstraction for all the data residing in all the sources. The method of integration also differs from one approach to another. The Local As View(LAV) approach describes the data sources as containing answers to views over the global schema (eg. TSIMMIS [12]).

A second approach, Global As View(GAV), describes a mediated schema containing answers to views over source relations (eg. Information Manifold [15], InfoMaster [10]).

The LAV approach allows new sources to be added and removed in a modular manner, while the GAV approach requires source descriptions to be modified when such changes occur. Query answering is straightforward in GAV (the answers can be obtained by composing the query with the views) while LAV requires a more sophisticated form of query rewriting. To get the best of both worlds a Both As View approach was proposed in the Automed project, based on reversible schema transformation sequences [14].

Given the drawbacks described above, we propose a integration method based on knowledge oriented specification of knowledge. We believe that this approach offers a new and novel way of thinking about integration. In the next section we give the full mathematical definitions of the structures employed to perform integration. We show that our formalism is rigorous and that it meets the methodological requirements of such framework.
2 Data source formal model

2.1 Preamble

This subsection first presents the main graph theoretical definitions that will be used throughout the remainder of the paper. Conceptual graphs and conceptual graph projection are then introduced as syntactical devices for representing knowledge. These two definitions are based on the CG formalisation introduced by [19]. We add semantics by using a model based interpretation. In the end of the subsection the notion of repository is introduced. This last definition is very important from our integration oriented point of view and is crucial for the next subsection which deals with the methodology of integration.

A bipartite graph is a graph where vertices can be partitioned into two sets such that no edge connects vertices in the same set. An ordered bipartite graph requires that for each node in one of the partitions, its neighbors (belonging to the other partition) are ordered. Working with ordered bipartite graphs as opposed to bipartite graphs facilitates the definition of CGs and their querying mechanisms.

Definition 2.1 (Ordered Bipartite Graph) A triple \( G = (V_C, V_R, N_G) \) is called an ordered bipartite graph if - \( V_C \) and \( V_R \) are finite disjoint sets, \( V(G) := V_C \cup V_R \) is the vertices set of \( G \), and
- \( N_G : V_R \rightarrow V_C^+ \) is a mapping, where \( V_C^+ \) denotes the set of all finite nonempty sequences over \( V_C \).

For \( r \in V_R \) with \( N_G(r) = c_1 \ldots c_k \), \( d_G(r) := k \) is the degree of \( r \) in \( G \) and \( N^i_G(r) := c_i \) is the \( i \)-neighbour of \( r \) in \( G \).

The multiset \( E(G) \) of edges of \( G \) is \( E_G = \{c, r | c \in V_C, r \in V_R \text{ and } \exists i \text{ such that } N^i_G(r) = c \} \). We further assume that for each \( c \in V_C \) there is \( r \in V_R \) and \( i \in N \) such that \( c = N^i_G(r) \) (\( G \) has no isolated vertices).

If \( G = (V_C, V_R, N_G) \) is an ordered bipartite graph with \( |V_R| = 1 \), then \( G \) is called a star graph.

A subgraph of a graph \( G \) is a graph whose vertex and edge sets are subsets of those of \( G \). A subgraph spanned by a set of edges \( A \) is a graph whose edges set is \( A \), and its node set is “spanned” by \( A \) in the original graph. This means that if an edge belongs to the subgraph then both of its defining nodes will be in the subgraph. For example, in Figure 3, the subgraph spanned by the node \( r \) and the edge \( c \) has the edge set \( \{i\} \), and the node set \( \{r, c\} \). This definition will be used when formalizing conceptual graphs querying.

If \( G = (V_C, V_R, N_G) \) is an ordered bipartite graph and \( A \subseteq V_R \), the subgraph spanned by \( A \) in \( G \) is the graph \( G[A] := (V_C^A, A, N_C^A) \), where \( N_C^A \) is the restriction of \( N_C \) to \( A \) and \( V_C^A = \{c \in V_C | \exists r \in A \text{ and } \exists i \in N \text{ such that } c = N^i_C(r) \} \).

If \( A = \{r\} \), then we simply write \( G[r] \), which is referred to as the star subgraph spanned by \( r \) in \( G \). Clearly, the graph \( G \) can be expressed as the union of its star subgraphs.

Visually, an ordered bipartite graph \( G = (V_C, V_R, N_G) \) is represented using boxes for vertices in \( V_C \), ovals for vertices in \( V_R \) and integer labelled simple curves (edges) connecting boxes and ovals (if \( c \) and \( r \) are such that \( c = N^i_G(r) \), then we have an edge with label \( i \) connecting the box representing \( c \) to the oval representing \( r \)). Figures 3 depicts such a graph.
The next two definitions introduce the concepts of support and conceptual graphs. A support is a structure that provides the background knowledge about the information to be represented in the conceptual graph. It consists of a concept type hierarchy (e.g. director, actor, theater etc.), a relation type hierarchy (e.g. directs, acts, improvises, shows etc.), a set of individual markers that refer to specific concepts (e.g. Woody Allen) and a generic marker, denoted by * which refers to an unspecified concept.

**Definition 2.2 (Support)** A support is a 4-tuple \( S = (T_C, T_R, I, *) \) where:
- \( T_C \) is a finite partially ordered set (poset), \( (T_C, \leq) \), of concept types, defining a type hierarchy (specialization hierarchy): \( \forall x, y \in T_C \ x \leq y \) means that \( x \) is a subtype of \( y \) and which has a greatest element \( \top \) the universal type.
- \( T_R \) is a finite set of relation types partitioned into \( k \) posets \( (T_{iR}, \leq)_{i=1,k} \) of relation types of arity \( i \) (\( 1 \leq i \leq k \)), where \( k \) is the maximum arity of a relation type in \( T_R \). Moreover each relation type of arity \( i \), \( r \in T_{iR} \), has associated a signature \( \sigma(r) \in T_C \times \ldots \times T_C \) \( i \) times, which specifies the maximum concept type of each of its arguments. This means that if we use \( r(x_1, \ldots, x_i) \), then \( x_j \) is a concept with type \( (x_j) \leq \sigma(r)_j \) (\( 1 \leq j \leq i \)). The partial orders on relation types of the same arity must be signature compatible, that is \( \forall r_1, r_2 \in T_{iR} \ r_1 \leq r_2 \Rightarrow \sigma(r_1) \leq \sigma(r_2) \).
- \( I \) is a set of countable set of individual markers, used to refer specific concepts.
- * is the generic marker used to refer to an unspecified concept (having, however, a specified type).
- The sets \( T_C, T_R, I \) and \( \{*\} \) are mutually disjoint and \( I \cup \{*\} \) is partially ordered by \( x \leq y \) iff \( x = y \) or \( y = * \).

A conceptual graph is a structure that depicts factual information about the background knowledge contained in its support. This information is presented in a visual manner as an ordered bipartite graph, whose nodes have been labelled with elements from the support.

**Definition 2.3 (Conceptual graph)** A (simple) conceptual graph (CG) is a triple \( SG = [S, G, \lambda] \), where:
- \( S = (T_C, T_R, I, *) \) is a support;
- \( G = (V_C, V_R, N_G) \) is an ordered bipartite graph;

![Figure 1: Simple ordered bipartite graph](image)
• $\lambda$ is a labelling of the vertices of $G$ with elements from the support $S$: 
  $\forall r \in V_R, \lambda(r) \in T_{R}^{dG(r)}$; $\forall c \in V_C, \lambda(c) \in T_C \times (I \cup \{\ast\})$ such that if 
  $c = N^G_c(r)$ and $\lambda(c) = (t_c, ref_c)$, then $t_c \leq \sigma_i(\lambda(r))$.

A sample CG is depicted in Figure 4. The graph represents that a movie (given 
by its id) has a director, a name, and a premiere date. On the left hand side 
the support is shown, with the concept / relation hierarchy. On the right hand 
side a simple ordered bipartite graph is represented, labelled with concepts and 
relations from the support. We have not depicted the set $I$, or the generic 
marker "*".

Conceptual graphs represent knowledge at a syntactic level. **Projection** 
(subsumption) is a syntactic mechanism that allows for a comparison of the 
knowledge contained in two CGs. This is done by preserving the order of the 
neighbors in the two graphs (the relation “eat” between “cat” and “mouse” 
is totally different from the relation “eat” between “mouse” and “cat”) and 
comparing the types and labels of the nodes / relations. Projection is the 
fundamental operation on simple conceptual graphs since it can be used to 
define a preorder on the set of SCGs based on the same support. Subsumption 
checking is an NP-complete problem [19]. When the size of graph used in 
practice is not large, subsumption can be done in polynomial time [7].

**Definition 2.4 (Projection, Subsumption relation)**

If $SG = [S, G, \lambda_G]$, and $SH = [S, H, \lambda_H]$ are two CG’s defined on the same 
support $S$, then a projection from $SG$ to $SH$ is a mapping 

$$\pi : V_C(G) \cup V_R(G) \to V_C(H) \cup V_R(H)$$

such that

• $\pi(V_C(G)) \subseteq V_C(H)$ and $\pi(V_R(G)) \subseteq V_R(H)$;

• $\forall c \in V_C(G)$, $\forall r \in V_R(G)$ if $c = N^G_c(r)$ then $\pi(c) = N^H_c(\pi(r))$;

• $\forall v \in V_C(G) \cup V_R(G)$ $\lambda_G(v) \geq \lambda_H(\pi(v))$. 

Figure 2: Simple conceptual graph
If there is a projection from $SG$ to $SH$ then $SG$ subsumes $SH$, denoted $SG \geq SH$. This subsumption relation is a preorder on the set of all CG's defined on the same support.

Usually, CGs are given semantics by translating them to existential first order logic formulae. We propose a semantics based on model theory, adapted for our integration purposes. In order to do this we will define what the interpretation of a support is, and how to assign that interpretation to the simple conceptual graph defined on that support. This leads to the notion of a repository of a CG on a model.

An interpretation (or model) for a support is a structure that assigns appropriate values from a domain (universe) to each concept type, relation type and marker. This assignment respects the way the relation /concept types are defined and also preserves their hierarchical order.

**Definition 2.5 (Interpretation)**

An interpretation or model $M$ for the support $S = (T_C, T_R, I, \ast)$ is a pair $M = (D, F)$ where

- $D$ is a set of objects called the domain or universe of $M$,
- $F$ is a function defined on $T_C \cup T_R \cup I$ such that $F(I) \subseteq D$, $F(T_C) \subseteq P(D)$, $F(T_R) \subseteq P(D^i)$ for each $i \in \{1, \ldots, k\}$ ( $k$ is the maximum arity of a relation type in $T_R$) satisfying:
  - $\forall t_c, t'_c \in T_C, t_c \leq t'_c \Rightarrow F(t_c) \subseteq F(t'_c)$,
  - $\forall t_r, t'_r \in T_R, t_r \leq t'_r \Rightarrow F(t_r) \subseteq F(t'_r)$.

An assignment allows to link the concepts of a CG to the domain (universe) of the model defined over its support.

**Definition 2.6 (Assignment)**

Let $M = (D, F)$ be a model for the support $S = (T_C, T_R, I, \ast)$, and $SG = [S, G, \lambda]$ be a CG, with $G = (V_C, V_R, N_G)$.

An assignment for $SG$ in $M$ is a function $f : V_C \rightarrow D$ such that

- $\forall c \in V_C$, if $\lambda(c) = (t_c, ref_c)$ then $f(c) \in F(t_c)$, and if $ref_c \in I$ then $f(c) = F(ref_c)$;
- $\forall r \in V_R$, if $deg_G(r) = i$ then $(f(N^G_1(r)), \ldots, f(N^G_i(r))) \in F(\lambda(r))$.

The set of all assignments for $SG$ in the model $M$ is denoted $A(SG, M)$. If $A(SG, M) \neq \emptyset$ then $SG$ holds in $M$ and is denoted $M \models SG$.

The soundness of projection now follows as a simple observation. Let $SG$ and $SH$ be two conceptual graphs on the same support $S$ such that $SG \geq SH$ and $M$ is a model for $S$. From each assignment $f$ for $SH$ in $M$ we can construct an assignment $f'$ for $SG$ in $M$ by defining $f'(c) = f(\pi(c))$, where $\pi$ is a projection from $SG$ to $SH$. Hence, if $SG \geq SH$ and $M \models SH$ then $M \models SG$.

Given a data source, we need to be able to link the information (set of tuples) contained therein with the conceptual graph and its model. To do this we introduce the notion of a repository. A repository is a set of tuples, each of which makes the conceptual graph true in a given model. The repository is intentional (as opposed to extensional); one needs to go through the data source to be able to build it. There is no need to materialize the repository in order to make use of it (in the manner of materialized views for databases). A repository contains all possible interpretations for the generic (marked with "*" )concepts in the graph.
Definition 2.7 (Repository)
Let \( M = (D, F) \) be a model for the support \( S = (T, T_R, \mathcal{I}, \ast) \), and \( SG = [S, G, \lambda] \) be a CG, with \( G = (V_C, V_R, N_G) \).
We set \( \Gamma_C := V_C(\ast) \cup V_C(\mathcal{I}) \), where \( \forall c \in V_C \) with \( \lambda(c) = (t_c, \text{ref}_c) \), \( c \in V_C(\ast) \) if \( \text{ref}_c = \ast \), and \( c \in V_C(\mathcal{I}) \) if \( \text{ref}_c \in \mathcal{I} \). We suppose that \( V_C(\ast) \neq \emptyset \) and that an ordering \( V_C(\ast) = \{c_1, \ldots, c_p\} \) is fixed.

A repository for \( SG \) in the model \( M \), is a set of tuples \( R(SG, M) \subseteq D^p \), such that \( \forall (d_1, \ldots, d_p) \in R(SG, M) \), the mapping \( f : V_C \rightarrow D \), defined by \( f(c_i) := d_i \), for \( c_i \in V_C(\ast) \), and \( f(c) := F(\text{ref}_c) \) for \( c \in V_C(\mathcal{I}) \), is an assignment for \( SG \) in \( M \).

2.2 Data source querying and integration
Once the data sources are defined, we need to be able to query and integrate them with other sources. In this subsection we define the main querying mechanisms for our model and how the results will be retrieved. We also formally introduce the notion of a knowledge oriented specification, and we present our integration methodology.

In order to be able to query the data sources, we introduce a structure called a query conceptual graphs. This structure was introduced in [9], but in this paper we present it in a new, graph theory oriented, light. A query conceptual graph allows one to represent a query over the sources in a conceptual graph like notation. Basically, to find all the information about a generic concept, we mark it by “?”. The “?” symbol stands for all the occurrences of a given type in the repository, which make the graph hold. The QCG has an associated simple conceptual graph, whose intuitive purpose is to represent the query graph without any “?”. Later on, when defining an answer over QCGs this graph will be important because it will help validate answers.

Definition 2.8 (Query Conceptual Graph)
A query conceptual graph (abbreviated qCG) is quadruple \( Q = [SQ, \text{arity}, X, \lambda'_Q] \), where
- \( SQ = [S, Q, \lambda_Q] \) is a CG with \( Q = (V_C, V_R, N_Q) \),
- \( \text{arity} \) is a positive integer,
- \( X \subseteq V_C(\ast) \), and
- \( \lambda'_Q : X \rightarrow \{?_1, ?_2, \ldots, ?_{\text{arity}}\} \) is a surjective labelling (with query marks).
\( SQ \) is the conceptual graph associated to qCG \( Q \), \( \text{arity} \) is the arity of \( Q \), and \( X \) are the query concept vertices of \( Q \).

If \( SG \) is a CG and \( Q \) is a qCG on the same support \( S \), then we say that \( Q \) is legal for \( SG \) if \( Q \), the conceptual graph associated to \( Q \), is obtained from a spanned subgraph of \( SG \) by replacing the generic marker of some concept vertices of this subgraph by individual markers (clearly, if \( SQ \) is obtained in this way from \( SH \subset SG \), then \( SH \geq SQ \)).

An answer to a QCG is the set of all data retrieved from the repository that validate the QCG. Intuitively, by taking all the instances from the repository that make the graph associated to the QCG true, one obtains its answer. This notion is very important because it helps us define a knowledge oriented specification for a given source.
Definition 2.9 (Answer to a qCG)
Let \( Q = [S_Q, \text{arity}, X, \lambda^Q_0] \) be a legal qCG for the CG \( SG = [S, G, \lambda] \), where
\[
G = (V^1_G, V^2_G, N_G), \quad V^1_G = \{ e_1, \ldots, e_p \}, \quad S_Q = [S, Q, \lambda_Q], \quad Q = (V^1_Q, V^2_Q, N_Q),
\]
\[
V^2_G \subseteq V^1_Q, \quad V^2_G = V_C([V^2_G]_C) \quad \text{and} \quad X \subseteq V^2_Q.
\]
Let \( M = (D, F) \) be a model for the support \( S \) and \( R(SG, M) \) be a repository for \( SG \) in the model \( M \).

We define the answer to \( Q \) over \( R(SG, M) \) as being the set \( \text{Ans}(Q, R(SG, M)) \) of all tuples \( (d_1, \ldots, d_{\text{arity}}) \subseteq D^\text{arity} \) that satisfy the following property: there is \( (d'_1, \ldots, d'_p) \in R(SG, M) \) such that, if \( \lambda_Q(e_j) \in I \) then \( d'_j = F(\lambda_Q(e_j)) \), and for each \( i \in \{1, \ldots, \text{arity} \} \), if \( \lambda^Q_0(e_j) = ?^i \) then \( d'_j = d_i \).

Consider the example in Figure 5. A simple query conceptual graph is represented, that asks for the year and titles of the movies directed by Woody Allen. In order to retrieve the complete answer we check the tuples from the data source (with the appropriate type) and return the ones that make the graph hold. In our example the answer will contain the tuples \{2005, Match Point\} and \{1975, Love and Death\}. Note that the number of records in the answer is the same as the number of question marks in the query.

All of the above definitions now lead to the formal definition of a knowledge oriented specification. This is the main contribution of the paper, allowing us to express our integration method in a rigorous, theoretical manner. A knowledge oriented specification of an information source is (i) a conceptual graph that visually describes that source, (ii) an interpretation for the support on which the graph is built, (iii) a repository for the graph (that contains all the data tuples) and (iv) a wrapper that ensures the communication in between the user queries and the repository.

Definition 2.10 (Knowledge Oriented Specification of an Information Source) Let \( IS \) be an information source. A knowledge oriented specification of \( IS \) is a quadruple \( \text{KOS}(IS) = (SG, M, R(SG, M), W) \), where
- \( SG = [S, G, \lambda] \) is a CG on the support \( S \), source support,
- \( M = (D, F) \) is a model for the support \( S \), source model,
- \( R(SG, M) \) is a repository for \( SG \) in the model \( M \), and
- \( W \) is a wrapper, that is a software tool which, for each legal qCG \( Q \) for \( SG \), returns the answer set \( \text{Ans}(Q, R(SG, M)) \).

Once formally defined, the data sources can be integrated. This is the purpose of a CG Mixer. As described in Figure 1, a CG Mixer depicts the integrated view, by the means of a conceptual graph, and provides the rules to allow for the translation of user queries to the appropriate data sources. The rules are defined by the relation vertices from the integrated view. As shown in Figure 1, for each relation in the integrated view, the proper translation is provided.
This translation has to preserve the order of nodes in the initial relation, hence the extra labelling of concepts (as depicted in greyed out rectangles).

**Definition 2.11 (CG Mixer)** Let \( IS^1, \ldots, IS^n \) be a set of information sources, and their knowledge oriented specifications \( KOS(IS^i) = (SG^i, M^i, R^i(SG^i, M^i), W^i), i = 1, n \).

A CG Mixer over the information sources \( IS^1, \ldots, IS^n \) is a pair \( M(IS^1, \ldots, IS^n) := (SG^0, R) \), where:
- \( SG^0 = [S^0, G^0, X^0] \) is a CG with \( G^0 = (V^0_R, V^0, N_{CG^0}) \), and \( R \) is a mapping which specifies for each \( r^0 \in V^0_R \) a set \( R(r^0) \) of rules providing descriptions of the relation vertex \( r^0 \) in (some of) information sources. Each rule in \( R(r^0) \) is a triple \( (IS^k, A, w) \), where:
  - \( IS^k \) is an information source specified by \( KOS(IS^k) \)
  - \( A \subseteq V^k_R \) (the relation vertices set of \( SG^k \))
  - \( w \in V^k_G(A)CG^k \) is a sequence of \( d_{CG^k}(r^0) \) concept vertices of the subgraph \( ACG^k \) spanned in \( G^k \) by the relation vertices in \( A \).

Rule \((IS^k, A, w) \in R(r^0)\) means that the star graph \( G^0[r^0]\), is translated in the source \( IS^k \) as \( ACG^k \) and if \( w = w_1 \ldots w_k \) \((k = d_{CG^k}(r^0))\), then \( w^0 \) corresponds to \( N_{CG^0}(r^0) \) \((i = 1, k)\). In other words, a rule interprets each relation vertex in the CG Mixer via a subgraph of the CG describing the appropriate local source. This is done by means of an ordered sequence of concept vertices (the relations’ vertex neighbours).

The CG Mixer is constructed manually by domain experts (who understand the knowledge oriented specifications of the sources). The quality of the integration system depends on the complexity of CG \( SG^0 \) and the quality of the rules.

The conceptual graph \( SG^0 \) represents the visual querying interface provided to the user, in order to describe the graphical language in which (s)he can interrogate the system.

### 2.3 Query answering and rules

In this subsection we describe the methodology of query answering. We explain how the rules associated to the integrated view are redirecting the user queries on the sources. We detail this process in an algorithmic manner and formally define the answer of a query over a CG Mixer.

**Definition 2.12 (Querying a CG Mixer)**

Let \( M(IS^1, \ldots, IS^n) := (SG^0, R) \) be a CG Mixer. A legal query over \( M(IS^1, \ldots, IS^n) \) is any legal QCG for \( SG^0 \).

Let \( Q = [SQ, arity, X, \lambda_Q] \), be a legal QCG for \( SG^0 \), with \( SQ = [S, Q, \lambda_Q] \), \( Q = (V_C, V_R, N_Q) \), and \( X \subseteq V_C(*) \). Let \( V_R = \{r^0_1, \ldots, r^0_m\} \) and \( H = \{[r^0_i, \ldots, r^0_m]\}CG^0 \) (the spanned subgraph of \( G^0 \) from which is obtained \( SQ \) by specialization).

From \( SQ \) a set \( R(SQ) \) of graphs is constructed as follows:

- For each \( r^i_0 \) \((i = 1, m)\) take a rule \((IS^{k_i}, A^i, w^i) \in R(r^i_0)\).
- In each graph \( ACG^k \), if the vertex \( w^j_0 \) has a generic marker in \( SG^{k_i} \), and in \( SQ \) the \( j \)-neighbour of \( r^i_0 \) has been replaced by an individual marker, then the generic marker of \( w^j_0 \) is replaced by this individual marker.
- Consider the union of all these graphs.
- Add to the graph obtained a special set of new vertex relations in order to describe the neighborhood structure of the original graph $H$. All these vertices have the special label (name) "$ = "$ and have exactly two neighbours (with the meaning that the corresponding concept vertices represent the same object). More precisely, if $N^t_H(r^0_i) = N^s_H(r^0_j)$ (in $H$ the $t$-neighbour of $r^0_i$ is the same concept vertex as the $s$-neighbour of $r^0_j$), then a new equality relation vertex is added to the graph already constructed, with the 1-neighbour vertex $w^1_i$ of $[A^i]_{G^v}$, and the 2-neighbour vertex $w^2_j$ of $[A^j]_{G^v}$.

The graphs from the set $R(SQ)$ can be considered as the set of all possible query rewritings of $Q$.

Each graph $RH \in R(SQ)$ can be expressed as a disjoint union of source subgraphs, interconnected (as described above) by the equality relation vertices. Let $RH_j$ be the (nonempty) subgraph of the graph $SG^j$ of the source $IS^j$. For each concept vertex $w^k_j$ in $V_C(RH_j)$ (which means that there is $r^0_i$ for which a rule $(IS^j, A^j, w^k_j)$ has been used in the construction of $RH_j$), if $N^k_{G^0}(r^0_i)$ has a query marker, then assign a query marker to $w^k_j$. The superscripts of these new query markers can be numbered such that they will form a set \{1, ..., arity'\} and also respect the meaning in $Q$ (that is, if the two original vertices have the same query mark, then their surrogates have the same new query marks). In this way, we have obtained a legal qCG $Q^j_{RH}$ for $SG^j$.

**Definition 2.13 (Answer to a qCG over a CG Mixer)**

Let $M(IS^1, ..., IS^n) := (SG^0, R)$ be a CG Mixer, Let $Q = [SQ, arity, X, \lambda' Q]$, be a legal qCG for $SG^0$.

We define the answer to $Q$ over $M(IS^1, ..., IS^n)$ as being the set $Ans(Q, M(IS^1, ..., IS^n))$ of all tuples $(d_1, ..., d_{arity}) \subseteq D_{arity}$ constructed as follows.

For each graph $RH \in R(SQ)$ consider the corresponding qCGs $Q^j_{RH}$, find the sets $Ans(Q^j, R^j(SG^j, M^j))$ (using the wrapper $W^j$), combine the tuples from these sets such that the equality relation vertices of $RH$ to be satisfied, and the set of all successful combinations is $Ans(RH)$. Finally,

$$Ans(Q, M(IS^1, ..., IS^n)) := \bigcup_{RH \in R(SQ)} Ans(RH).$$

### 3 Discussion

In this paper we introduce a novel knowledge-oriented specification technique for data integration. We believe that this technique can be used in different areas such as data and ontology integration, information management and retrieval.

Data sources in our approach to integration are represented using conceptual graphs (CGs). A Knowledge Oriented Specification (KOS) for a knowledge source is a conceptual graph that syntactically describes the data. The specification does not try to exhaustively describe the sources, but it provides a description of the accessible data. These sources are linked to their KOS by means of a wrapper and encapsulated within an integrated schema. A set of rules associated with the schema directs every query to the appropriate sources during the querying process. Individual query results are then combined and presented to users.
No existing approaches have represented the integration problem in the way
described above. We believe that this approach offers a new and novel way
of thinking about integration. This approach is similar in spirit to techniques
introduced by Ehrig and Staab for QOM [11].

While we have not yet created a concrete implementation of our approach,
its formal presentation, as described in this paper, should simplify the imple-
mentation process. In the future we intend to integrate the KOS specification
with the LCG formalism introduced in [6].

References


tion in open and dynamic environments. In *Proc. of the ACM SIGMOD

tions of peer-to-peer data integration. In *Proc. of the 23rd ACM SIGACT
SIGMOD SIGART Sym. on Principles of Database Systems (PODS 2004)*,

information integration. *Computational Logic: From Logic Programming
into the Future (In honour of Bob Kowalski)*, Lecture Notes in Computer

databases using conceptual graphs. In W. M. Tepfenhart, J. P. Dick, and

integration using layered conceptual graphs. In Springer, editor, *Prec. of 3th
To appear.

Comp. Society’s Specialist Group on Artificial Intell. (AI’2004)*, pages 130–

Mineau, editors, *ICCS*, volume 1867 of *Lecture Notes in Computer Science*,

[9] F. Dau. Query graphs with cuts: Mathematical foundations. In A. Black-
well, K. Marriott, and A. Shimojima, editors, *Diagrammatic Representation
and Inference. Third International Conference, Diagrams.*, volume 2980.


